ENG 1 - Method Selection and Planning

Jacob Tangye
Joseph Wipat
Isaac Williams-Mayo
Hasan Syed
Huzaifa Thakur
Alex Worsley
Charlie Taylor

Software Engineering Methods and Tools

Methods: The team adopted Agile methodologies to accommodate iterative development and constant feedback. Agile emphasises flexibility, collaboration, and regular communication, which suited the team's structure and project scope. By breaking down the project into smaller tasks, assigning team members specific roles, and holding regular reviews, the team was able to adapt swiftly to changes, ensuring that all tasks met their deadlines while maintaining quality. We considered using the Waterfall model, which involves taking each section one by one as a group, (i.e. we would all work on requirements, then all on risk assessment etc), however, this approach felt too stagnant, as it would result in 7 people all trying to work together on a single document, which would lead to a lot of clashes and the ability for more disagreements, so as a result, we chose to stick with Agile.

Communication: The team used Instagram group chat for communication due to its accessibility and familiarity. This choice provided fast response times, vital for maintaining fluid communication during development. The team considered using Discord, a platform widely used for collaboration and file sharing, but as most members were more active on Instagram, this increased efficiency, especially because the file sharing aspect of Discord wasn't needed due to our use of other software for file management. The use of Discord had its benefits though, such as the ability to have multiple different channels if we created a server, so we could keep all "requirements" stuff in a single tab, but the pure ease of creating an Instagram group chat deemed it simply more effective for us to create one of those instead.

File Management: Google Drive was used to organise files for each aspect of the project, including requirements, architecture, and risk assessments. The benefits of Google Drive over alternatives like Dropbox or OneDrive include:

- **Collaboration**: Real-time file sharing and editing allow multiple users to work simultaneously, essential for Agile workflows.
- **Organisation**: Google Drive's folder system provided structure, making it easy to access relevant project documents.
- **Accessibility**: Drive is accessible across multiple platforms, enabling team members to access files from any device. Other alternatives, such as Dropbox, do not offer as seamless collaborative editing tools as Google Docs does.
- **Set up:** We all already have access to Google Drive, and are all familiar with using the Google software, so it wouldn't require us to learn how to use any new software
- **Tracking:** On Google Drive and the related apps, you can see who updated things when and where, allowing us to keep track of which bits of work have been done by which people, to both show what works we did do, and point out what work others didn't do

Dropbox and OneDrive were both good alternatives that we considered using, especially OneDrive, due to it being a Microsoft program and so being well optimised to use the other Microsoft programs (mainly word), but due do the accessibility and ease of setting up and using Google Drive, we decided to go with the Google option for our project.

Game Development: The team used **libGDX**, a Java-based game development framework, for the game's development. It was selected due to its cross-platform support, specifically working across Windows and other operating systems. Unlike Unity or Unreal, libGDX is lightweight and designed specifically for Java developers, making it a better fit for the team's expertise. It also provides:

- **Customizability**: The framework offers low-level access to OpenGL for custom game rendering.

- **Flexibility**: It supports 2D and 3D development without the overhead of a heavy game engine like Unreal, which is more suited for large-scale, graphically intensive projects.

Unity, while more user-friendly, adds extra overhead and licensing costs for certain features, which makes it less appealing, and also doesn't natively support Java to program in. Unreal was considered too complex for the game's needs and yet again doesn't natively support Java, and JMonkey was discarded due to its steeper learning curve for those unfamiliar with it and focus on the development of 3D games.

Website Hosting: The team hosted the project website on **GitHub**, taking advantage of GitHub Pages, which allows the hosting of static websites directly from a repository. The main benefits of GitHub for the team included:

- **Version Control**: Git allows team members to track changes, make commits, and use pull requests to propose modifications. This ensures that everyone works on the most updated version and reduces the risk of conflicting changes.
- **Collaboration**: A shared repository ensures all team members can contribute, view, or modify the website. Other free services like GitLab offer similar version control, but GitHub is more widely used, ensuring broader community support.

We considered using Netlify to host our website, as Netlify has a lot of positives to it, such as automatic deployments from Git repos, a higher level of customization, and a global Content Delivery Network, allowing users from all around the world to be able to access it faster, but it also has a higher learning curve to implement things, and several limitations on the free tier, whereas GitHub would be completely free to use. As a result, we decided to use GitHub to host our website in the end.

Assets: In this stage of the project, we decided against using assets and to just use placeholder coloured tiles to represent each different tile or building, but we still considered which websites we would use when it came to using assets, and the main 2 were Itch.io and CraftPix.net. Itch.io has a decently large asset selection set, but that isn't the website's main purpose, and the licensing and allowed uses of the assets you get from the website are up to the control of the person who uploaded the assets, meaning that it's different for each asset. CraftPix.net however is a dedicated 2D asset marketplace, so it has a much larger catalogue of available assets, and the licensing for them are done by CraftPix themselves, and unless specifically stated otherwise, includes royalty-free and non-exclusive licences, meaning for the case of our project, we wouldn't need to worry about getting the licence to use the assets from CraftPix in our project.

IDEs: There were a lot of different IDEs which our team considered using for the project. The recommended IDE to use with LibGDX is Android Studio, however, due to none of our team having ever used it, we decided against using it, and instead debated over whether to use VS Code or IntelliJ IDEA. VS Code was more lightweight and customizable, however it doesn't natively support Java. IntelliJ IDEA however is designed for Java development and is at default a full featured IDE, and multiple members of our team were already familiar with IntelliJ IDEA, so the decision seemed easy to make between which IDE we would use. Although you can use multiple different IDEs simultaneously and there won't be any outright compatibility issues, we decided to all use the same IDEs to avoid any confusion and clashes with different Java versions or unavailable functions, and so we all decided to use IntelliJ IDEA as our IDE.

Team Organization Approach

The team adopted a hierarchical structure with a designated leader, Jacob. His role involved making sure we don't get sidetracked and helping ensure meaningful progression. This leadership structure was effective for the team for several reasons such as acting as a central point of contact, distributing work and ensuring that no one was overwhelmed.

In addition to appointing Jacob as the team leader, the roles were clearly divided to streamline the workflow and maximise efficiency. Each team member took responsibility for specific aspects of the project:

- **Hasan and Isaac** handled **risk management**, analysing potential project risks and preparing mitigation strategies. Their work was vital for ensuring that issues such as delays, technical hurdles, or resource limitations were anticipated early and addressed.
- **Alex and Hasan** were tasked with **game architecture**. They designed the underlying structure of the game, ensuring it was scalable, efficient, and adaptable for future changes. This task involved defining core components such as game logic, graphics systems, and input handling.
- **Huzaifa** took charge of the **website development** and **method selection and planning**. He was responsible for hosting the project on GitHub and ensuring that it was easily accessible to the client and team. His role in method selection helped define the Agile approach that guided the team's workflow..
- **Jacob and Joe** worked on the **requirements gathering**, ensuring that all client expectations were clearly understood and documented. They translated the client's vision into concrete game features and deliverables.

Parallel Development and Team Coordination: While each member focused on their individual roles, they worked in parallel to ensure that no time was wasted. For example, as Hasan and Isaac worked on risk management, the others progressed with their tasks. The deadlines were staggered to allow for dependencies, such as the architecture deadline preceding the start of the full implementation and as certain people members were free they helped with the others tasks.

The team's approach was appropriate for both the project and team because it:

- Allowed each member to focus on their area of expertise, maximising output quality.
- Facilitated early detection of problems in risk management, enabling timely adjustments.
- Ensured smooth handoffs between phases, particularly between architecture and implementation.

The team held **two meetings per week** to review progress, resolve challenges, and brainstorm improvements. These regular meetings were essential for:

- **Transparency**: Ensuring everyone was on the same page and that all aspects of the project were progressing in sync.
- **Flexibility**: Early identification of potential blockers allowed the team to shift priorities and adapt quickly.

This organisation method was appropriate for both the project and team because it emphasised collaboration, task division, and continuous feedback, all of which were necessary for timely delivery.

Project Plan

The project was divided into key tasks, each with its own timeline, priority, and dependencies:

1. Website Development (Huzaifa)

- **Start Date**: October 1
- **End Date**: October 17
- **Priority**: High
- Huzaifa worked on the project website, hosted on GitHub, and he ensured the website was up-to-date with the project timeline

2. Requirements Gathering (Jacob and Joe)

- **Start Date**: October 1
- **End Date**: October 20
- **Priority**: High
- **Dependency**: None
  This phase focused on gathering and documenting the game's functional and non-functional requirements, ensuring alignment with the client's needs.

3. Risk Management (Hasan and Isaac)

- **Start Date**: October 3
- **End Date**: October 10
- **Priority**: Medium
- **Dependency**: None
  Hassan and Isaac identified potential risks (technical, resource-related) and devised mitigation strategies.

4. Game Architecture Design (Charlie and Alex)

- **Start Date**: October 5
- **End Date**: October 24
- **Priority**: High
- **Dependency**: Requirements must be clear before architecture design begins.
  This task involved defining the game's structural components, including class hierarchies, game mechanics, and system interactions.

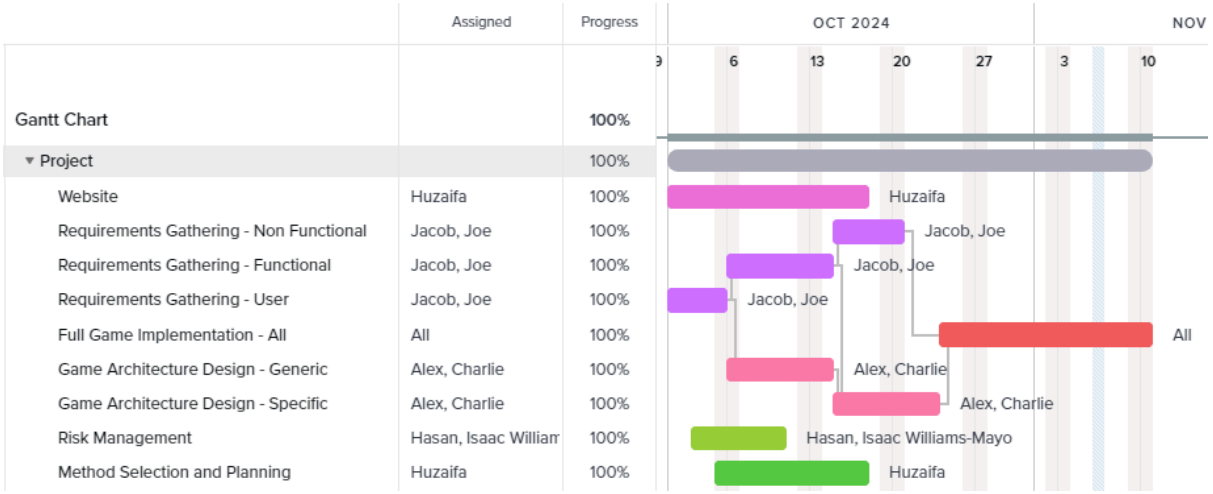
5. Method Selection and Planning (Huzaifa)

- **Start Date**: October 5
- **End Date**: October 17
- **Priority**: Medium
- **Dependency**: None
  Huzaifa convened with the team which selected the Agile method for coordination.

6. Full Game Implementation (All Team Members)

- **Start Date**: October 25
- **End Date**: Before client's final deadline
- **Priority**: High
- **Dependency**: Completion of the architecture design
  Once Charlie and Alex completed the architecture, the team began full implementation, working in parallel on different game components such as gameplay logic, graphics integration, and testing.

Gantt Chart (Isaac)



The above Gantt Chart is a visual representation of our plan, the start and end dates for each process, and the connections between them. The main links are between the requirements and the architecture, as the architecture required the requirements to be complete in order for it to be undertaken. This was not however realised at the beginning, as the previous plan stated, as we assumed that the architecture could be made before the completion of the requirements, however, when we came to understand that requirements must be done before the architecture could be begun, we changed our plan to accommodate for this.

The requirements and architecture have been split up in this Gantt Chart in order to highlight that some parts of the architecture could be completed without the full set of requirements being done, but the overall critical path of the project required all the requirements to be completed, with the user and functional requirements being more so important to be completed first, as the process of developing the architecture relied on them being done in order for them to be undertaken.

The risk assessment and planning, alongside the risk management, were not members of the critical path, as even though they are crucial to the completion of the project, no other part of the project relied on them being completed before it could be done, as a result, they were only put on medium priority (as shown by being shades of green), whereas everything else was critical to being completed, as they are being shown in shades of purple and red. The website may not have had anything else hinging on its completion along the critical path, but due to the nature of the project requiring everything to be displayed through the website, it was of utmost importance that it was completed as well, thus we put it on High priority too.