Method Selection And Planning

Group 14

Tecch Titans:

Bradley Mitchell

Daniz Hajizada

Ellie Gent

Joel Crann

Keela Ta

Leo Crawford

Lukas Angelidis

Software Engineering Methods

The software engineering process can consist of a set of activities that will take place in a certain order typically linearly until more recently in industry where more rapid or otherwise agile approaches are taken to software development. More agile approaches are often geared towards the benefit of receiving more customer feedback throughout a project but are usually not best suited for smaller projects.

Software engineering methods allow for different ways of approaching both software development and the delivery of such software. For example, the Waterfall approach is one of the most well known methods that is very rigid and best suited to smaller development projects.

It can be of the utmost importance to choose the correct engineering method and collaboration tools for a team within a project to make sure that assigned deliverables are created to a high standard within the timeframe and resource space of the project at hand.

Methodologies such as test driven development were considered as this would allow for an extremely reduced chance for failures and unnecessary code that would create inefficiencies and a reduced readability for other students choosing to take the project further, however, as a group we did not feel that we had the experience with software development to most effectively implement this.

The team, instead, have chosen a short agile development strategy based upon an order driven by the weightings of our task priority set within a greenfield development setting with, although, given the short timeframe, the methodology will not shy too far from what would resemble a rigid structure.

We were able to justify the order of completion within our agile methodology based upon the dependencies between the deliverable development aspects and their precedence in terms of the marks that these attain and, thus, where the team's time and capable resource would be best spent forming this into a WBS diagram and then a Gantt diagram to make sure that we would be able to complete what we needed to in an order best suited to our team, our capabilities and our experience with the best orders to software development.

In terms of collaborative tools to facilitate the group work of our teams, we chose to use discord for the main segmentation of our communications for role specific applications and Instagram for more casual communications and organisations. We also kept a logbook to keep track of and better share the progress and updates taking place across our project also helping with the updates to our WBS and Gantt diagrams taking place across the project. Of course, we also used git as our communal version control system allowing for effortless implementation between multiple members while, in some cases, we were able to make use of pair programming also.

We did not choose to use Slack as a team as we were very much unfamiliar with the platform and using platforms that we were already very comfortable with, we knew would save good time for us especially since we were aware that there was already going to plenty for us to learn also especially under the implementation branch of deliverables.

On the whole, this engineering methodology has worked very well for the team's suitability and compatibility and has allowed us to learn some valuable insight into the approaches that can be applied within software development projects while also allowing people to work with their strengths and comfortabilities at different points within the project.

Team Organisation

Throughout the project, teamwork and team organisation was crucial to presenting deliverables of the highest standard. Our approach from the start was to assess each individual's strengths, weaknesses and comfortability with tasks, based on previous group work modules and personal choice. The two most crucial aspects of team building, forming and storming, were taken into consideration early on in the project. These helped form the basis for communication, collaboration and designation of responsibilities.

The team initially focused on collaboration in the early stage of development, as many later stages of development were dependent on Requirements. Key team values, such as communication for lateness/absences, and a responsibility to catch up on missed work, were established. A logbook was developed to track progress across all deliverable areas, with a team member from Method Selection and Planning assigned this responsibility, alongside a group leader role. This ensured task completion between meetings, and provided a detailed timeline great for assisting the updates of WBS and Gantt diagrams.

Roles and responsibilities were assigned, after a discussion about strengths and weaknesses in each deliverable area. It was collectively decided to assign three team members to deliverables worth at least 20 marks, two members to the deliverables with approximately 10 marks and one team member to the Website, worth 3 marks ensuring each team member contributed at least 15 marks to the final submission, as required. Roles were assigned based on strengths, with confident coders on Implementation.
Some took the opportunity to develop their weaker areas, such as research and written-based deliverables like Requirements. This fostered new perspectives during the development of these deliverables.

Decisions were made unanimously, with each member voting on suggestions. All members contributed to developing questions for the customer meeting to gain deeper insight into each deliverable. This was crucial for the Requirements and Implementation team, as most tasks were dependent on these two deliverables. For each high-mark-weighted deliverable, a "leader" was assigned to make sure that no aspect of the deliverable was left overlooked.

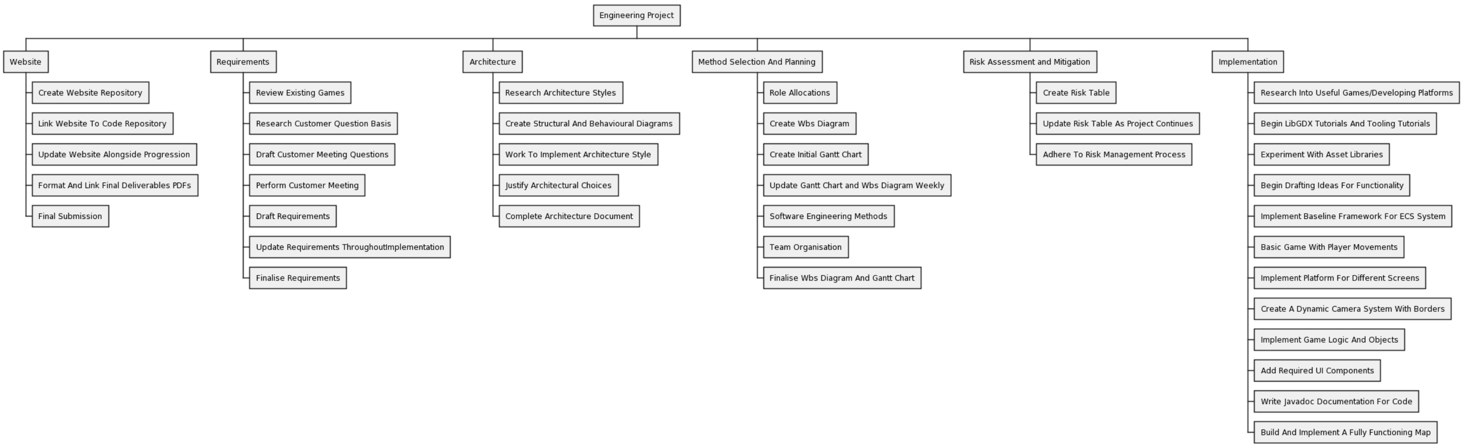
Communication within the group was made through separate Discord channels based on deliverables and developments in each task were presented to the other group members during our meetings. Instagram was also used as a general communication channel for more casual conversation and organising whole team meetings. These platforms proved useful for the group and project as a whole, as it allowed for a clear flow of communication for those working on the same task, and deliverables that were dependent on each other could receive frequent updates on progress.

In summary, the team's organisation and communication not only suited the team, but also aligned with the project brief as it fostered clear work structures and deliverable communication. This meant that when modifications were made to the original project plan, it allowed the reassignment of individuals from their current tasks to those that required immediate attention, without any confusion. The assigned yet flexible leadership developed trust among members and led to the excellence of deliverables.

Project Plan

The task assigned was to build a single-player game simulation of a second-year computer science student a week before their finals. Our engineering project consists of 6 main deliverables; website, requirements, architecture, method / planning, risk assessment and implementation. The importance of each section ties into the whole project, with implementation, risk assessment and architecture contributing the bulk weight of marks for the final submission. Our objective was to successfully present these deliverables, in line with the customer's needs, as well as fitting the initial product brief.

The 6 key tasks for the project are presented in the **Work Breakdown Structure (WBS)** below, each of which are split into individual tasks to complete throughout the duration of the project. The WBS helped us to split workload equally, via a hierarchical breakdown of all project deliverables, and also, plan for the weeks ahead with a better understanding of the task at hand. From this structure, it was easier to identify components such as task priority and dependencies. Roles were then assigned to each task, with greater precedence for requirements, architecture and implementation, requiring more time for completion. Deliverables, such as the website, planning and risk assessment required constant update throughout development and, therefore, also great consideration when planning our project.

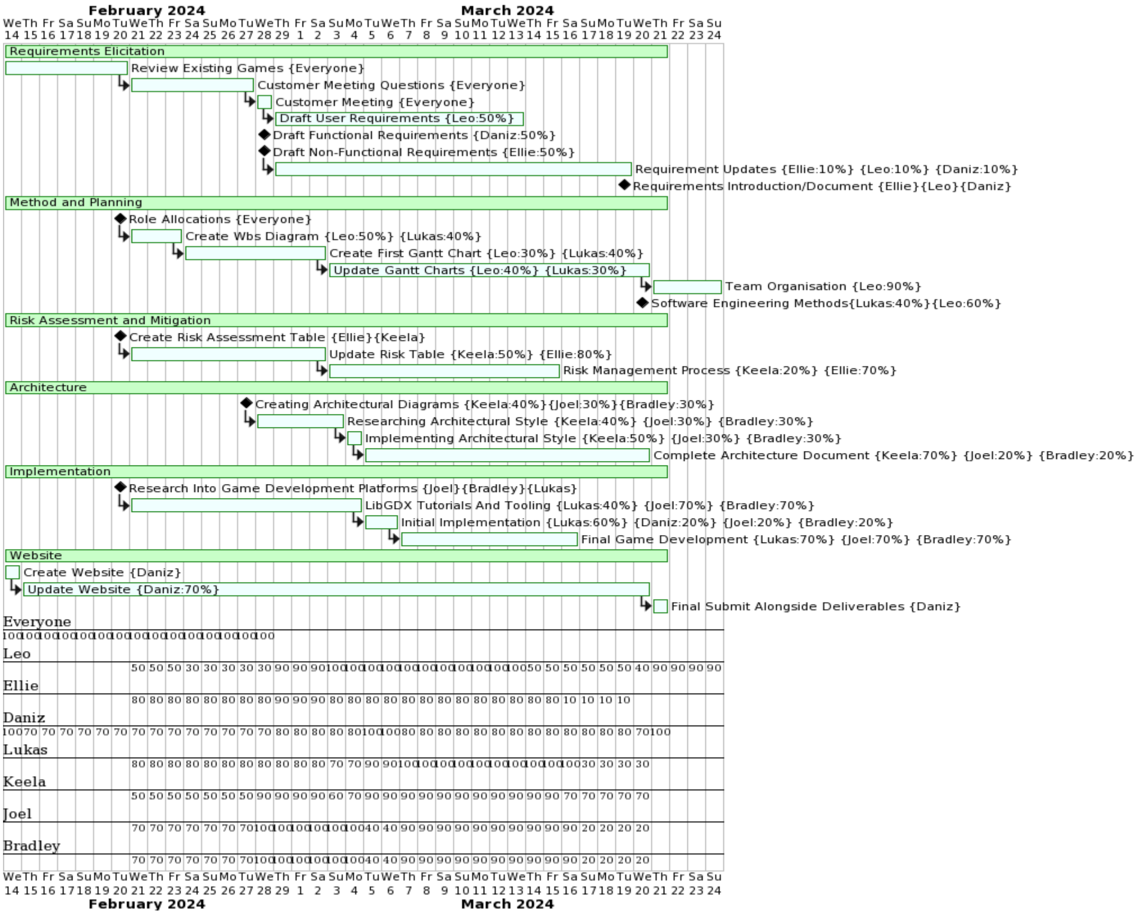

Engineering Project
Website
Create Website Repository
Link Website To Code Repository
Update Website Alongside Progression
Format And Link Final Deliverables PDFs
Final Submission
Requirements
Review Existing Games
Research Customer Question Basis
Draft Customer Meeting Questions
Perform Customer Meeting
Draft Requirements
Update Requirements ThroughoutImplementation
Finalise Requirements
Architecture
Research Architecture Styles
Create Structural And Behavioural Diagrams
Work To Implement Architecture Style
Justify Architectural Choices
Complete Architecture Document
Method Selection And Planning
Role Allocations
Create Wbs Diagram
Create Initial Gantt Chart
Update Gantt Chart and Wbs Diagram Weekly
Software Engineering Methods
Team Organisation
Finalise Wbs Diagram And Gantt Chart
Risk Assessment and Mitigation
Create Risk Table
Update Risk Table As Project Continues
Adhere To Risk Management Process
Implementation
Research Into Useful Games/Developing Platforms
Begin LibGDX Tutorials And Tooling Tutorials
Experiment With Asset Libraries
Begin Drafting Ideas For Functionality
Implement Baseline Framework For ECS System
Basic Game With Player Movements
Implement Platform For Different Screens
Create A Dynamic Camera System With Borders
Implement Game Logic And Objects
Add Required UI Components
Write Javadoc Documentation For Code
Build And Implement A Fully Functioning Map

Following the WBS, a **task priority table** was created to better understand the deliverables that were of most important focus during development of the project. Each deliverable is ranked in a low-high scale for priority, and urgency. The deliverables that were of the highest importance carried the most marks in the project and the most urgent were those that other deliverables had dependencies upon.

Task	Importance	Urgency
Website	Low	Low
Requirements	High	High
Architecture	High	High
Method Selection and Planning	Medium	High
Risk Assessment and Mitigation	Medium	Medium
Implementation	High	High

After a group discussion, we decided to analyse these **task dependencies** in further detail. We concluded that Implementation was the most important deliverable to develop since it had the most other deliverables with this as a dependency. Requirements were also found to be a very important deliverable in order to develop the Architecture and Implementation. Method Selection and Planning allowed for consistency to follow schedule and ensure that the deadline was met with a high-standard of presentation. Highlighting the dependencies of each task was essential when it came to developing our first gantt chart for the project.

The **Gantt chart** below is our first implementation of the project plan as a whole. It is a visual representation of our project schedule, showing each task, their duration, individual contribution, dependencies and progress over time. Each dark green bar represents each of the 6 deliverables, and the light blue bars represent the individual tasks needed to complete them. Timeline was taken into consideration to allow all deliverables to be completed and ready to submit for the 21st of March.



As the project evolved and was carried out over time, as also seen by our Gantt diagram snapshots, our plan was able to remain mostly in line with our original schedule, however, our timings were often influenced by external obligations which was sometimes beneficial to finish many tasks within our project earlier than expected. Many aspects of implementation, although, ended up taking longer than expected forcing us to amend our plans slightly and so, we also added the help of an additional team member to this deliverable to ensure completion in time and completion to a good standard. Of course, we made sure to do this early enough to avoid Brooke’s Law’s and as a team, we also made sure that awareness of what was taking place amongst the rest of the team was kept clear throughout the project.