Engineering 1

Assessment 1

Cohort 3

Team 23

Seif Hussein

Joshua Mckean

Harry Draper

Sebastian Armstrong

Thomas Maalderink

Tikhon Likhachev

Software Engineering Methods and Tools

After looking at various software engineering methods and frameworks, discussing with each other their advantages and disadvantages and which suits our team and our project the best, we chose the Agile method to be our software engineering method.

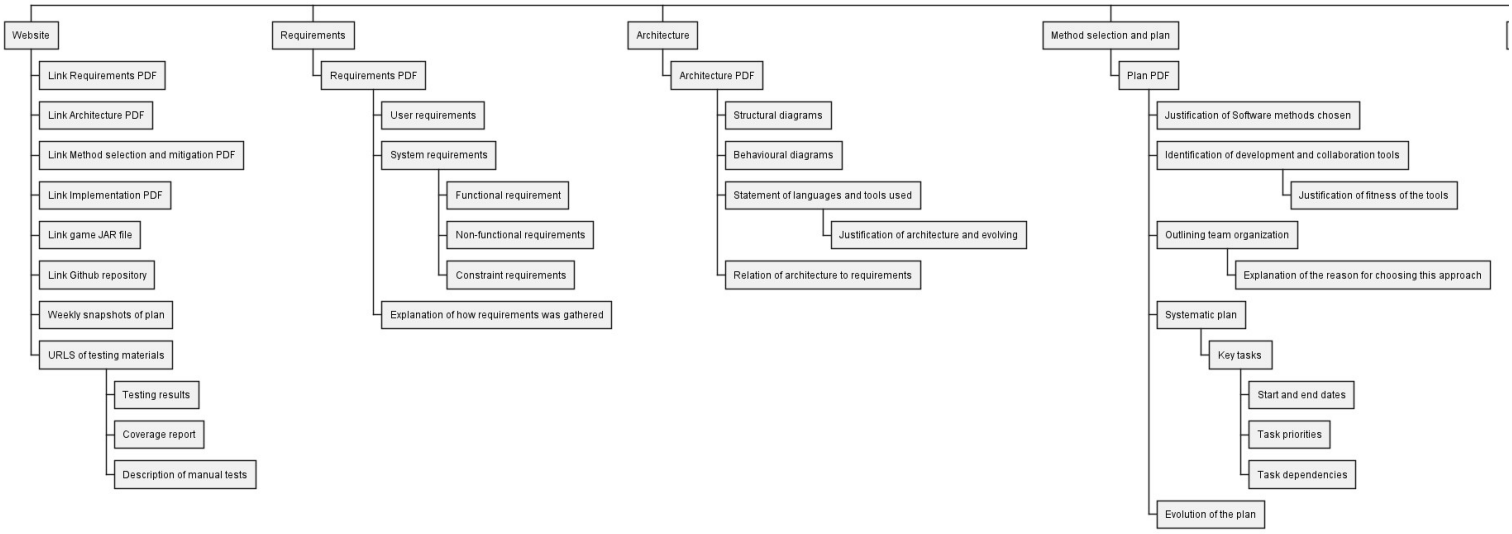
Our choice was based on various reasons. Agile software engineering method is an iterative and incremental approach to software development that focuses on the idea of flexibility and on the importance of collaboration between team members and between team members and client. Because our project can be seen as a flexible project environment, we felt it would be better to use agile methods to allow for rapid iterative development in order to allow us to test different features and implementations quickly during development, allowing us to find what would be best for our project. This is allowed due to the iterative and incremental development that the method involves - developing and delivering software in small increments, that allow rapid feedback from testing and allow us to rapidly respond in any change to the project requirements. Moreover, collaboration is a really important aspect of agile methods, this involves collaboration between team members with focus on face to face communication and working. Agile methods are also designed to be flexible and adaptable that allow us to respond quickly to any changing requirements or feedback. There is an emphasis on working software in agile methods by prioritising the delivery of working software as quickly as possible to receive early feedback and to be able to adapt to changes and not giving much priority into comprehensive documentation. In agile methods it allows us to measure our progress through working software so we can be able to know where we are and what needs to be done. Our team was self-organised because we felt that a good balance of ability and responsibility could be struck this way, while also producing the results necessary for the project. Due to this structure we ensured that we were checking on the progress of team members regularly.

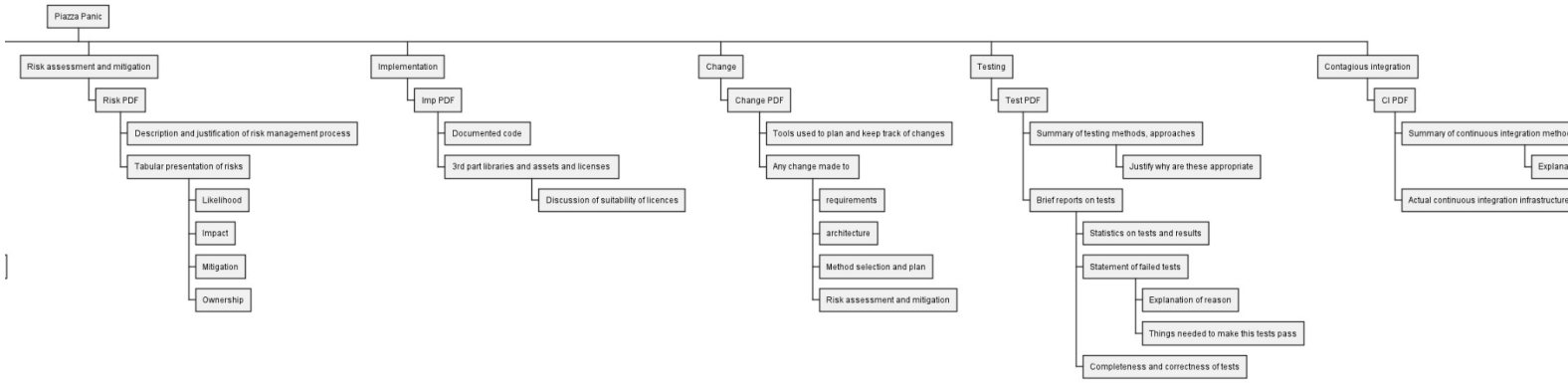
We used a variety of development and collaboration tools. For our version control softwares, we used Github to create a repository which all the team members have access to. This allows the managing and tracking of changes to the code, and allows collaboration between team members on the same code simultaneously. This increases collaboration and communication between team members and allowed us to deliver working software as quickly as possible, which are both important factors in our method of software development. For issue tracking we used Github Issues. This allowed the team to manage and track tasks and issues related to the project, and allow us to assign tasks to relevant team members, which increased allowed for successful self-organisation, and allowed us to measure our progress. We used Discord for communication which allowed us to communicate and collaborate remotely via multiple options such as text chat or video chat. We used it to conduct virtual regular meetings to discuss our progress and work if we could not conduct in-person meetings. We looked at many game engines and discussed various options until we decided to use libGDX as our game engine. libGDX allows us to build our game and test it more efficiently. It provided us with libraries and tools that helped us to handle many tasks, this allowed us to focus on the game itself rather than focusing on low level code. The game engine helped us to implement high quality working software which aligned with our software development method, and allowed us to deliver this software to the client. We considered using other game engines such as LITIengine and Slick2D, but we chose libGDX for many reasons - it has support for cross-platform

development; it is a Java-based game engine which has an active community and high quality documentation that could help us if we faced any problems or troubles; because of the fact that libGDX is highly customizable which allows us to tailor the framework depending on our requirements and needs; and because it is relatively easy to learn which saved us a lot of time.

Work Breakdown Diagram

This Diagram shows the basic work breakdown structure for the entire product cycle for the game.
(Had to be split into 2 PNG's to fit on the page.





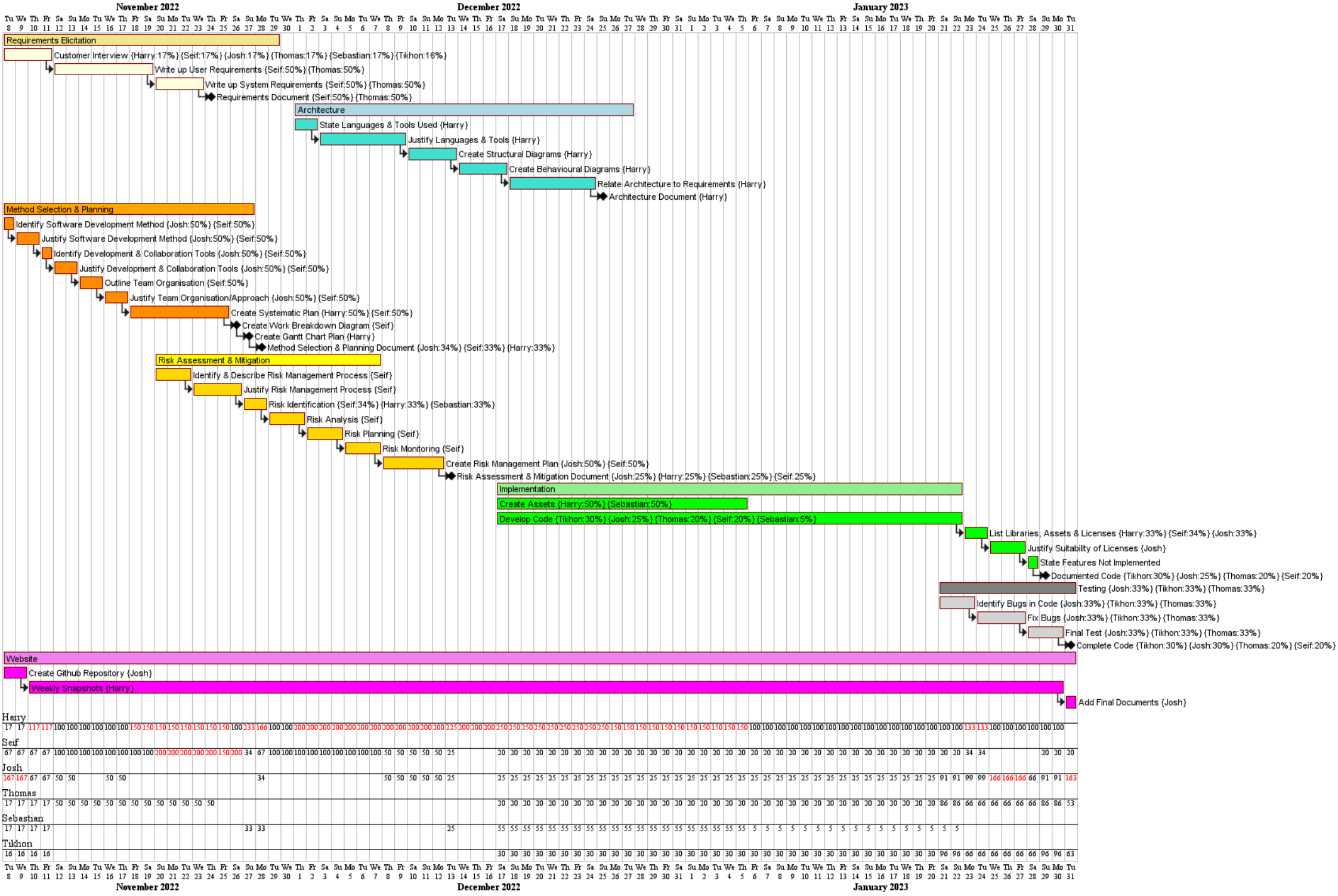
Approach to Team Organisation

Our team's approach to team organisation was self-organising. We chose this approach to align with our agile method and SCRUM framework. In each stage, we organised ourselves into teams for different purposes, for example one team for the charts and diagrams, one for the requirements, one for the risks and mitigation and one for implementation. In every stage, the people in each team could change to ensure each member could take part in every part of the process to maximise the learning output and experience. All the teams together would have regular meetings where progress of tasks were discussed and if any problems arose then other team members could help. Additionally, in each stage, we assigned a member of the team to assess progress on all tasks of all teams and make sure that the team as a whole is not behind. We would change which team member carried this out at every stage to allow all team members to help develop their leadership skills. Also, this helped us ensure that we did not put pressure on only one team member to do this throughout the entire project. Every team should report to the leader of this stage on the progress of the tasks and discuss any problems faced and the tasks completed. The team leader would set up an urgent meeting if any team requested one, or would set up regular meetings if no urgent meeting is requested.

We followed a basic SCRUM framework which focuses on the fast delivery of working software by the use of short development cycles that are called sprints. By following this framework, we divided ourselves into teams that each focused on and were responsible for a specific part of the development process. Each team had a list of predefined goals accompanied with specific time frames which were the sprints that needed to be done. At the end of each sprint, the teams reviewed the progress made and adjusted their priorities and tasks accordingly. With SCRUM, transparency is very important, to help ensure that the team is always aware of the current status of the project. Regular progress check ups and reviews of work were also done to keep the rest of the team updated. An Agile method with this basic SCRUM framework worked well for our small team and project environment with many changing priorities, so we chose to stick to this framework throughout the project.

We have two stakeholders, a client which will try to market and sell our game, and the University of York Communications Office who is interested in our game for promotional reasons. The team and the client have agreed on terms of communication. The team can email the client and can set up a meeting with a client but the client has to be given a 1 day before notice. Meetings can be held in person in the client office or online through Zoom. Moreover, if the team has any questions, it can be asked to the client during the weekly practical hours.

Gantt Chart Plan


November 2022
December 2022
January 2023
Requirements Elicitation
Customer Interview {Harry:17%}{Seif:17%}{Josh:17%}{Thomas:17%}{Sebastian:17%}{Tikhon:16%}
Write up User Requirements {Seif:50%}{Thomas:50%}
Write up System Requirements {Seif:50%}{Thomas:50%}
Requirements Document {Seif:50%}{Thomas:50%}
Architecture
State Languages & Tools Used {Harry}
Justify Languages & Tools {Harry}
Create Structural Diagrams {Harry}
Create Behavioural Diagrams {Harry}
Relate Architecture to Requirements {Harry}
Architecture Document {Harry}
Method Selection & Planning
Identify Software Development Method {Josh:50%}{Seif:50%}
Justify Software Development Method {Josh:50%}{Seif:50%}
Identify Development & Collaboration Tools {Josh:50%}{Seif:50%}
Justify Development & Collaboration Tools {Josh:50%}{Seif:50%}
Outline Team Organisation {Seif:50%}
Justify Team Organisation/Approach {Josh:50%}{Seif:50%}
Create Systematic Plan {Harry:50%}{Seif:50%}
Create Work Breakdown Diagram {Seif}
Create Gantt Chart Plan {Harry}
Method Selection & Planning Document {Josh:34%}{Seif:33%}{Harry:33%}
Risk Assessment & Mitigation
Identify & Describe Risk Management Process {Seif}
Justify Risk Management Process {Seif}
Risk Identification {Seif:34%}{Harry:33%}{Sebastian:33%}
Risk Analysis {Seif}
Risk Planning {Seif}
Risk Monitoring {Seif}
Create Risk Management Plan {Josh:50%}{Seif:50%}
Risk Assessment & Mitigation Document {Josh:25%}{Harry:25%}{Sebastian:25%}{Seif:25%}
Implementation
Create Assets {Harry:50%}{Sebastian:50%}
Develop Code {Tikhon:30%}{Josh:25%}{Thomas:20%}{Seif:20%}{Sebastian:5%}
List Libraries, Assets & Licenses {Harry:33%}{Seif:34%}{Josh:33%}
Justify Suitability of Licenses {Josh}
State Features Not Implemented
Documented Code {Tikhon:30%}{Josh:25%}{Thomas:20%}{Seif:20%}
Testing {Josh:33%}{Tikhon:33%}{Thomas:33%}
Identify Bugs in Code {Josh:33%}{Tikhon:33%}{Thomas:33%}
Fix Bugs {Josh:33%}{Tikhon:33%}{Thomas:33%}
Final Test {Josh:33%}{Tikhon:33%}{Thomas:33%}
Complete Code {Tikhon:30%}{Josh:30%}{Thomas:20%}{Seif:20%}
Website
Create Github Repository {Josh}
Weekly Snapshots {Harry}
Add Final Documents {Josh}
Harry
Seif
Josh
Thomas
Sebastian
Tikhon