Method Selection and Planning

“Lucky” Team 13

Team 13

Yuxin Wu
Bailey Findlay
Elizabeth Edwards
Chenxi Wu
Alex McRobie
Yihong Zhao

Method Selection and Planning

Software Engineering Method Outlined and Justification

In order to develop a game that meets the requirements set by our client, an appropriate software engineering method was required. To choose a method we had to take into account time constraints, our client and their requirements including user, system and any possible future changes.

We chose to use SCRUM which is an Agile method as it is good for small teams, short iteration cycles, accommodating change with low documentation overhead which was beneficial to us due to our time constraint. Other engineering methods such as plan-driven methods were not as advantageous to us as they require a stable environment and extensive documentation in order to communicate and progress the project. Plan-driven methods are not suitable for this project as new requirements from the client will be produced causing a need for change. Also, due to the time constraints a large amount of paperwork would detract focus from development.

When using SCRUM we would have one weekly SCRUM meeting, potentially more depending on deadlines, to see what work has been completed and delegate more tasks. This way we were able to ensure requirements were met due to the regular meetings and changing any plans if tasks took longer to complete than anticipated.

We had considered other Agile methods such as XP however we, as a team, wanted to be able to prioritise which tasks were done first which you cannot do with XP. XP also has technical practices such as pair programming which you have to follow and due to our time constraints and small team size we wanted to be able to have more flexibility with our resources. Therefore we chose to use SCRUM.

Development and Collaboration Tools

UML: Plant UML
We chose to use PlantUML for our UML architecture diagrams and Gantt charts. We chose to use PlantUML as it was free, intuitive and easy to make changes and update diagrams. It also has an extension that works with Google Docs to easily add the diagrams into documents and edit them in the document too. Compared to graphical chart creators the process of extending and modifying the charts was much simpler as PlantUML automatically determines the layout.

Team Communication: Discord
For team communication we chose to use Discord as everyone was familiar with it and had access to it. We created multiple messaging channels that had different functions in order to keep related information together. We had a channel for meeting reminders, a main channel for general communication and an implementation channel to discuss topics relating to implementation. We also created multiple voice call channels for subgroups to hold meetings

and for the entire team to meet. We had a general channel to host whole team meetings, an architecture channel for the subgroup working on the architecture to meet and then a programming channel for the programmers to meet. We considered other messaging platforms however the ability to separate discussions into channels and the ability to hold meetings on the same platform made it more beneficial.

Version Control: Git with GitHub
In order to support the entire team having access to the code as it was developed, we used github to store the code, as well as to host the website for the project. Github was the obvious choice here for a few reasons. It often has built in compatibility with a lot of the other tools we wanted to use, meaning instead of having to manually copy across files and code, there was an option to update code as well as open the project directly from github. This was really useful to us as it meant we could always easily access the latest version of the project from one central location that we all had access to.

Documentation: Google Drive
We used Google Drive to store our documentation of the project as it is accessible and familiar to everyone. Google Drive is also cloud based which we required as we wanted everyone to be able to contribute, even concurrently. Since Google Drive is cloud based there is a very low risk of losing documents as it is all stored remotely.

Task Management: Trello
In order to track and delegate tasks for our team we used Trello. We created two boards, one for documentation tasks and one for implementation. This way the board would not become overloaded with tasks relating to different aspects of the project. In each board we created sections to show how far a task had progressed such as a to do section, a doing section, a done section and an issues and problems section. This way everyone was able to easily communicate how their tasks were progressing and state any problems that arose. We considered using Github projects which would have kept all of our work on the same platform, however we found that Trello was easier to use and visualise.

Java Game Framework: libGDX
For the engine itself, we had a few options before deciding on LibGDX. We had originally considered LITIENGINE and Mini2Dx. LITIENGINE has good documentation for getting started, but unfortunately there weren't that many guides after that or much support for the engine. Mini2Dx was similar in that it didn't have many examples or guides to help us learn how to use it. We decided on LibGDX for a few reasons, one being there are a lot of guides and support for how to use it, which is really important to us as new users of this engine as we didn't want our work to be stalled trying to figure out an issue with the tools we were using. It has official and unofficial tools for creating assets which streamlines our development even further. Although some of the official documentation was a little more complicated than the other engines, we decided that this one was still the best suited to our project.

IDE: IntelliJ IDEA
We chose our IDE in relation to our version control system. Originally, we didn't all use the same IDE, but as more people started needing access to the code to work on their own sections, we ran into a few difficulties accessing the project. Most of us were originally using

Eclipse and manually transferring the project across - however, after some experimentation we found that IntelliJ IDEA was a much easier program to open the project with. IntelliJ IDEA also has fantastic Github integration which completely changed how we could share the project with the rest of the team, so for the remainder of the project we all used this IDE.

Team Organisation

Assessment 1
We chose to approach team organisation as a collective. By this I mean we had regularly scheduled meetings, where we discussed; what work needs to be done, how we should do the work, then split it up between the team depending on multiple factors (such as someone with libGDX experience doing the implementation).

We found this both appropriate to the team and project as it allowed all users to voice the way they would approach the work. We are all working through the same course with the same information so we have a similar starting knowledge but a possible different view on how to apply it, making it important that all views are acknowledged. This helps with the project a lot, as when you acknowledge all views, in most cases you pick the best out of the options presented, giving a better final product you have all agreed on.

When dividing the work we also gave the freedom and flexibility on when we want the work for review in order to work around the teams very conflicting schedules, outside of our weekly 2 hour meeting.

Apart from these organisational options we chose, we had our aforementioned weekly team meeting which was allocated for 2 hours every Wednesday first term and every Friday second term. We used this to do all the above scheduling, attempting to agree on as much as possible and then going away to do the work where needed.

To develop our game, we all worked together to develop the initial ideas for the requirements of the game. Work for documenting each part was split between individuals. Initial ideas for architecture were done by the team as a whole but as the implementation was started and the architecture involved it was done more by individuals.

Implementation itself was mainly done by one person with the rest of the team having access and weighing in on how the game's base architecture should evolve. We thought it would be easier in a project of this size for one person to do the actual coding to avoid any minor issues with misunderstanding someone else's code. In a bigger project, this would not be possible as the project would then be slowed down waiting for one person to code the entire project, but we thought it suitable in our situation.

Assessment 2
Our team organisation was managed using Trello. Through Trello we created and assigned tasks, allowing us to visualise the current workload of the project. Using the multiple sections: to do, doing, done and issues and problems we could effectively estimate where focus was required to keep to deadlines. We used the issues and problems board to keep track of any unexpected issues which we would discuss during the next meeting to discuss whether they needed to be actioned. When cards on the board were updated we would notify each other on the Discord to ensure everyone is up to date.

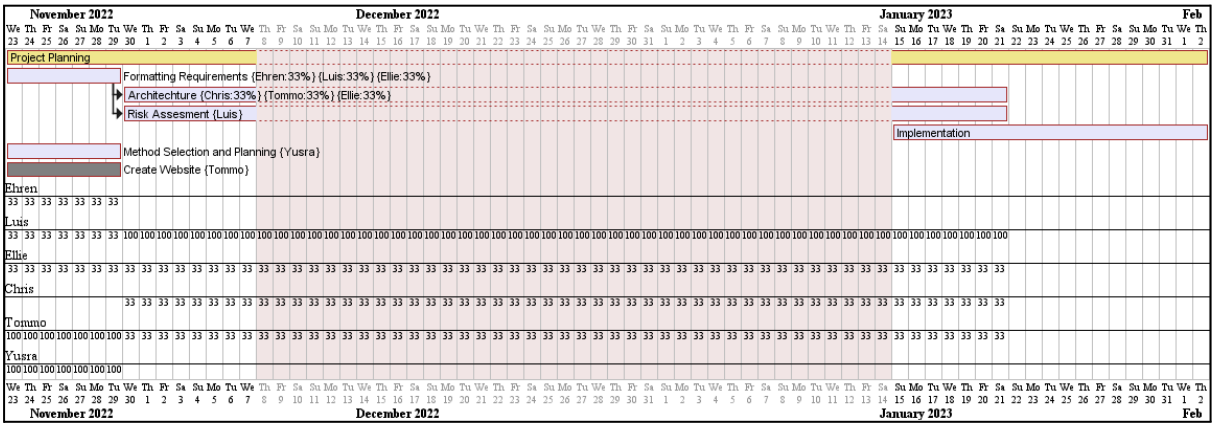
We have also assigned a project owner, Elizabeth Edwards, and a product owner, Alex McRobie. This allows for the planning of the documentation and meetings to be led by one person and the planning for the implementation to be led by another. We assigned three people to the implementation and testing and three to work on the documentation. This way the people coding had support from each other and the documenters would still have enough people to complete the required work. To delegate tasks, each member's strengths and weaknesses were taken into consideration. For example those most familiar with Java were assigned the majority of implementation tasks. Where possible we assigned two people to each task to improve the bus factor of our project.

As previously mentioned our chosen software engineering method is SCRUM. We met at least once a week either in person or online through discord to have sprint planning meetings. These sprints would be a few days to a week in length. During each meeting we created a list of tasks that we needed to do from reviewing the requirements and the previous sprint's work. This way we can keep up to date with issues in the code, unfinished work and ensure requirements are met.

Project Plan

Assessment 1

Gantt Chart 23/11/22:



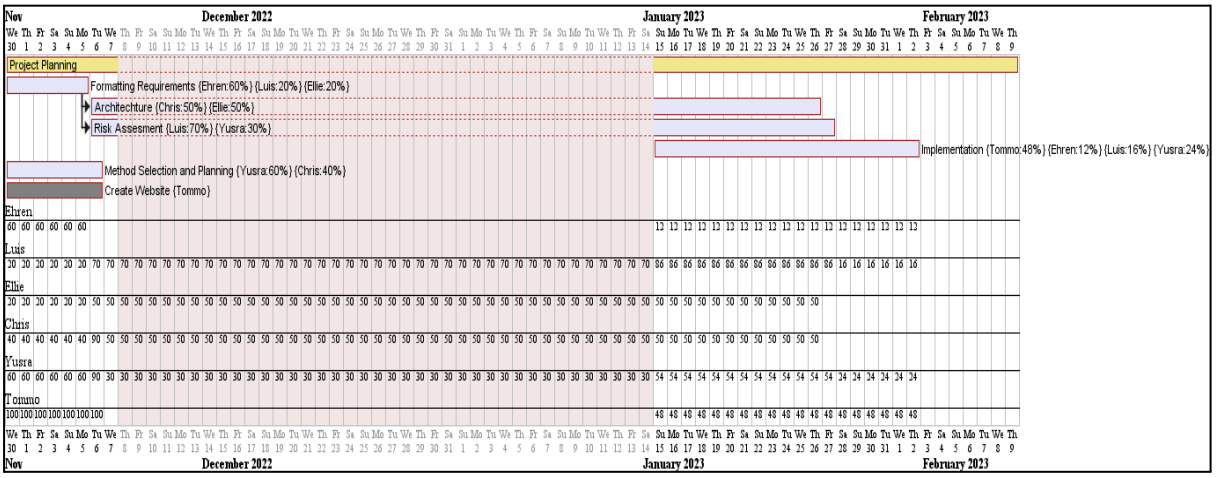
At this point, the requirements have been noted down from the meeting with our customer. We are not allocating time to work during the Christmas holidays, due to exam revision and other work, and not during exam week either. With this the majority of sections should be either completed or semi-completed by the end of this term, with some of architecture and risk assessment carrying over into next term. All of the implementation will be left for next term. The current work split is to allow each person to gain a significant amount of marks, with the rest being made up for in the implementation section when the work is completed and the real mark split is known.

As the actual implementation of the project relies on the previous sections, it is left till last. Architecture is also left till later as the initial implementation phase will heavily influence the architecture section due to the new understanding of how the chosen game library works.

The other sections aren't dependent on each other and can be done at any point during their assigned time frame.

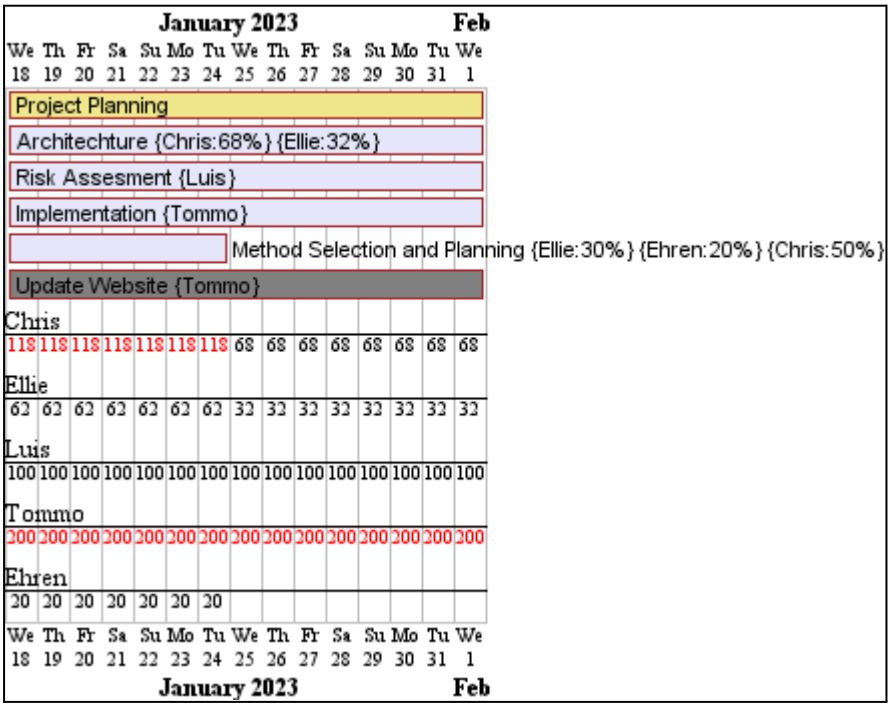
The gantt chart will be updated with any major changes to the decided work plan of the project and added to the document.

Gantt Chart 30/11/22:



Some work was done on the sections over the week, with some minor reformatting of the requirements document, but other than that no huge progress was made, so the timeframes have been pushed back in accordance with this. Proper splits have been added to the chart to signify everyone's workloads and to properly show the allocation of marks.

Gantt Chart 18/01/23:

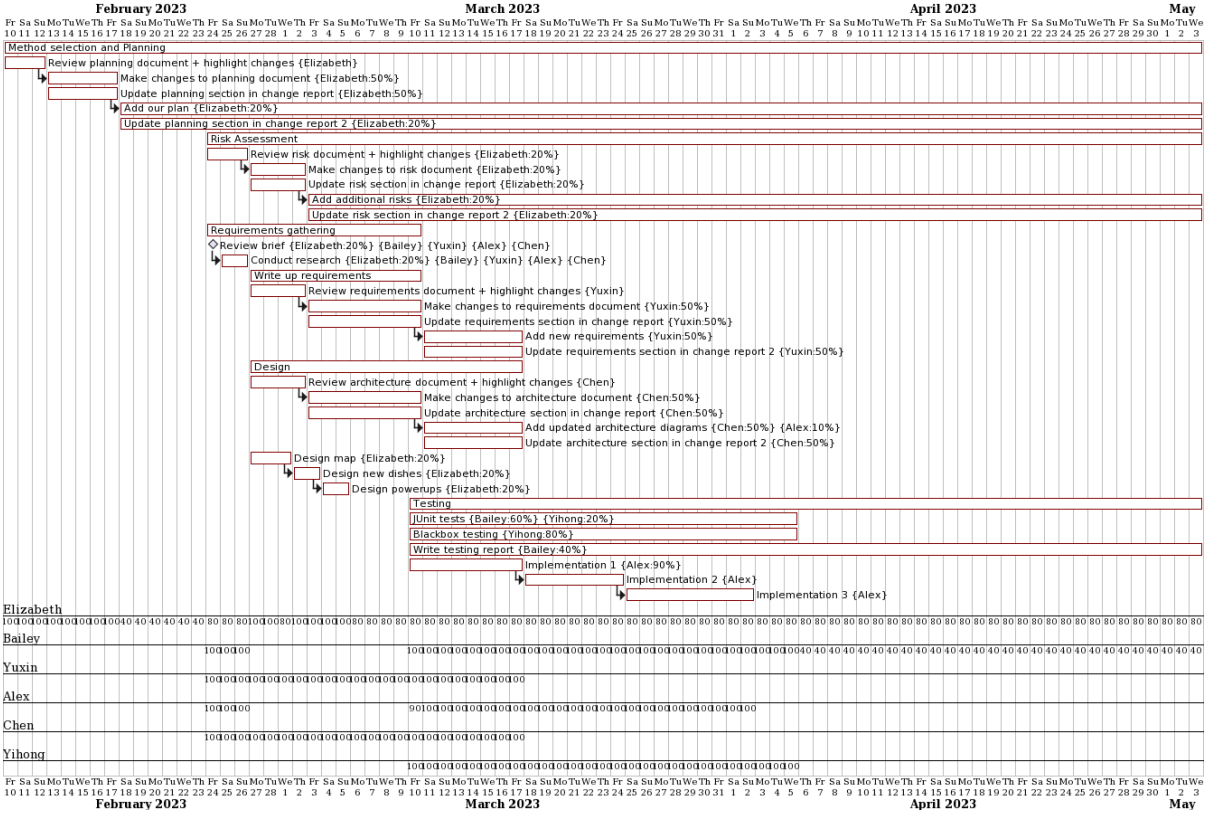


Entering the final weeks of the project, the workload assignment shifted heavily. The website will continue to be updated with each completed document by the creator. Implementation

will now be taken on by one person, who most fully understands the game engine and has undergone the beginning stages of the development already. Requirements are completed and the remaining tasks are split between members.

Assessment 2

In our first week of project development we produced an initial plan:



In the initial planning stages we wanted to complete the majority of documents first before focussing on implementation. This way we would be able to implement the project without missing requirements and have an idea of how we want it structured. The method selection and planning document as well as the risk assessment would be continuously worked on in order to be able to state all stages of our plan and thoroughly manage risks. All of the Gantt charts for our project can be found here : Gantt Charts 2 and on our website.

The first 3 weeks consisted of taking on the planning document, risk assessment and requirements. Before moving forward to beginning architecture. Once we had an idea for the architecture, testing and implementation could begin. Implementation was done in 5 sprints over the course of 5 weeks. Each sprint compromised a breakdown of the requirements. Both testing types were started however, whitebox tests were completed first as this could be completed alongside implementation. Whereas only some blackbox tests could be completed alongside implementation so this took longer to complete.

Having gained experience from the first assessment, times to complete tasks were well-judged and our initial plan is fairly similar to our final plan. However there were still a

couple of times where tasks took longer such as architecture diagrams and the CI document. However, due to our plan we were able to give extra time for tasks. One of the team members also became unavailable due to unforeseen circumstances and so their work had to be completed by others. This can be seen in the final Gantt chart.

The final Gantt chart for the project can be found below: