Method Selection and Planning

“Lucky” Team 13

Team 13

Yuxin Wu
Bailey Findlay
Elizabeth Edwards
Chenxi Wu
Alex McRobie
Lawrence Li

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: PlantUML and Smartdraw
We chose to use a combination of PlantUML and SmartDraw 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. However, SmartDraw was used for the component diagram as it was more intuitive for the team member who worked on it.

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
We decided to use Git for version control and hosted our Git repository on GitHub. We chose GitHub because it allows multiple people to code at the same time and allows changes to be compared to previous versions and restored if needed. We are also able to create and host our website with GitHub which is extremely useful.

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
We decided to use libGDX instead of LWJGL as due to its many built in functions and supported libraries, such as box2D, it is easier and quicker to implement and develop a game. libGDX is built on top of LWJGL, simplifying common tasks such as creating a window or handling inputs. This is beneficial as due to our limited time constraint set by the client, we can put more time towards implementing the requirements instead of implementing the foundations from scratch.

Team Organisation

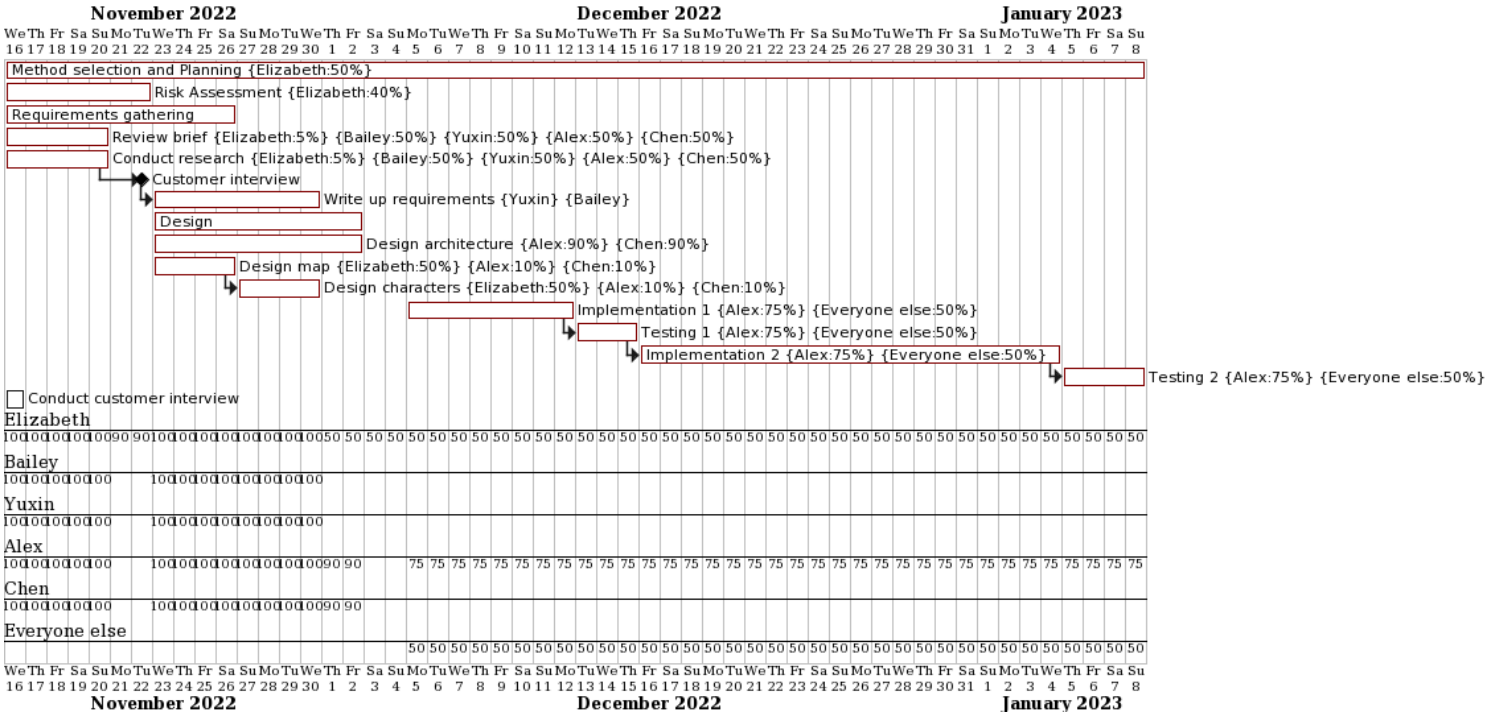
Our team organisation was managed using Trello. Through Trello we created and assigned tasks, allowing us to visualise the current workload of the project. Using the multiple sections: to do, doing, done and issues and problems we could effectively estimate where focus was required to keep to deadlines. 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. Since we only had 5 people for this part of the project we assigned two people to the implementation 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. We were unable to fully keep to our meeting plan due to team members being unavailable especially during the Christmas holiday when members were away seeing family or revising for upcoming exams.

Project Plan

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 an idea of how we want it structured. The method selection and planning document would be continuously worked on in order to be able to state all stages of our plan. All of the Gantt charts for our project can be found on our website.

During the first 2 weeks we had managed to complete the risk assessment document, review and research the brief, have a meeting with our customer to discuss requirements and completed our requirements document. It was crucial that we completed these as the rest of the project was dependent on it.

The next 3 weeks consisted of focussing on the architecture as the start of the implementation was dependent upon it. Once we had a good understanding of how we were structuring our program we began implementing it. We decided to get small tasks done in the early stages of implementation as we were still familiarising ourselves with libGDX and so allowed for more time per task due to this. Our first task was to create a chef and basic movement. This was followed by creating a background and adding collisions.

Our initial plan was optimistic to allow for contingency time and to minimise the impact of project risks. Having the contingency time was very useful as on multiple occasions it took longer than expected to complete the tasks. We also decided during the Christmas holidays that we were going to take three weeks off in order to rest and focus on our upcoming exams.

The next three weeks were heavily focused on finishing the implementation and ensuring documents were properly proofread and correct. This included creating classes for all of the workstations so the user could interact with them, recipes for the user to be able to combine ingredients together and a timer.

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