Methods Selection and Planning

Outline of team's software engineering methods and collaboration tools.

For our project, we decided as a group to use the scrum methodology, which is a branch of the agile methodology [1]. The organisation of these scrums were as follows:

1) **Sprint Planning:** We would plan what we would be working on during our practical meeting, which acted as the start of each weekly sprint. This would include going over the backlog of documents that have not been completed during the previous sprint and adding that to the work that we would be working on in the next sprint. We would assign members to each bit of work for the current sprint, ensuring that everyone has got an adequate amount of work to do.
2) **Sprint:** Using Discord we would communicate throughout the sprint as we are working on the documents, whilst also using Notion to produce a "Kanban" style board to communicate what has been completed.
3) **Sprint Review:** The day before the next practical we would review what has been done during the sprint and if anything can be completed between this review and practical.
4) **Sprint Retrospective:** We would review how the last sprint went. Focussing on individuals, tools and processes and whether everything that has been 'completed' is up to the quality of the project.

Each Tuesday we would carry out the Sprint review and retrospective for the previous sprint, then once that has been completed, we would start planning the workload for the upcoming sprint . This would make our sprints a week long, allowing us to be very flexible and be constantly changing, so we are not stuck in a rut with certain documents.
This was good for us as we would do this after our ENG1 lecture, so that we were all already on campus.
The idea of using the scrum methodology for our team was that we had someone (George), who was a natural leader who could act as our scrum master and lead the meetings and hand out tasks to each member. With the agile methodology, we can accept any changes in requirements from our customer or from our requirements document, which we felt was good as our requirements document would be changed quite a bit from the start of our project. Also the fact that we could have week long sprints made it good for our team, so that we can quickly go through our tasks and ensure that we can change what we are working on every week.

The other method that could be considered for a project like this would be the waterfall methodology. This would've worked through getting all the requirements written out before even touching the other sections. Then completing each entire section before moving to the next one. This would mean that, if we mess up one small part of our section, we have to ignore it and move onto the next part.

However we have decided against using this methodology as it wouldn't allow us to have regular meetings and updates for the team. Another disadvantage is that you must fully complete each phase before you move on to the next phase, so this would mean that everyone is only focusing on one phase at a time rather than part of the team working on one phase and another part on the next phase.

We decided to use Discord as our tool for communication, as we all had experience with it. Also the fact that Discord can allow us to make multiple channels allows us to communicate about different parts of our sprint at the same time without being confused with everything on the same chat.

The other tool for communication that we use is Notion, which we use to show our progress during each sprint on each task. This is good as we can see which parts of the sprint need work on and during the sprint we can move to another task if we have completed ours.

For our work we would store everything in a shared GoogleDrive folder, this allowed multiple of us to work on the same document at once, while seeing the changes in real-time. This allows us to have multiple people on a task at a time, so that if someone gets ill we can make sure the task will still be completed.

For our IDE of choice we decided on using IntelliJ, as we have done a lot of research on different IDEs and along with good reviews from previous groups, we found it the easiest to learn and use in comparison to the other IDEs, so that all of our team can use this IDE to implement the game if we need to (perhaps due to illness of other team members).

An alternative method of communication that we discussed using was WhatsApp. However we thought against it as the main chat could get confusing if we are discussing different tasks. The way around it would've been creating multiple different chats however we felt Discord makes it a lot easier to do this and that we didn't want to waste too much time trying to create all these chats.

Our Approach To Organisation
In order to ensure every team member contributes something to the organisation of this project, everyone has been assigned an important organisational role:

- George and Seyi are responsible for scrum management. They ensure that our weekly meetings result in the most effective use of our time, keeping us focused on high priority tasks rather than useless tasks. In doing so, they maximise our productivity and the progress made between sprints.
- Alex and Will are responsible for the product backlog. They maintain the Notion database the group uses to select and track the tasks of the current sprint. This facilitates smooth team collaboration by keeping everything in one place where it can be easily referenced and updated in real time.
- Jamie and Ben are responsible for secretarial tasks. They take notes from each of our meetings to help us record their purpose and view the overall direction of our project. They also maintain other personal notes used by the group to ensure general organisation.
- Meg and George are in charge of report editing. They ensure any documentation we produce is kept to a high standard by identifying mistakes and any areas of improvement.

These roles are inspired by the scrum guide [3] and all facilitate the scrum method by maximising the effectiveness of each sprint, ensuring they are all efficient, clearly defined, well organised and recorded. The benefit of this approach is that it encourages team engagement, everyone feels like they are adding something to the project. Furthermore, to reduce the bus factor [4], we have ensured multiple people are responsible for each role, so that this approach can be married with flexible circumstances such as a team member falling ill. To do this with every role having two people taking control, the first person in the list is the main one that will do this role and the second on the list being the understudy for if the first person is absent.

Within each sprint, the team has a flat structure with each team member having equal control. Tasks are delegated to teams of two or three on a case by case basis, for example two people may be responsible for writing up a document, whilst three are responsible for implementing the basic features of the game.

This allows us to maximise our adaptability to unforeseen circumstances, particularly important for a team of students who may travel and have dynamic lives and for a high risk project such as this one which involves learning a new game engine. No task is left to one person, to reduce the bus factor. Team members who wish to switch roles can do so week by week, increasing motivation and morale.

The team meets twice a week, once after our ENG1 lecture on a Tuesday and in our weekly practical on Wednesday. We believe in-person communication facilitates a more lively discussion, where back and forths can happen and everyone can feel welcome. To communicate during the week, we have set up a Discord server where we can all message each other or directly message a specific group member. Whilst the most important things we need to do are communicated in the meetings themselves, this option allows for quick updates and changes throughout the week if necessary.

Project Plan

Week 1 - To start out we set a series of tasks, mainly starting requirements and researching the game engines and IDEs.

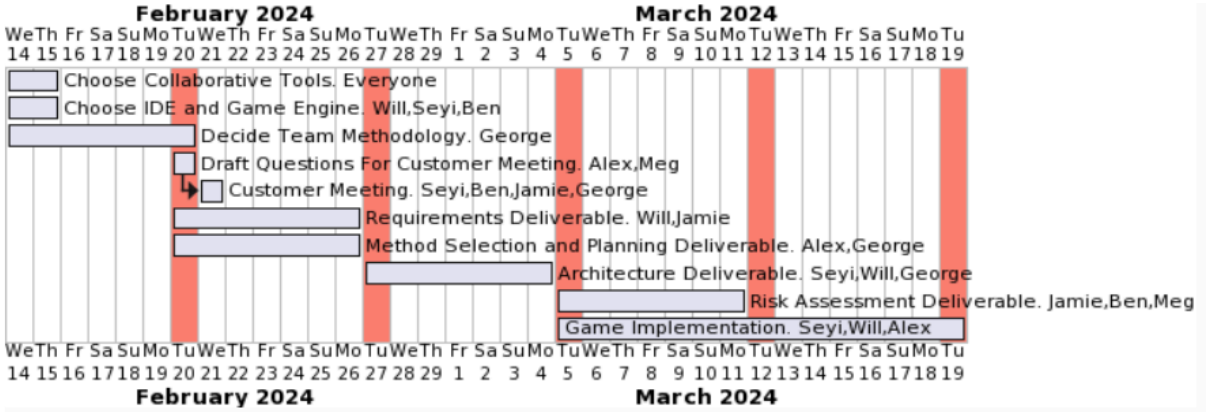
One part of the team worked on the decision on the game engines and IDEs. We had created a table with a bunch of different engines and IDEs with some characteristics to help us choose which one to use. Our initial thoughts were about using LibGDX and IntelliJ. This part was started on the date of the first practical on 14/02/2024 and the aim was to complete it within the second sprint with a deadline of 27/02/2024.

The other part of the team worked on the requirements deliverable, by creating a file with a brief overview on what the game was going to be like as well as creating separate tables for each type of requirement. Our aim was to initially fill in the tables based on the product brief and then have our customer meeting and finish off the rest of the tables. The requirements deliverable was started on 14/02/2024 with the aim of finishing it by 27/02/2024.

We decided on the tools that we would use to collaborate as a team, deciding on Discord and GoogleDocs as our sharing and communication applications and deciding to use an agile scrum methodology to manage our project. This was completed in the first practical .

Week 2 - Scheduled our customer meeting for the afternoon on 21/02/2024, so we would be able to add in any requirements to our existing document. For this customer meeting we started and finished drafting the questions that we would be asking our customer during the sprint review on 20/02/2024.

We also finalised the game engine and IDE that we were going to use in the sprint review (20/02/2024). (LibGDX and IntelliJ). The following Gantt chart shows our initial progress and thoughts on timing for the whole project. This Gantt chart shows the key areas of the project and doesn't go into the specifics of each area (e.g. the architecture deliverable is broken into 4 smaller tasks.

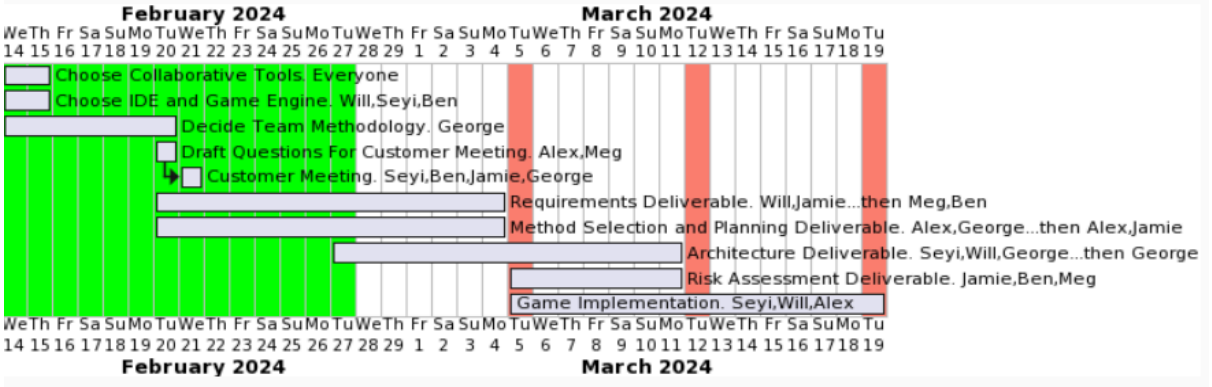


Week 3 - We finished off writing up the requirements from the customer meeting in the existing document.

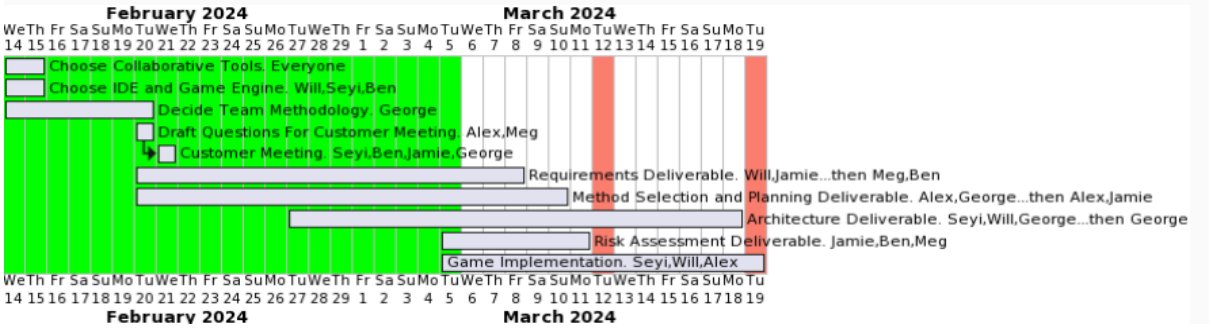
The website has been set up and everything in it is functional.

We split into 3 teams to sort out the next parts of the assessment. An architecture team, a team to complete the method selection and planning section and a team to finish up polishing the requirements deliverable. This was going to start on 27/02/2024 and would hopefully last until the end of the next sprint on 06/03/2024. We then realised that we would have to update our Gantt

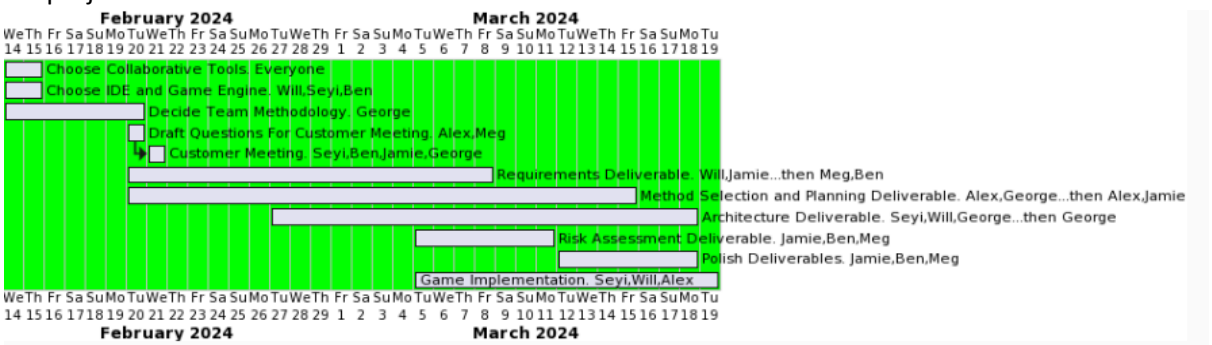
chart with the green showing the days that had passed so far.



Week 4 - We finished off the writing up of the method selection and planning section.
We split into two teams. One team to start the implementation of the game and the other to start the risk assessment deliverable and finish off the architecture deliverable.
The game implementation team will complete the game over 2 sprints. Starting on 06/03/2024 and finishing 20/03/2024.
The other team will be starting and finishing the risk assessment deliverable and also finishing the architecture deliverable, starting 06/03/2024 and finishing at the end of the sprint on 13/03/2024. This team will then polish off the deliverables to make sure that they are in line with the assessment objectives and to correct anything that is missing or is erroneous. This will be over the final sprint starting 13/03/2024 and finishing 20/03/2024. Once again we underestimated the sizes of certain deliverables, so we updated our gantt chart as shown below.



Week 5 - We finished off the method selection and planning, and the risk assessment deliverable on 13/03/2024. The parts of the team that were on them worked, polishing up all the deliverables for 20/03/2024. The implementation team made steady progress on the game looking to finish for 20/03/2024. The architecture deliverable was almost completed with the aim to complete it within the next day or two. This would allow us to change our gantt once more to show the final stretch of the project as shown below.



References

[1] "Agile Software Development – Software Engineering", GeeksforGeeks, 2024, Mar. 7.
Available: https://www.geeksforgeeks.org/software-engineering-agile-software-development/
[2] "Waterfall Model - Software Engineering", GeeksforGeeks, 2024, Mar. 12.
Available: https://www.geeksforgeeks.org/waterfall-model/
[3] "The 2020 Scrum Guide", scrumguides, 2020, Nov.
Available: https://scrumguides.org/scrum-guide.html
[4] "What's the bus factor and 7 ways to increase it", medium, 2020, May. 13.
Available: What's the bus factor and 7 ways to increase it | by Sébastien Dubois. | Management Matters | Medium