Method Selection and Planning

Group 27 - WaddleWare Studios

James Smith

Amanda Ling

Fran Medland

Hannah Vas

James Kellett

Malik Tremain

Mischa Zaynchkovsky

4a)

We decided that an agile software engineering methodology would fit best for the project, specifically we chose to use Scrum.
Scrum allows for continuous, iterative development of our game, making it ideal for the project. At the start of the process, a product backlog is created, clearly outlining what the product needs to include/achieve. In our case this was our requirements elicitation process. Then from this a sprint backlog is created, choosing specific requirements to be met within each sprint. Usually a sprint lasts 2-4 weeks but due to our restricted time to complete the project our sprints were shorter, each resulting in a new iteration to the game. During sprints, bi-weekly meetings are held to share progress and highlight any issues that the team have encountered. This ensures that the team is organised and deadlines are met. In addition it helps with risk management; problems that are raised within the meetings can be tracked with our risk assessment table, allowing us to take measures to prevent them. After each sprint, feedback is gathered and the cycle continues, creating a new iteration of our game. This iterative approach is necessary for the development of our game as when new requirements are discovered they can be added into a future iteration (sprint).

For collaborating on the documentation side of the project, we decided to use the google services available to us (google docs, drive etc.). We created a shared google drive for easy file sharing and access, allowing us to keep track of documents and easily examine each other's work as necessary. In addition google software has version control. Although not as important in the documentation as it is in the programming, it makes it easier to see what changes have been made and by who. If mistakes were made we could easily revert back to earlier versions.
We briefly considered using Notion, a software that is specifically designed to aid in managing projects and working collaboratively in a team. It facilitates managing deadlines, creating documents etc., making it ideal for our project. Although Notion had some interesting and helpful features, we decided against using it. Due to the restricted timeframe we had for the project, we felt it best to stick with software that everyone was familiar with rather than trying to learn and navigate something new.

With regards to the programming side of things, we have decided to use GitHub, a widely used version control system in collaborative software development. GitHub was the obvious choice for us, as those allocated to the programming of the game have prior experience with using it. Creating a group repository allowed the team to individually work on separate parts of the game in isolation from one another through the use of branches. GitHub does very well in outlining issues when merging changes made onto the main branch. Pull requests ensure that errors when combining code are omitted.

Communication within the team was very important. We started with a whatsapp chat for general conversations, and then moved the majority of our discussions to a Discord server. Discord can be very useful with collaborative work. It allows for voice/video calls with screen sharing, meaning meetings can be held virtually. Within the server separate channels were created, for example we had a separate channel for implementation and general messaging. This helped us keep our work organised and prevented confusion. Although this separate channel was created for the programmers, everyone could still view it ensuring all group members were up to date with the work that was being completed.

There were other options available to us, for example Google Meet or Microsoft Teams, however Discord had all the facilities of these softwares and more, all in one place, hence this is the communication medium we chose.

4b)

Team organisation was crucial in this project due to the large number of different tasks needing to be completed. Dividing the work between team members was obviously necessary to ensure each member completed an equal amount of work and ensure the project was completed on time. The struggle once this had been done was bringing the individual work back together and ensuring everything made sense as a whole. We held a meeting a few days before submission where we discussed all the work completed as a group and made any changes necessary for submission.

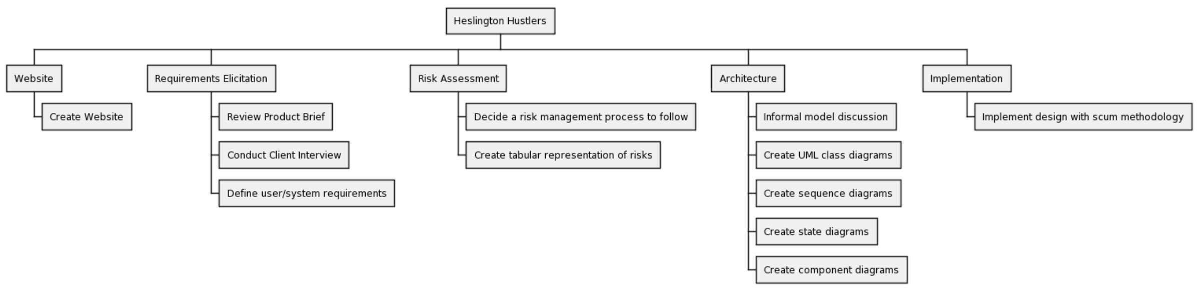
To ensure everyone had a clear idea of what they were doing, we assigned people to different parts of the project in our first meeting after the interview. Doing this prevented any confusion down the line regarding what parts of the project were who's responsibility. It also meant that individuals did not need to research different parts of the project in depth and could mainly focus on their own section and getting it to a high standard.

Although everyone had their own roles and responsibilities, everyone in the team still needed to be kept up to date with what was going on in different parts of the project. The meetings we held (either in person or online) were used to discuss such matters. In each meeting, team members would explain the work they had completed. They would also explain any problems that arose whilst completing their assigned job and any problems that they felt may arise in the future. As a group we then discuss how these risks can be mitigated. Finally we share ideas of how to move forward and decide what each individual will complete for the next meeting. It was decided to hold the meetings in this way to ensure we were sticking to our Scrum development methodology.

By sticking to the Scrum organisational plan, we have ensured that we keep track of work that needs to be completed and kept on schedule throughout the project. It forces regular meetings facilitating clear communication within the team.

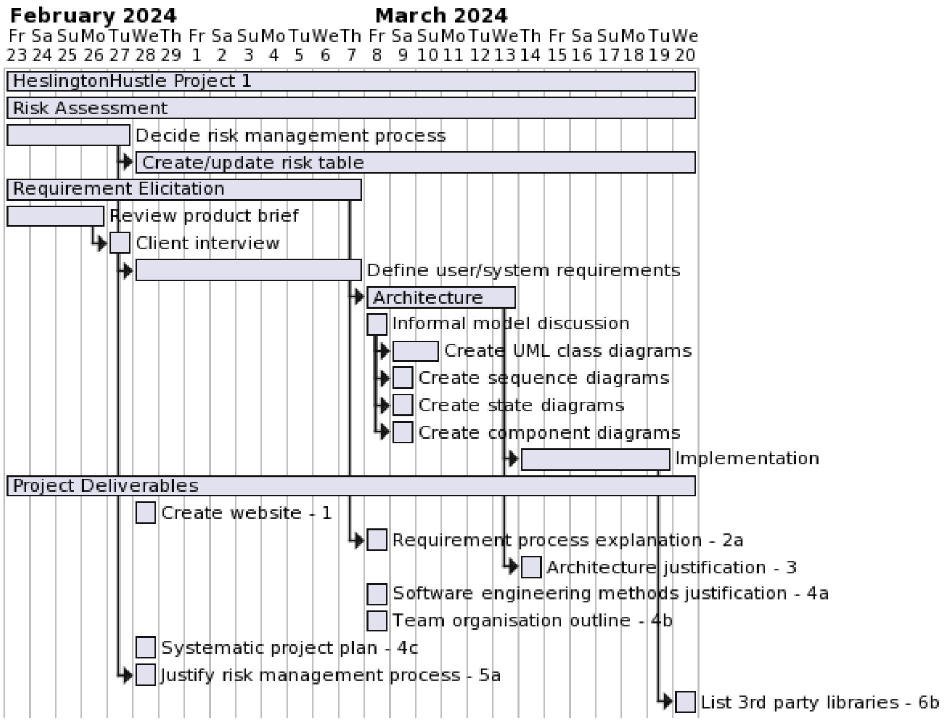
4c)

Work Breakdown:


Heslington Hustlers
Website
Create Website
Requirements Elicitation
Review Product Brief
Conduct Client Interview
Define user/system requirements
Risk Assessment
Decide a risk management process to follow
Create tabular representation of risks
Architecture
Informal model discussion
Create UML class diagrams
Create sequence diagrams
Create state diagrams
Create component diagrams
Implementation
Implement design with scum methodology

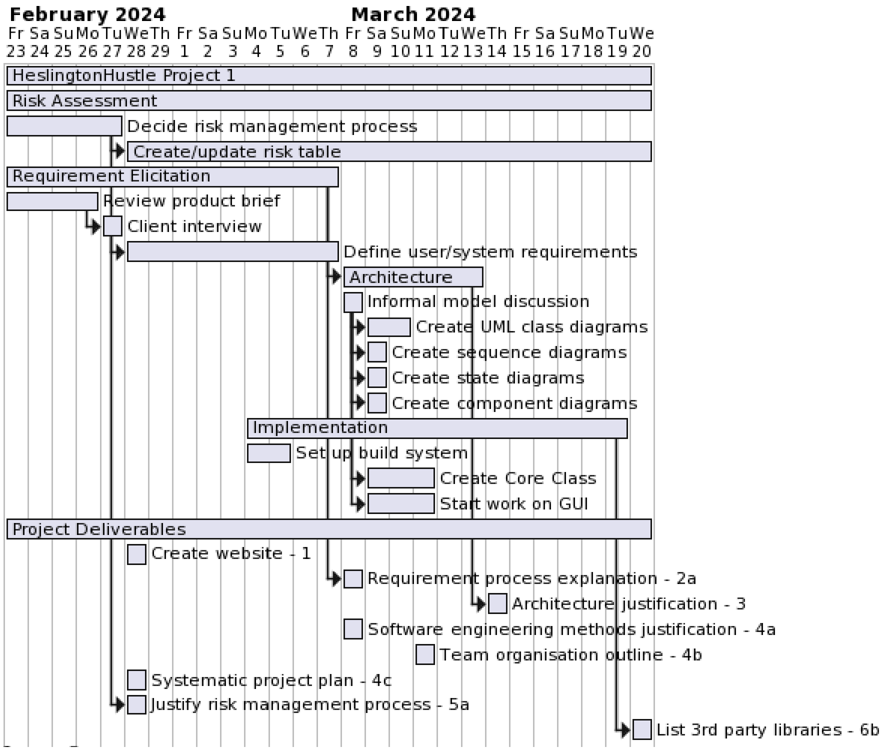
Week 2 & 3 Plan:

At the start of the project (week 2 and 3) we created a rough plan outlining when we wanted each section to be completed by. At this stage we knew exactly when the requirements elicitation and risk management would need to be done by. As you can see, the Risk Assessment spans across the entire project. This is because as we worked on the project, there were likely to be new risks discovered that would need to be added to our risk management tableau. Requirements elicitation would take until our practical session at the end of week 4. We planned to spend more time on this section as it is arguably the most important. Without well defined requirements the project would suffer.



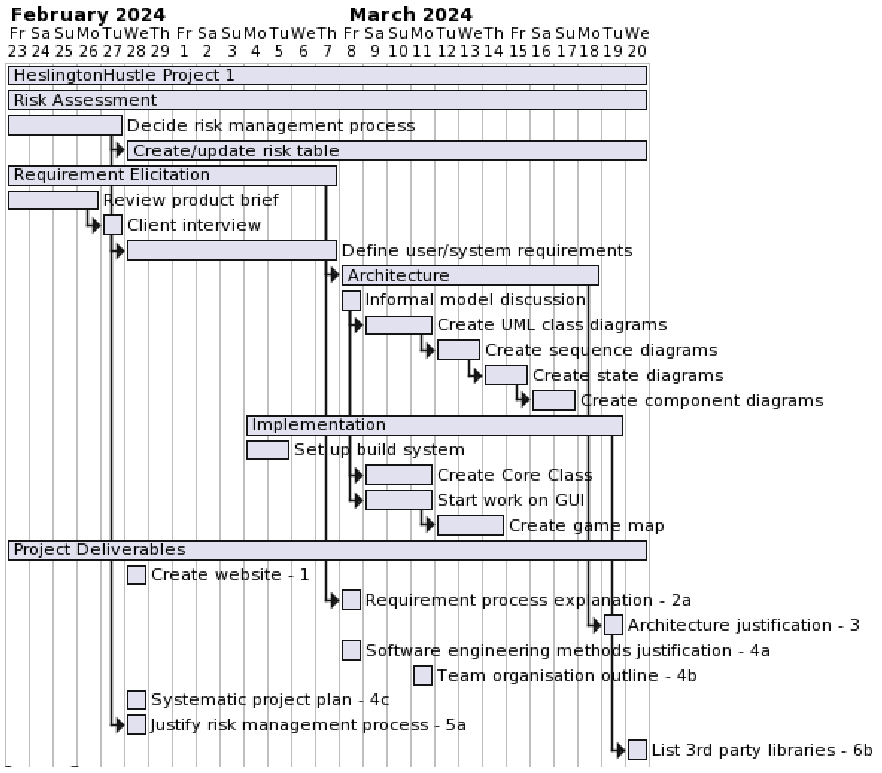
Week 4 Plan:

Our week 4 plan shows updates to the Implementation section. Initially this section came after Architecture, however we realised that these two could be happening in parallel. After our Informal model discussion, both the architecture and implementation team knew the structure of the game so could both work on it at the same time. This was an important change to make, as without it the quality of our game would likely have suffered, due to insufficient time for implementation.

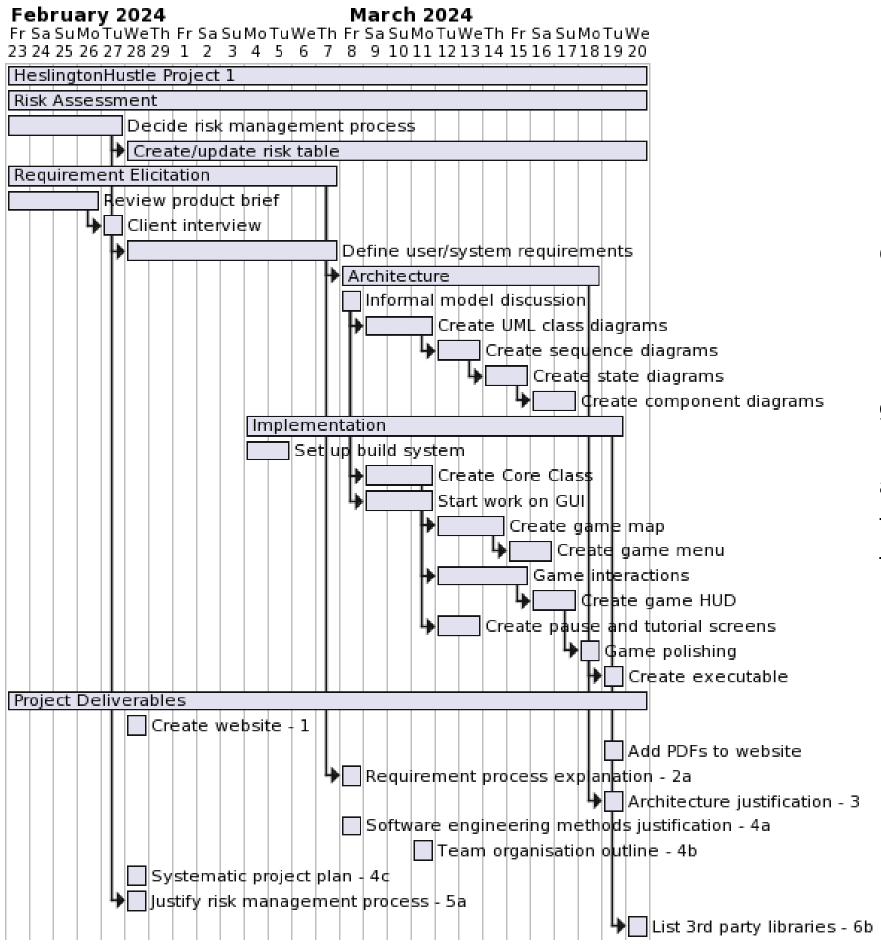


Week 5 plan:

By week 5 we realised that we had not assigned enough time to the architecture section of the project. This is something we had outlined as a risk of the project at the start, and it can be found in the Risk Assessment. Although we had initially planned to only spend just under a week on Architecture, we left plenty of time to complete it after the initial assigned time, allowing us to extend the architecture section of the plan to the start of week 6, giving the architecture team ample time to create all the necessary diagrams. At this point implementation was also updated to include the next steps in the programming.



Week 6 Plan:



In the final week of the project, we see the complete plan. Implementation has been updated to plan the programming of the game to completion. Using our plans we were able to stay on track with the project deadline, finishing a day early, ready to submit.