Method Selection and planning

4a: Justification of Software engineering methods

Software engineering methods

In terms of planning we used an agile method, more specifically following a scrum approach. Each development in the project was divided into sprints, getting together for a daily meeting reporting what each person had accomplished and what the plan for the rest of the day would be.

We decided on this approach as we wanted to stay flexible around the changing requirements of the project. There were constantly changing factors due to new game ideas we'd brainstormed or new requirements we faced after our multiple customer meetings. For example we decided to not use graphic violence after our client stated the game was to be played on open day. Another significant change was our decision to switch from Box2D to Scene 2D, all while still programming the game. However due to our agile method planning this was a seamless transition and did not take a heavy toll on time management.

Trello

Trello was an extremely helpful tool when it came to integrating the scrum approach into our project. We created a board for each specific task (architecture, risk assessment, etc) where we split the page into 3 sections consisting of what to do, what we're doing, and what we've done. This just aided in staying organised with each daily task and making sure no one was working on the same job.

GitHub

A few of our members had used GitHub before and were fond of the site. We ultimately decided on using it as it was perfect for the situation we currently find ourselves in. As there was no possibility of meeting up GitHub was ideal for remote collaboration as we all coded from separate locations.

Another aspect that attracted us to using GitHub was its security. We'd all have versions of the code and if anything was to go wrong or astray we could very simply revert back to old versions of our code or use someone else's branch, a possible problem that we had noted down in our risk assessment, mitigated through the use of GitHub.

Drive/Docs

Google drive was one of the first tools we decided on and was a very simple decision among the group. Everyone is very familiar with it, especially when it comes to group projects. We knew it was important to stick what has worked for us in the past and so Google Drive was dropped into the conversation immediately.

An alternative we had considered was OneDrive, however as students we have unlimited storage space on Google Drive, and we were more familiar with Drive.

It's easily accessible to everyone in the group and a good way of tracking which tasks have been completed or need to be assigned, which helps keep everyone motivated to work as you know your fellow team members can see who has or hasn't been keeping up with the tasks.

We knew it would be particularly useful at the beginning of the project where the focus is on documentation as we would split up in pairs to work on requirements and google drive and docs allows multiple users to work on the same document.

Intellij

In terms of alternative tools we had considered, the most significant decision came down to choosing between Eclipse and Intellij. Ultimately we came to the conclusion that Intellij was the best choice for our project. We had members of the group who had used one or both so we had a team discussion on the positives and negatives of the two choices.
Whilst both have their advantages the majority voted for Intellij. Fraser gave us the strongest argument as he had attempted to create a game recently, specifically with Intellij and found that it had been much easier to learn the finer details of it if you had never used it before, in comparison to Eclipse which had a steeper learning curve.
We had already discussed the importance of time management and staying as efficient as possible, which led to us picking Intellij in the hope it saved us time and stress, due its user friendly interface.

4b: Team organisation

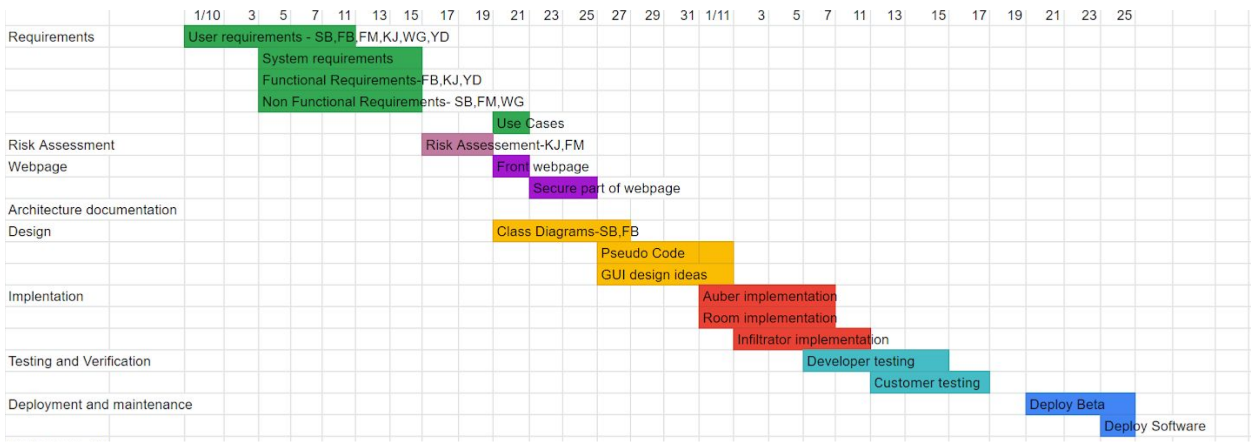
Our group has set up a workspace on Trello, an online collaborative teamwork platform. We have created separate rooms for each deliverable, within which there are 3 sections: to-do, doing and done. This approach is appropriate for the team as we are unable to meet in person due to the current COVID-19 lockdown situation, and it keeps all team members up to date with what tasks have and haven't been completed yet.

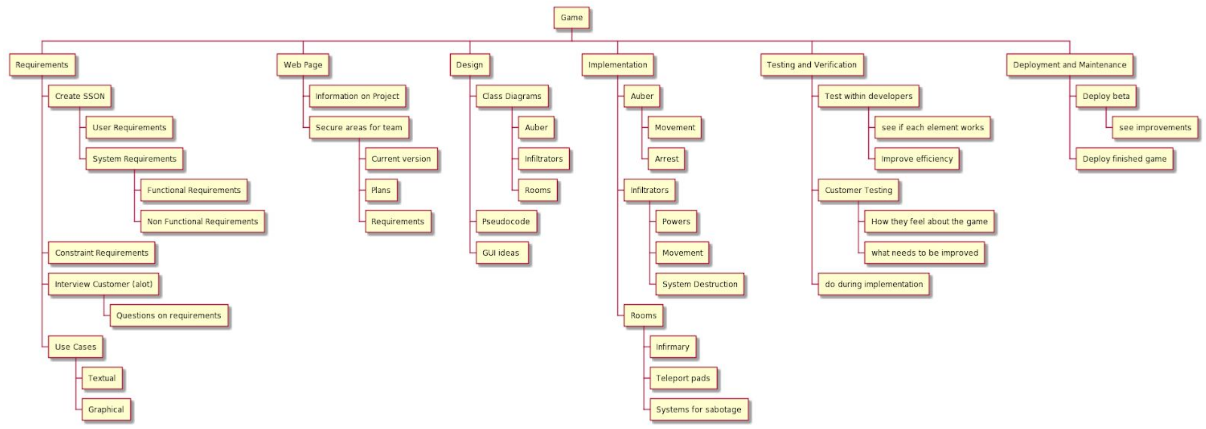
We have also set up a Google Drive, to which each member can upload work they have done to a Google Docs file. This is where all of our work was uploaded, and from there it was added to the Github Pages

We have set deadlines for each deliverable and their respective subsections using a Gantt chart. This is suitable for the project as time management allows us to keep on track with deadlines and not fall behind.

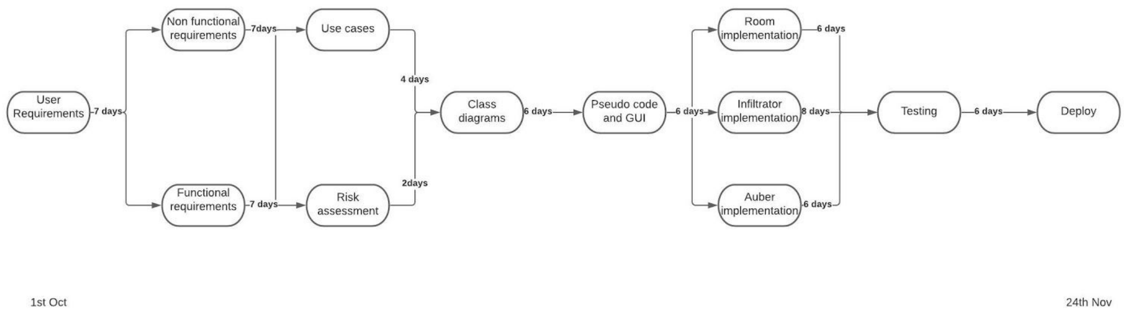
For each small task, roles are assigned during our group meetings, and each group member can assign themselves any desired roles by annotating the Gantt chart with their initials. Letting group members to assign themselves tasks allows for work to be allotted without the need for a group meeting, and can increase productivity if each group member is working on a task they want to complete.

4c: Project plan





<u>Our critical path</u>



Our critical path shows the estimated minimum possible amount of time. From this we can deduce the project will take at least 44 days by adding the paths together (taking the larger from the parallel path). This optimistic estimation suggests we may finish on the 13th of November which gives us 11 days extra time if anything goes wrong.

<u>Our plan</u>

1/10/20-16/10/20

In the first week, we will discuss with our stakeholder their needs. This task priority is very important as every other task depends on it. Our user requirements from this meeting should be finished on the 11th of October. From our stakeholders needs we will decide our game requirements which the implementation depends on.

17/10/20-20/10/20
When we finish the requirement we will be looking at the risk assessment. We will identify the risks that are likely to threaten the project and we will analyze the consequence of it. In addition we will discuss how to minimise or avoid the causes of the risks and throughout the project we keep monitoring the risks to see if the risks are happening. This is medium priority as if something goes wrong in our project we will be better equipped to deal with it.

21/10/20-26/10/20
After we finish the Risk Assessment, we will start doing the Use case and the web page.The web page has a low priority at this stage as nothing depends on it and can be done any time during the project.

21/10/20-28/10/20
Simultaneously we will begin the class diagram design. Our class diagram design is high priority as the implementation depends on it. If our class diagram is not detailed enough or badly engineered this will affect the outcome of our game. For the class diagram we will design the classes we need, their attributes and methods as well as the relationship between classes.

27/10/20-2/11/20
We will then begin the pseudo code which will give us a general idea of what to code in the implementation state. We will also begin to design the GUI which may inform us on extra things we need to implement for our game to be played easily. The pseudo code is medium priority as the code depends on it however it is not essential to write the code.

1/11/20-12/11/20
We will then start coding, starting with Auber and the rooms. After that has been tested we will begin coding the infiltrators. The infiltrators depend upon the room as it needs a destination for the AI (the ship systems)

7-11/20-18/11/20
We will then begin the test cycle in unison with our coding, testing what we have already done before moving on. This is a high priority and the rest of the code is dependent on it as we cannot keep writing code until it has been tested.

21/11/20-24/11/20
We have left a gap between testing and deployment to allow for room if any part of this project requires more time. By the 23rd we should have completed the entire project.