Method Selection and Planning

TEAM 29

Team Members:

Michael Beal, Lucus Soloman, Ryan Early,

Daniel Pearce, Thomas Auburn, Inna Strogonova

Assessment 1 Team Members(Team 32):

Ethan Higgins, Lakhan Mankani, Clara Hohenlohe, David Thomson, George Burke, Adam Gurdikyan

Method selection and Planning

Outline and Justification

After researching and discussing various team working methodologies, we decided to use Scrum. At the beginning of the project we prioritised the task of choosing a team working framework as it would lay out the foundations of how the teams would operate from the very beginning. Various options were suggested by team members and we discussed the benefits and drawbacks of each method. Given that the project had to be completed in a short period of time and a quickly changing set of requirements, factors such as rapid prototyping and flexibility were the most important aspects of choosing the right methodology. The most popular methodologies that were considered were Plan driven methods and Agile. Plan driven methods were quickly identified as not being suited for our project as it required us to make a detailed plan at the beginning and did not accommodate changes in the project requirements very well. Scrum was the best suited option as it was designed for smaller teams and encouraged smaller incremental updates through shorter development cycles. Shorter development cycles adapted well to changes in the requirements as it didn't require us to revise the entire plan.

Various frameworks based on the Agile such as Scrum and Kanban were available to use. Through research, we identified the following key differences between Scrum and Kanban[1]:

- Scrum's implementation of sprints required us to lock-in the tasks for the upcoming week and had a higher overhead of planning as opposed to Kanban which allowed more flexibility.
- Scrum promoted delivery at the end of each sprint. Whereas, Kanban primarily focused around continuous delivery of features. In this respect, Kanban was best suited for our team.
- Kanban's organisation of tasks was arranged in a more visual manner through the use of a Kanban board which highlights what the status of a given task is.

To facilitate communication among team members we decided to use Discord. The application met our requirements as a team and everyone was already familiar with it so it was easy for everyone to get setup. Other options such as Slack were considered but Discord was chosen due its ease of use. Through the use of communication channels, we were able to structure our communication and have designated places to talk about certain topics. The ability to easily share content such as images and links enabled us to share ideas and design concepts. We hosted our team meetings on Discord using the voice chat feature as it provided us with a more integrated solution to our communication needs such as screen sharing.

We opted to use Git after researching various Version Control Systems such as Mercurial and Apache Subversion. The decentralised approach of Git and Mercurial ensured that we did not have to worry about maintaining a central server where our repository would be hosted. Although Git had a steeper learning curve as opposed to the other options, the wider availability of tutorials and documentation made it a good choice. We decided to use GitHub as the central repository as it offered us additional functionality that worked alongside Git for example pull requests and issue tracking. Other options to host the repository such as GitLab were explored but provided no significant advantage in achieving the task at hand.

IntelliJ IDEA was chosen as the primary development environment as it had good support for writing Java code. The game engine that we used in our implementation of the project had out of the box support for IntelliJ so we didn't have to spend a lot of time configuring. Tools provided by the code editor such as debugging and refactoring support were very useful in the development of the game. Another alternative that was considered was Eclipse but we found the interface complex to navigate and lacked features that were available in IntelliJ. Although Eclipse does contain a number of features not included in Intellij such as the ability to create UML diagrams from code. This is a useful feature but not required while coding the actual project.

Team Organisation

We decided to designate Michael as the leader of the group by being the only person who was willing to take the position. Michael demonstrated his excellent team management skills through having a deep understanding of how all the tasks related to each other and the timeline of the project plan as well as the ability to coordinate group members to complete the required activities.

Our approach to team working was based on the idea of having clearly distinct parts of the deliverable and designating a person who is responsible for that section. This person was considered to be the 'owner' of that section. The owner delegated tasks, ensured that the work done by team members was up to standard and set the timeline for tasks. We identified each team member's key strengths at the beginning of the project and assigned them roles that best suited their skills and expertise.

During each week's meeting, we began by assessing the tasks that need to be completed. Tasks that were the most pressing were assigned immediately to a team member who was best equipped to tackle it. These meetings were also used as a way for members to update the team on progress on previously assigned tasks. Any tasks that were 'stuck' would be discussed among the team to come up with new suggestions or a task might be reassigned to another team member who would attempt to solve the issue from a different approach.

New Teams roles

Team member	Role	Key responsibilities (ownership)
Michael Beal	Programmer and change report	Implementing difficulty and saving features into the game as well as a pause screen and writing up these changes. Also writing up the change report and carrying out relevant changes.
Lucas Solomon	Deliverable Changes	Updating the Abstract and concrete architecture and creating relevant diagrams.
Ryan Early	editor and continuous integration.	Reviewing and editing updated deliverables and helping to write the continuous integration deliverable.
Daniel Pearce and Thomas Auburn	Programming	Implement the users abilities and write up the implementation.
Inna Strogonova	Website, software testing	Maintaining the website and uploading relevant deliverables to it when required as well as helping to produce diagrams. Designing and implementing tests, writing testing report.

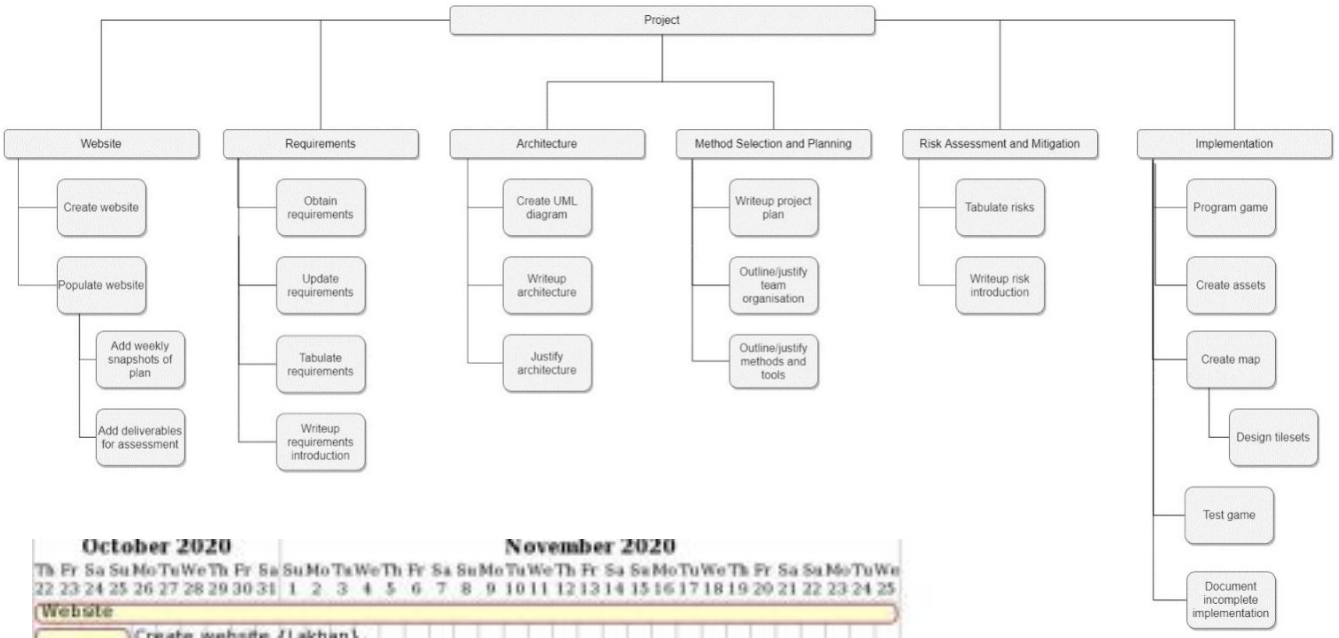
Old teams roles

Team member	Role	Key responsibilities (ownership)
George Burke	Programmer	Core systems programming, map design, asset creation
Adam Gurdikyan	Helper	Report writing
Ethan Higgins	Report editor	Code review, NPC design
Clara Hohenlohe	Secretary	Revising project report, user interface design, keep meeting log
Lakhan Mankani	Librarian	Website management, character design, GitHub management

Bibliography

[1]C. Drumond, "Scrum - what it is, how it works, and why it's awesome", *Atlassian*, 2020. [Online]. Available: https://www.atlassian.com/agile/scrum. [Accessed: 08- Nov- 2020].

Systematic Plan - HD images available from our GitHub repository here



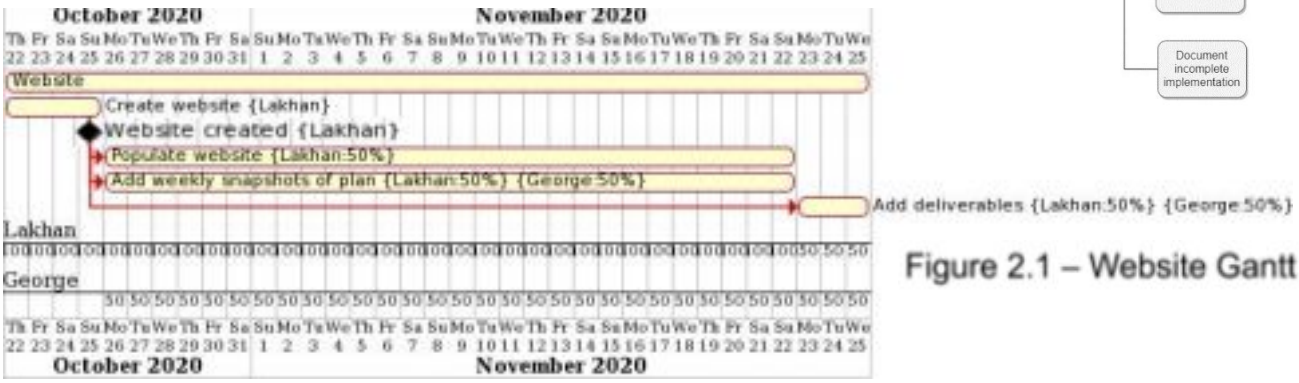


Figure 2.1 – Website Gantt chart

Figure 2.2 – Requirements Gantt chart

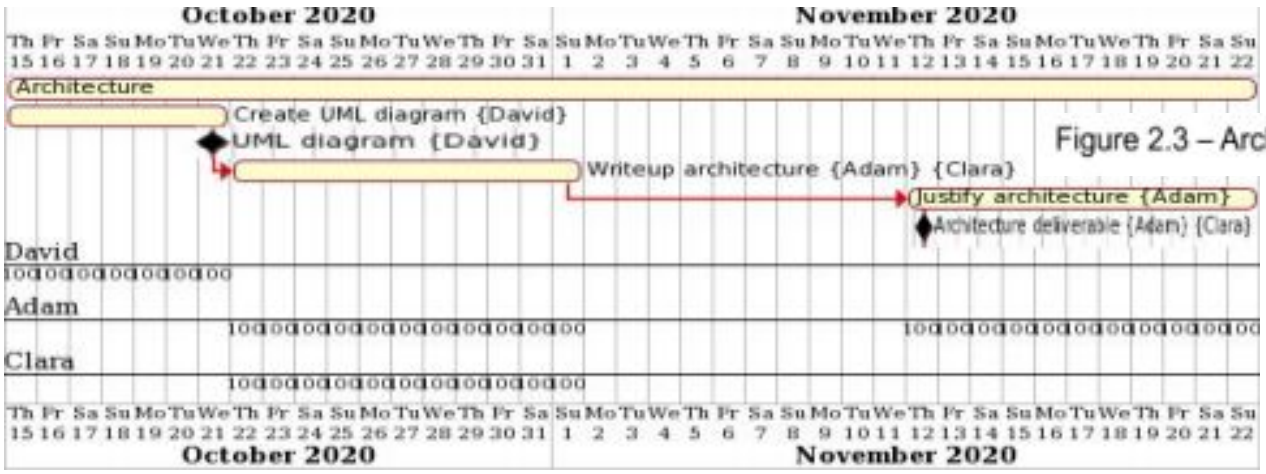
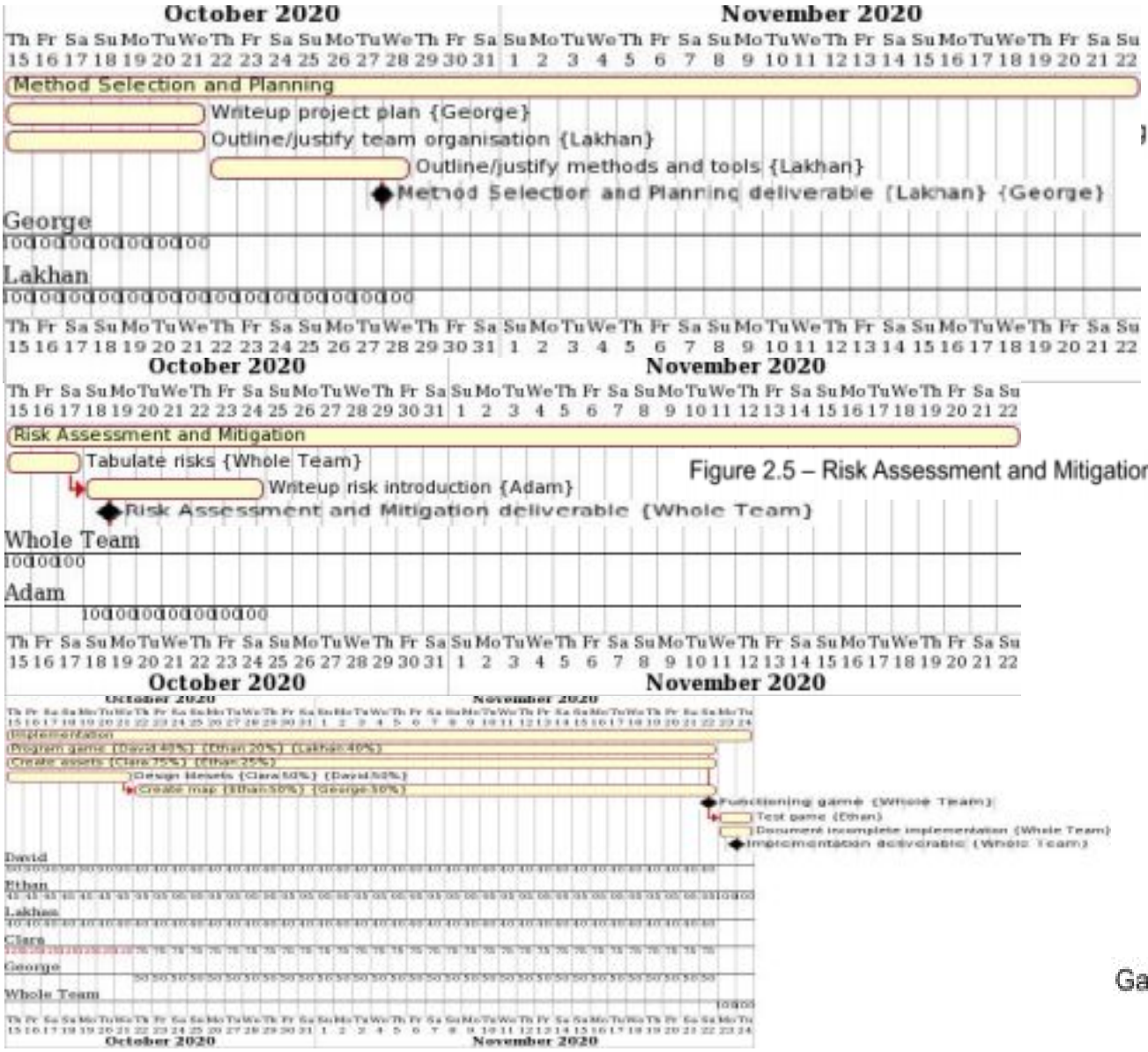


Figure 2.3 – Architecture Gantt chart

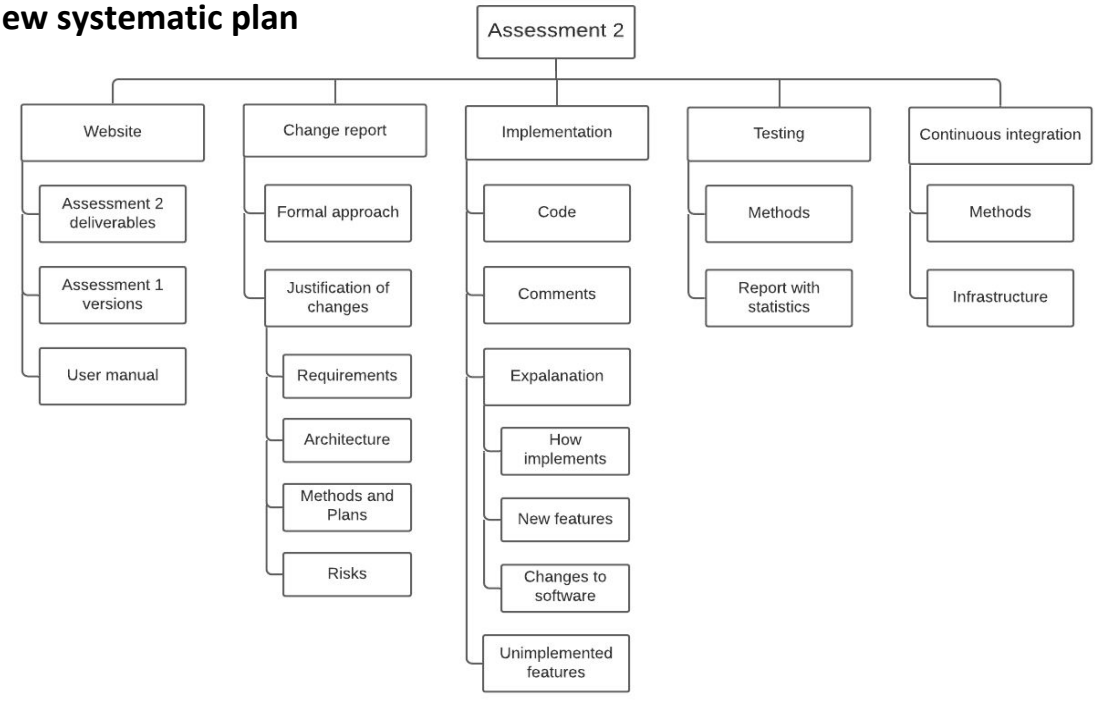


Gantt chart

Figure 2.5 – Risk Assessment and Mitigation Gantt chart

Gantt chart

New systematic plan



The assessment 2 gantt chart is available here.