Method Suggestion and Planning

Group 8 - Delta Ducks

Yumis Zyutyu

Zac Challis

Viktor Atta-Darkua

Dandi Harmanto

Harsh Mohan

Danny Burrows

a)

- Agile software development was chosen as the team's software engineering method. *Agile* is a methodology adaptive to changing environments centred around the idea of iterative development over multiple cycles, such as requirements gathering, analysis, design, coding, testing and partially implemented software delivery. [4] All evolve through the collaboration between self-organising cross-functional teams and a customer-centred design. [3] Design requirements and implementation of such are central activities in the software process. Letting the customer evaluate the development process and getting rapid feedback makes requirements and design develop together, which increases customer satisfaction. [5] The informal communication between the customer and the team reduces the formal documentation process used in other methods. The system requirements are constantly ready for changes, driving it more likely to meet customer satisfaction. Due to agile's iterative development, the small teams can quickly deliver new requests from the user, incorporating requirement elicitation and testing into design and implementation. The non-productive work is reduced as there is frequent work distribution and immediate addressing of any issues. That further increases efficiency and user satisfaction. [2, pp. 74–77]

- Scrum was chosen as the framework of the project. It uses a structured supervisory approach to teamwork, enabling the team to constantly accommodate the possibility of changing requirements such as the alteration of customer expectations or shift in available resources.[2, pp. 75–77] The team was split into two smaller teams, based on the abilities of the team members - programming team and writing team; scrum meetings were held regularly in which: the workload was defined by the product owner, decided upon and distributed to the team. Focusing on the highest priority tasks emphasised the completion and delivery of fundamental features. Productivity was sustained, on account of the frequent updates on completed and undertaken assignments during each scrum meeting, in addition to any issues that had arisen, enabling the scrum master to reassign resources before the next development sprint.

Collaboration tools used:

- **Discord:** Discord is an instant messaging and digital distribution platform. Consisting of the ability to create different channels (text or voice) for dedicated topics, allowing users to stream their screen live and share files.[6] Its primary role in our team was to create stable and relevantly simple communication between the team members. The ability to split the server into different channels and work simultaneously on different tasks was a significantly important feature as it supported the framework we chose - scrum. Discord's specific search abilities allowed us to trace past conversations and files for guidance easily. The chosen framework required weekly team meetings to keep the different teams up to date with the progress; they were hosted on Discord due to convenient screen sharing, making it easier to directly showcase the team members' progress. Discord also consists of different features that can be downloaded and implemented into the server; the one used by the team was *webhooks* allowing us to receive notifications regarding any GitHub updates on the project.

- **Version Control**: (Git) Having a Git repo made it easy to keep track of the project. All files necessary for specific analyses can be held together, and people can add in their code and graphs as the project develops. Each file has a history, making it easy to explore the changes that occurred to it at different time points. Other people's code can be reviewed; comments are added to certain lines or the document to allow suggestions for changes. Allowing you to keep track of your work and easily navigate among the many versions of the files you create while maintaining an online backup.

GitHub Project: Used for the organisation of tasks allocations. The project board is a collection of card-like columns consisting of pull requests, notes and issues designed for teams for practical project assessment, documentation, monitoring and reporting work in progress. We had a 5 column Kanban board for a broad distinction of stages, with custom labels to identify the priority and nature of the task. We also allocated the task here, which notified and created the log of every stage to trace the project's timeline. Since we used GitHub for a significant portion of our project, we had everything in one place, which was very convenient, and we applied to use GitHub at its peak potential.

- **UML Diagram**: (Gleek & IntelliJ/XML) - used to create UMLs for the abstract and concrete architecture overviews. Visualising how our system was designed to support any future or current developments.

- **Google Drive** was used as a cloud server; the primary usage was to keep every file created and who has edited or helped write every document. Collaboration was made more accessible, owing to the ability to accommodate multiple simultaneous edits, allowing multiple sections of the same document to be completed at once. The real-time updates enabled quick and convenient review of sections others had completed; this was favoured instead of alternatives in which the document needs to be downloaded, edited and then reuploaded to shared space.

- **Planning**: (OfficeTimeline) - Used by the team as a Gantt chart generator to lay out the key tasks and their starting and finishing dates. Leading to an efficient layout of all the workload. Giving the team a realistic expectation of when the project is finished and layout dependencies. To represent dependencies, in our Gantt chart, we have clearly shown that the dependent task's start date will not begin until the previous completion of the task it depends on. For different levels of predicted time, e.g. optimistic and pessimistic, we can have more than one bar in a row, with different colours to make them distinct (e.g. a large blue bar and a narrower black bar that are distinct from each other).

- **IntelliJ:** Originally, it was decided that Visual Studio Code would be the IDE used to develop the game; however it was discovered there were issues regarding its compatibility with the chosen game engine LibGDX; therefore, IntelliJ was chosen as a concrete alternative. IntelliJ made implementing the game faster owing to its built-in Gradle support.

- **Implementation**: (Java with libGDX)

- **Graphic design**: (Piskel) - the team used Piskel to design images for the game due to it being free, and the user-friendly interface enabled it to be picked up faster, allowing game graphics to be designed more quickly. Accommodating for the limited time available.

b) The initial approach we took to our team organisation was not to select an outright leader. Instead, we let the natural leader emerge as we went on with the project, implying we ended up with the person best suited to lead specific parts of the project. That decision was taken because different group members have different strengths.

We settled on a participative leadership style where the leader would offer guidance to the group and participate in the group the same as the other members.[8] The leader would take input from all other group members and engage in the discussion assuring all team members are involved in the decisions and making the leader act more as an overseer, leaving the big decisions to be made as a team.

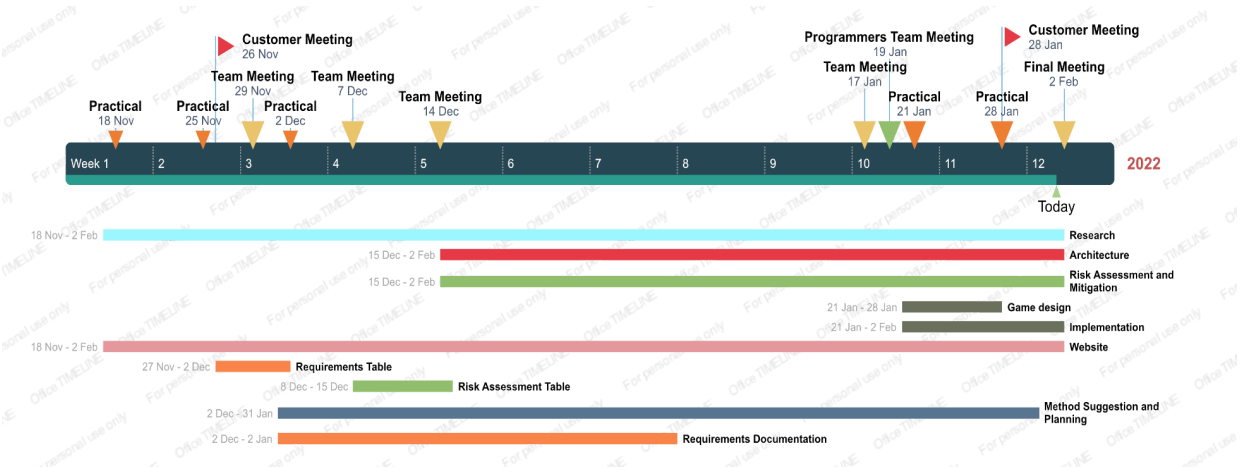
As a result of this approach, the team members felt more involved in the project overall and thus more motivated and productive.
The approach towards assigning tasks was to divide each task into smaller tasks that would then be assigned to different team members. How these tasks are split up would be decided by the whole group over a discord call and allocated to suit each individual's strength, we would then set realistic deadlines for these tasks to ensure we do not fall behind. This approach allowed us to work more efficiently. Each team member worked on a section they are particularly good at whilst providing more overall flexibility for the project as each member can work on their section in their own time. This approach can also allow members to easily switch to other sections of each task if it is found that they are better at a different section. For more extensive and critical sections, a secondary member shadows that section of a task as a backup just in case. Furthermore, the team will review all sections before the task is classed as complete to ensure everyone is happy with the outcome.
These methods do come with some risks; however, one of these risks is that cutting tasks into sections for individual people requires a higher level of trust that they will complete that section correctly and on time. This risk is decreased by having someone shadow certain sections and having the group review the section as a whole, but this takes extra time. The method will also rely on good communication, so deadlines are met in time, and we are not held back; for this, we will use GitHub to assign tasks to people with deadlines that are Discord in our meeting over Discord.
Further on, closer to the deadline, the team's approach towards team organisation changed around maximising the members' strengths and dividing the team into programming and writing sub-teams, which allowed us to work on implementation and documentation simultaneously.

c) Systematic Plan for the Project

Priority: 1-5 Where 1 is most important

ID	Description	Start date	Finish Date	Priority	Dependencies
T1	Setting up the website.	18/11/21	25/11/21	3	
T2	Drafting initial requirements and risks questions.	18/11/21	25/11/21	1	
T3	Initial customer meeting.	26/11/21	26/11/21	1	T2
T4	Forming initial requirements.	26/11/21	28/11/21	1	T3
T5	Project planning.	6/12/21	28/01/22	1	T4
T6	Assessing risk and mitigation.	29/11/21	4/12/21	1	T4
T7	Consolidating plan and requirements.	2/12/21	28/01/22	1	T4
T8	Architecture & design discussion.	21/01/22	31/01/22	2	T7
T9	Initial prototype.	21/01/22	28/01/22	1	T8
T10	Second customer meeting presenting prototype	28/01/22	28/01/22	1	T9
T11	Refactoring and replanning/refinement customer meeting.	28/01/22	Ongoing	1	T10
T12	Make images required for game	6/12/21	24/01/22	2	T8
T13	Research	18/11/21	02/02/22	3	

Please find the Gantt Chart below:



Snapshots Descriptions (Please note snapshots can be found on the website.)

- **18/11/21**- The team got to know each other and discussed basic ideas for approaching the project. [T5]
- **25/11/21 -** The team brainstormed questions to ask at the customer meeting to get to know the customer better and understand what the customer desired the product to be satisfying. We then familiarised ourselves with the product brief and set up a Github project.
- **2/12/21 -** The first draft of requirements and risks was completed. We assigned tasks for the project planning section and decided on the game engine we would use.
- **7/12/21** - We have just completed the initial forming of requirements (T4) based on the answers provided in T3. We will continue to refine the requirements and risks as the project progresses. Furthermore, we have started on project planning, although as of yet is mainly in the discussion phase; written deliverables are to come soon. So far, we are on track, potentially even ahead of schedule.
- **14/12/21** - Implementation of the game discussed. The progress of the method suggestion and planning draft previously assigned was reviewed, and further updates were done.
- **17/01/21** - The meeting was used to catch up and further divide the team into two sub-teams to maximise efficiency.
- **19/01/21** - The programming team brainstormed ideas on what the game would look like. Moreover, discussed any software they will require to use.
- **21/01/22** - *The programming and writing team met to discuss any updates on the allocated tasks. The programming* team got familiarised with LibGDX. *Furthermore, concrete and abstract architecture was discussed.*
- **28/01/22** - The primary purpose of the meeting was to make sure there was *a prototype of the game ready for the customer meeting in order to be able to present the prototype to the customer and adjust any customer requirements if need be. The team stayed extra to discuss any changes that might need implementing.*
- **02/02/22 -** The team's final meeting. We met to go over all the documentation and implementation. The purpose was to make sure we all agreed on the assessment and that nothing was missing before submitting it.

Bibliography

[1] Agile Manifesto, “Principles behind the Agile Manifesto,” *Agilemanifesto.org*, 2019. https://agilemanifesto.org/principles.html.

[2] I. Sommerville, *Software Engineering*. Hallbergmoos/Germany Pearson, 2018, pp. 75–77.

[3] K. Trapani, “What is Agile/Scrum,” *cPrime*, May 22, 2018. https://www.cprime.com/resources/what-is-agile-what-is-scrum/.

[4] S. Sharma, D. Sarkar, and D. Gupta, “Agile Processes and Methodologies: A Conceptual Study,” May 2012. [Online]. Available: https://www.yashada.org/yash/egovcii/static_pgs/TC/IJCSE12-04-05-186.pdf.

[5] B. W. Boehm and R. Turner, *Balancing agility and discipline : a guide for the perplexed*. Beijing: China Electric Power Press, 2004.

[6] “Discord Blog,” *discord.com*. https://discord.com/blog (accessed Jan. 31, 2022).

[7] Bluescape, “UML Diagrams - Everything You Need to Know About Process Visualization,” *Bluescape*, Sep. 12, 2019. https://www.bluescape.com/blog/uml-diagrams-everything-you-need-to-know-to-improve-team-collaboration/.

[8] Harappa, “Participative Leadership: Definition, Characteristics, and Examples - Harappa,” *harappa.education*, Oct. 11, 2021. https://harappa.education/harappa-diaries/participative-leadership/#:~:text=Participative%20leadership%20refers%20to%20leaders,decisions%20depend%20on%20the%20leader..

[9] Project Manager, “Gantt Chart – Long Form Info Page,” *ProjectManager.com*, Mar. 26, 2019. https://www.projectmanager.com/gantt-chart.