Module	Engineering1 (Eng1) - COM00019
Assessment Title	Assessment 1, Cohort 2
Team	Dragon Boat Z (Team 18)
Members	Robert Dalgleish, Benjamin Jenner, Joseph Lonsdale, Richard Upton, James Wilkinson, Xinyi Zhang
Deliverable	Method Selection and Planning

Software Engineering Methods

In order to meet the constraints of the stakeholder's outlined requirements, a methodology that would assist this development in a beneficial way was needed. When deliberating on which methodology should be chosen, the following points were considered: time constraint of the project and its size, the requirements that must be fulfilled and the need for flexibility. This narrowed our decision to three types of methods: Plan-driven, Soft Systems and Agile methods.

Plan-driven methods provide a structured way to create software by predicting and anticipating all future features that may be implemented. Planning of the project can be promoted by providing comprehensive explanations of tasks, workflows and responsibilities [1]. There is an emphasis on continued review and risk management analysis. This detailed level of planning requires the project to be developed in a stable environment and is mainly used in projects that have a long working period [4]. Plan-driven methods that were researched include Capability Maturity Model and Team Software Process (TSP). The focus of TSP is to create an environment that supports disciplined work and maintaining a self-directed team, that would help to establish an effective management and structure to the project. However, with TSP requiring the need for experience of managerial skills and Plan Driven Methodology not catering for the project's need for flexibility, the use of Plan-Driven methods was not chosen [7].

Soft System methods look to break down complex problems in order to truly understand the problem outlined by the stakeholder, and use this to develop multiple conceptual models that can be compared to the original real world problem [2]. The time taken to understand this problem and discuss potential solutions is a lengthy procedure that takes weeks. The model has limitations in comparing the conceptual model to the real world problem and it does not provide a framework for how the design should look, instead focusing on the search process for the best solution [9]. The time-constraint and potential need to integrate other Models meant this plan was not a rewarding model to use for this project [6].

Agile methodology focuses on the "concept of ongoing waves or sprints of project planning and execution" enabling the project team to adapt the design, scope and execution of the project deliverable [5]. Code is distributed in minor releases with short periods, where the project team works together with the Customer in close cooperation. The approach is easy to understand, and its core principles were aligned to the scope of the project and the stakeholders requirements. The focus on frequent software delivery allows for the development team to produce the product in a shorter time frame, when compared to other methodologies, and its allowability to iterate and innovate on functionality of the system would help to maintain a higher motivation within each individual of the team. The scope of the project and its delivery fell in-line with this methodology and was decided on.

With the Agile methodology chosen, **Scrum** was chosen as the framework as it iterated on the continued benefits of the Agile methodology, such as the ability to cope with change through short iteration cycles, allowing for quick feedback, and usually low documentation overheads. The Scrum method outlined a clear and concise managerial structure to teams that would help the team to adapt to expected changing requirements (customer requirements, time and resources). The structure involved defining a Product Owner, Scrum Master, and Development Team [3]. Sprints will be used to organize and distribute the workload, which are decided on in regular Sprint Planning sessions, within the short time frame [8]. This structure will help to make sure there is a focus on delivering the fundamental features [11].

Development and Collaboration Tools

Scrum management tool: Jira was chosen to manage task assignments in Sprint Planning Sessions and offered feedback for review sessions. Sprints with task allocations are simple to set up and comment on, adhering to Scrum's principle of team organization of member roles and responsibilities, and task priorities can be assigned. It offers a feature set that allows integration of other tools such as Slack for notifications regarding updates to the project. With some team members familiar with Jira, it would allow the team to adapt and make use of the full functionality. Trello and GitHub Project were noted as alternatives, but Jira's simplistic design and integration of Scrum methodology in different project forms stood out.

Documentation tool and Deliverable Repository: Google Drive acts as the central location to store all the documents related to the project. It allows for simultaneous work on single or multiple documents. With support for different document types such as Docs, Sheets, Slides and third-party tools, such as UML diagrams, it stood out over alternatives. These alternatives included Microsoft Word which was not chosen as it only allowed for asynchronous collaboration and the transferring of this document type required additional services. Dropbox and OneDrive (with its integration of further Office tools) were noted but their feature sets are not as vast as those offered by Google Drive. Confluence was also researched (which integrates with Jira) but its cost was too great.

Communication tool: Slack and **Discord** is used to communicate with other team members. Slack's feature of multiple channels can be utilised, which is useful for localizing communication between those who are assigned to a specific task. It pushes notifications to ensure every team member is kept informed about any issues or queries surrounding the project. As Slack does not offer voice chat functionality in the free version, Discord is used for voice communication, as it provides voice channels and the ability to share the current user's screen to other users, useful when tackling tasks collaboratively. The group members are familiar with these tools, which ensures that everyone can maintain and keep track of ongoing activities within the project sections.

Version Control tool: GitHub will be used for the version control tool for the project. It works as an online repository and allows us to collaborate on code development. The feature of branching will prevent unnecessary confusion and damage to the project, when team members are coding at the same time. As GitHub is a widely used VC tool and offered large amounts of integration with other software, it was chosen.

UML diagram tool: Lucidchart will be used to create the architecture diagrams for the game. It allows for early discussions of basic entities to be conceptualised and the relationships between them, before the implementation process is started. It allows for flexibility when adding or removing functionality in later iterations of the diagram. The tool is easy to learn, free and allows us to work on the UML diagram synchronously with integration with Google Docs.

Implementation Tool: Java using the **LibGDX Module** will be used as Java is a customer requirement stated in the Product Brief. LibGDX will be used as the game library, as the documentation and online resources about it are vast (including tutorials). Additionally, there was a consensus within Cohort 1 and 2 over its use, allowing for easy transfer of projects in Assessment 2. Alternatives considered were Slick2d and Joge (Java OpenGL Game Engine).

Team Organisation

Organisation - Jira is used for managing tasks, to help us rationally finish all the parts of the assessment and review sprints. The project tasks are stored within a board that uses Jira's built-in Scrum project style. Tasks will be assigned during Sprint Planning sessions inside the corresponding board with deadlines. The Sprint length will be one week. The Sprint Planning session will involve understanding each individual's time management and assigning tasks for the next sprint including who the task is assigned to, description of the task at hand and a priority value (determined by the defined Gantt Chart). To monitor progress of a Sprint, tasks will be stored in three columns: To Do, In Progress and Done. This will enable the team to maintain an understanding of each individual's progress with their allocated tasks.

To further this, two meetings will be attended by team members each week to plan Sprints, monitor current tasks and review the completed Sprint. Jira's functionality of Reports should help to offer further insight into these sessions such as Sprint Reports (understand work completed and issues that have been pushed to the backlog). To support the use of Jira, Google Docs should be used to record meeting notes for these sessions and record attendance. These were expected to prove useful for those who could not attend and allow for reflection on issues or points discussed.

Roles and Tasks Allocation - To understand and play to each team member's strengths and potential weakness, a personality test, suggested by James, was conducted by each team member. Using this alongside the team's individual experience and confidence in programming, tasks were assigned. Everyone in the team will be assigned to tasks during the whole process and every task should be allocated to at least two team members who prefer to do the task and have strengths towards it. This will help to mitigate any potential backlogs that may build up at the end of Sprints.

Roles assigned were a Product Owner, who will ensure that the most value is delivered to the business by the development team, and a Scrum Master, that coaches every team member and ensures they are adhering to the Scrum method [7]. The Scrum Master was chosen to be Richard U due to his experience with Jira prior to the project, and displayed aptitude for driving meetings and team organization. The Product Owner is Joe L because of his experience with coding games in a 2D environment with Java. This means that when it comes to making sure requirements have been satisfied, through means of testing, this can be led by an individual with experience. The Development team was not specified as it will allow the team to respond quicker to volatile situations throughout the project.

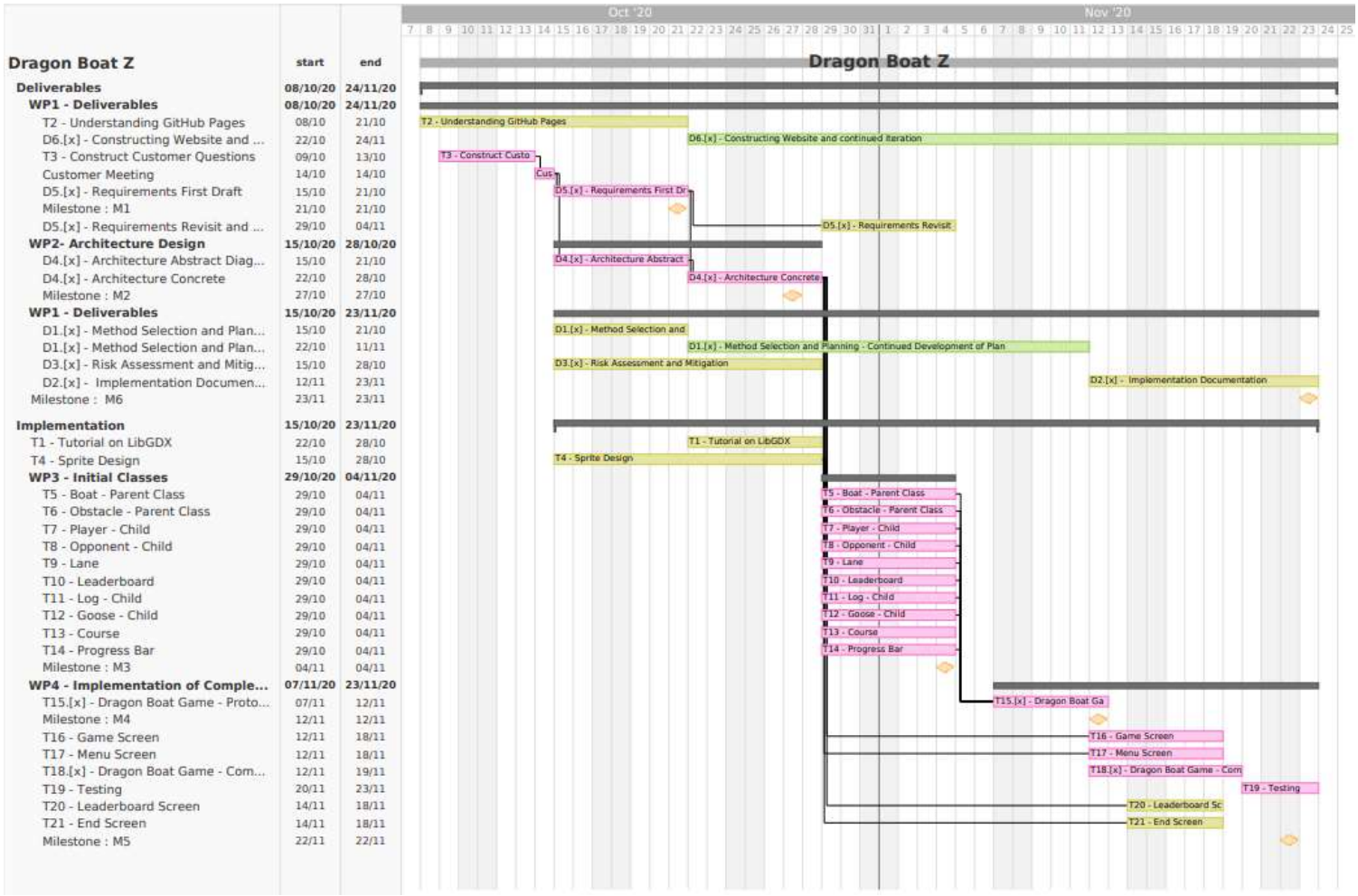
Development Architecture - Using the principles outlined in research of the SCRUM method, meetings are fixed on Tuesday mornings and Thursday afternoons each week using Discord. Fixed times were decided on to further ensure tasks will be completed on time and to maintain a routine. Time constraints of the team mean that daily stand-ups will not be possible, so Slack channels will help to mitigate this by enabling team members to continue to communicate regarding current progress with others. If work cannot be completed/solved in a given timeframe, additional team members can be assigned to the task or members may switch the task that they are working on.} This is to maintain the constant delivery principle of SCRUM.

Project Plan

Task priority was decided based on:

- The time taken to compete.
- The number of marks it was worth and the number of people it requires, if related to **documentation.**
- The impact of the task not being completed and the priority value assigned to it in the Requirements deliverable, if related to **coding.**

Using this, an initial Gantt chart was constructed to display the various parts of the project and highlight any anticipated dependencies between tasks. An online tool [10] was used as this allowed for synchronous collaboration between team members and allowed for adjustments to be made in response to volatile requirements. This allows for a clear understanding of the implication of any delays in the project.



Each task of the project was assigned a colour, representing the tasks priority and the expected time taken to complete. Green for low priority, Yellow for medium priority, and Red for tasks critical to the completion of the project. These critical tasks were used to define a critical path (the longest sequence of dependent tasks that must be completed to produce a game that is defined by the SSON). The dependencies (tasks that require prior tasks to be completed) were demonstrated using links between them. To compensate for a potential

backlog between sprints building up, the Gantt Chart allows for a period of three days at the end of the project to compensate for delays in task completion.

The Gantt Chart was split into separate sections for Deliverables and Implementation, to allow for readability. During weekly Sprint Planning sessions, the Gantt Chart will be used as an indicator of tasks that should be completed and prioritized during that time frame. When designing the chart, IDs (Work Package IDs, Task IDs, Deliverable IDs and Milestone IDs) were used to maintain a clear structure to the project. This will be used later on within Jira boards to link to tasks stated in the Gantt Chart.

Tasks are assigned a unique identifier, a description and a reference to its Work Package.

Work Packages are used to define combinations of Project's tasks that are related to each other.

- WP1 → Deliverables
- WP2 → Architecture Design
- WP3 → Initial Classes
- WP4 → Implementation of Complete Game

Deliverables use identifiers with an iterable version number for easier referencing. Work related to Deliverables will be assigned in Jira as a story.

- D1.[Version] → Method Section and Planning
- D2.[Version] → Implementation
- D3.[Version] → Risk Assessment and Mitigation
- D4.[Version] → Architecture
- D5.[Version] → Requirements
- D6.[Version] → Website

Milestones are used to define specific points in the project where a defined goal has been reached (based on completion of tasks belonging to a Work Package).

- M1 → Customer Meeting and Requirements Completion — Due Date: 23/10/2020
- M2 → Architecture Abstract and Concrete Completion — Due Date: 27/10/2020
- M3 → Initial classes Completion — Due Date: 04/11/2020
- M4 → First Prototype iterated — Due Date: 11/11/2020
  - This will include a working 'leg' of the game
- M5 → Completion of Testing and Game — Due Date: 22/11/2020
- M6 → Final Deliverables Completed — Due Date: 23/11/2020

Project Evolution

By developing a concise and well laid out project plan in the early stages of the project, few changes were made over the course of the project. The first development was the move, within Jira, from one board to two separate boards representing Deliverables and Implementation. This was to avoid clutter that had started to appear between Sprint 4 to Sprint 6. To further get a stronger understanding of the critical path on the Jira board, tasks used the link field to represent where dependencies occurred between completion of tasks. During Sprint Week 4 a backlog appeared of two tasks, T14 and T11, because the tasks assigned were greater than anticipated. To compensate for this, they were assigned the highest priority within Sprint 5, as other critical tasks were dependent on the completion of T14 and T11. A similar issue appeared between Sprints 5 and 6 for T15; this was again solved by extending this task into Sprint 6 and assigning it a top priority.

References

[1] H. Svensson, *Developing support for agile and plan-driven methods*, PhD dissertation, KTH, Stockholm, 2005, p.26. [Accessed: 23/10/2020]

[2] S. Burge, "An Overview of the Soft Systems Methodology", *System Thinking: Approaches and Methodologies*, 2015, pp.1-14. [Accessed: 23/10/2020]

[3] *What is Scrum?*, Scrum.org. [Online]. Available: https://www.scrum.org/resources/what-is-scrum [Accessed: 15/10/2020].

[4] *[No title].* [Online]. Available: http://www.se.rit.edu/~se456/slides/PlanDrivenMethodologies.pdf. [Accessed: 18/10/2020]

[5] *[No title]*. [Online]. Available: https://www.wrike.com/project-management-guide/agile-methodology-basics/. [Accessed: 05 Nov. 2020].

[6] *Soft Systems Methodology*. [Online]. Available: https://www.umsl.edu/~sauterv/analysis/F2015/Soft%20Systems%20Methodology.html.htm. [Accessed: 02 Nov. 2019].

[7] *Scrum- what it is, how it works, and why it's awesome*, Atlassian.com [Online]. Available at: https://www.atlassian.com/agile/scrum [Accessed: 26/10/2020]

[8] A. Özerten, (2016, December.26), *Benefits and Challenges of Self-directed Teams in Software Projects*, Commencis. [Online]. Available: https://medium.com/commencis/benefits-and-challenges-of-self-directed-teams-in-software-projects-fe8df2d842e8. [Accessed: 28 Oct. 2020].

[9] (2018, September.11), *Soft Systems Methodology (SSM)*. [Online]. Available: https://www.toolshero.com/problem-solving/soft-systems-methodology-ssm/. [Accessed: 24 Oct. 2020].

[10] *Online Gantt Chart Software & Project Planning Tool*. [Online]. Available: https://www.teamgantt.com/ . [Accessed: 15 Oct. 2020].

[11] U. Eriksson, (2015, July.15), *Agile Software Development and Requirements*. [Online]. Available: https://reqtest.com/agile-blog/requirements-in-agile-software-development/ . [Accessed: 1 Nov. 2020].