Module	Engineering1 (Eng1) - COM00019
Assessment Title	Assessment 2, 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 2

Software engineering methods

We choose to do an agile “SCRUM” (Atlassian and Drumond) method approach [1]. We believe it is the ideal candidate for this for the following reasons:

● Since this was a first for a lot of us and there would be a lot of continuous learning, and the requirements that we have defined earlier might need to be revised, which is great about scrum where the “product backlog” is constantly revisited and maintained to ensure that the requirements are still correct. ● Gave us the opportunity to focus every sprint by allocating a certain amount of tasks to our “sprint backlog” and we would attempt to have that fundamental goal implemented by the end of the sprint.
An alternative perhaps, could have been a plan-driven approach. But due to how we work as a group this wouldn’t be nearly as efficient and would handicap our performance. Since our requirements were changing throughout as we better understood the tools we were using, the excessive documentation would be of no use since we could have changed our approach entirely and rendered time loss. If we were working as a larger group or been given a larger scope project, then it would become more appropriate to use. But since the scope isn’t too large it makes agile scrum ideal.

Development and Collaboration Tools

Having all these tools is great, but the main thing that is needed in any project or team is communication. Unfortunately, with COVID-19, there are no real ways to have a lab to work in together in person. And even if we did, it wouldn’t be enough. So we spread our communication across three areas, each being played to their strength:

1. Our instant messaging application, Discord, provided the nice in-between the two extremes, providing both text and voice functionality, and letting you also organise them into channels, which is quite powerful for keeping our communication organised. It will allow us to share screens so we can collaboratively work on current issues and allow us to switch voice channels when working within sub-teams on specific tasks. Discord allows team members to alert other users in the server when assistance is required so we can quickly respond to current issues. As team members are familiar with Discord it was chosen as our primary communication tool.
2. Our website acted as the long term memory, important information that occurred in the other platforms were eventually merged onto it, So it gave quick access to documentation, meeting minutes and links to our repositories.

To aid development, we took advantage of a couple of tools:

- Git, provided version control, the branching functionality allowed features to be developed separately from the stable codebase, then after being safely implemented could be merged back. This prevents confusion when independently working on features. The commit history is also useful as it made all committed code still accessible even after being deleted. An alternative could be Mercury, which does have a simpler user interface, git has also become ubiquitous in the industry, and provided a better real-world experience.
- To allow for the tracking of issues within the codebase and subsequent testing, Github’s Issues system will be used as it directly links to specific code and allows specific information to be assigned to that issue within GitHub Projects (Assignee, Progress, Comments). These are represented in a board that is split into sections on To Do, …. GitHub Projects was not exclusively used for all project deliverables because of Jira’s greater featureset and ability to evaluate progress on a Sprint to Sprint basis. GitHub Issues was to be used when in testing when highlighting

bugs and other critical issues

- Jira was used as the Scrum Management tool as it allowed for multiple Sprints to be run at the same time (allowing for a clear distinction between Implementation and Deliverable work) and a variety of additional information Stories and Tasks could contain (estimated time, Assignee 1 and 2). Jira also offers a Roadmap Feature that will act as an iterable version of the initial Gantt Chart to reflect how the Project evolves and allow for easy monitoring of story progress as more tasks are elicited. Trello was acknowledged as an alternative but Jira appeared simplistic in its integration of the SCRUM Methodology and members of the team had experience with it.
- Google Drive, provided collaborative document editing, this was particularly useful since we often did paired work. Also, the version history is useful for being able to retrieve previous versions of work. Google Drive offers a further set of tools (Sheets, PDFs, Slides) that allows us to work on all non-implementation work within one central location. With the support for additional tools such as LucidCharts GitHub stood out over other alternatives.These alternatives are Office 365 and OneDrive, but this wouldn't probably wouldn't work since our University accounts only provided the basic tier and we needed to be able to share with the lecturers. Furthermore, the tools supplied with Office 365 would not allow for synchronous collaboration.
- Lucidchart was used to create the architecture diagrams needed for creating the game (Abstract) and representing its core functionality (Concrete). The diagrams can be easily modified in response to a change in requirements and integrate with the chosen documentation tooling, Google Docs.
- Specific IDEs were used, VSCode and Intellij, to provide consistency when working collaboratively on issues. Team members were comfortable with these two environments so when issues potentially arise, the team should be quicker at resolving them. IntelliJ and VSCode both provide strong support for Unit Testing local code

Those tools were great at help for assisting. But for developing the game we also used some other tools to accelerate the progress:

- LibGDX and Java were used for Implementation. Java was used to reflect the Customer's Requirement. Libgdx provided an easy to use API built on top of openGL and quite cross compatible. We chose to use libgdx since there was a rich amount of tutorials available to beginners and let us get up to speed in a rather short amount of time. A possible substitute that we considered was lwjgl, which is lower level and could offer more functionality but that does come with a lot more learning than what we had time for. So ultimately decided against it due to us wanting to get familiar with what everyone else was using so our skills would be more transferable.
- JUnit is an additional open-source framework that enables us to work and write WhiteBox Testing that will enable the team to write functional code that will continue to correctly provide its existing functionality.
- GitHub Pages, which allows us to host our own website for a more public view of our project, without the lengthy and hard process of making our own website.

Team organisation

Since we are a small group of students where all our work is being done remotely, and with one of our members being in a largely different timezone it required us to be quite flexible with how we were organised as a group. Consequently we used an agile method, specifically a scrum as it suited our small team size, need for flexibility and use of customer meetings.

To be effective, we set out to accomplish a few preliminary goals before getting into the scrum cycle. Firstly, we had ice-breaking activities so we could find out what people were best at doing and also seeing how we interacted with each other as a team. This proved useful as it allowed us to delegate our roles out easily and efficiently, although we didn't always strictly follow the roles that we set out for ourselves, and people gravitated towards what initially wasn't expected, but ended up performing well at the task. Specific sub-teams of Testing and Implementation were stated to enable us to better manage and complete the workload but a Deliverable Sub-Team was not specified because this would involve all team members participating. To ensure that Sub-teams would not cause conflicts in current work, sprint sessions were extended to review each other's work and identify possible conflicts.The key roles of Scrum were allocated: Product Owner (ensures the most value is delivered to the business) and Scrum Master (ensures team members are adhering to the Scrum method).

- Scrum Master was assigned to Richard U due to prior experience with Jira and demonstrated an aptitude for organizing team meetings and organisation.
- Product Owner was Joe L due to his experience in coding games in Java.

So at the base we had our set roles, and we used that to our advantage by assigning tasks to people who had the most appropriate role, allowing for further flexibility in development. Furthermore, we also organised set dates occuring every week, allowing us to define sprints, and partake in a scrum. We also held sub meetings where people working on similar tasks would meet and work together to bounce ideas off each other and help develop the project further.

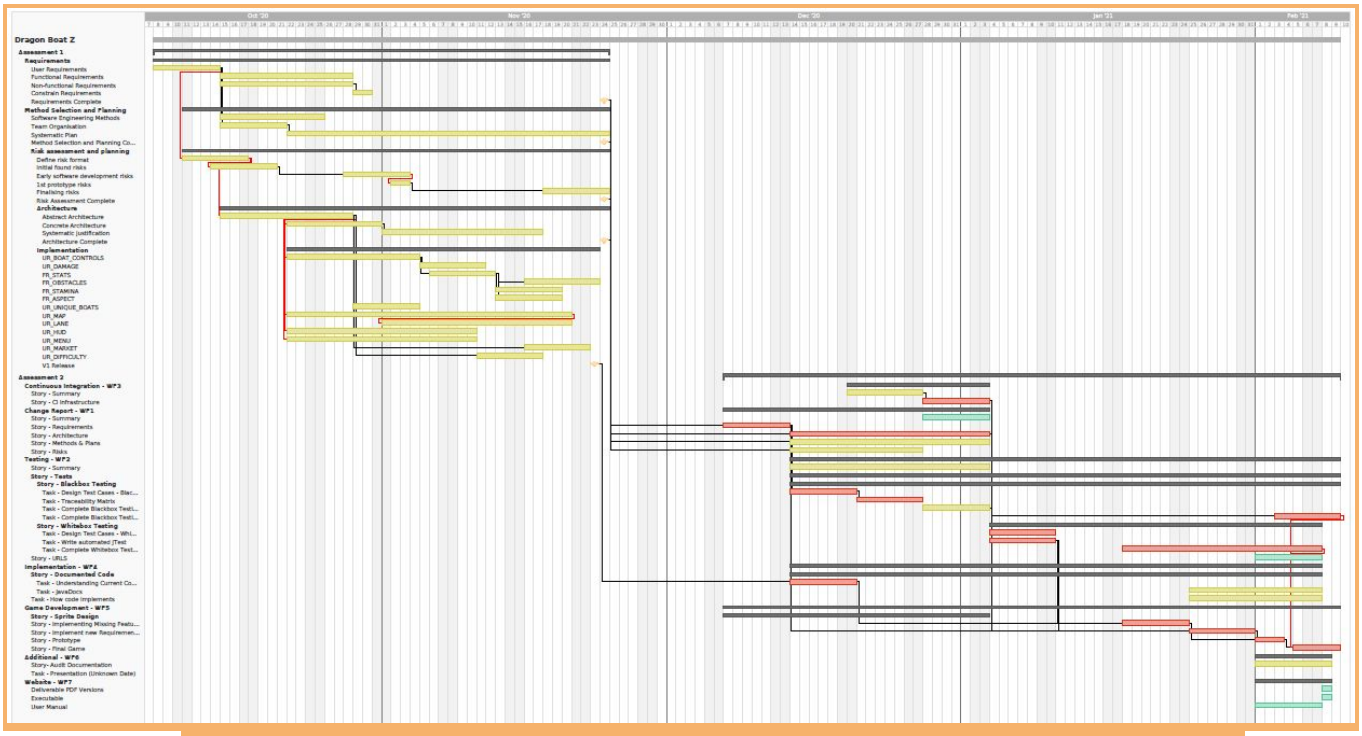
These meetings provided some core functionality to how we organise ourselves and allowed us to be the most productive.

- It allowed us as a team to make large, educated, decisions about the approaches / tasks we set out to do due to the scrums, and gave everyone a chance to raise concerns regarding risks and resolve any conflicts before we went ahead and implemented that item.
- We also used them to catch-up and check progress with what people had been working on since the last meeting, enabling us to document progress in the minutes and let the group reflect on what was produced and see if any improvements could be made. This provided a good incentive to create high-quality work, acting like a positive feedback loop, whilst also fitting in well with the need for frequent releases in agile methods
- The most important factor from these meetings was giving us a solid platform to delegate tasks out, which aided us heavily in completing the majority of the project. After discussing a task, and breaking it down, we each would choose a sub-task. In some cases two people would collaborate on the same task in a pair programming fashion, which was useful for larger and complex tasks and allowed us to complete them within a set timeframe we assigned it. Jira was used to manage these tasks allocated in the form of sprints and help to finish all parts of the assessment and review sprints. Tasks and their priority were based on the initial Gantt chart constructed at the start of the Assessment, which outlined dependencies and tasks that lay on the critical path.

Overall, these meetings helped us focus on what needed attention, where we would define clear goals, keep within scope, and maintain a high level of efficiency throughout the project life.

Project Plan

The plan for the project was to build the game using an agile method approach, allowing us to modify existing elements as we progress through the project. So with our agile approach, we decided to put ourselves on one week sprints, that would contain two meetings during the sprint; one meeting was dedicated to a Sprint Review and Planning session and an additional meeting to discuss task progression.



For readable version visit: [Sprints - DragonBoat Z (dragon-boat-z.github.io)]

The Gantt Chart was constructed to reflect Assessment 2 of the Project and initial thoughts regarding the different work packages, stories, and tasks. Each Story and Task were assigned a priority level based off:

- The time taken to complete
- The number of marks worth and number of people required, 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

The Gantt Chart was constructed using the online tool [2] and detailed any clear dependencies between tasks. Task priority was indicated by the colour associated with it. Green represents low priority, Yellow indicates medium priority and Red indicates high priority. This was used to form the critical path, based on the set of tasks that must be completed according to the SSON, and implement a contingency plan of several days on the off-chance a backlog of tasks built up.

Work Packages were used to reflect a combination of stories that were related to each other and were needed to achieve the milestone of achieving the Work Package goals.

- Change Report
- Continuous Integration
- Testing
- Implementation
- Game Development
- Additional
- Website

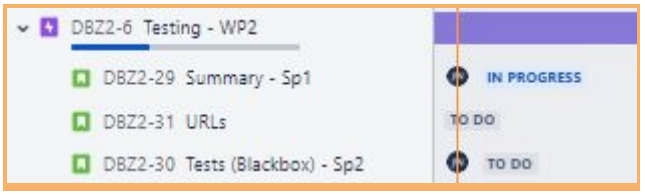
As this plan was constructed at the start of Assessment 2, a high level approach was taken through

the use of stories. Stories reflected the necessary parts of a Work Package that must be delivered. These were indicated through ‘Story -’. If we had initial thoughts on the tasks required to complete that story, subtasks were elicited within the Story groups these were indicated as a set of subtasks.

The Gantt chart will be used to indicate Stories that need to be completed during Sprint Planning sessions. When transferring this information to Jira, the team elicited the sub-tasks required to complete that story and these were assigned to individuals.

Within Jira, each story and task were assigned an expected time to complete, multiple team-members per task, a priority level based on the information present in the Gantt Chart and assigned to the Work Package they relate to in the Gantt Chart. In Jira, this is referred to as Epics.

To indicate Product Evolution, Jira’s beta functionality Roadmaps was used to help monitor progress across all Epics in the form of an iterable Gantt Chart that directly integrated into Jira Sprints. This allowed us to visualise the current progression of Stories over Sprints and the completion of Work Packages (represented as Epics).



Example section of Roadmaps. Final version: [Sprints - DragonBoat Z (dragon-boat-z.github.io)]

After defining the initial Gantt Chart, alterations occurred because the time given to preparing for exams was not enough. This meant that certain tasks were moved back past the exam period including

- Completing the first iteration of Black Box Testing
- Designing and Writing White Box Testing
- Implementation of new features (because this was dependent on the required writing of White Box Testing)

These were assigned a high priority to reflect their importance as other tasks were dependent on their completion and the results they would output. To maintain the organisation required we extended the number of weekly meetings from two to three meetings.

However, no further changes were made to the Roadmap apart from the time constraints on tasks were reduced to accommodate the delays caused by not completing the above tasks.

The progression of weekly sprints can be found here: [Sprints - DragonBoat Z (dragon-boat-z.github.io)]

Bibliography

1 Atlassian, and Claire Drumond. “What is Scrum?” *Scrum - what it is, how it works, and why it's awesome*, https://www.atlassian.com/agile/scrum. Accessed 24 11 2020.

[2] S. Burge, “An Overview of the Soft Systems Methodology”, System Thinking: Approaches and Methodologies, 2015, pp.1-14. [Accessed: 7/12/2020]