Method Selection and Planning

Original Assessment 1 Team

Team 15

Joe Wrieden

Benji Garment

Marcin Mleczko

Kingsley Edore

Abir Rizwanullah

Sal Ahmed

Assessment 2 Team

Team 12

Richard Liiv

Umer Fakher

William Walton

James Frost

Olly Wortley

Joe Cambridge

Method Selection and Planning

Part A: Software Engineering Methods

Scrum is our team's software development method of choice, as this agile methodology is suitable for small teams, and shall allow us to deal with changing requirements in the future, especially in relatively short time constraints. Along with this, it is in our best interests to reconvene in a weekly Scrum meeting, in which we reflect on our progress throughout the week, checking that every requirement in our previous stage maps onto at least one requirement in our current stage, and address issues which may have been initially overlooked before greater complications can arise, as well as have frequent team walkthroughs of our artefact throughout its incremental development stages - from the initial requirements to the solution we will pose.

- Other agile frameworks such as XP were considered, but Scrum was chosen due to it being up to the team to prioritise work according to that which they know is needed, as well as having less of a focus on strictly following particular engineering methods (e.g paired programming) that XP advocates.

● Development is carried out in weekly sprints, culminating in the artefact's current iteration for the week being provided as a deliverable to the client so they may be included in the development process should they wish to have access and provide feedback at any point.

- Focusing on the principles of Scrum rather than the practices (e.g. daily Scrum standups, which we are unable to do due to workloads from other modules; user stories are not a viable option as we knew the system requirements from the onset), allows us to be flexible with how much each person can contribute at a time depending on their circumstances, but also ensuring equal overall contribution through the project schedule, which means our bus factor remains high (more on this later in "Team Organisation").

Part A: Development and Collaboration Tools

The Java framework we decided on is libGDX.

- We had considered using LWJGL, a Java library that provides access to native APIs for graphics and audio. However, libGDX uses this library in its framework, and since code reuse is an essential part of Software Engineering, we decided we would use the higher level methods that libGDX provides.

- This also allows us to focus our time during each Scrum sprint on solving issues that relate to the task given to us by the client, rather than spending the time to learn to use LWJGL as it provides low-level access.

- On that note, libGDX is quite seamless in that the majority of the program consists of regular Java, which our team is familiar with. The high-level libGDX methods may simply serve to render graphics to the screen, for example.

Our IDE of choice is IntelliJ.

- IntelliJ IDEA is the recommended IDE by the libGDX team, and so it has the most support for the framework.

- We had considered Visual Studio Code as an alternative, however, we had difficulties with VS Code during our findings, specifically with using the installer from libGDX, which creates projects using gradle.

- IntelliJ on the other hand does not have this issue, and thus in learning how to use libGDX, IntelliJ is often referenced, making it the ideal choice.
- Furthermore, IntelliJ lends itself to a project-style of development. It has features that aren't present in VS Code that make it easier to work on large scale projects with multiple classes such as automatic class refactoring.

For version control, we are using Git with our repository stored on GitHub.

- This allows us to avoid collisions and other inconsistencies when merging different members' works together.

- It allows members the ability to access the code after each sprint to understand how it works in order to reduce the effects of the bus factor, especially since members are working remotely.

- Furthermore, hosting the repository on GitHub allows us to easily maintain version control history and even gives us access to GitHub Pages which allows for easy access to documentation.

- Finally, Git has native integration in IntelliJ thus making it easier to pull and push changes.

For online meetings, we use Zoom.

- A weekly scheduled Zoom Scrum meeting allows us to communicate via voice and text chat as well as sharing our screens for input from other members for any problems encountered.

- Zoom also gives us a platform to engage in Team-Customer meetings should we have any questions or queries for the client or vice versa.

For further collaboration, we use Discord.

- Separate text channels allow for easy communication on the relevant issues in each channel, along with support for uploading various file types.
- Voice channels allow us to communicate with each other in a less structured way when we need to collaborate before the Scrum meeting.

For task organisation and keeping track of tasks, we used Trello.

- Boards and Calendar showing tasks and team members allocated to each one, with detail in text and images, organised into different sections.

- Checklists on individual tasks as well as links to resources such as the Github repository.

For UML Class Diagrams, we use PlantUML.

- That lets us create a very high level, abstract representation of the game's Software Architecture providing a graphical model that allows people regardless of their technical background to grasp the gist of complex concepts that code aims to carry out.

Finally, for the storage and collaboration of documents, we use Google Drive, making it easy for several members to update the same document and edit in real time through the version control functionality.

Part B: Team Organisation

Our tasks, as described below, have initially been distributed in a way that allows two of our members, Umer and Will, to focus on learning the technical aspects of the development and programming, while the other four members take on refining the artefact's successive design versions and its documentation. This meant that while the programming team familiarise themselves with libGDX, the other members are to define the software architecture from the requirements which had been elicited, carry out planning, as well as identify risks and their respective mitigations.

To respect the Scrum development we had a Scrum master (Umer) which handled team dynamics, arranging meetings, maintaining task plans and adapting to changing productivity levels of the team over time. We shared feedback to make sure the bus factor on certain tasks was not too low and made sure to communicate feedback to each other when changing priorities and owners of tasks. The Scrum master would coordinate the sub-teams and evaluate project progress.

The reason we feel it is appropriate to split the team into subteams is so that people can work with their strengths and productivity can be boosted. The documentation is worth three times more than the code, which is why we have decided to put more people on writing documentation. We decided against putting more than 4 people on documentation as this would greatly decrease the bus factor, since there would be only one focusing on code.

Following their findings, the programming team is to inform the rest of the group on how to lay out the concrete architecture, such as detailing technology-specific methods. They will then continue to implement the product while the rest of the team write-up the other tasks as laid out below.

For the first half of the project, the documentation team decided to spread the work equally amongst themselves, with each member focusing on one or two sections. There are a few reasons for this: first of all, we manage to get four times the amount of ideas on each section, secondly, the layout/style of each document is going to be the same. Thirdly, having four people working on a section each week increases the bus factor. And lastly, people can combine their written-communication and technical abilities.

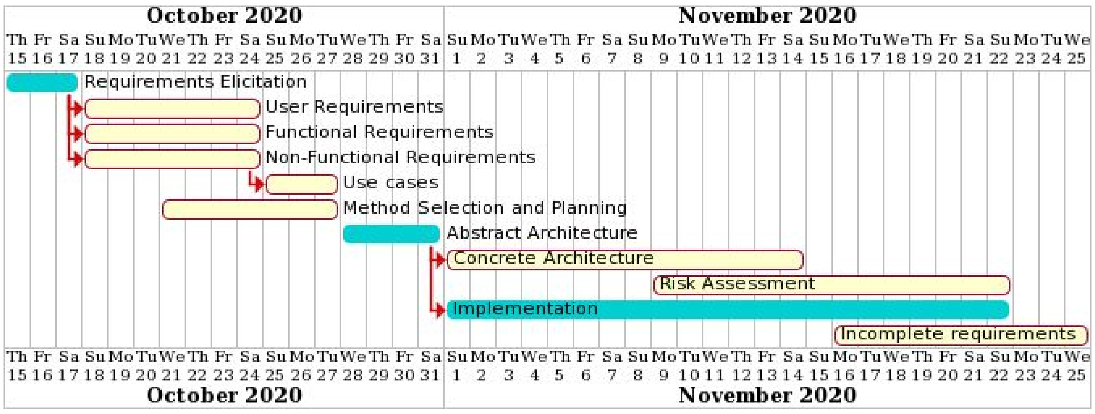
For some tasks, however, we decided to have a few members of the team work together on a single part of a document, especially during the closing stages of the project. During the end of the project, we had less and less to do with some of the team members having nothing to do so having all (or most) members work together would boost team productivity.

Part C: Project Plan

The plan above describes each individual task that must be completed and in what order. **High priority** tasks are coloured in **dark turquoise**.

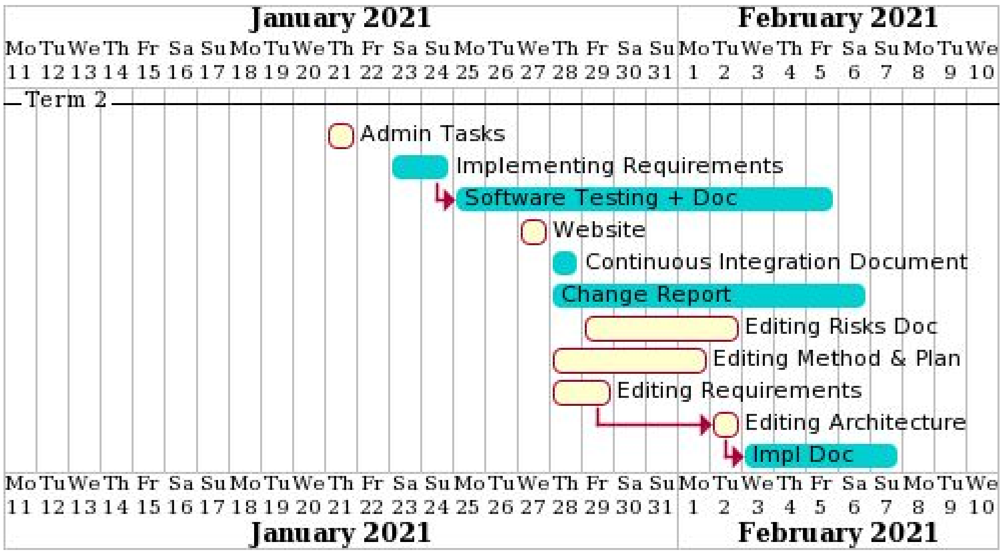
Tasks that are **dependent** on others are denoted by a leading red arrow.

Assessment 1 Gantt Chart



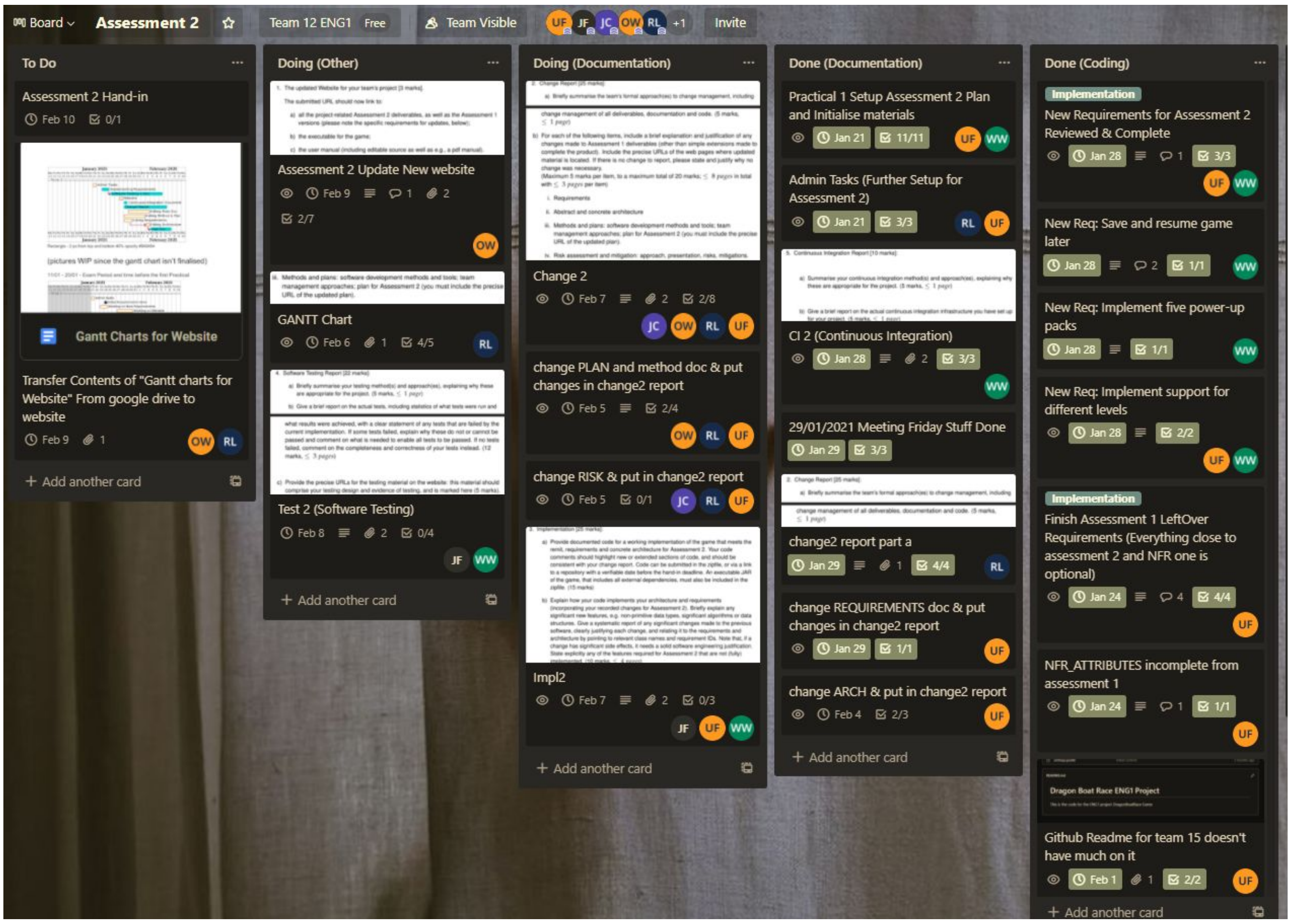
The critical path for Assessment 1 consists of [each of the User, Functional and Non-Functional] Requirements → Abstract Architecture → Method Selection → Implementation. The plan above assumes that each task takes the longest possible amount of time to allow room for any risks that may arise. Changes to how long each task takes and when they begin/end is discussed on the website.

Assessment 2 Gantt Chart



The critical path for Assessment 2 consists of Implementing Missing Requirements → Software Testing → Continuous Integration → [Editing Original Team 15 Documents →] Change Report → Implementation 2 Document. The plan above follows the real time taken to do each of the tasks.

Trello Board Screenshots


Board · Assessment 2 · Team 12 ENG1 Free · Team Visible · UF JF JC OW RL +1 · Invite

To Do
- Assessment 2 Hand-in — Feb 10 · 0/1
- Gantt Charts for Website — (pictures WIP since the gantt chart isn't finalised)
- Transfer Contents of "Gantt charts for Website" From google drive to website — Feb 9 · 1 · OW RL
- + Add another card

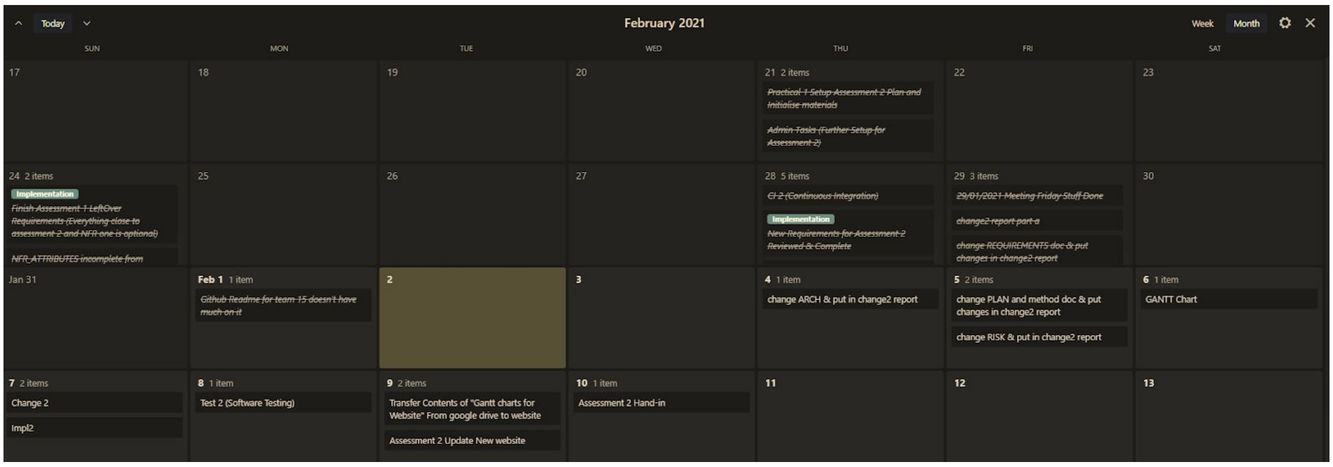
Doing (Other)
- Assessment 2 Update New website — Feb 9 · 1 · 2 · 2/7 · OW
- GANTT Chart — Feb 6 · 1 · 4/5 · RL
- Test 2 (Software Testing) — Feb 8 · 2 · 0/4 · JF WW
- + Add another card

Doing (Documentation)
- Change 2 — Feb 7 · 2 · 2/8 · JC OW RL UF
- change PLAN and method doc & put changes in change2 report — Feb 5 · 2/4 · OW RL UF
- change RISK & put in change2 report — Feb 5 · 0/1 · JC RL UF
- Impl2 — Feb 7 · 2 · 0/3 · JF UF WW
- + Add another card

Done (Documentation)
- Practical 1 Setup Assessment 2 Plan and Initialise materials — Jan 21 · 11/11 · UF WW
- Admin Tasks (Further Setup for Assessment 2) — Jan 21 · 3/3 · RL UF
- CI 2 (Continuous Integration) — Jan 28 · 2 · 3/3 · WW
- 29/01/2021 Meeting Friday Stuff Done — Jan 29 · 3/3
- change2 report part a — Jan 29 · 1 · 4/4 · RL
- change REQUIREMENTS doc & put changes in change2 report — Jan 29 · 1/1 · UF
- change ARCH & put in change2 report — Feb 4 · 2/3 · UF
- + Add another card

Done (Coding)
- Implementation — New Requirements for Assessment 2 Reviewed & Complete — Jan 28 · 1 · 3/3 · UF WW
- New Req: Save and resume game later — Jan 28 · 2 · 1/1 · WW
- New Req: Implement five power-up packs — Jan 28 · 1/1 · WW
- New Req: Implement support for different levels — Jan 28 · 2/2 · UF WW
- Implementation — Finish Assessment 1 LeftOver Requirements (Everything close to assessment 2 and NFR one is optional) — Jan 24 · 4 · 4/4 · UF
- NFR_ATTRIBUTES incomplete from assessment 1 — Jan 24 · 1 · 1/1 · UF
- Dragon Boat Race ENG1 Project — Github Readme for team 15 doesn't have much on it — Feb 1 · 1 · 2/2 · UF
- + Add another card

Screenshot of the Trello board.
Used for dynamic task management and deadline adjustments. Major and minor tasks have embedded checklists.


Today · February 2021 · Week · Month

SUN	MON	TUE	WED	THU	FRI	SAT
17	18	19	20	21 2 items ~~Practical 1 Setup Assessment 2 Plan and Initialise materials~~ ~~Admin Tasks (Further Setup for Assessment 2)~~	22	23
24 2 items Implementation ~~Finish Assessment 1 LeftOver Requirements (Everything close to assessment 2 and NFR one is optional)~~ ~~NFR_ATTRIBUTES incomplete from~~	25	26	27	28 5 items ~~CI 2 (Continuous Integration)~~ Implementation ~~New Requirements for Assessment 2 Reviewed & Complete~~	29 3 items ~~29/01/2021 Meeting Friday Stuff Done~~ ~~change2 report part a~~ ~~change REQUIREMENTS doc & put changes in change2 report~~	30
Jan 31	Feb 1 1 item ~~Github Readme for team 15 doesn't have much on it~~	2	3	4 1 item change ARCH & put in change2 report	5 2 items change PLAN and method doc & put changes in change2 report change RISK & put in change2 report	6 1 item GANTT Chart
7 2 items Change 2 Impl2	8 1 item Test 2 (Software Testing)	9 2 items Transfer Contents of "Gantt charts for Website" From google drive to website Assessment 2 Update New website	10 1 item Assessment 2 Hand-in	11	12	13

Screenshot of the Trello board calendar.