UNIVERSITY OF YORK

DEPARTMENT OF COMPUTER SCIENCE

Method Selection and Planning

Group 20

Formerly Group 16

Group Members:

Group 16

Charlotte MacDonald
Hollie Shackley
Luis Benito
Kaustav Das
Sam Hartley
Owen Gilmore

Group 20

Leuay Dahhane
Max Irvine
Sam Butler
Flynn Gadsden
Jacob Wharton
Billy Moore

Outline and Justification of our Software Engineering Methods

Due to the short time frame to complete the project, it was clear that multiple stages of the software engineering process would have to be completed at the same time. This fitted an agile approach. It was also reasonable to suspect that plans would change significantly each week as it would be hard to plan how long each task would take when all team members had other commitments and relatively little experience of creating a game or using game engines. Agile fitted this and also the concept of having flexible interactions with the customer - for example, after the initial customer meeting, there were informal conversations that happened in each practical. Agile approaches prefer shorter timeframes, encourage face-to-face conversations and encourage regular reflection on efficacy [1]. These fitted well to our organisation of having a face-to-face meeting with all team members each week. As these meetings were 2 hours long, they provided ample time to reflect on the previous week, plan for the following week and have detailed discussions about the deliverables. The assessment format of releasing a partial solution first also fitted well to the agile manifesto [1].

The main inspiration for our methods and organisation came from scrum. As we were already committed to weekly meetings, it was natural to create weekly sprints. The start of each meeting was then dedicated to a review and retrospective of the sprint using ideas from scrum to reflect on what did and didn’t happen and which tasks went well, were challenging or were problematic. This allowed us to reflect on anything that needed to change for the next sprint before planning the sprint and adapting overall project plans as needed. Work that was categorised as not done was able to be pushed back to a more suitable time. The retrospective also allowed for reflection on unforeseen dependencies that had limited progress which allowed for the re-ordering of tasks and prioritisation of what was left.

We also drew inspiration from the spiral lifecycle as this produces good documentation control [2] and a large part of the project is based around documentation. This documentation is also reviewed in every loop which was very similar to how we created new plans and reviewed the risk assessment each week. However, we only used certain parts of this lifecycle as adapting it well would not have fitted our size of project and scheduling was extremely important to the project [2].

Identification and Justification of Development and Collaboration Tools Used

The customer briefing placed the constraint of using only Java for the implementation and using a gaming engine written in Java. The requirements for the game included that it would be 2D. Research of various 2D Java game engines was undertaken and LibGDX was chosen as this has specifically been built for 2D games and interlinks well with Github. Other alternatives, including J Monkey and LWJGL, were considered. Many team members liked the look of J Monkey however it seemed much more suited to 3D game development and the assessment brief specified that the game must be 2D. By looking at reviews of LWJGL, it seemed that it was difficult to learn at first so it was felt that this would be an unnecessary delay and inefficient to choose [3]. IntelliJ was selected as the IDE as LibGDX relies heavily on Gradle to assemble projects. Originally, the plan was to use VSCode as all team members had used it before including for Java and it was already on all department machines but this does not interact well with Gradle and would have created unnecessary difficulty during implementation. After finding that IntelliJ was also available on the department machines and would work much better, it was agreed upon.

Git in conjunction with Github was chosen for the code base and website as several team members had prior experience and it is the standard tools. Other alternatives were briefly considered such as Apache Subversion but no team members had prior experience with these so it was felt that they would add unnecessary delays and extra work of becoming experienced with using them first. In addition, git encourages small commits as well as branching and merging which worked well with our agile approach. Github was also selected to host the website as all team members either had prior experience or would be gaining experience through its use in the implementation. Google Drive was used for collaboration for the other deliverables due to the live collaboration features and all team members having access and prior experience. Using Google Docs for the deliverables also had the advantage of version control so any changes could always be reverted if necessary. Being able to add comments easily to Google Docs also allowed collaboration on documents without having to switch between multiple platforms. Google Slides was also used for scrum reviews and retrospectives as it provided easily movable shapes and text boxes to categorise if targets for the week had been met or not. It also allowed all team members to add in their thoughts and for this to easily be presentable in meetings. To track the progress and ownership of tasks we used a Jira KAN board. This allowed us to list all tasks, track ownership of them and their progress, one of “to do”, “in progress”, “review” or “done”. This made it really easy to see what was left to do, what everyone was working on and what tasks no one had claimed.

For collaboration such as suggesting ideas outside of meetings, Discord was agreed as the platform to use. Slack was considered but no team members had experience of using it whereas all had experience of Discord. Whatsapp was also considered but Discord was felt to be more suitable as it was better suited for sharing larger amounts of text and channels would allow different deliverables to be discussed in their own areas to help organisation. Using Discord fit well with our scrum-inspired methods as it allowed for very frequent communication between team members. For architectural diagrams, the decision was made to use PlantUML. This was because this works well with Google Docs and so it would be easy to add diagrams to documents but also to change them during the evolution of the document and the project. PlantUML was also used for other diagrams including those in the planning process. This reduced the bus factor as it meant that more people were aware of and experienced with the language being used for the architectural diagrams. Alternatives considered included Mermaid and Graphviz but these did not have the benefits of PlantUML.

As well as the constraint of only using Java for implementation, there was also the constraint of only using tools available on department machines so that if team members weren’t able to use a personal device or access the tools personally, they would still be able to access the full project and contribute well. Available expertise was taken into account for each decision.

Team Organisation

Deliverable	**‘Owner’**	**Other involved members**
Website	Billy	-
Change Report	Billy	Leuay
Implementation	Flynn	Billy, Max, Jacob, Sam
Testing	Leuay	Jacob, Billy, Flynn
Continuous Integration	Max	Max, Sam
User Evaluation	Jacob	Sam
Presentation	Sam	Max, Jacob

We decided to have a team member in charge of each deliverable, we considered the presentation to count as an additional deliverable. We looked at areas of success from the first deliverable and the preferences of team members in assigning the responsibilities. This ensured all team members were working on work that both interested them and they were able to complete to a high standard.

To reduce the bus factor each team member explained their work to the group each week. For example if a change was made to the code the responsible team member would show the changes to the game along with the changes to the code in enough detail for other team members to understand the changes. In addition where possible we had multiple people work on each deliverable, as you can see in the table above.

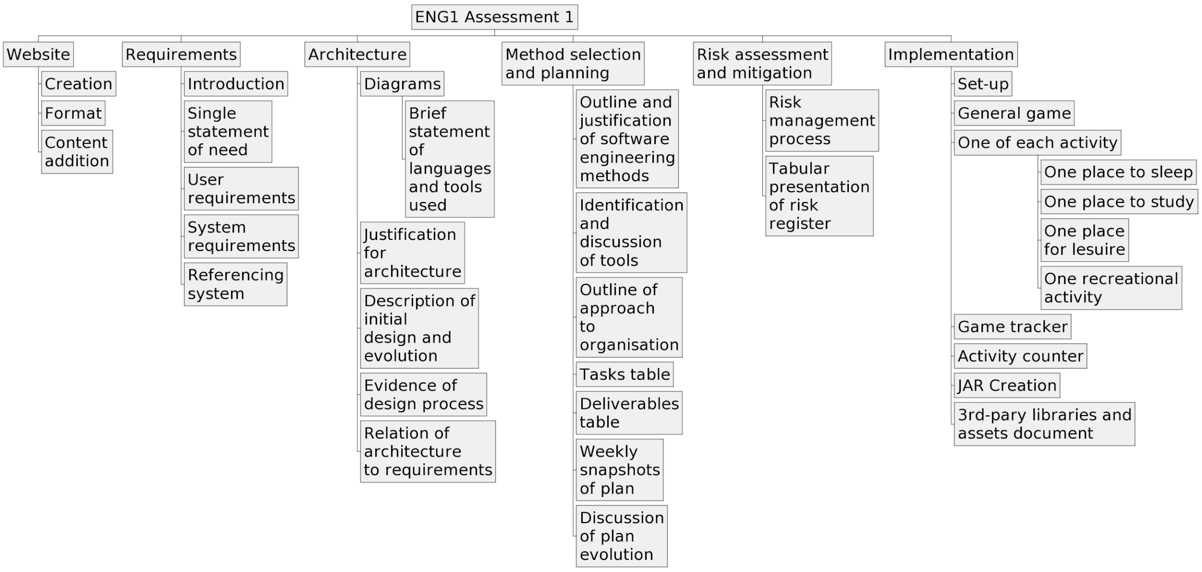
This approach was suitable for the project as it was made clear that equitable work allocation was expected and so no team member should be given more responsibility than another. This approach avoided shared leadership of any area of the project. Deliverable leaders were responsible for splitting the work in that deliverable between the team members assigned to work on it.

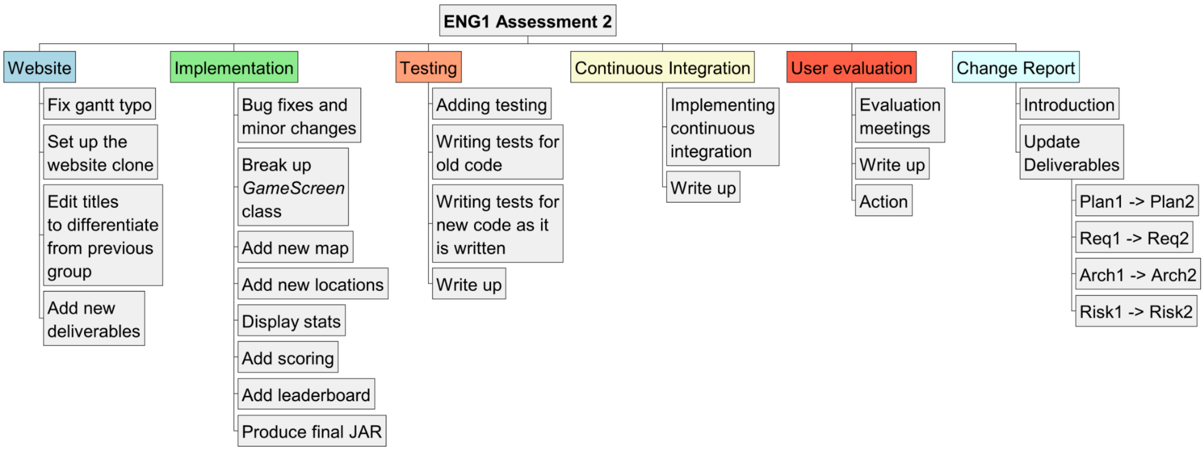
There was also the role of upper management provided by the customer. This was available if it was needed to solve team disputes or any other issues but wasn’t needed. The customer was also the main stakeholder and the only stakeholder who decided requirements. Communication with the stakeholder was first through a formal client meeting to gather more information about requirements. It was then continued through weekly discussions during practical sessions where smaller questions were clarified and updates on progress were given.

Decisions were mostly made through unanimous decision as there was very little disagreement. However, where there was any disagreement, the decision was first attempted to be made through the majority opinion. If opinion was equally split, the decision was made by the leader of the deliverable it related to.

Work Breakdown

The work breakdown structure was created using the assessment document to split into deliverables which were then further broken down. The product brief was used to break down the implementation deliverable.


ENG1 Assessment 1
Website
Creation
Format
Content addition
Requirements
Introduction
Single statement of need
User requirements
System requirements
Referencing system
Architecture
Diagrams
Brief statement of languages and tools used
Justification for architecture
Description of initial design and evolution
Evidence of design process
Relation of architecture to requirements
Method selection and planning
Outline and justification of software engineering methods
Identification and discussion of tools
Outline of approach to organisation
Tasks table
Deliverables table
Weekly snapshots of plan
Discussion of plan evolution
Risk assessment and mitigation
Risk management process
Tabular presentation of risk register
Implementation
Set-up
General game
One of each activity
One place to sleep
One place to study
One place for lesuire
One recreational activity
Game tracker
Activity counter
JAR Creation
3rd-pary libraries and assets document



Deliverables Table

ID	Title	Due date	Description	Visibility	Relevant tasks
D1	url1.txt	21/3	Website	Shared	T1
D2.1	Req1.pdf	21/3	Requirements	Shared	T2
D2.2	Questions for client	29/2	Preparation of questions for client interview	Internal	T2.2
D3	Arch1.pdf	21/3	Architecture	Shared	T3
D4	Plan1.pdf	21/3	Methods and planning	Shared	T4
D5	Risk1.pdf	21/3	Risk assessment and mitigation	Shared	T5
D6.1	Impl1.pdf	21/3	Implementation	Shared	T6.6

D6.2	Code	21/3	Implementation	Shared	T6.1-T6.5
D6.3	Executable JAR	21/3	Implementation	Shared	T6.1-T6.5
D7	url2.txt	24/5	Website	Shared	T7
D8	Arch2.pdf	24/5	Updated architecture	Shared	T8
D9	Plan2.pdf	24/5	Updated methods and planning	Shared	T9
D10	Risk2.pdf	24/5	Updated risk assessment and mitigation	Shared	T10
D11	Req2.pdf	24/5	Updated requirements	Shared	T11
D12	Change2.pdf	24/5	Change Report	Shared	T12
D13	CI2.pdf	24/5	Continuous Integration report	Shared	T13
D14	Test2.pdf	24/5	Testing report	Shared	T14
D15.1	Impl2.pdf	24/5	Implementation	Shared	T15.8-T15.9
D15.2	Code	24/5	Implementation	Shared	T15.1-T15.7
D15.3	Executable JAR	24/5	Implementation	Shared	T15.1-T15.7

Tasks Table

Task ID	Description	Start date	End date	Dependencies	Priority
Assessment 1 Tasks					
T1.1	Create and format website	21/2	27/2		High
T1.2	Add all content and links needed to website	12/2	12/3	T1.2	High
T2.1	Create requirements referencing system	28/2	5/3		High
T2.2	Prepare for and have client meeting	21/2	29/2		High
T2.3	Give statement of user requirements	28/2	5/3	T2.1, T2.2	High
T2.4	Give statement of system requirements	28/2	5/3	T2.1, T2.2	High
T2.5	Introduction to requirements	6/3	12/3	T2.2	High
T2.6	Single statement of need	6/3	12/3	T2.2	Medium
T3.1	Diagrammatic representations of product's architecture	21/2	12/3		High
T3.2	Statement of languages and tools	21/2	12/3		High
T3.3	Justification for architecture	21/2	12/3		High
T3.4	Initial design and evolution	21/2	12/3		High
T3.5	Evidence of design process followed	21/2	12/3	T3.1	High
T3.6	Relation of architecture to requirements	28/2	12/3	T2.2, T2.3	High
T4.1	Outline and justification of methods and tools including alternatives considered	6/3/24	12/3		High
T4.2	Outline and explanation of team organisation	6/3/24	12/3		High
T4.4	Work breakdown diagram with explanation	21/2	27/2		High
T4.5	Deliverables table	21/2	27/2		High

T4.6	Tasks table	21/2	27/2	T4.5	High
T4.7	Discussion of plan evolution and Gantt charts	21/2	12/3	Meetings, previous charts	Medium
T5.1	Risk register	21/2	27/2		High
T5.2	Create mitigation and contingency strategies	21/2	27/2	T5.1	High
T5.3	Describe and justify risk management process and format of risk register	21/2	27/2	T5.1, T5.2	High
T5.4	Continued risk reassessment	21/2	12/3	T5.1	Medium
T6.1	Set-up implementation	21/2	27/2		High
T6.2	Create one of each activity location	28/2	12/3	T6.1	High
T6.3	Create game tracker	28/2	12/3	T6.1	High
T6.4	Create counter	28/2	12/3	T6.1	High
T6.5	Document code and create JAR	28/2	12/3	T6.2, T6.3, T6.4	High
T6.6	List 3rd-party libraries and assets with licences and discussion of licence suitabilities	28/2	12/3	T6.2, T6.3, T6.4	High
Assessment 2 Tasks (It is assumed that all assessment 1 tasks are complete below this point therefore they will not be included in dependencies)					
T7.1	Update the website to reflect change of ownership of the project	8/5	14/5		Low
T7.2	Add new deliverables	15/5	21/5	T8,T9,T10,T11,T12	High
T7.3	Add new gantt charts to the website	15/5	21/5	T9.4	High
T7.4	Add updated architecture diagrams to the website	15/5	21/5		High
T8.1	Update diagrams to show architecture changes	18/5	20/5	T15.2	High
T8.2	Update architecture write up	18/5	20/5	T8.1	High
T9.2	Update team organisation for new team	5/5	12/5		High
T9.3	Create work breakdown for assessment 2	5/5	12/5		High
T9.4	Create new gantt charts and discuss	5/5	22/5	Meetings, previous charts	High
T9.5	Improve existing planning and methods text where necessary	2/5	12/5		Low
T10.1	Add additional risks unique to assessment 2	8/5	16/5		High
T10.2	Rewrite sections of risk plan specific to group 16	8/5	16/5		High
T10.2	Improve existing text of risk write up where necessary	8/5	16/5		Low
T10.3	Add any additional unforeseen risks encountered along the way	8/5	22/5		Medium
T11.1	Add the additional requirements for assessment 2	24/4	30/4		High
T11.2	Change existing requirements as appropriate	24/4	30/4		Low
T11.3	Update requirements introduction where required	24/4	30/4		High
T12.1	Change report introduction	24/4	30/4		High

T12.2	Change report for requirements	1/5	4/5	T11	High
T12.3	Change report for architecture	20/5	22/5	T8	High
T12.4	Change report for method selection and planning	5/5	12/5	T9	High
T12.5	Change report for risk assessment and mitigation	17/5	20/5	T10	High
T13.1	Add continuous integration to the project	16/5	18/5		High
T13.2	Write up the continuous integration report	19/5	22/5	T13.1	High
T14.1	Set up testing	24/4	27/5		High
T14.2	Add tests for existing code	18/5	20/5	T14.1	High
T14.3	Add tests for new code	16/5	20/5	T14.1	High
T14.4	Write up testing report	16/5	20/5	T14.1-T14.3	High
T15.1	Big fixes and minor changes	24/5	27/5		Medium
T15.2	Break up GameScreen class	15/5	18/5		High
T15.3	Add new map	1/5	7/5		Medium
T15.4	Add new locations and activities	8/5	18/5	T15.3	High
T15.5	Display stats to the player as they play	8/5	9/5		Low
T15.6	Calculate a score for the player, ideally in the form of a university result	14/5	20/5		High
T15.7	Add a leaderboard to the game	14/5	20/5	T15.6	High
T15.8	Continue documentation code through javadoc comments	24/4	22/5		High
T15.8	Update Impl1 to include consideration of any new assets used	15/5	22/5	T15.4	High
T15.9	Update Impl1 to include discussion of any requirements not met	20/5	22/5	T15.1-T15.7	High
T15.10	Produce an executable JAR	20/5	22/5	T15.1-T15.7	High

Discussion of Plan Evolution

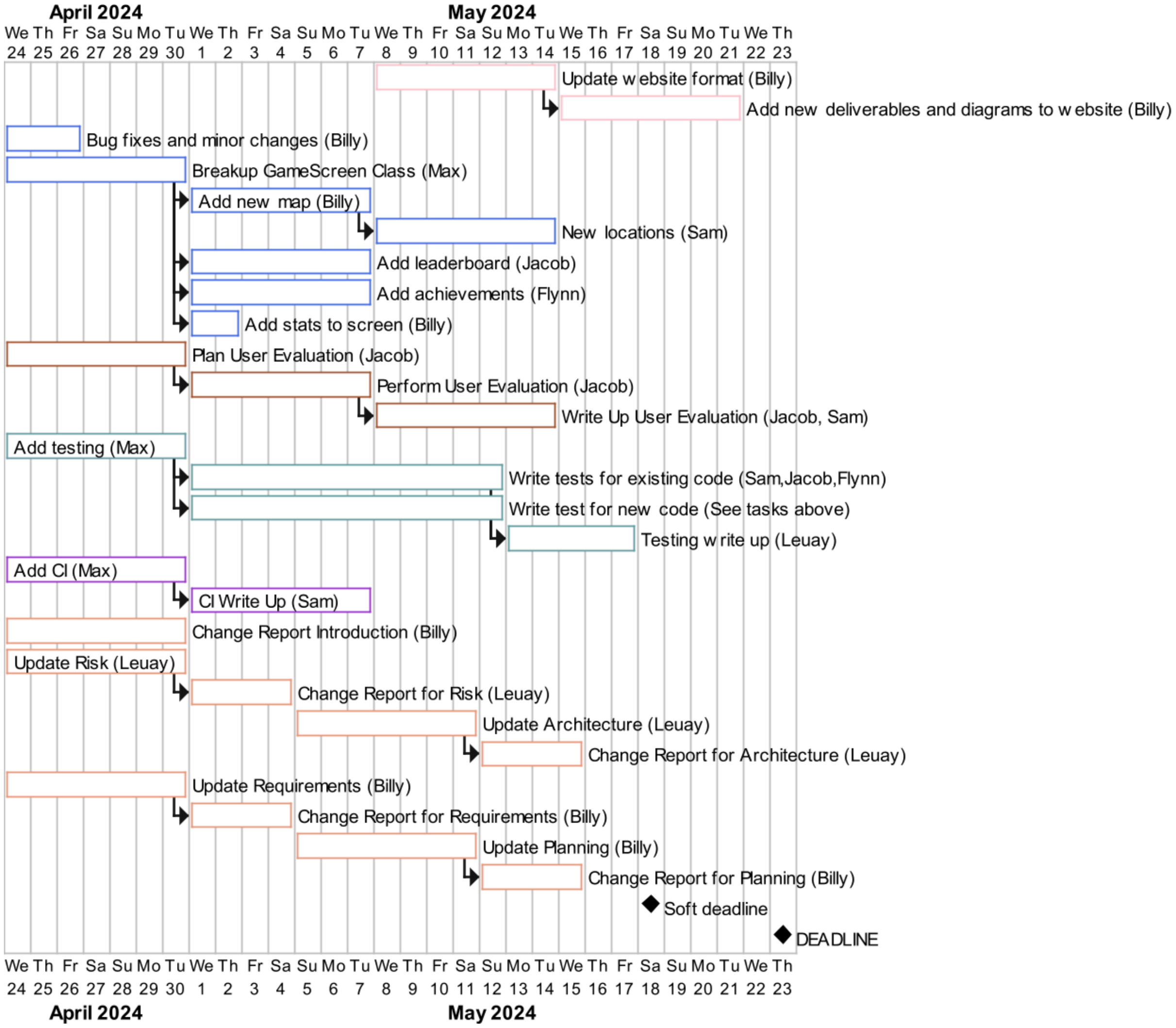
Assessment 1

The Gantt charts [please see Gantt Charts website tab] were updated after each weekly meeting to reflect changes in the plan and variations between the work that was intended to be completed and what was actually completed. As we'd adapted a scrum methodology, each meeting started with a sprint review and retrospective to identify what work had been completed and any issues that team had faced. A new plan was agreed for the remainder of the project. Small changes were required each week. These mostly involved extending the number of days required for tasks or pushing back tasks due to unforeseen dependencies. We started by leaving a spare week for anything that ran over and for proofreading. After the week 2 meeting, the website hadn't been created and the progress with architecture was behind. Luis asked to be moved off implementation so Sam was moved to this and Luis took on methods selection. After the week 3 meeting, the website, user requirements and non-functional requirements needed to be pushed back. User requirements being pushed back restricted the ability to finish functional system requirements and also held back architecture and implementation. After the week 4 meeting, it was clear that some deliverables needed a bit of extra time so these were extended and the proof reading time shortened to accommodate this. It was also necessary to add a task of a week 5 re-plan as things hadn't been finished as hoped. It was also necessary to push back the methods selection write-up.

Assessment 2

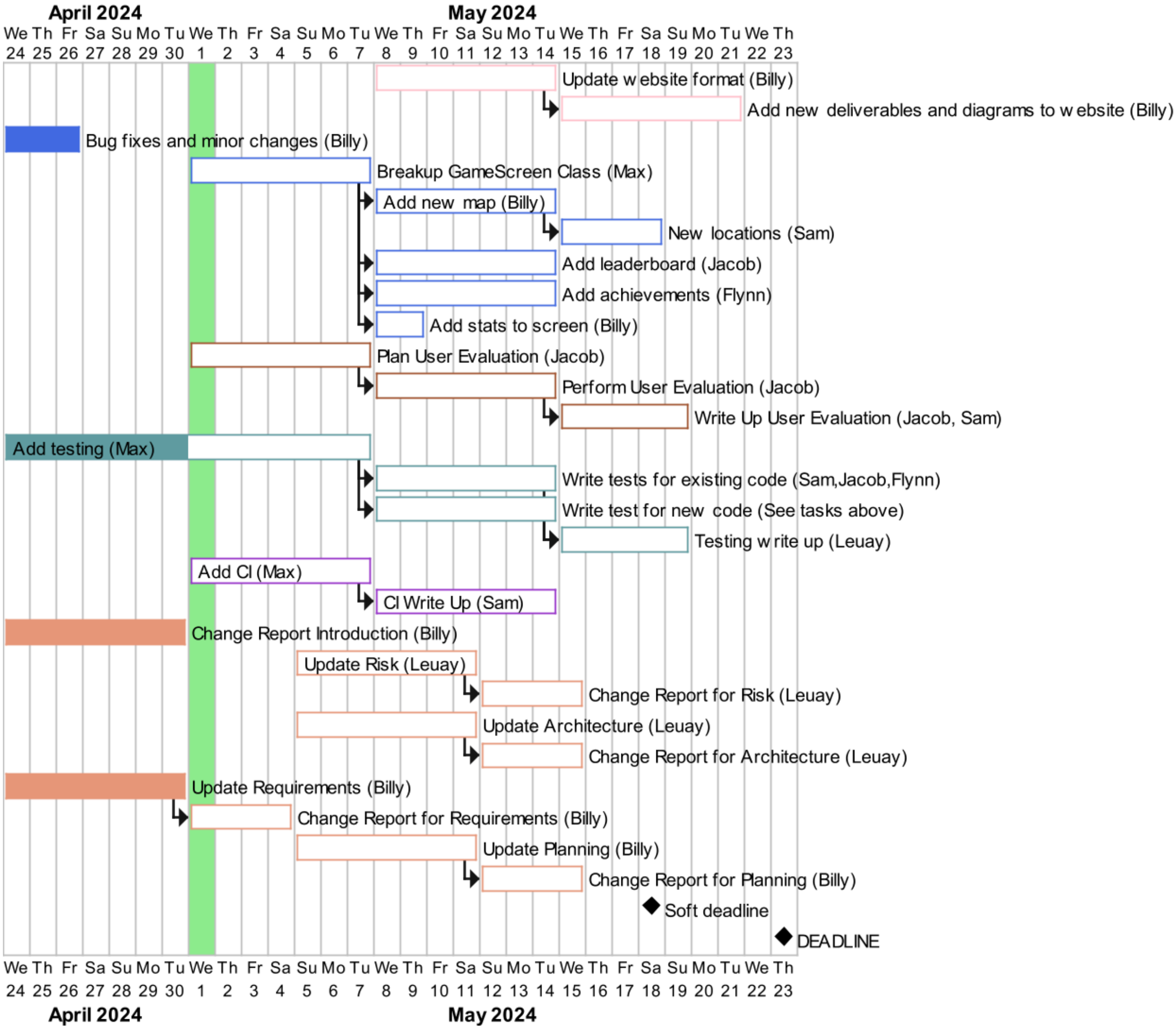
We continued this into the second assessment, we produced a single gantt chart each week. The chart for each week shows the progress that has been made up to the start of the meeting, and any changes in the future plan related to the meeting. The gantt chart omits certain dependencies between tasks for the sake of readability, for example updating architecture is dependent on the GameScreen class having been broken up but to show this an arrow would have to cross a large part of the diagram. As there is no longer a page limit requirement and gantt charts aid readability we have included them below, they can also be found on our website.

Week 0



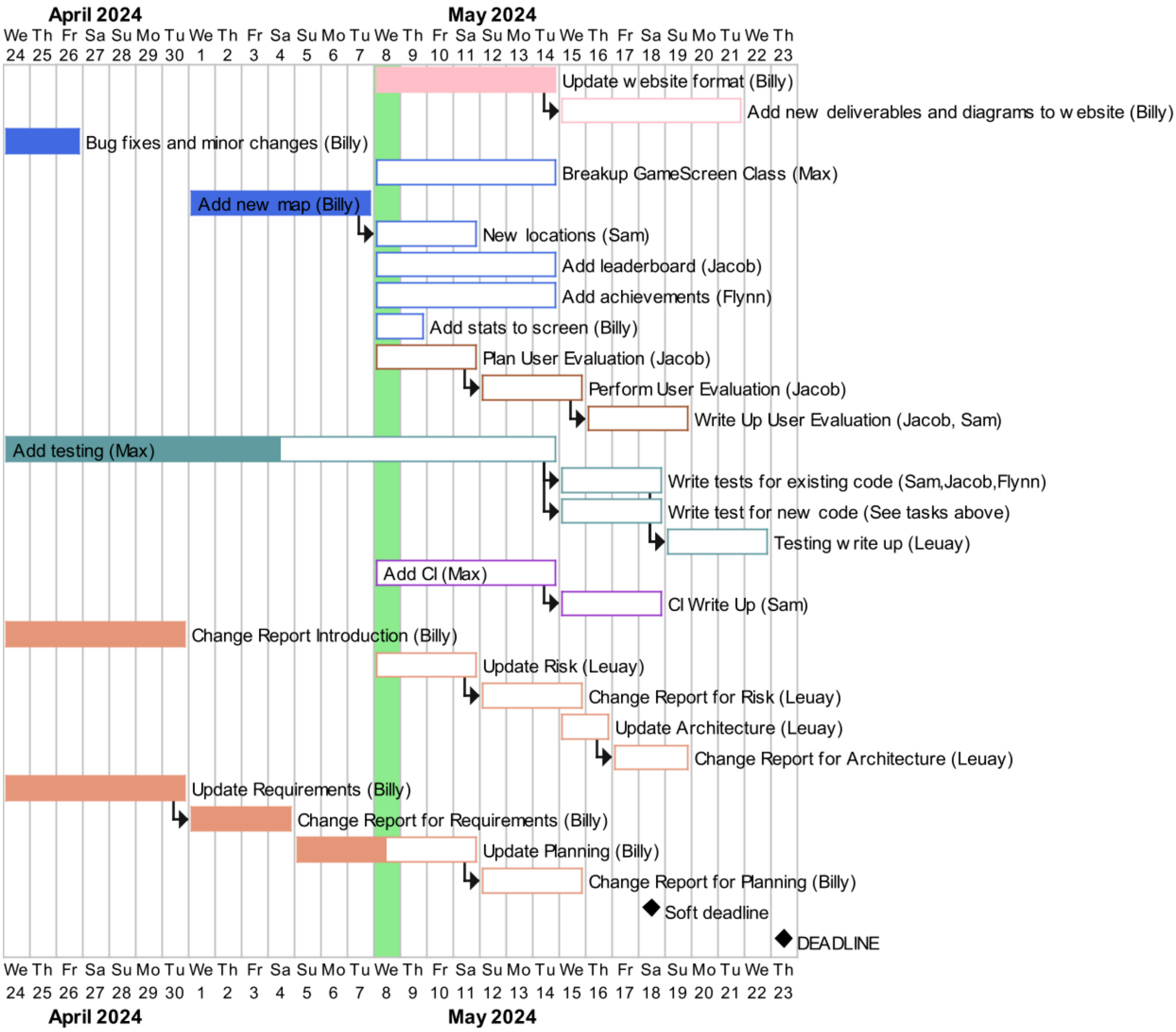
Week 1

In the first week we encountered setbacks, Max found setting up testing required more troubleshooting than anticipated as a result more time was allocated to getting testing set up, this ate into the time allocated to write tests in future weeks. Additionally this caused a delay in setting up CI as we felt it would make sense to set CI up when testing was functional, especially as this was also planned to be completed by Max. We also chose to delay starting on our user evaluation until we better understood its purpose and best practices, to allow for this we shortened the time to write up the user evaluation. Updating risk was also delayed in this case due to the team member responsible, Leuay, temporarily being unable to work for medical reasons.



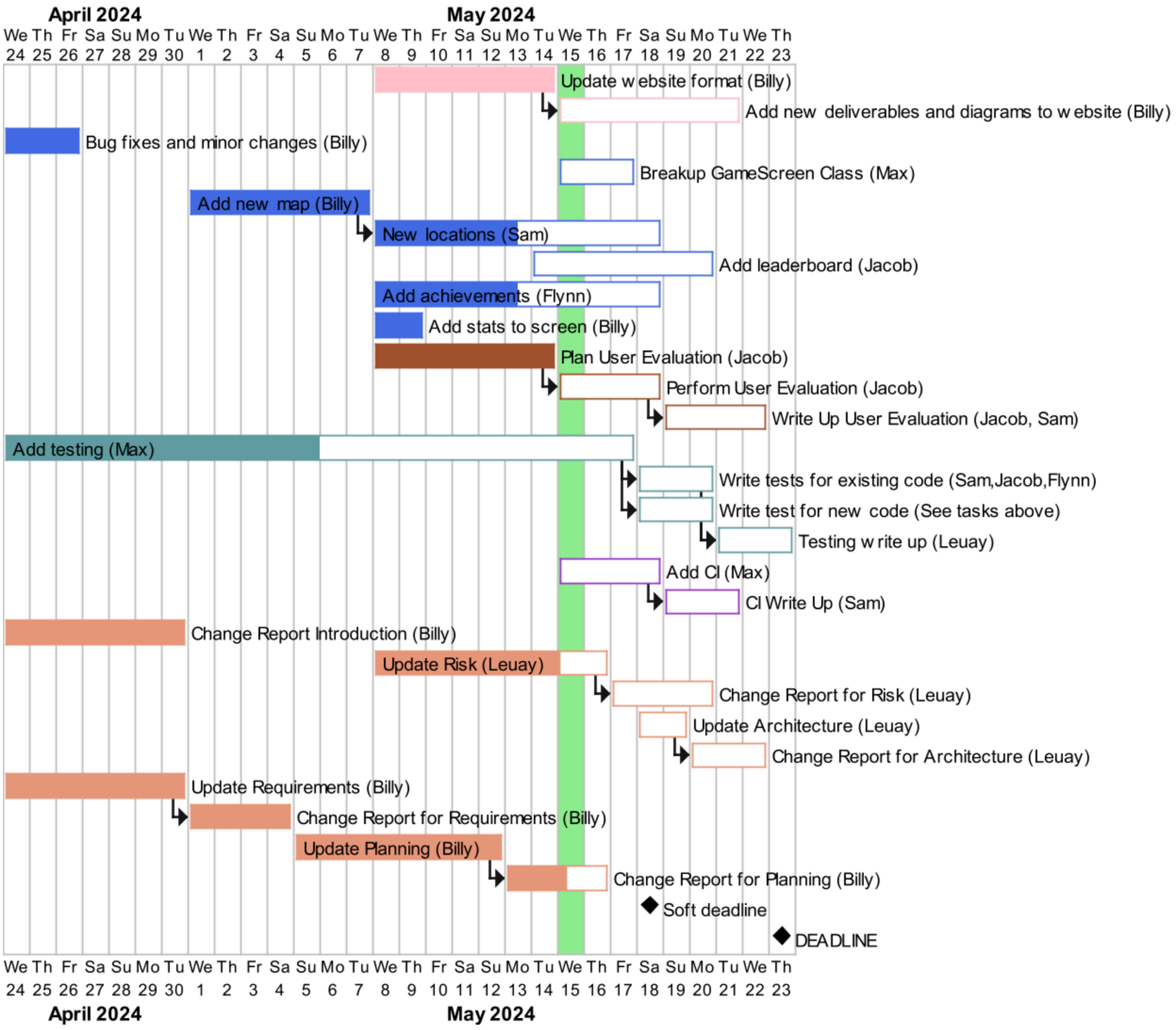
Week 2

During the second week we made some more major changes to the plan, we decided to break the dependency of our implementation tasks on breaking up the GameScreen class. Initially we felt that it would be best to break the class up into multiple smaller classes before working on additions to the game. However we decided during week two that this would take too long and hamper us getting started working on new development so it was decided that the splitting of the GameScreen class would have to be taken up alongside other implementation work. We made this decision mid week and Billy was able to work on adding a second map during the week We were confident that if this led to merge conflicts in git we would be able to resolve them satisfactorily. Our user evaluation was delayed due to personal sports commitments of the team member responsible, as a result the time frame for completing each user evaluation task was reduced. In addition, the time allowed for *add testing* was extended as Max, who was working on it, was unable to work on it for personal reasons, if we were to do this project again someone would have taken over this responsibility.



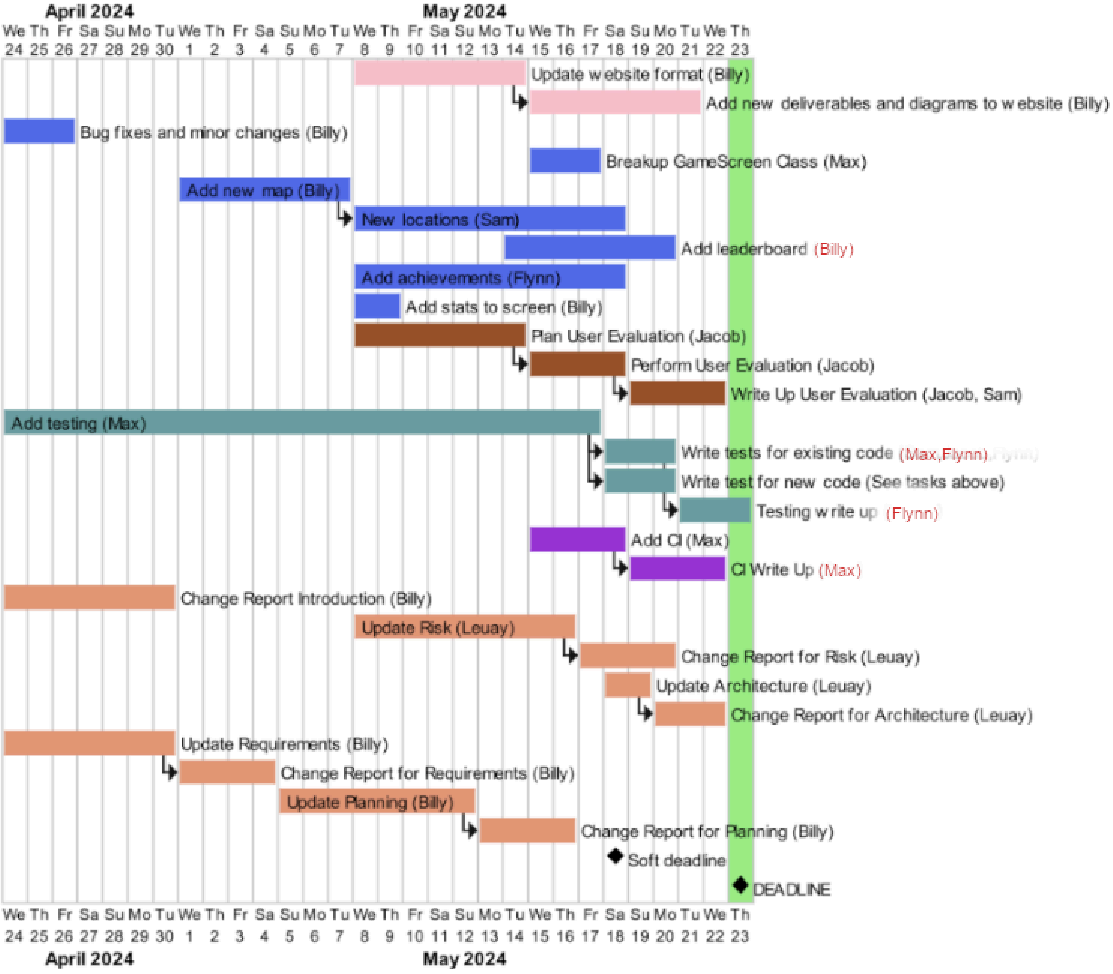
Week 3

During the third week the team member responsible for breaking up the GameScreen class (Max) continued to be unable to work for personal reasons, we chose to leave this task an additional week so as to avoid to reassigning of responsibilities, however this had a knock on effect on testing as some tests cannot be written until the class has been broken up and on updating the architecture documentation. If we could do the project again we would have had another team member take on this responsibility so that the workload was more evenly distributed between weeks. Some implementation tasks took longer than expected for example *New locations*. Planning user evaluation was slightly delayed. Testing was still not complete at this point but Max was planning to get it finished in the next couple of days. As a result of this Adding CI was delayed to week 4. Update planning is shown as complete, this excludes small tasks that couldn't be completed yet such as the week 4 gantt chart.



Week 4 (Final)

The time allowed for some tasks ended up being quite small as a result of team member absences and other factors. All tasks were successfully completed in the end. Quite a few tasks were taken on by different team members at this point as there was a team push towards getting everything done.


April 2024
May 2024
We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th
24 25 26 27 28 29 30 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Update w ebsite format (Billy)
Add new deliverables and diagrams to w ebsite (Billy)
Bug fixes and minor changes (Billy)
Breakup GameScreen Class (Max)
Add new map (Billy)
New locations (Sam)
Add leaderboard (Billy)
Add achievements (Flynn)
Add stats to screen (Billy)
Plan User Evaluation (Jacob)
Perform User Evaluation (Jacob)
Write Up User Evaluation (Jacob, Sam)
Add testing (Max)
Write tests for existing code (Max,Flynn),Flynn)
Write test for new code (See tasks above)
Testing w rite up (Flynn)
Add CI (Max)
CI Write Up (Max)
Change Report Introduction (Billy)
Update Risk (Leuay)
Change Report for Risk (Leuay)
Update Architecture (Leuay)
Change Report for Architecture (Leuay)
Update Requirements (Billy)
Change Report for Requirements (Billy)
Update Planning (Billy)
Change Report for Planning (Billy)
Soft deadline
DEADLINE
We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th
24 25 26 27 28 29 30 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
April 2024
May 2024

References

[1] K. Beck, et al. (2001). Principles behind the Agile Manifesto. Manifesto for Agile Software Development. [Online]. Available: https://agilemanifesto.org/principles.html [Accessed: 13 March 2024].

[2] A. Garg, R. K. Kaliyar, and A. Goswami (2022). PDRSD-A systematic review on plan-driven SDLC models for software development. 8th International Conference on Advanced Computing and Communication Systems, Coimbatore, India, Mar. 25-26, 2022, IEEE, 2022

[3] B. Refi (2023, Aug. 3) Java Game Engines: Top Choices For Game Development. Bluebird. [Online]. Available at: https://bluebirdinternational.com/java-game-engines/ [Accessed: 14 February 2024].