UNIVERSITY OF YORK

DEPARTMENT OF COMPUTER SCIENCE

Method Selection and Planning

Cohort 2 - Group 17 (Rich Tea-m 17)

Group Members:

George Jopson
Ben Slater
Meg Tierney
William Potts
Jamie Burgess
Seyi Towolawi
Alex Staicu

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.

Github was chosen for the code base and website as several team members had prior experience and it is the standard tool. 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, Github 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.

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. Another tool we use for communication is Notion, which helps us keep track of our progress on our tasks during each sprint. This is beneficial to us as it means we can schedule team members to different tasks easier if they are taking longer to complete. 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

Team members were assigned to ~13 marks from the 6 deliverables. Most team members wanted to be on more than one deliverable to gain more experience. As the website had so few marks allocated, this was assigned to just one person. However, all other deliverables had a minimum of 2 team members working on them to lower the bus factor. Assignment started by assigning team members to areas that they had experience of in order to keep the project as efficient as possible. Team members were then assigned to anything they particularly wished to do and finally the gaps were filled.

As there were 6 deliverables and 7 team members, each team member took on leadership for one deliverable, with joint leads for the implementation deliverable. The website was assigned to Meg so she took on the leadership role for this deliverable. George (50%) took on leadership for the Change report, while he, Ben, Seyi/Will and Meg updated the Risk1, Plan1, Arch1 and Req1 documents for the report respectively (12.5% each). Implementation was split between Seyi (50%) and Will (50%), who took on joint leadership of the deliverable. User evaluation was assigned to Ben, who took on leadership for this deliverable. Software testing was split between Jamie (70%) and George (30%). As George already had a role, Jamie took on the leadership role. Continuous Integration was assigned to Alex (100%), who took on leadership for the role. 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 also avoided joint leadership for any deliverable. Deliverable leaders were responsible for splitting the work in that deliverable between the team members assigned to work on it.

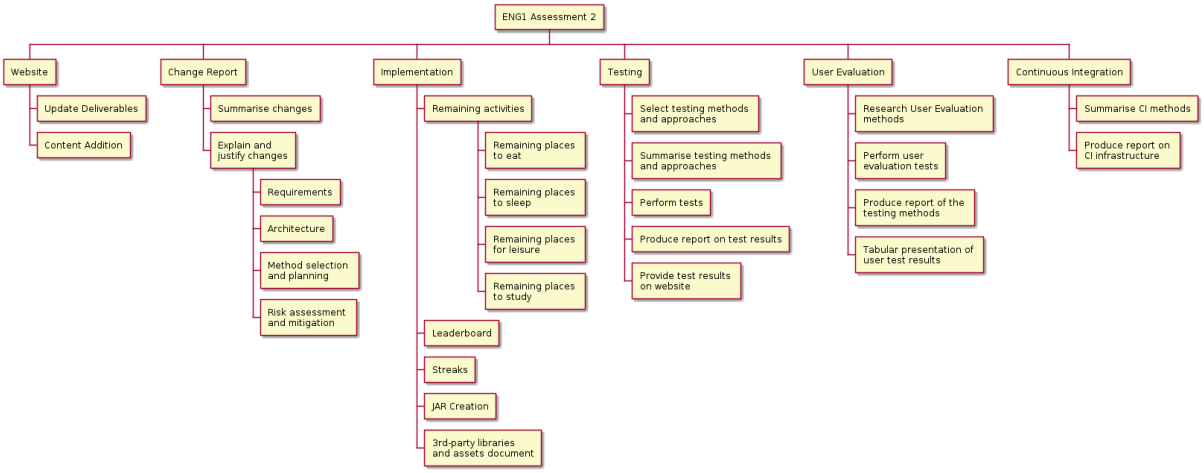
In addition to deliverable leadership roles, each team member took on an organisational role, in order to ensure that each team member contributed something to the team's organisation. George and Seyi are responsible for scrum management and team leadership. They ensure that our weekly meetings result in the most effective use of our time, keeping us focused on high priority tasks rather than useless tasks. In doing so, they maximise our productivity and the progress made between sprints. Alex and Will are responsible for the product backlog. They maintain the Notion database the group uses to select and track the tasks of the current sprint. This facilitates smooth team collaboration by keeping everything in one place where it can be easily referenced and updated in real time. Jamie and Ben are responsible for secretarial tasks. They take notes from each of our meetings to help us record their purpose and view the overall direction of our project. They also maintain other personal notes used by the group to ensure general organisation. Meg and George are in charge of report editing. They ensure any documentation we produce is kept to a high standard by identifying mistakes and any areas of improvement. These roles are inspired by the scrum guide [3] and all facilitate the scrum method by maximising the effectiveness of each sprint, ensuring they are all efficient, clearly defined, well organised and recorded. The benefit of this approach is that it encourages team engagement, everyone feels like they are adding something to the project. Furthermore, to reduce the bus factor [5], we have ensured multiple people are responsible for each role, so that this approach can be married with flexible circumstances such as a team member falling ill. To do this with every role having two people taking control, the first person in the list is the main one that will do this role and the second on the list being the understudy for if the first person is absent.

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.



Deliverables Table

ID	Title	Due date	Description	Visibility	Relevant Tasks
D1	url2.txt	22/05	Website	Shared	T1, T4.5
D2	Change2.pdf	22/05	Change Report	Shared	T2
D3	Impl2.pdf	22/05	Implementation	Shared	T3
D3.1	Code	22/05	Implementation	Shared	T3
D3.2	Executable JAR	22/05	Implementation	Shared	T3.4
D4	Test2.pdf	22/05	Testing	Shared	T4
D5	User2.pdf	22/05	User Evaluation	Shared	T5
D6	CI2.pdf	22/05	Continuous Integration	Shared	T6

Tasks Table

Task ID	Description	Start Date	End Date	Dependencies
T1.1	Update deliverables	16/04/2024	14/05/2024	
T1.2	Add new deliverables to website	16/04/2024	14/05/2024	

T2.1	Summarise changes	16/04/2024	14/05/2024	
T2.2	Explain and justify changes to deliverables	16/04/2024	14/05/2024	T2.1
T3.1	Implement remaining activities	16/04/2024	23/04/2024	
T3.2	Implement leaderboard	16/04/2024	30/04/2024	
T3.3	Implement streaks	16/04/2024	08/05/2024	
T3.4	Upload JAR File	16/04/2024	14/05/2024	T3.1-3.3
T3.5	3rd-party assets document	16/04/2024	14/05/2024	T3.1-3.3
T4.1	Research and select testing methods	16/04/2024	23/04/2024	
T4.2	Summarise testing research in report	16/04/2024	14/05/2024	T4.1
T4.3	Perform testing methods	16/04/2024	14/05/2024	T4.1
T4.4	Produce report on testing results	16/04/2024	14/05/2024	T4.1-4.3
T4.5	Upload testing results to website	16/04/2024	14/05/2024	T4.1-4.4
T5.1	Research user evaluation methods	16/04/2024	05/05/2024	
T5.2	Perform user evaluation tests	16/04/2024	08/05/2024	T5.1
T5.3	Produce report on user testing results	16/04/2024	14/05/2024	T5.2
T5.4	Tabular presentation of user test results	16/04/2024	14/05/2024	T5.1-5.3
T6.1	Research and summarise CI methods	16/04/2024	01/05/2024	
T6.2	Produce report on CI Infrastructure	16/04/2024	14/05/2024	T6.1

Work progress

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 planning and delivering our team presentation, and reviewing the Assessment 2 specification. After the week 2 meeting, we had selected the team whose project we would be taking over from, and we had assigned each other our team roles. After the week 3 meeting, the CI report had been finished, and the Req1 and Risk1 deliverables had been updated, with Plan1 and Arch1 still in progress. The old requirements from the previous team had also been implemented, and unit tests were being written and carried out for these requirements. By week 4, the first Arch1 draft has been completed whilst Plan1 had to be put on hold so that the user evaluation deliverable could start being prepared. The new requirements given in the brief had also been implemented and tests were being written and carried out for them. After week 5, the user evaluation had been fully planned and interviews were ready to be carried out. The user interviews were carried out during week 6, and the user1 deliverable had its first draft completed. During week 7, we had to push our internal deadline ahead so we could polish off the remaining deliverables and upload them to the website. We used the remaining time before the deadline to plan our presentation.

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].

[4] "The 2020 Scrum Guide", scrumguides, 2020, Nov.
Available: https://scrumguides.org/scrum-guide.html

[5] "What's the bus factor and 7 ways to increase it", medium, 2020, May. 13.
Available: What's the bus factor and 7 ways to increase it | by Sébastien Dubois. | Management Matters | Medium