UNIVERSITY OF YORK DEPARTMENT OF COMPUTER SCIENCE

Method Selection And Planning

Group 14

Tecch Titans:

Bradley Mitchell

Daniz Hajizada

Ellie Gent

Joel Crann

Keela Ta

Leo Crawford

Lukas Angelidis

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].

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. 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.

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. **Instagram** 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 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 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 15 marks from the 6 deliverables. Most team members wanted to be on more than one deliverable to gain more experience **in weaker areas**. As the website had so few marks allocated, this was assigned to just one person. However, all other deliverables had a minimum of **3** 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. **Roles during assessment 1 were also taken into account for this, and those who worked on the Impl1 deliverable, continued with the Impl2 deliverable as they felt the most comfortable continuing on with the previous teams' code.** Team members were then assigned to anything they particularly wished to do and finally the gaps were filled.

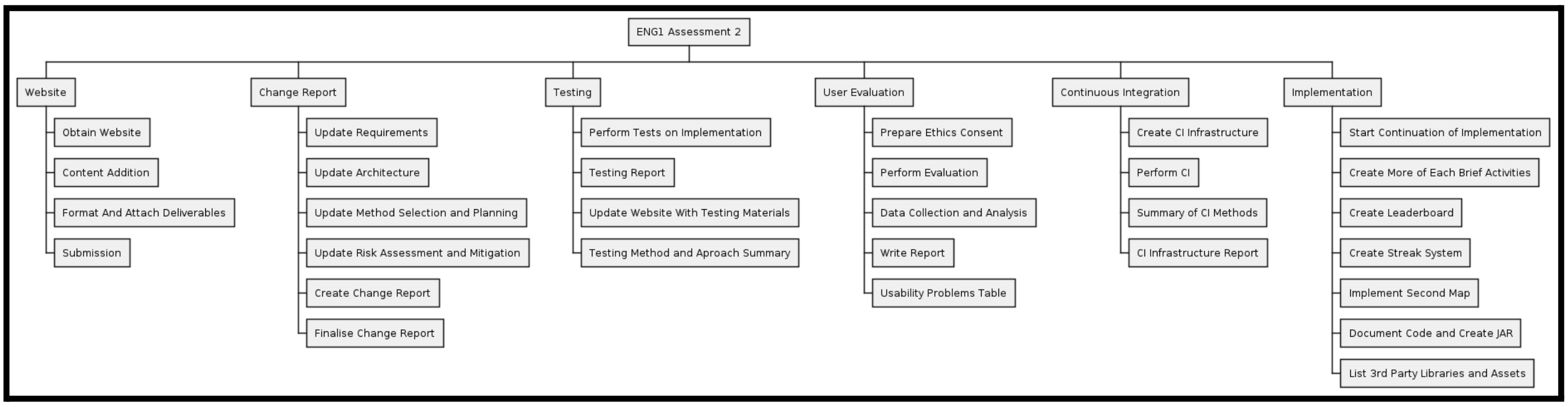
As there were 6 deliverables and 7 team members, the roles were split up to ensure that each member completed enough of their designated marks. The website was assigned to Lukas so he took on the leadership role for this deliverable. Implementation was split between Bradley, Joel, Keela and Lukas. The Change Report was split between Ellie (30%), Leo (30%), Daniz (20%) and Bradley (20%). Testing was split between Keela (30%), Leo (20%), Daniz (30%) and Lukas (20%). User Evaluation was split evenly between Ellie (25%), Joel (25%), Daniz (25%) and Leo (25%). However, all team members got involved to perform a user evaluation with one user each to fulfil the requirements for this section. Continuous Integration was split evenly between Lukas (50%) and Ellie (50%). 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.

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.

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	Relevant tasks
D1	url**2**.txt	**23/5**	Website	T1.1,T1.2
D2	**Change2.pdf**	**23/5**	**Change Report**	T2.1-T6.7
D3	**Test2.pdf**	**23/5**	**Testing**	T6.1, T7. -T7.4
D4	**Eval2.pdf**	**23/5**	**User Evaluation**	T6.6, T8.1-T8.5
D5	**CI2.pdf**	**23/5**	**Continuous Integration**	T6.1, T9.1-T9.4
D6.1	**Impl2.pdf**	**23/5**	Implementation	T6.7
D6.2	Code	**23/5**	Implementation	T6.1-T6.6
D6.3	Executable JAR	**23/5**	Implementation	T6.1-T6.6

Tasks Table

Task ID	Description	Start date	End date	Dependencies	Priority
T1.1	**Obtain and reformat** website	**22/4**	**23/4**		High
T1.2	Add all content and links needed to website	**22/4**	**22/5**	T1.1	High

T2.1	**Amend** requirements referencing system	**22/4**	**23/4**		High
T2.2	**Amend statements of user and system requirements**	**22/4**	**23/4**	T2.1	High
T2.3	**Amend the** introduction to requirements	**22/4**	**23/4**	T2.1, T2.2	High
T3.1	**Amend** diagrammatic representations of product’s architecture	**15/5**	**21/5**		High
T3.2	**Amend** justification for architecture	**15/5**	**21/5**		High
T3.3	**Amend** initial design and evolution	**15/5**	**21/5**	T3.1	High
T3.4	**Amend evidence of design process followed**	**15/5**	**21/5**	T3.1	High
T3.5	**Amend** relation of architecture to requirements	**15/5**	**21/5**	T2.2, T3.1-T3.3	High
T4.1	**Amend** methods and tools including alternatives considered	**30/4**	**30/4**		High
T4.2	**Amend** team organisation	**7/5**	**7/5**		High
T4.4	**Amend** work breakdown diagram	**30/4**	**30/4**		High
T4.5	**Amend** deliverables table	**30/4**	**30/4**		High
T4.6	**Amend** tasks table	**30/4**	**30/4**	T4.5	High
T4.7	Discussion of plan evolution and Gantt charts	**7/5**	**20/5**	Meetings, logbook, previous charts	Medium
T5.1	**Updates of** risk register	**31/4**	**31/4**		High
T5.2	**Update** mitigation and contingency strategies	**31/4**	**31/4**	T5.1	High
T5.3	**Update of risk management process and risk register justifications**	**31/4**	**31/4**	T5.1, T5.2	High
T5.4	Continued risk reassessment	**31/4**	**20/5**	T5.1	Medium
T6.1	Set-up implementation	**22/4**	**25/4**		High
T6.2	**Create at least one of each activity location, with at least three leisure activities**	**22/4**	**26/4**	T6.1	High
T6.3	Create **leaderboard**	**22/4**	**25/4**	T6.1	High
T6.4	Create **streak system**	**30/4**	**4/5**	T6.1	High
T6.5	**Create and implement second map**	**25/4**	**29/4**	**T6.1**	**Medium**
T6.6	Document code and create JAR	**5/5**	**8/5**	T6.2-T6.5	High

T6.7	List 3rd-party libraries and assets with licences and discussion of licence suitabilities	**16/5**	**18/5**	T6.2-T6.5	High
T7.1	**Perform tests continuously throughout implementation and document results for write-up**	**22/4**	**18/5**	**T61.1**	**High**
T7.2	**Write testing report with references if needed**	**19/5**	**22/5**	**T7.1**	**High**
T7.3	**Add testing materials to website**	**13/5**	**18/5**	**T7.1**	**High**
T7.4	**Create testing method and approach summary**	**19/5**	**22/5**	**T7.1-T7.3**	**High**
T8.1	**Prepare ethics consent/fast track forms**	**1/5**	**1/5**		**High**
T8.2	**Create user evaluation prompts and perform user evaluation**	**5/5**	**11/5**	**T8.1**	**High**
T8.3	**Data collection and analysis of research gathered**	**12/5**	**17/5**	**T8.2**	**High**
T8.4	**Write report of user evaluation and research conducted**	**17/5**	**22/5**	**T8.2, T8.3**	**High**
T8.5	**Create usability problems table**	**3/5**	**19/5**	**T8.2, T8.3**	**High**
T9.1	**Create continuous integration infrastructure**	**22/4**	**27/4**		**High**
T9.2	**Perform CI throughout implementation**	**28/5**	**15/5**	**T9.1**	**High**
T9.3	**Summary of CI methods**	**16/5**	**21/5**	**T9.2**	**High**
T9.4	**CI infrastructure report**	**16/5**	**21/5**	**T9.1**	**High**

Discussion of Plan Evolution

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 had chosen to use logbooks for meeting organisation, each**

meeting started with a logbook review to identify what work had been completed and any issues that the 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 setting the 22nd of May as our internal deadline, as we knew we were to be having our presentation on the morning of the module submission deadline. After week 2, several changes were needed. The change report plan had been finished early, and the requirements document had been completely updated ahead of schedule. This meant that all change report tasks could be moved forward. The testing team also requested longer time to create tests for the implementation, and so this was adjusted too. This did restrict the implementation team as we had planned to do these two deliverables in parallel. After the week 3 meeting, it was decided that the change report planning and architecture sections needed longer to be updated as the diagrams were time-intensive to replicate and adjust. In the week 4 meeting, Leo asked to be moved off testing to focus on completing the change report and to start working on the presentation on behalf of the group, so that this is not forgotten. The presentation is not displayed on the Gantt charts but started being worked on from this week. Lastly, the CI report time was pushed back as the CI team had yet to begin on this.

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].