UNIVERSITY OF YORK

DEPARTMENT OF COMPUTER SCIENCE

Method Selection and Planning
Cohort 2 - Group 16 (Skloch)

Group Members:

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

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. 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 15 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 6 team members, each team member took on leadership for one deliverable. The website was assigned to Luis so he took on the leadership role for this deliverable. Implementation was split between Sam (60%) and Owen (40%). As Sam had the biggest proportion of marks, he took this one on. Method selection and planning was split evenly between Hollie (50%) and Luis (50%). As Luis already had a leadership role, Hollie took this on for this deliverable. Requirements were split between Hollie (50%), Luis (25%) and Kaustav (25%). As Luis and Hollie already had roles, Kaustav took on the leadership role. Risk assessment and mitigation was split evenly between Kaustav (50%) and Charlotte (50%). As Kaustav already had a leadership role, this was taken on by Charlotte. Architecture was split between Charlotte (45%), Luis (9%), Kaustav (23%) and Owen (23%). As Charlotte, Luis and Kaustav already had roles, Owen took the leadership 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.

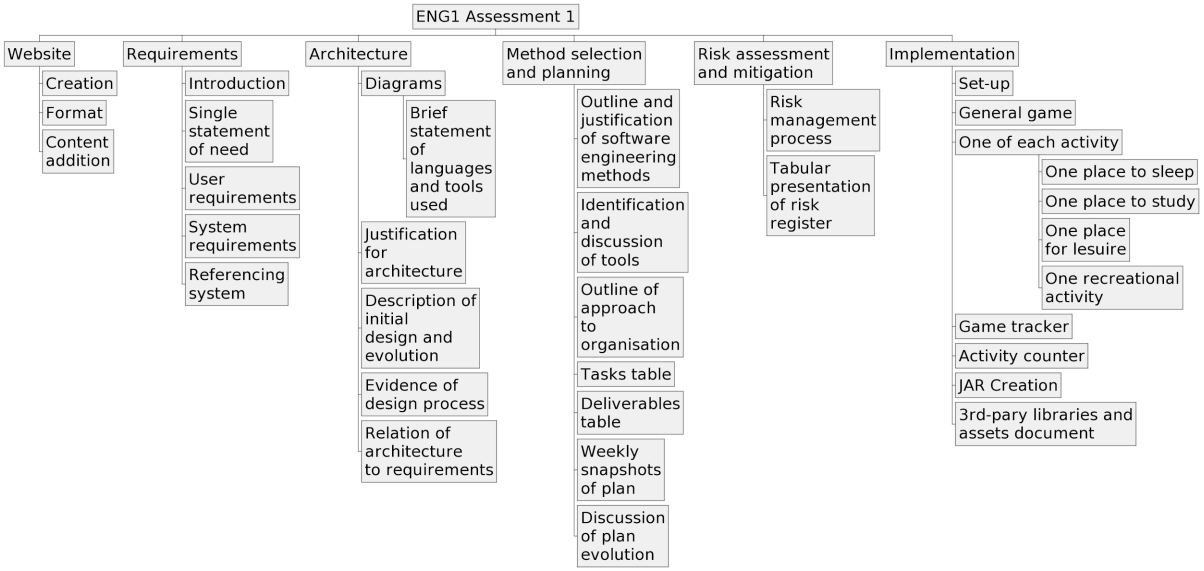
The risk management process required that 3 roles be designated. These were project manager, product owner and team leader. To again ensure equitable work allocation, each role was allocated to 2 team members. As the primary product is the game, Owen and Sam were selected as product owners as they were completing the implementation deliverable. Charlotte and Kaustav were selected as team leaders as they were responsible for the risks and mitigation deliverable and so were naturally leading that process. Hollie and Luis were selected as project managers as they were responsible for the planning deliverable which fits into the scope of project management.

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	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

Tasks Table

Task ID	Description	Start date	End date	Dependencies	Priority
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

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'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.

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].