Method selection and planning

Cohort 4 Group 6

Javangers

Braithwaite, Max
Faruque, Amber
Fu, Zhuoran
Kocaman, Melike
McDermott, John
Rissen, James
Scott, Charlotte

Justification of the team’s software engineering methods, development and collaboration tools:

When deciding upon which software engineering approach to adopt, our group chose an Agile methodology, as it best suited the nature of the project. Agile is an iterative and flexible software development approach that focuses on incremental development, collaboration, and continuous feedback. It encourages teams to adapt to change quickly and deliver working software in short, manageable cycles rather than completing all development in a single phase. Agile methodologies provide flexibility, promote continuous delivery, and reduce the overall risk of project failure in comparison to non-Agile methodologies.

The primary Agile methodology we decided to adopt was Scrum. Scrum is a methodology that organises work into short iterations called sprints. Each sprint involves reviewing, planning, and development, allowing teams to continuously improve and deliver increments of the project regularly. Scrum meetings commonly take place daily; however, we intend to conduct weekly sprints as we do not plan to meet daily. A weekly sprint structure allows us to balance our academic workload while still maintaining regular progress and consistent communication. Although we will meet in person twice a week, we plan to maintain daily contact through our WhatsApp group to discuss progress.

The use of Scrum provides us with significant flexibility. Since game development requires constant improvements and prototypes, testing and evaluating as we progress, a flexible approach is required. If changes need to be made midway through the project, for example if a team member wants to change roles, or new user requirements are released, flexibility is key in the project's success.

Scrum focuses on development divided into sprints, with each sprint planned in a meeting. This is suitable for our team, as we intend to meet at least once a week outside of timetabled sessions, to plan the sprint for that week. Each time we meet, we discuss task priorities and everyone’s progress on their assigned task from the previous week. We discuss risks and anticipate potential issues that may arise. The flexibility allows us to reallocate roles within the team if any issues arise. Finally, we assign team members responsibilities for the next sprint for the upcoming week.

Scrum allows us to adapt quickly and make changes to our initial plans, unlike other non-Agile methodologies like plan-driven development, in which plans aren’t interchangeable and adaptable, and is therefore unsuitable for evolving projects like game development. The outcome of game development very rarely matches the initial plan exactly. Changes tend to be made to the initial plan throughout the game's development if new, more efficient ideas and solutions are created. We considered other Agile methodologies such as XP, however we believe it is less suitable for a short-term, small-group game development project due to its intensive technical practices such as programming and continuous integration, which can be time-consuming and hard to manage within a short timeframe. In contrast, Scrum’s clear roles and structured sprints are better for keeping small teams organised and focused on deliverables.

Agile methods like Scrum and XP are the best suited for small to medium sized teams. This is because they allow for clearly assigned roles for each team member, so each member has designated responsibility and therefore has a clear understanding of their contributions. This

is suitable since the project is short term, in small teams. Each team member can be primarily assigned a specific section of the project to be primarily responsible for.

We haven't just limited ourselves to using Scrum alone. We intend to use Kanban Boards to assist in project management. Kanban boards offer visual tracking of tasks and workflow, allowing the team to see progress at a glance and quickly identify constraints.

As a team, we are utilising shared Google Drive as a key collaboration tool for our group, where we can collaboratively contribute to brainstorming ideas, and writing up the project. Each team member has been assigned a primary section of the markscheme to work on, consisting of separate Google Docs for each section. However, team members can access all of these docs, and add and contribute to any, for example if we decide to prioritise a specific task, we could assign more team members to work on it. This tool supports Scrum since it promotes transparency and collaboration.

WhatsApp is another collaborative tool we have utilised. Within our group chat channel, we can communicate with each other on a daily basis to keep track of each other's progress, ask questions and plan in person discussions. This supports Scrum software methodology as it allows us as a team to communicate and to plan our next team meeting and sprints. It also supports flexibility as we can discuss progress and problems, and re-allocate roles if needed.

We are also using GitHub as both a developmental and collaborative tool. There are many benefits to using GitHub. Initially we made an organisation for this project, which creates a team workspace, allowing us to create a repository, assign roles and push changes to the game. GitHub will track every change made to our code. Branches can be used to work on features separately, and merged once completed. It provides us with solid version control which is essential for group coding projects. This software also fits in nicely with our software development methodology Scrum since we push new changes each week, and use the provided task board to organise work allocation.

When deciding upon which game engine to use, we researched a few different options. The main game engines we found were LibGDX, gMonkey and LITIENGINE. Our final decision was to go with LibGDX. LibGDX is one of the oldest and most mature java development frameworks, therefore it has a large community with plenty of tutorials for supporting our team in our game development, unlike others such as gMonkey which are much more niche with a smaller community. LibGDX also provides us with more extensive tools and libraries than Gmonkey and LITIENGINE, which can better support our game development.

The IDE which the majority of the team decided to use was Intellij IDEA. IntelliJ IDEA provides smart code suggestions, error checking, and easy debugging, which help speed up development and reduce coding errors. IntelliJ also integrates well with GitHub, allowing us to manage version control, push changes, and merge branches directly from the IDE. Its robust features make collaboration easier, as team members can work on different parts of the project efficiently, while maintaining code quality. Overall, IntelliJ IDEA supports our Scrum workflow by allowing quick, organised development

Organising the Team:

Since our team is organised using Scrum, our strategy for allocating tasks was to assign each team member responsibility for a specific section of the project. We went round the group and discussed each other's main strengths and weaknesses, allowing each member to request their preferred role in which they would feel comfortable and confident in their ability to fulfil it. For example, we identified which team members would prefer to focus on coding the game, or gathering user requirements and conducting the client interview.

Our initial identification of tasks was based on the deliverables section of the mark scheme so we assigned roles accordingly. Regardless, no team member was strictly limited to their role - there are no restrictions. Instead they were given primary responsibility for that role. All team members are able and encouraged to contribute to other tasks within the group provided their primary focus is on their initial role. This approach aligns with Scrum, and allows us to distribute team members to tasks of higher priority if needed. We felt like this approach is best suited for the team as each member can focus on areas where they are strong and confident, but also allows them to experience and assist in other areas of the project where needed, keeping the element of flexibility. This approach encourages accountability and collaboration within the team.

Our team's approach to organisation and communication involves in-person weekly meetings outside of practical sessions, at least once a week to perform our weekly sprints, as well as continuous online interaction. Timings of meetings are discussed and organised in our WhatsApp group prior to the meet. Outside of meetings, we maintain daily communication through WhatsApp to share updates, coordinate work, and raise any quick questions or issues. This allows the team to stay connected, monitor progress, and respond promptly to changes. This approach is appropriate for both our team and the project because it supports consistent collaboration, accountability, and flexibility. Weekly meetings keep us organised and aligned with project goals, while ongoing online communication ensures continuous progress even when we cannot meet in person.

Our team's organisation correlates directly with the project plan, which shows a start to finish timeline of all the tasks to be completed. The plan and requirements will change throughout the project, and so how we organise our team and allocate roles may change, and can be discussed in the weekly sprint meetings or via the WhatsApp group. As a team, we have set our own internal deadline, being a week prior to the actual deadline. This is to ensure that we have enough time to peer assess each other's work for quality assurance and to avoid the risk of anyone not understanding the requirements.

Overall, this organisational strategy provides the structure needed for clear task ownership while remaining flexible enough to adapt to challenges, making it well-suited to both our team's working style and the project's requirements.

Systematic plan for the project:

We broke the project down into a series of **Work Packages** (WPs):

WP1: Team Organisation, **WP2**: Requirements, **WP3**: Method selection and Planning, **WP4**: Architecture, **WP5**: Risk Assessment, **WP6**: Game implementation, **WP7**: Website.

Each Work Package will contain a sequence of sub-tasks in the form: Task (per WP). These sub-tasks, along with their dependencies, priority, start and end dates, can be found on the project task table on the website.

Milestones: These mark important achievements and checkpoints in the project timeline, representing key points where significant progress or completion happens.

M1: Establish and organise the team - Due Date: 29th September
M2: Establish user requirements post-interview analysis to ensure team familiarity - Due Date: 13th October
M3: Finalise the project's work packages before the peer review - Due Date: 5th November.
M4: Finish game development for testing - Due Date: Wednesday 5th November.
M5: Project completion and submission - Due Date: Monday 10th November.

We have also broken down the project into a series of deliverables. Deliverables are concrete outputs of the project. These can be either documents (like Requirements or Architecture) or software (like prototypes). Each deliverable is given an ID and can have multiple versions, such as interim drafts and final versions, each with unique IDs, e.g. (D1.1, D1.2). The deliverables we've created can be seen via the deliverables table on our website.

The following write-up outlines the changes made to the project plan throughout the project timeline and references the weekly Gantt charts, which are available on our website.

Initial Gantt Chart: This Gantt chart shows our initial timeline for the project, which we aimed to follow as closely as possible however, we anticipated some adjustments as minor diversions from the plan were inevitable. The first week (Team Organisation) was to focus on team introductions, identifying individual strengths and weaknesses, and assigning primary roles. Once completed, we planned to move on to Requirements Elicitation by brainstorming interview questions prior to our interview with the client and to start thinking about our approach to the Method Selection and Planning section. Our Interview Analysis depended on the preparation and execution of interviews, which was planned for around the 9th October. Once the analysis was complete, we planned to begin work on the Risk Assessment and to develop initial Architecture Designs for the game, both of which would rely on the client's requirements to guide decision-making. Game Development would begin shortly after the architecture phase to allow time for designing the game's core structure (e.g., primary classes) before starting to code. The website development was also planned to begin alongside the game development however this ended up getting delayed. Finally, Game Testing and Peer Reviewing would take place at the end of the project. Testing depends on the completion of game development, while peer reviewing relies on all team members finishing their assigned sections. Finally, note that the main sections: requirements, planning, risk assessment and architecture all continue right up until the end of the project since team members will be continuously adding and updating to these sections.

Week 2 chart: As planned, organising our team took one week. In week 1, we became familiar with our team members, discussed and distributed roles within the group. In week 2 we scheduled a meeting with our client to gather requirements. Many subsequent tasks

depend on this interview, as it provides the information needed to guide our decision-making moving forward. To maximize time for post-interview tasks such as Interview Analysis, Risk Assessment, and Architecture Design, we aimed to conduct the interview as early as possible. We successfully booked it for Tuesday, 7th October, moving it two days earlier than originally planned. As shown in the week 2 Gantt chart, we decided to assign these extra days to the interview analysis, as this is a really important section and it's crucial that we fully understand the client's requirements.

Week 3 chart: In Week 2, we began Requirements Elicitation by brainstorming interview questions as planned. We also started Method Selection and Planning, deciding on a software engineering methodology to follow and setting up collaboration and development tools, such as GitHub, for the project. No changes were made to the Gantt chart for Week 3, as we remain on track with our initial plan.

Week 4 chart: In week 3, we conducted the interview with our client, and organised our findings into tables outlining user requirements, functional and non-functional requirements. Those that conducted the interview informed the other team members on the information they received, and as a group, we performed the interview analysis. Again no changes were made to the Gantt chart for Week 4, as we remain on track with our initial plan.

Week 5 chart: In week 4, we began our risk assessment by outlining our chosen risk management process and brainstorming potential risks that could arise during development. At the same time, we created initial architecture designs for the core functionalities of the game. Our coding team then worked in parallel but slightly behind the architecture phase, implementing each component after its design was completed. We plan to maintain this pattern throughout the project to ensure smooth handovers and steady progress. Our team member who was originally assigned to begin the website development was unable to start as scheduled due to unforeseen family matters. This pushed the start date back by seven days to Monday the 20th of October (Shown in the week 5 Gantt chart). Our choice of Agile scrum methodology provided us with the flexibility to adapt to this unforeseen change. The website development task was reassigned to another team member and elevated in priority. To remain on schedule, we also prepared to involve additional members if needed to recover lost time.

Week 6 chart: in week 5, we continued to develop our architecture designs, and coders continued to implement them. With the core game functionalities established, we began adding the hidden, positive, and negative events. We continued to structure and add diagrams and charts for sections such as architecture and planning to our website. This week we were on track, and there weren't any unexpected events to change the chart for week 6.

Week 7 chart: During Week 6, our team focused on finalising and refining each written section in preparation for peer review. We successfully completed the game implementation three days ahead of schedule, enabling us to commence testing earlier than planned on Monday, 3rd November. This provided additional time to thoroughly evaluate and ensure the game meets the required functionality. Based on this progress, we introduced a new task, refactoring the code. The refactoring process is directly dependent on testing results, as any identified issues or inefficiencies will guide our improvements.This will allow us to further optimise, refine, and finalise the game, ensuring it's up to quality standards.

Final Chart: In week 7, we conducted our testing, identified any errors with our code and refactored the code to resolve these issues. We also conducted our peer reviews on the written sections for quality assurance ready for submission. The week 8 chart as well as the Final chart show our progress in the final week and the final completed plan.