Method Selection and Planning

Group 26

devCharles

Ross Holmes
Sabid Hossain
Thomas Pauline
Joel Paxman
Andrey Samoilov
Louis Warren

Part a)

Within our team we discussed the tools that we thought were the best fit to optimize collaborative work. The tools we decided to use were:

- Github (Including Github projects and actions)
- Intellij
- Vscode
- Discord
- Google drive
- PlantUML

This was for the following reasons:

- GitHub provides:
  - Convenient way for developers to host and share our code
  - Possibility to track changes
  - Easy way to Collaborate with each other on our project

  It also offers a variety of tools for managing and reviewing code, such as:

  - Pull requests
  - Issue tracking

  The team creates Pull requests that can then be assigned to issues which can be created in the projects, making it visible as to which feature is being worked on and its current status. The pull requests can then be reviewed by a different team member when they are complete and once reviewed it can be merged with the main branch, with our continuous integration running tests on the code and running a prettifier. This then makes sure that the code follows a specific style and functions with the current code base.

  Additionally, GitHub is widely used in the industry, making it a popular choice for developers to share and discover open-source projects and a great way for us to familiarise ourselves with such a

  Alongside the features mentioned above we also decided to use the newly released github projects. This massively streamlines our planning as it contains:

  - Gantt diagrams
  - Tasks
  - Road map

  This means as we create issues we can give deadlines as well as descriptions for what needs to be done. When these tasks are completed, we can merge pull requests that solve them and then github projects will automatically complete the tasks. Then when looking at our roadmap we can see the road map as it evolves, allowing us to see how our plan is working and whether it needs to be changed or if we are progressing on schedule.

  We also chose to use PlantUML to generate gantt charts to help track the progress of our project. We can then update this every week to show the weekly status updates. However after the first several weeks we realised that our roadmap from github was simpler and clearer to use.

  Following all this Github Actions has enabled us to introduce Continuous Integration that automates:

  - The build
  - The test
  - The deployment pipeline.

We chose Github actions mainly since we were using so many other github features, allowing us to integrate our work with github further and learn more about the industry tools we will be using in our careers.

- IntelliJ and VSCode are the IDE's that we chose as they provide advanced features such as:
    - The ability to set breakpoints
    - Step through code
    - Inspect specific variables
    - Code completion
    - Code refactoring

  This makes it easier to fix bugs in code as well as helping us write code more efficiently and with fewer errors. Intellij was chosen specifically because it is less finicky with testing than VSCode. VSCode was also used as its live share feature enabled us to collaborate far more easily when having virtual meetings. Several other key benefits include that they both have large and active communities, so there is support available and they are available on all OS systems.

- Discord was our method of communication outside of group meetings. Due to:
    - Channels can be organised by topic or purpose
    - Multiple meeting rooms can be created allowing for simultaneous meetings
    - Direct messages can be sent to any team member
    - Screen sharing allows for multiple team members to see what's happening on an individual's screen
    - Discord allowing integration with third-party tools such as GitHub, which allows us to easily update all members of the team as to our progress of the game

  In addition to channels being usual for separating information about different topics this can also allow for different members being part of different channels and therefore team members can ping one channel and everyone on the team that is part of that channel will receive the notification.

  Slack was not chosen as Slack channels are organised by team or project. This would have not been utilised as we did not have multiple projects or teams as we are working on one project with only a small team. As well as this discord was used by all team members before the project started so was familiar with every member. Finally, Discord has better calling facilities compared to Slack, and since a significant portion of our meetings are conducted online, Discord was the better choice..

- Google Drive was our primary system for sharing and working on different files mainly:
    - PDFS,
    - Google Docs
    - Google Spreadsheets

  This is an extremely useful tool for a Java coding project involving large amounts of documentation as it allows for:

    - cloud-based file storage
    - real-time collaboration
    - version history
    - access control
    - integration with other tools

  This makes it easy for all team members to work together on our documentation, even when we were not in the same physical location. And it is also a good complement to other tools like GitHub to improve the collaboration of our Java game.

OneDrive was an alternative we considered but decided not to choose as it's:

- not as easy to collaborate with team members
- not as robust
- known to cause delays on live collaboration
- relatively slow to show updates to collaborators.

Google Drive comes with many cloud based productivity tools like:

- Google sheets
- Google slides
- Google docs

All of which allow you to:

- Create
- Edit
- Share documents

Onedrive does not provide cloud based applications and therefore if we were to use it we would need to install applications. And since it is not cloud based these applications are more difficult to work on simultaneously.

All of the tools above aided our collaboration on our game development as we looked for the following attributes to aid us in improving team 32's game:

1) Version control: Using a version control system allows team members to collaborate on code and track changes, with the ability to revert to previous versions if needed.

2) Efficiency: Using Continuous Integration allows the development to be more automated, improves code quality and speeds up the development cycle.

3) Communication: A real-time communication platform can be used for team members to discuss and collaborate on code, as well as share files and screenshots.

4) Cloud-based file storage: A cloud-based file storage platform allows team members to access project-related files and documents from anywhere with an internet connection.

5) Project management: A way to keep track of tasks, deadlines, and progress, and keep team members informed of the overall project status.

6) Code review: Establishing a code review process allows team members to review and provide feedback on each other's code, which can help improve the overall quality of the codebase.

7) Access control: Control of access levels for different team members, allowing for managing who can view, edit, or share files and documents.

8) Security: Ensuring that all tools used are secure and that sensitive data is protected by encryption, two-factor authentication, and other security measures.

Part b)

To effectively work as a team, it is important that we play to the individual strengths of each team member, such that everyone feels comfortable in the role that they have been assigned to, thus performing to the best of their abilities.

We held a group discussion about the group's prior software experience to determine who was suitable and willing to take on specific roles within the group. Furthermore, we had to have a Forming discussion about the roles we would need in the lifecycle of the project and therefore every member of the team had a clear idea of their role and what they were supposed to do.

During the discussion, the notable roles for members to play were: Meeting Chair, Secretary Librarian, and Report Editor.

- Meeting Chair: They ensure that the group remains on task during our meetings and is responsible for assigning tasks at the end of the meetings.
- Secretary: They must record decisions made in meetings for future use through meeting minutes.
- Librarian: Their role is to keep documents and other resources and to oversee version control.
- Report editor: Their role is to oversee and organise the production of reports.

  We also decided to divide some roles between two teammates as this meant that we could have a higher bus factor and enabled more of our group to be aware of what to do.

Roles in assessment 2 were re-assigned as we had already worked together during assessment 1 and so we already had an idea of what each of us were good at and could use that to decide on roles. Using our knowledge from assessment 1 combined with each member's personal preference we decided on the roles in the table below.

Sabid	Report Editor	Junior Developer
Ross	Secretary	Senior Developer
Andrey	Meeting Chair, Librarian	Senior Developer
Tom	Secretary	Junior Developer
Joel	Librarian	Junior Developer
Louis	Report Editor	Senior Developer

In assessment 1, 3 group members (Ross, Andrey and Louis) did most of the programming for the game whilst the other members (Sabid, Tom, and Joel) focused more on the other deliverables.

For this assessment we decided to involve everyone to do some element of programming which led us to create the senior developer and junior developer roles to each team member. The senior developers are the group members who worked on implementation in assessment 1, and therefore have more experience with the libraries and game development than the junior developers do.

Therefore we decided the best way to account for this difference in experience would be to pair each junior developer with a senior developer as they got used to the programming so they had someone to ask questions to and learn from. We decided to pair up:

- Ross with Sabid
- Andrey with Tom

- Louis with Joel

Although occasionally these pairs worked with other individuals depending on what tasks were being done at the time.

Throughout the project Andrey (and very occasionally Louis) would plan out tasks and meetings for the group. Allowing every member to have a clear task to complete by the next meeting, as well as knowing when the next meeting would be held.

Part c)

Beginning the project, we looked at the key tasks for each deliverable that needs to be completed. This is demonstrated in the table below:

		Start Date	End Date
Requirements	● Change to fit with our team	10/02/2023	17/02/2023
Architecture	● Research how game is designed	18/02/2023	04/2023
Risk Assessment	● Research into appropriate ways to demonstrate a risk register	19/02/2023	20/02/2023
	● Create our risk register	20/02/2023	20/02/2023
Implementation	● Code new game features	18/02/2023	04/2023
	● Expand list of 3rd party libraries and assets	18/02/2023	04/2023
Method and Planning	● Software engineering methods	22/02/2023	22/02/2023
	● Team organisation	20/02/2023	22/02/2023
	● Create a systematic plan for the project	22/02/2023	?????????
Testing	● Write tests for existing code.	18/02/2023	03/04/2023
Change Log	● Keep documentation clear about edits and new items.	24/02/2023	03/05/2023
Website	● Add the necessary details for the website	04/2023	04/2023

At this point in the since it was so early in the project the end dates for these tasks were so far away we had no specific dates. This was therefore just set as a specific month and as we continued the project our plan was to refine this to specific dates.

As part of our team-forming session, we created a plan for when each part of the project would need to be completed. The first thing on the agenda was to identify whether we would need to work over the easter break. The group decided that it would benefit the project if we continued to work during the break, however, allowing a short period of recession over Easter Sunday. Considering these aspects, we can develop an initial plan.

The initial plan worked as follows:

- Start editing all the deliverables except architecture as we decided it would be more efficiently done towards the end of the project.
- Simultaneously start changing the Requirements, Method Selection and Planning (MS&P) and Risk Management & Mitigation by splitting up tasks amongst the group.
- During the period of 18/02/2023 to 22/02/2023 we would complete the requirements and risk management deliverables subject to mid-project changes.
- Two thirds of the MS&P would be completed by 22/02/2023
- The 3rd/last section (Systemic plan for the project), as in this part, would be updated as we progress through the project and be fully completed by the end of it.

- Start creating the tests for Team 32s code on 20/02/2023 and test them alongside our initial implementation work that we started two days earlier.

  With a better understanding with time of how the life-cycle of the project is going to look, we can formulate a number of new risks that could impact the development of the project.

So far the project met the group's expectations and was completed on schedule. Therefore, there is not much differentiation between the first iteration and the second iteration of the plans in terms of timings.

As a result of completing prior work, we have a clear understanding of the project as a whole and can now continue onto more technical aspects of the system.This is when we delve deeper into the classes needed for the new system requirements we received for Assessment 2 on 16/02/2023. As mentioned previously of course, the justification of the architecture could not be completed until the architecture was developed. As a result, writing the report was not a top priority. Thus, the Architecture report was neglected until the previous work was fully completed.

Updated plan (15/03/2023):

In general our initial plan worked as expected, every team member had clear tasks and we created a regular meeting schedule:

- Mondays, Wednesdays and Fridays
  - 12:00 - 14:00
  - break (1 hour)
  - 15:00 - 17:00

This enabled everyone to have clear times when ENG work was expected to occur, along with this as everyone was in the meeting it meant anyone who needed help could immediately ask somebody for help. This generally occurred virtually as well as the occasional in person meeting.

This meeting schedule was kept for about a month, but some members started becoming busy during meeting times, and it became inconvenient to be required to be free every monday, wednesday and friday, so we decided to change the structure of our meetings.

Our new plan was as such:

- Mondays, Wednesdays:
- A quick (15 min) meeting, discussing what was done between the last meeting and the current one, and what to do before the next meeting.
- The timing of this was also flexible, allowing every member of the group to attend every meeting
- Fridays:
- This was a longer meeting (around 2-3 hours) planned between 12:00 - 15:00 such that everyone can work and communicate if they come across any issues

This then meant that work could be done between those meetings at team members convenience allowing for greater autonomy within the group and so hopefully greater productivity.

This was all implemented with our second stage of our plan which was as such:

		Start Date	End Date
Requirements	● Further improve with feedback from last assessment	10/02/2023	17/02/2023
Architecture	● Continue designing the game so this document can be created later on	24/04/2023	02/05/2023
Risk Assessment	● Further improve with feedback from last assessment	19/02/2023	20/02/2023
		20/02/2023	20/02/2023
Implementation	● Continue implementing new game features	18/02/2023	24/04/2023
	● Continue to expand list of 3rd party libraries and assets	18/02/2023	25/04/2023
Method and Planning	● Further improve with feedback from last assessment	22/02/2023	22/02/2023
Testing	● Continue writing tests for existing code.	18/02/2023	01/04/2023
	● Write tests alongside new code	18/02/2023	24/04/2023
Change Log	● Keep documentation clear about edits and new items.	24/02/2023	25/04/2023
Website	● Add the necessary details for the website	9/03/2023	26/04/2023

This system worked up until 6/04/2023 when team members had holidays planned and so could not participate in meetings. Alongside this others were behind on several modules so couldn't focus on eng, which meant nobody was keeping people responsible. This led to 2 weeks of no productivity, massively disrupting our plan.

Since eng work started back up after 20/04/2023 this led to severe cramming. As given in the github road map (which was updated to reflect what actually happened) it shows how our focus increased massively to testing and implementation, with little to no work done on documentation until the last few days.