Method Selection and Planning

Galin Dzhumakov

Aisyah Firoz Khan

Faran Lane

Samuel Nicholson

Jack Polson

Alana Witten

4. Method selection and planning

Part a)

Within our team we discussed the tools that we thought were the best fit to optimise collaborative work. We decided that Github, intellij, discord and google drive would be the best tools to aid our game development. For the following reasons:

GitHub is the platform we chose because it provides a convenient way for developers to host and share our code, track changes, and collaborate with each other on our project. It also offers a variety of tools for managing and reviewing code, such as pull requests and issue tracking. Additionally, GitHub is widely used in the industry, making it a popular choice for developers to share and discover open-source projects.

IntelliJ is the IDE that we chose as it provides advanced debugging features, such as the ability to set breakpoints, step through code, and inspect specific variables. This makes it easier to fix bugs in code. IntelliJ also provides intelligent code assistance, including code completion, error highlighting, and refactoring. This helped write code more efficiently and with fewer errors.

Another reason why we chose IntelliJ is that it is available for Windows, Mac, and Linux, making it accessible for the whole team to collaborate on the different operating systems. IntelliJ IDEA has a large and active community of users and developers, which means support is readily available when a problem arises.

We chose IntelliJ over VS Code as it is a more powerful and feature-rich IDE that is best suited for larger scale Java projects, while VS Code is a lightweight IDE and is more suited to other projects.

Discord was our method of communication outside of group meetings. Discord servers are divided into channels that are organised by topic or purpose which makes it easier to find information. Whereas our team did not select Slack as Slack channels are organised by team or project. This would have not been utilised as we did not have multiple projects but we had multiple topics ongoing. Also Discord has better calling facilities compared to Slack.

Discord is designed to help teams collaborate and stay organised by allowing us to create channels for specific projects or topics. For example, we created a channel about the implementation. Team members can also send direct messages to one another and make calls or start video chats. A useful feature was screen sharing as this allowed us to easily work on code and mention specific parts of the project without confusion. Discord also allows users to receive notifications for specific channels or mentions, which means that team members will be alerted when there is new activity or important information that they need to know. Discord allows integration with third-party tools such as GitHub, which can be useful for version control.

Google Drive is a useful tool for a Java coding project as it allows for cloud-based file storage, real-time collaboration, version history, access control, and integration with other tools. It is also a good complement to other tools like GitHub to improve the collaboration of our Java game.

Google Drive provides multiple users to edit and collaborate on a document in real-time. This makes it easy for all team members to work together on our game, even when we were not in the same physical location.

We used Google Drive over DropBox as Google Drive can be integrated with a variety of Google Suite apps, such as Google Docs, Sheets, and Slides, which can be useful for project management, documentation, and collaboration. Dropbox also offers integration with third-party apps, but the list of apps is more limited than Google Drive. Also every member within the team has a Drive set up that was readily available to use due to our York university accounts.

All of the tools above aided our collaboration on our game development as we looked for the following attributes within these tools:

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

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

3) Cloud-based file storage: A cloud-based file storage platform like Google Drive or Dropbox allows team members to access project-related files and documents from anywhere with an internet connection.

4) Project management: Can help keep track of tasks, deadlines, and progress, and keep team members informed of the overall project status.

5) 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.

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

7) 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 prior software experience and group experience to determine where each member would fall in which role. Furthermore, we had to have a discussion about the roles we would need in the lifecycle of the project.

During the discussion, the notable roles for members to play were: Meeting Chair, Secretary, Librarian, and Report Editor. The purpose of the Meeting Chair is to ensure that the group remains on task during our meetings and to assign tasks at the end of the meetings. The purpose of the secretary is to record decisions made in meetings for future use through meeting minutes. The librarian's role is to keep documents and other resources and to oversee version control. The purpose of the report editor is to oversee and organise the production of reports. Some roles were divided between two teammates.

These roles were divided up to the team based on individuals strengths. To identify their strengths, each team member wrote down three of their strengths and then as a group we decided who would fit which role. Ensuring that we consider each of the team members preferences.

We also incorporated the use of a shadow into our project planning, where within each role there would be an accompanying shadow of the same position to ensure that if the original person assigned became unavailable to complete the role there would be someone already up to speed and with the knowledge of what to do that can step in, to prevent any delays in the project. Furthermore, for every task assigned during the project, a shadow member was assigned to make sure the work was completed even if the originally assigned member couldn’t do it on time.

Role:	**Skills:**	**Assigned:**	**Shadow:**
Secretary	Organisation, Planning, Communication	Alana	Jack
Meeting Chair	Problem solving, Motivated, Organised	Sam	Alana
Librarian	Problem solving, Listening, Adaptability	Faran	Galin
Report Editor	Problem solving, Communication, Dedication	Aisyah	Galin

Figure 1 : Role Assignment Table

Commencing the project, it was decided that all members would contribute to the code. However, in practice, we soon realised that it would benefit the team for only a few select members to work on the code, while the rest dedicated their time to the deliverables. This was because, in order for all the members to be involved in the coding, tasks were split up incredibly small, which resulted in some tasks relying on the completion of tasks by other

team members. This violated one of the risks in the risk register, R_PROJECT_06. Therefore, the owner, Faran, decided it would be best to only allow three members of the team to code. The split was decided by preference and experience. At this point, Faran, Galin and Sam continued with the code and Alana, Jack and Aisyah continued with the deliverables.

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	● Conduct interviews	21/11/2022	21/11/2022
	● Research for requirements	24/11/2022	26/11/2022
	● Write up requirements	26/11/2022	27/11/2022
Architecture	● Research into architectural designs	28/11/2022	16/01/2023
	● Responsibility Driven Design	28/11/2022	28/11/2022
	● Structural Diagrams and Behavioural Diagrams	29/11/2022	22/01/2023
	● Justification of architecture	28/11/2022	26/01/2023
Risk Assessment	● Research into appropriate ways to demonstrate a risk register	24/11/2022	25/11/2022
	● Create risk register	25/11/2022	01/12/2022
Implementation	● Code	05/12/2022	31/01/2023
	● List of 3rd party libraries and assets	23/01/2023	30/01/2023
Method and Planning	● Software engineering methods	23/01/2022	29/01/2023
	● Team organisation	23/01/2022	29/01/2023
	● Systematic plan for the project	23/01/2022	29/02/2023
Testing	● Against the requirements	20/01/2023	29/01/2023
Website	● Create website	27/11/2022	27/11/2022
	● Add the necessary details for the website	27/11/2022	31/01/2023

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 autumn term 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 Christmas and New Year's. Considering these aspects, we can develop an initial plan. Seen here under the heading first iteration.

The initial plan worked as follows: complete a draft of the requirements and risk management simultaneously by splitting up tasks amongst the group, then move on to the Architecture of the software and complete a draft before starting to code. During the period of 05/12/2022 to 06/01/2023 we would complete the development of the code in its entirety, as well as write a first draft of the Method and Planning as much as we could at that point in time. In the New Year, test and ensure that each of the requirements has been met. Finalise copies of the deliverables so that they are ready to be submitted. This plan can be seen here.

It detailed that we would first complete our client meeting as a team. From there, we can develop the requirements using the notes from the client meeting and the product brief. With a better understanding of how the life-cycle of the project is going to look, we can formulate a number of risks that could impact the development of the project while simultaneously completing the requirements. This is because requirements and risk register do not depend

on each other and can be completed independently. This part of 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 weekly plans in terms of timings. However, we felt it necessary to add extra detail to the chart. Such as specific tasks that were completed that week with the initials of the team members who were completing them.

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 start considering the software architecture and delve deeper into the classes needed for the system. During this part of the project, we prioritised generating the Sequence and Behavioural diagrams, thus ensuring that there is a detailed description of the project that the group can follow. This is because the implementation of the project was dependent on the completion of the UML diagrams. 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.

This was not in our initial plan; this is because the generation of the diagrams took longer than we expected. There were a lot of aspects of how the code would work that needed to be discussed, changed and constantly updated, such as how the chef will be controlled, where the verification of the correct recipe will take place in the code (with the cook or service station), and so on. As a result, the Architecture part of the initial plan overran, and therefore the report aspect was neglected to ensure that we did not fall further behind on the project. This is not reflected in the Gantt Charts (third iteration) however since the architecture and UML diagrams were technically complete and team member Megan Miles had written about the Responsibility Driven Design process during this time, which is reflected in the chart.

Next to be completed is the actual coding and development of the Piazza Picnic. Originally, we devised the plan such that each member of the team would contribute to the code. However, very early on, we realised that this was not feasible, and we would have to split the team into two parts: one part completing the code and the other part continuing with the deliverables. Therefore, more work can be done simultaneously. This correction was added to the Gnatt chart here where the fourth iteration displays every member contributing to the code and then the fifth iteration shows that only half of the team are continuing with the code.

Unfortunately, in this rendition of the plan (fifth iteration), we concluded that we would like the prototype to be completed by 15/01/2022 which did not come true. Due to other commitments (other modules and exams), there was a stagnation in the development of the code and the deliverables. As a result, the plan needed further editing. These can be seen here. The difference is that we had less time to ensure that the report flowed properly and that the code was well documented. However, the time added at the end was there in the event that we fell behind and needed extra time to complete certain tasks.

According to the project's final plan (sixth iteration), all ongoing tasks (whether coding or report writing) must be completed by January 23 before our meeting. During this meeting, we would tidy up the website, adding all the necessary information and ensuring that it was readable. At the end of the meeting, we would assign each person a deliverable that they had no involvement in writing. Then over the weekend, each person would read over the deliverable to ensure that it met the needs of the project. Thus, in the succeeding meeting on 30/02/23 we can then discuss any necessary additions or edits.

On 13/02/2023, we took over this project and initially laid out a plan of what tasks needed to be completed, the priorities and dependencies of these when the tasks need to be completed by. 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	● Analysis and adjust previous requirements document	17/02/2023	03/03/2023
	● Draft new requirements	24/02/2023	03/03/2023
	● Formalise new requirement deliverable	03/03/2023	10/03/2023
Architecture	● Analysis and adjust previous architecture documents	17/03/2023	21/04/2023
	● Add new architecture diagrams for new features	21/03/2023	28/03/2023
Risk Assessment	● Analysis and adjust previous risk document	24/02/2023	03/03/2023
	● Add new risks to risk register	24/02/2023	03/03/2023
	● Formalise risk document	03/03/2023	06/03/2023
Implementation	● Analysis of inherited code	03/03/2023	17/03/2023
	● Add all missing requirements from assessment 1	17/03/2023	14/04/2023
	● Implement new assessment 2 requirements	14/04/2023	28/04/2023
	● List of 3rd party libraries and assets	28/04/2023	02/05/2023
Method and Planning	● Analysis and adjust previous planning document	17/02/2023	28/02/2023
	● Create systematic plan for the project	28/02/2023	03/03/2023
Testing	● Write a test plan	03/03/2023	17/03/2023
	● Carry out testing	17/03/2023	28/04/2023
Website	● Analysis and adjust previous website	17/02/2023	28/04/2023
	● Add the new necessary details for the website	28/04/2023	02/05/2023
Change Report	● Write change for Requirements document	21/04/2023	28/04/2023
	● Write change for Requirements document	21/04/2023	28/04/2023
	● Write change for Requirements document	21/04/2023	28/04/2023
	● Write change for Requirements document	21/04/2023	28/04/2023
Continuous Integration	● Implement CI into code	17/03/2023	21/04/2023
	● Formalise a report on outcome of CI	21/04/2023	28/04/2023

Our first task when initially taking over this project from the other team was to go through each of their deliverables to understand and familiarise ourselves with them. We then had to make all necessary changes to these deliverables to adapt them to our team and the next stage of the project. The analysis of these documents can be worked on simultaneously as none of them depend on each other.

However, it was essential that these tasks were completed first as for all the other tasks to be completed, it depended on the group's understanding of the project we had overtaken as well having the relevant deliverables to refer to and use.

It was essential to our project to create and carry out a testing plan. This was to ensure that the code we inherited worked fully with no bugs, that any code we add to the inherited repository didn't break any of the logic already in place and all the code we were adding did not have any bugs. Once we had this in place we could move onto looking at the code.

The next task was to analyse the inherited code and compare this to the requirements established by the original team and the initial product brief. This showed us parts of their implementation were missing/incomplete and hence had to be implemented before we could code any of the new requirements.

In order for us to make any progress with new elements of the game. This is the creation of the new architecture diagrams for the new features of the game. This is a task that was dependent on the completion of understanding and adapting the previous deliverables regarding the requirements, architecture and implementation. This task is essential to being able to actually start coding the new features of the game and must be in place before any code can start to be written.

Once this had been completed and we felt our implementation was now up to a standard where we felt our project had all of the game features to satisfy the assessment 1 criteria. As well as we now had the architecture in place, we could move onto assessment 2. Using our new requirements table and architecture diagrams, we knew exactly what had been implemented and what we still had left to do, hence we could move into the actual code.

Furthermore as we begin to code, simultaneously we have to implement continuous integration (CI) into our coding. It was essential for us to implement CI practises into GitHub when coding our game as it helps to ensure code quality, facilitates collaboration and reduces risks. This task is not dependent on the completion of previous tasks but is essential to be completed at the same time as our implementation as CI will ensure that the game’s codebase is continuously tested and validated using our test plan. This is important to automate the testing process and catch all potential bugs early.

Finally, the task of the website is partly dependent on other tasks. The website we inherited is required to be updated throughout the project and once all the deliverables have been finalised placed onto the website. The updating of the website can take place alongside all of the other tasks and will run the duration of the project. However, another requirement is that all the generated deliverables are placed on it and therefore these need to be produced in order for them to be put onto the website.

The order of our tasks, or number of tasks did not change through the project. However, the duration of different tasks changed over the course of the project. The full changes of the plan can be seen here https://eng-25.github.io/assessment2.html#gantt. With the final gantt chart showing exactly how long each task had taken, including the different start and end dates.