ENG1 Group 9
Method Selection And Planning

Cameron Fox
Liam Burnand
Jack Cameron
Eoin O’Connor
Adam Penny
Ben Brown

PART A

Plan Driven Method

At the beginning of the project development cycle, we agreed as a team that we would be using a plan-driven method to bring 'Piazza Panic' to life. A plan-driven method would allow us to discuss, document and prepare for all aspects of the game we create. PDM's also give us a comprehensive and thorough roadmap from start to finish of what is needed to be done and on what time scale. This comes at the cost of being heavily bureaucratic. Only one of us, Liam, had worked on a collaborative software development project before, so it would be better to over-plan than under-plan.

We thought it would be most appropriate for us to use the 'Jackson System Development' plan-driven method. We liked the 3 obvious phases it contained and believed this would provide a simple yet effective roadmap from start to finish. This starts with the modelling phase in which we had to identify the entities in the system, the actions they can perform and the attributes of each entity. The initial diagrams from the modelling phase were later integrated into the 'Architecture' document.

The specification phase was next. This involved breaking down the project into key tasks which can be shown in 4 c). We then came up with timelines of when we expected each task to be completed by. As the project progressed, the team leaders would assign individuals to specific tasks week-by-week at our in person meetings. We would hold scrums at each meeting to discuss the progress that had been made and update the plan when we needed to.

The implementation phase is final. This is when we had to figure out how to achieve the required functionality and then actually implement it. We all looked into libGDX - the java library we were using - and watched basic tutorials on using that library. Analysing the entity structure diagrams and learning the basics of libGDX was all we needed to begin the implementation of the functionality.

Agile Methods

Agile methods can vary, but are all largely based on some of the following principles defined in the 'Manifesto for Agile Software Development': continuous delivery of working software; work with volatile requirements; customers play a role in working with the developers; value face-to-face communications; self-organise responsibilities and reflect on team performance. The project will not be volatile and although we will be interviewing our client, we will not be working closely with them. The task is well defined and static in nature. Therefore, strictly using an agile method would be redundant.

However, we acknowledge that some of these principles will be paramount in ensuring success. For example, as a group we will have to organise into teams and evenly spread out the tasks between us all, ensuring to self-reflect and critique as the project progresses. We also recognise the value of face-to-face interactions and want to maintain a high attendance rating at each Wednesday meeting. We will therefore be

taking on board some principles from agile development but primarily be using a plan-driven method.

Scrums

At every face-to-face meeting we held a scrum to discuss what was done in the previous week, what needed to be done in the following week, and whether or not we were on schedule. This is how we implemented the self-reflection and critiquing principles of 'Agile Software Development'. We could also update the plan as the project developed to ensure it reflected what progress had been made. The notes from our scrums are displayed in the plan. Over the Christmas period we decided not to hold weekly meetings as the project was ahead of schedule and we felt it would be more appropriate for people to focus more of their time and energy on the January assessment period. Work was still set for people to complete during the break, but it only needed to be completed for the first meeting back on the 18th of January. One discord meeting and scrum was still held just before Christmas day.

Development and Collaboration Tools

Github was our primary collaborative and development tool for the 'Game' key task. We felt GitHub was most appropriate as it allows for simultaneous editing, keeps track of previous versions of software, and most people in our team were familiar with it. It also keeps track of who makes edits and when the edit was made. This is useful for tracking the progress of development and updating our plan when need be.

libGDX was the java library we used to develop the game. We felt this was fit for use as there is a plethora of information, tutorials and documentation online in relation to libGDX. There is also an active community of users with new information and help being talked about daily.

For communication we used Discord and Snapchat. Discord was useful for hosting any online meetings in which we could discuss issues we found with certain tasks and elicit help from our peers. Snapchat was used for more general communication. Snapchat group chats have a simpler interface than Discord's mobile app so for more general communication such as informing the other members if we would be late to a meeting or getting small questions answered from the comfort of our phones was handy.

Google Drive was also used to provide simultaneous document editing and sharing. Everyone was familiar with Google Drive and like GitHub, we could use it to keep track of previous versions of documents and see who edited what and when. Our University of York emails can be used for Google Drive, so no sign-up is necessary.

PART B

During the forming stage of team performance, we spent time getting to know one another and equally dividing the responsibilities between us all. Once we knew each other a bit better and had devised the initial plan, we thought it would be a good idea to go through everyone's strengths and weaknesses so we could assign everyone the appropriate roles.

Name	Strength	Weakness
Liam	Strongest coder	Less confident writing skills
Ben	Strongest writer	Not as strong a coder as others
Cameron	Graphic design experience	Weaker literacy skills than coding skills
Eoin	Organised	Must commute from Leeds to meet face-to-face
Jack	Strong literacy skills	Not as strong a coder as others
Adam	Strong at both coding and literacy skills	Quieter than other members

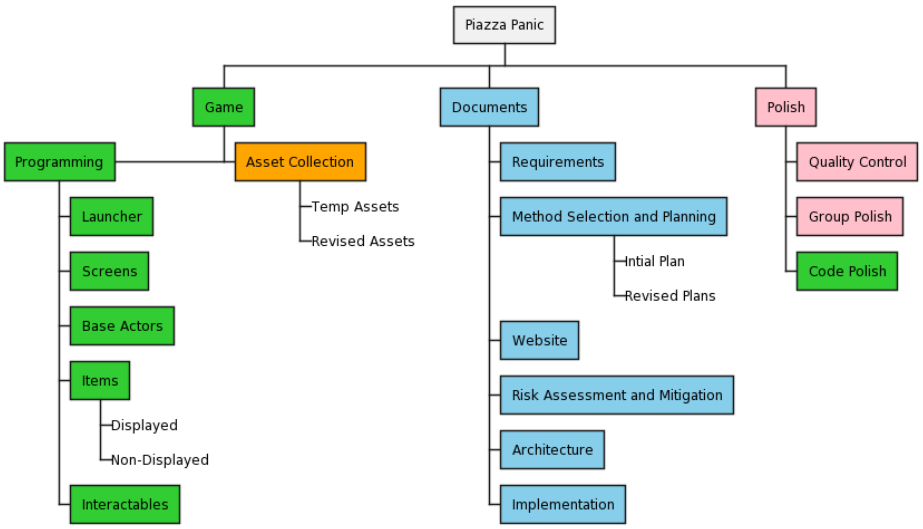
It was clear that the two biggest key tasks were game development and document writing. We therefore split the team in two and assigned each key task a team leader. The team leader would remain constant throughout the entirety of the project. The individual workers could change between teams depending on what work there was to be done at different stages of development. Roles were also assigned based on what we thought was most appropriate given people's characteristics.

Name	Role	Team
Liam	**Game Development Leader** – Oversees all work on the development side. Responsible for delegating programming tasks.	Game
Cameron	**Head of Graphics** - Responsible for graphics in game as well as the written Implementation deliverable, and coordinating the UML diagram ensuring it reflects the state of the code.	Game
Ben	**Document Writing Leader** – Oversees all work on document deliverables. Responsible for delegating writing tasks.	Document
Jack	**Risk Assessor –** In charge of conducting the risk assessment and ensuring team members are aware of risks and challenges present.	Document
Adam	**Quality Control** - Ensures correct grammar, spelling and vocabulary used. Checks for redundant code/inefficiencies. Makes suggestions and criticisms for improvements.	Document
Eoin	**Planner** – Keeps the plan up to date. Ensures both teams are following the plan. Updates plan when necessary.	Document

This team organisation is most appropriate as it plays to our individual strengths. The document deliverables are also worth 70% of the marks available, so that is why more people are on the 'Document' team than the 'Game' one. The in person scrums we held ensured that everyone was aware of the progress that was being made.

PART C

We have broken our assessment down to 3 key tasks: ‘Game’, ‘Documents’ and ‘Polish’. All key tasks have then been further divided into subtasks.

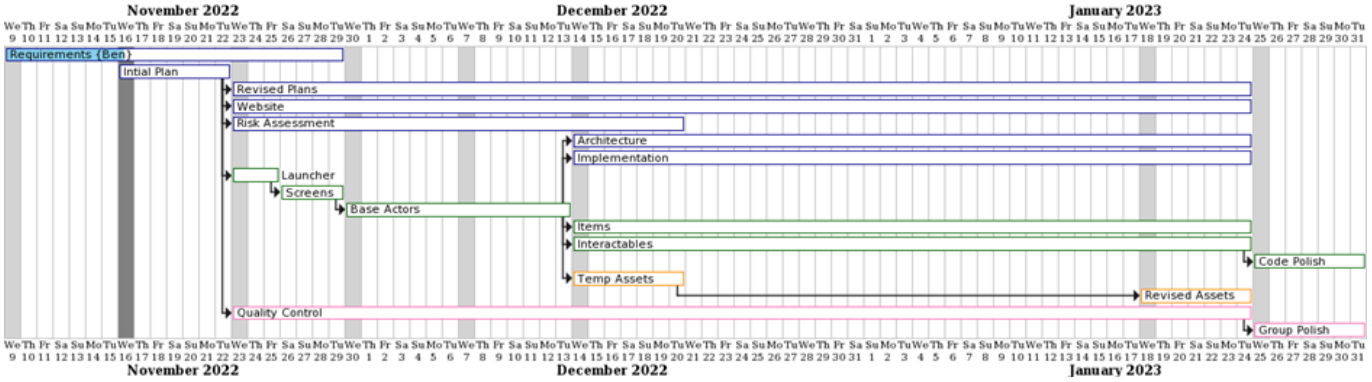


The ‘Game’ key task consists of the ‘Piazza Panic’ programming and assets. We decided that programming took a higher priority than asset collection. Therefore, we are using temporary assets to begin with. These would be revised later in the project's development. As you can see from our Gantt charts below, the ‘Base Actors’, ‘Items’ and ‘Interactables’ subtasks are dependent on the ‘Screens’ subtask being completed, which is itself dependent on finishing the ‘Launcher’ subtask.

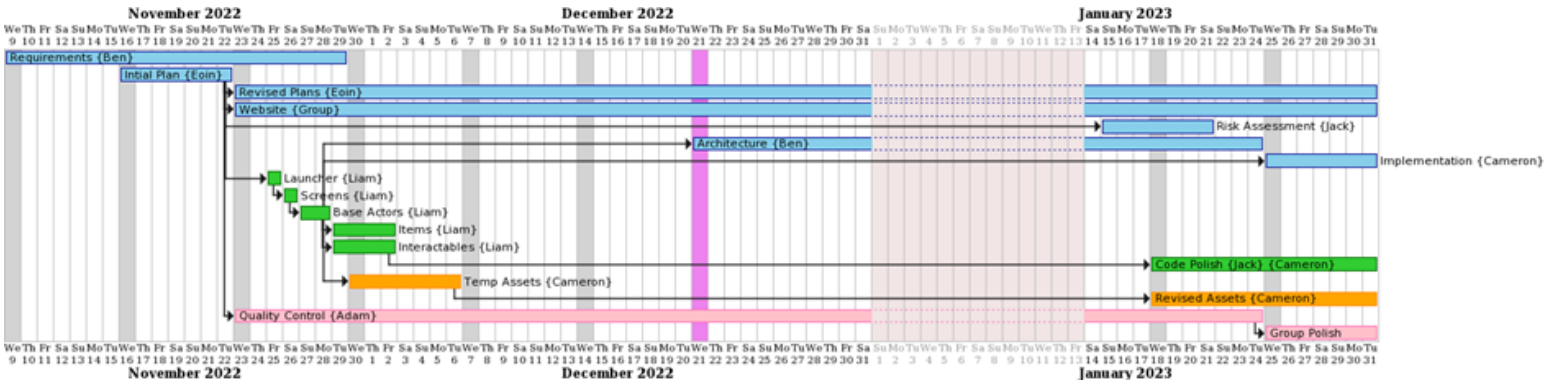
The ‘Documents’ task consists of the deliverable documents we must produce. The ‘Requirements’ sub-task must be started first. This is so we can get a feel for what the client wants and discuss with ourselves what is needed. Once we have started the ‘Requirements’, we can put together an initial plan. The plan will evolve alongside the project. The ‘Website’ and ‘Risk Assessment’ are dependent on the initial plan being completed. The first iteration of the plan is important to use as a jumping-off point. ‘Architecture’ and ‘Implementation’ are dependent on the ‘Base Actors’ subtask. We felt it would be redundant to plan the architecture attempting the code as it's easier to figure out what things the game needs by actually trying to practically tackle the problem. Planning too much of the code in advance would lead to certain aspects being overlooked or many changes being made to the architecture.

The final polishing is going to be completed last. As outlined in part b), Adam oversees quality control during the project. He will have the ongoing task of checking pieces of work for potential errors and improvements. Once the whole project is nearing completion, every group member will carry out proofreading and we will collectively contribute towards polishing.

Initial Gantt Chart (16/11/22):



Final Gantt Chart (01/02/23):



How did the plan evolve?

We originally overestimated how long the game development tasks were going to take. Liam began the work on the game and ended up getting the bulk of it completed in just over a week. We all agreed it would be unfair for him to do any more work. He would remain as 'Game' team leader but would take more of a backseat role going forward.

We simultaneously underestimated how quickly we could get the document writing completed. We were met with many delays in the document writing process. Ben was busy with external work commitments, so 'Architecture' was started a week late. Jack was behind on some of his uni work, so 'Risk Assessment' was delayed until after the assessment period.

Cameron needed an extra week to finish the revised asset collection as he was also busy helping polish the code and some of the document work. Group polish was also pushed back as some of the deliverables were not finished a week in advance.

The Discord group meeting, represented by the purple day, and exam period break were also not originally planned for. We wanted to meet at least once over the Christmas period to track peoples progress. The exam break also seemed necessary as it wasn't realistic to expect people to work on the project during this period.