Method Selection & Planning

Assessment 2

Team 10: Hard G For GIFs

Dragos Stoican

Rhys Milling

Samuel Plane

Quentin Rothman

Bowen Lyu

Change Report

Part A: Method and Tool Selection

For our project we decided to use the Scrum method as we are delivering a small piece of software with a small team, and it is essential for us to have a flexible working environment due to team members' different schedules. Scrum is an agile method, and we chose agile because there may be changing requirements, and we believe it is crucial to have constant team reflection and development. There are certain aspects of Scrum that we chose to change to better adapt to our team organisations, which we'll touch upon later.

Scrum involves dividing development into 'sprints', each of which are a week long and preceded by a planning meeting. At the end of each 'sprint' and often much more often, there is a team review of what everyone has done and any problems they came across. "Scrum employs an iterative, incremental approach to optimize predictability and control risk."[2]

To aid development, we took advantage of a couple of tools:

- Git, provided version control, the branching functionality allowed features to be developed separately from the stable codebase, then after being safely implemented could be merged back. The commit history is also useful as it made all committed code still accessible even after being deleted. An alternative could be Mercury, which does have a simpler user interface, git has also become ubiquitous in the industry, and provided a better real-world experience.
- GitHub offers git as a base, but it adds a lot more functionality that makes software development easier. One feature would be the issue system, as it allowed us to convert our system requirements into a trackable object, where we could comment on it, assign it different statuses so we could see how complete the issue is. This integrates excellently with GitHub projects which allows you to put those issues on boards, so we could visualise how each issue has been progressing, who has been assigned, how old, etc. An alternative could be Trello, which does have the card system and lets you transfer it between boards also, but we decided not to adopt it since GitHub allows a much tighter integration with the code, and being able to reference directly and with some provided assistance is very useful.
- Google Drive, provided collaborative document editing, this was particularly useful since we often did paired work. Also, the version history is useful for being able to retrieve previous versions of work. There are alternatives like Office 365 / OneDrive, but this wouldn't probably wouldn't work since our University accounts only provided the basic tier and we needed to be able to share with the lecturers.
- GitHub Pages, which allows us to host our own website for a more public view of our project, without the lengthy and hard process of making our own website.

Our main tool of communication was Discord. It was a tool some of us were already familiar with and filled all the jobs we required it to. We were able to use it to track progress and talk to other members of the team regularly. We also did use Zoom when the meetings happened on our time table, but Discord is what we used most of the time. Our server has specific channels for planning meetings, listing tasks and talking about organisation.

Eclipse and Intellij IDEA were the main programs we used for coding our project .While each team member had their own preferences on which to use, they all offered a wide range of

tools and features that would serve to help us out in making our project. We all had experience with Java programming, but none of us really had much experience with LibGDX, so having this program to help us along the way was a nice tool to have.

There were also some other more minor programs that weren't used a lot or by each team member, but that still played an important part in the project.

- The first one is PlantUML, a handy tool that we used for several UML Diagrams for the Architecture document, as well as Gantt charts. It's essentially a markdown language for UML.
- For the graphics side of our project, we used CS6 Adobe Photoshop for most if not all of the assets we created, using the programs wide variety of features to make sure we delivered good quality. Paired with this, we used TexturePackerGUI for when we had animated assets.

Part B: Team Organisation Approach

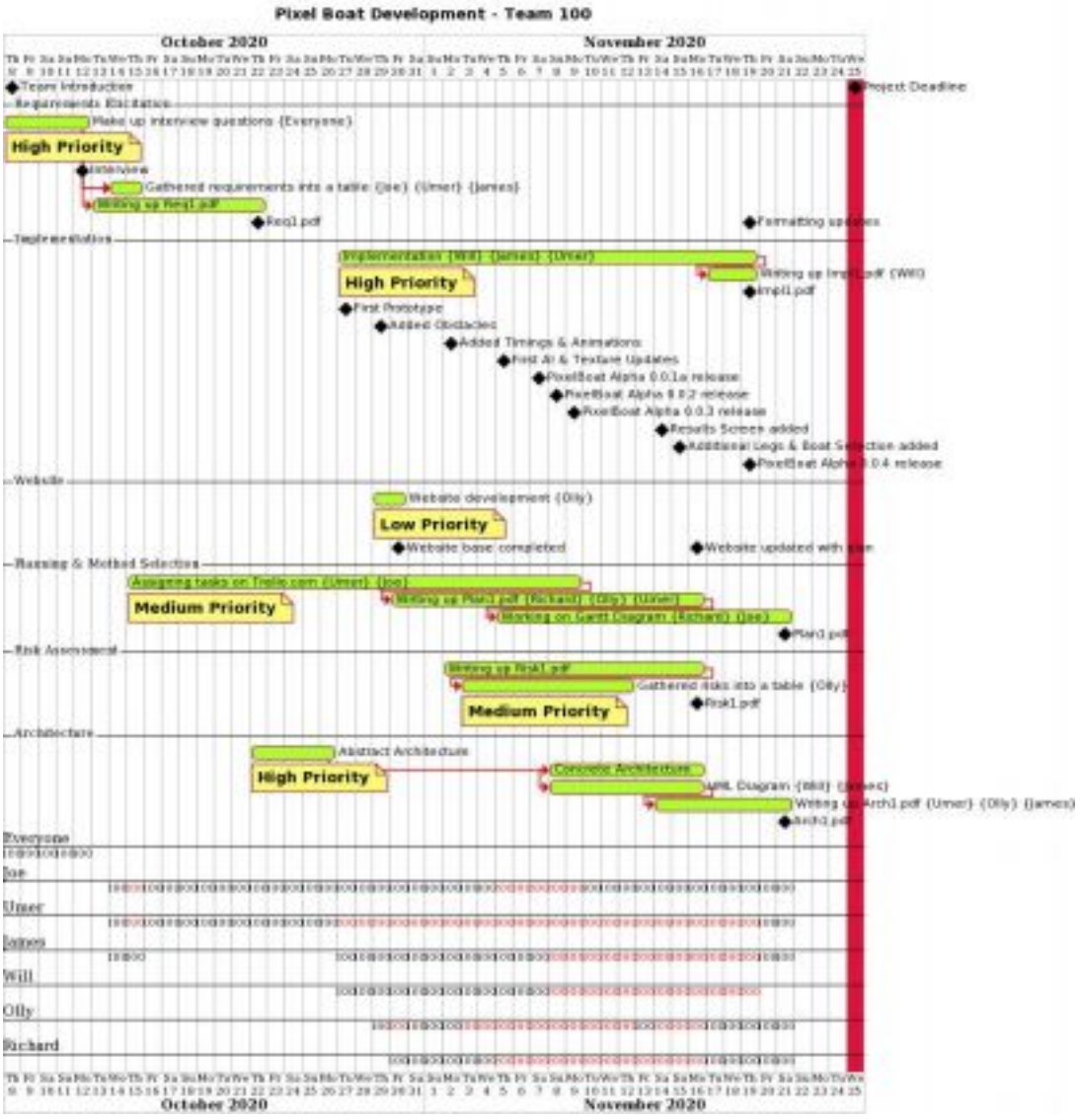
Our approach remained the same, the SCRUM approach, but not the official to the letter SCRUM. In a certain way, we adapted it to better suit our team and personalities. For example, while we were advised to have a team leader, we never had a vote or appointed one, as it just never came up. Our team worked together fairly well from the beginning and we all did what we were supposed to, and all gave in regular updates to our work with regular meetings. We did give some roles, for example Quentin would usually be the meeting leader and guide them along, and Sam would be the one to record what we did in the meeting and note it down.

We would have regular meetings, every Thursday and usually on Mondays, however these would increase as time moved on and so did urgency. At each meeting, we'd sit down and take a look at what was done, what still needed to be done, and who could do what. Tasks would be given out and each person would keep track of their own tasks by noting it down in a discord channel.

Our approach to using SCRUM may not seem appropriate to people out the team, however it proved to be more than effective. It relied on the team members being able to trust each other and communicate effectively, which we did as best as we could. The project is small, but we are also a small team with many different parts. Breaking this project down into smaller problems and solving them one at a time worked well.

Doing it this way allowed our team to focus on their areas of expertise, or where they wanted to improve. But due to our lower numbers, everyone had to participate in each aspect at least in some way, so we all made some code as well as helping on the documentation, but we did it as a team, by communicating and not being a liability to each other.

Part C: Project Plan



Gantt Diagram of the project

Plan Strategy & Outcome Review

Our plan remained fairly consistent throughout the project and each meeting following. After having finished Assessment 1, we all got some well deserved time off, but once the next steps were given, we chose to not sit on our hands too long and to try and get a headstart, although since it was still the vacation and we obviously wanted to enjoy them, we gave a small amount of tasks and we chose to take it slowly. Our main concern was also to balance this and the exams, choosing to start applying ourselves more seriously after we were all done. Once the exams were up, a meeting was held to plan out how we were going to spend the rest of our weeks until submission date and what we were going to do.

Key Events List

These key events originate from the meeting logs we have on our Discord, as well as the Gantt Chart. Highest priority events were the new requirements to be implemented, as well as the new documentation. The critical path of our project was Requirements reassessment,

Change Report, Risk Assessment, Planning, Continuous Integration, Architecture, and finally, Implementation.

First Week of Vacation:

- We had our first meeting post-Term 1 final, in which we first had a bit of a chat of our thoughts about the previous assessment and how it went
- Then, we began talking about the work we would do during the holidays as to start ahead
- We chose to begin our focus mainly on the old essential requirements that the previous team failed to implement, while also working on the new requirements of the Save & Load game on the side.
- We also emailed a few questions to our module leader to make sure we understood all the new elements of the assessment
- These tasks were to be completed slowly, as there was no rush and we wanted to enjoy our vacation time.

Fourth Week of Vacation:

- Our second meeting during the vacation. This was more of a check on the progress each person made on their tasks.
- We would again plan for the next meeting, after the exams, and give more tasks if people were done with theirs

Second and Third week of term:

- This is where we started shifting into a faster gear. We started to take a look at documentation, mainly their old ones.
- Like last term, we assigned teams to each new document that needed to be made.
- The main coder of our team would focus on making the new requirements and polishing the old ones that needed to be done.
- Others would focus on having a look at the old documents and updating them, giving them a fresh perspective to match our vision and the current assessment
- We also began to look into testing

Fourth week of term:

- Our developers had made great progress towards making tests, as it took significantly less time to make them than anticipated. This meant that we were able to thoroughly test the game and solve the issues that were had.
- Most of the important requirements were implemented, so now the name of the game was polishing and debugging, to make our product as complete as we could
- This was also the time where we began the website revamp.
- Documentation had also made great progress, with some already being done and others being close to.

Fifth week of term:

- All tests were done, and all important requirements were finished, so the developers focused on implementing missing features that were simply an extra, and improving other areas that could use it, while also working on bugs.
- The rest of the team finished up any documentation that needed to be done.
- The website was also finished up.
- The presentation was also scripted, recorded and edited within the week.

Bibliography

[1] I. Sommerville, Software Engineering, Pearson Education, 2008, pp. 74-98. [2] K. Schwaber and J. Sutherland, The Scrum Guide™, 2017 pp. 1-19