SEPR 2019/20 Assessment 1

Team CheatCodez

Jonathan Grout, George Lesbirel, Cheuk Wang Wu, Lauren Quarshie, Muaz Atif, Lillian Coultas

Method Selection and Planning

Software engineering method and tools

As a team we decided to use the Scrum methodology as our approach to the project, it is a type of agile development and has several advantages over other methodologies which make it suitable for our project. The major advantage of Scrum is the amount of work done by the team is managed and timed, we have the ability to set a given amount of tasks to each person to do per period of time (sprint). As part of the sprints, we will be iterating over our game and project report; we will have the flexibility to review and change documentation and work on the game simultaneously. To do this, we are using a piece of software called Jira to help with the organisation of tasks and sprints. Scrum uses sprints, set periods of time to work on tasks and deliver an iteration of the product. As a team, we decided that working on weekly sprints was suitable due to the length of time between each deliverable. By following the Scrum approach, we will conduct regular meetings reviewing how the sprint has gone and discussing what should be done in the next sprint.

We decided to steer away from a more traditional approach like the waterfall methodology, made popular by Winston W. Royce for a number of reasons. The waterfall approach follows a very strict layout of what needs to be done, it doesn't provide the fluidity that we need in this project to be able to react to changes in the requirements of the project or other matters, this is something that Scrum bypasses. Waterfall also doesn't give a proper measure of progress, with sprints we can clearly see what has/hasn't been completed. This is theoretical and practical, using pieces of software we can visually track our progress and measure each team members input when using the Scrum methodology. Another disadvantage to waterfall would be it's lack of being able to adjust if the output is incorrect at a later stage due to it not being clear earlier in the project. Even though the waterfall approach would allow us to have well understood milestones. For example, we would clearly know when we have completed all testing. Regardless, using Scrum, allows us to test alongside the development. So there is no need for another stage which would cost and time.

The main concept of the Scrum methodology comes in at the "Review Draft" decision in the center of the image. This is when we will host the sprint Reviews, if there are changes that need to be made, then we will make tasks and move them to the next sprint. Within the changes cycle we will be able to work on all aspects of the project mentioned this is not possible in the traditional Waterfall model.

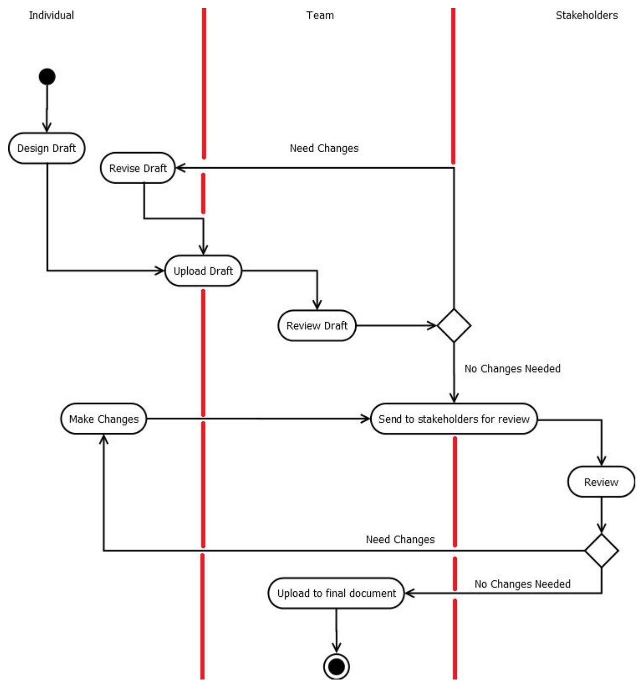
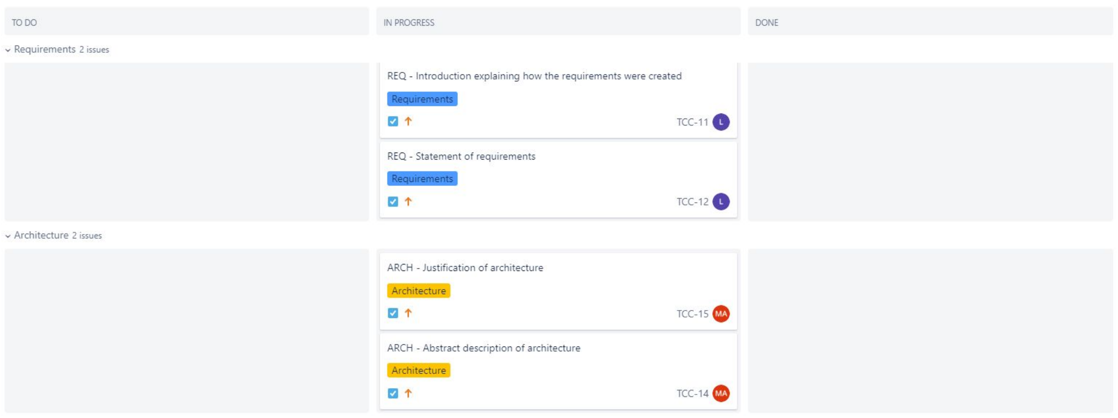


Figure 1.1, a diagram on how we will work using Kroy

Our choice of version control system is GitHub. This is mainly because of the decentralised approach. Each developer has a local directory which they can commit to online and offline. This allows each developer to maintain their feature(s) independently of others. Systems like Apache SVN use a centralised method, there is only one repository for all developers. This limits accessibility offline and doesn't promote small commits, like a bug fix for example. Our team members are all familiar with GitHub which eliminates any time being wasted on learning a new system.

Other tools we are using for collaboration include 'Slack'. 'Slack' allows users to communicate the details of the project to one another by providing different channels for every area of each assessment, for example one channel can be used to discuss the requirements and another for providing feedback on the website. Team members can see what others are working on in other channels and can provide feedback and advice where necessary. 'Slack' allows for excellent discussion on specific areas of the project, but where more general communication is needed 'WhatsApp' is more helpful. 'WhatsApp' allows for team members to quickly send messages to each others phones, this is helpful for organising team meetings or other small issues.

While 'Slack' and 'WhatsApp' are fantastic tools for discussion, they don't provide much value when trying to track what stages of the project have been completed or in progress, so in addition to those we also decided to also use 'Jira' as an online task management board. 'Jira' allows us to break down each stage of the project and assign each task to a different member of the team, this ensures that multiple tasks aren't undertaken by different people and provides a clear way of seeing what tasks are left to be completed before each assessment. 'Jira' is designed specifically for assisting with agile methods of development, which is perfect for helping us see the progress of our sprints. It also allows for version control, so we can review our progress during previous sprints should we need to. Overall 'Jira' is an excellent tool for managing each stage of the project and ensuring each team member knows what tasks they must work on.


TO DO | IN PROGRESS | DONE

Requirements 2 issues

REQ - Introduction explaining how the requirements were created
Requirements
TCC-11 L

REQ - Statement of requirements
Requirements
TCC-12 L

Architecture 2 issues

ARCH - Justification of architecture
Architecture
TCC-15 MA

ARCH - Abstract description of architecture
Architecture
TCC-14 MA

Figure 1.2, Jira "Active Sprint" page.

Team organisation

Great organisation is crucial for good project development. It is needed to make sure every team member knows exactly what tasks they need to complete, and what the goals for future development of the project are. Good planning will allow the team to ensure all of the stakeholder's requirements have been met. For these reasons, organisation plays a key part in the development of our project.

We have also allocated ourselves roles based on *Belbin* team roles model (*Appendix 3.1*):

Team Member	Primary Role	Secondary Role
George	Shaper	Implementer
Jonathan	Specialist	Monitor Evaluator
Lauren	Team Worker	Completer Finisher
Lillian	Co-ordinator	Team Worker
Muaz	Plant	Completer Finisher
Vincent	Implementer	Co-ordinator

Each member of the team has been designated a role, while some of the secondary roles may cross over the primary roles are all unique. The primary roles are defined as:

- Shaper - the team motivator.
- Specialist - provides specialist skills/knowledge to the team.
- Team worker - helps the team work together well.
- Co-ordinator - helps clarify goals and designate work.
- Plant - creative problem solver.
- Implementer - team strategist.

While these roles do not specify each member's technical ability, they do provide a good idea as to what each team member's strengths and weaknesses are. Knowing what areas of the project team members can work best in helps to aid with organisation by assigning each task to the most qualified person.

As part of our sprints we will also be conducting regular team meetings in person which will be recorded in our log book. In these meetings members will discuss tasks which have been completed, any issues that certain members are having, and we can plan what to do in the upcoming sprint. This will ensure that every team member knows what to do and will mitigate any confusion regarding assigned tasks.

Project Plan

In this section we will outline the key tasks for the project and more specific software engineering tasks for Assessment 2. We have planned out and identified the tasks for Assessments 2, 3 and 4 in a table and given them a description of what it entails. This will allow us an easy way to know what we have to do for each assessment.

When it comes to starting a new assessment, we will read through the tasks and allocate them to who suits each one best. We don’t want to assign tasks too early in the development progress, more specifically assessments 3 and 4, as people may find themselves better suited to a different task when it came to it. To help make these decisions easier, we will use the tables that outline the key tasks for that given assessment.

Plan for software development

This section is concerned with the software engineering design and implementation. As mentioned above, we have made a table *Appendix 3.3* which outlines the overall goals. We have also made a Gantt Chart, targeted specifically at the software elements. Overall, we have allocated ourselves 3 days of tolerance. Leaving 3 days at the end means we can either have 3 days of debugging, fixing or improving the game if it is complete. Otherwise, we will be able to delay parts of the project up to 3 days. Our Gantt chart has been made with only 3 tasks being completed simultaneously. This is because we don't want to have too much going on at any point in time, and 3 tasks split between us will be manageable.

Figure 1.3 shows our Gantt chart for developing the game. Each orange box represents one day of work that we think the task needs. The time allocated is a rough estimate based on the difficulty and complexity. For example, we have allocated a lot of time to graphics because we know that there will be a particular style that we want to meet and it will take time to match the graphics and make sure they sit well together. The other colours on the chart are representing the tolerance we have, e.g. latest starts and finishes. These have been grouped together for tasks that need to move with each other, or are dependant on one another. This will make it clearer when one task is delayed, we can see the space that we have available for the others when they need to be moved. A higher quality version is available in *appendix 3.2*.

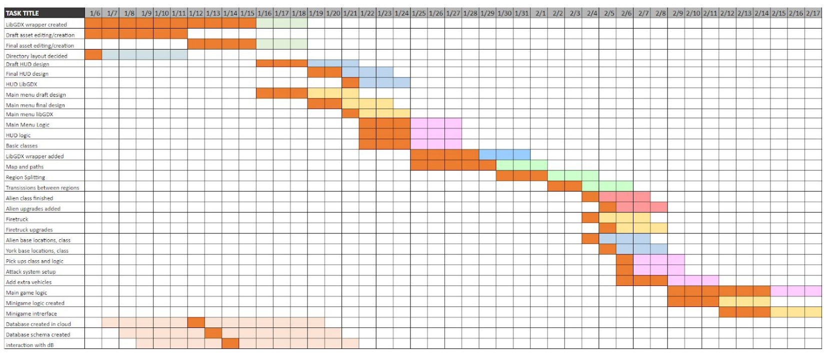


Figure 1.3 - Gantt chart showing essential tasks for game development