Software engineering methods

Development method

When choosing our development method, we took into consideration what is already being used by similar existing software development teams in industry. With this in mind, after group discussions, we chose to elect an agile method as “virtually all software products and apps are now developed using an agile approach” [1]. After viewing the agile methods manifesto, it was clear that an agile method would be suited to our project. One important aspect of this was that “business people and developers must work together daily throughout the project” [2], which was very similar to our project where we would maintain regular contact with the customers to ensure our software is what they expected.

Agile methods makes use of user stories where scenarios can be storyboarded and then broken up into tasks for the software to perform. These can then easily be organised into implementation tasks for developers to undertake. Storyboarding allows us to understand how the game will play out, especially through the use of prototypes like paper mock-ups, which also makes explaining the game to the customer much easier. Being an agile method, requirements for the software may change over time but the stories can easily be changed and reflected in the software being produced. Specific requirements to the game, like how many units spawn per turn or how often the PVC will spawn on the map, are likely to change after playtesting to make the game balanced.

Other methods were considered at the start of the project including plan driven methods like IBM’s RUP [3]. However, after some consideration it was decided that it would not be suitable because of the way a plan driven method works. Should any requirements change during development it can cause problems, having to alter various documents for example the cornerstone of the method, the development roadmap that is more unnecessary work. One strong point of the method is that it begins with storytelling, meaning we could easily plan and understand what we wanted the software to do. However, our decision to adopt scrum was largely down to its flexibility, the sprinting element of the method and of course its adoption in industry.

After being sure our team will adopt an agile method, our attention turned to which specific agile method we would adopt. Right away we were drawn towards adopting the Scrum method due to it being the “most widely used method” [1]. Scrum also seemed to suit how we would be approaching this project very well. Work is split up into sprints where each member of the small team knows what they should achieve by a given date at a meeting before the sprint, selected from the product backlog. After the sprint the team meets again to discuss what they have done and what should be done in following sprints. Importantly, tasks for each sprint are prioritised to ensure important features are added first. Sprints would normally last much longer than we will be deploying, typically they “are limited to one calendar month” [4], but we felt this is lasts far too long. Our sprints will last around a week to ensure they do not last too long and that tasks are not left too long to be completed. With our sprints lasting a shortened amount of time, the tasks set for each sprint will be much less demanding and time consuming, allowing for other studies and commitments to still be possible. We will also not be having daily scrum meetings as most uses of scrum would, but we will still hold regular meetings to maintain organisation and face-to-face communication. Should any problems arise between scrum meetings, each team member is contactable by email for our Facebook messenger group.

Scrum also works best when the team size is quite small, “having more than nine members requires too much coordination” [4]. Transparency within the team means that everyone knows what everyone else in the team is doing; being a small team makes this possible. The advantage to knowing what everyone is doing is that tasks are not done twice by two different members, wasting time. It also means that communication between team members is more efficient because they each member knows who to contact regarding any issues.

Development tools / Collaboration

Version Control

We needed somewhere to store the game as it is in its development phase, preferably where multiple people could work on separate sections of the game at the same time. Version control is therefore essential to this project and so a repository was required. After discussing our options, we settled on GitHub.

One benefit of using the Git VCS is that it is widely used in industry meaning there is a wealth of knowledge, both on how it works, as well as how to integrate it into effective software development methods. Another benefit is that Git supports branching. This means that team members can develop an entire feature, complete with commit history, without affecting the master branch.

File Sharing

Aside from the programme itself, we also required somewhere to store documents, preferably in the cloud to enable everyone to reach the documents. Services like Microsoft's OneDrive and Google Drive were considered.

We have setup a Google Team Drive that allows use to share documents, access the most recent copies of files and collaboratively work on the same document simultaneously. The benefits of using Google Drive for our team is that all documents can be centrally stored in a place meaning we do not need to chase up people for copies of documents and that we can provide realtime feedback on any documents. Google Drive also has the team drives feature where any in a team, like ours, can add and edit documents rather than have a document owner. The university also has the Google suite meaning we have unlimited storage and accounts already set up.

Communication

A Facebook Messenger group chat has been set up and is for the purpose of coordinating work, meetings and sprints. From the very beginning, our team has been and will be using Facebook messenger to communicate with each other. We use the group chat to coordinate meetings, raise any questions and discuss ideas when we are not meeting in person. We also have a group Discord that can be used for non-face to face group meetings where we can work and discuss elements of the project over VOIP or even just to talk to each other whilst working.

Team organisation & structure

Organisation

Organisation for any team project is essential to ensure that work is done at the correct times and to the highest quality. For organising our team, a Trello board was set up so we had a single, easy to understand area where organising could take place. We have a number of sections to our Trello board including a scrum backlog to keep track of what is to be done in the current sprint. We then have an area dedicated to work that is in progress where members can see who is doing what, and who they are working with. Once a task is complete, it can be ticked off the in progress list and added to the review list. This is where completed tasks are queued up to be reviewed by the team. This usually consists of the entire meeting in person or through Discord and going through completed tasks, checking and/or making small changes before the whole team deems that task is ready to be ticked off as fully complete and ready to be assessed.

When organising the team, it was clear we would need a central point for organisation, especially due to out adoption of scrums. A shared document would have sufficed but could very quickly become cluttered. Trello made more sense as it is basically a digital pin board where everything can be found in a quick glance instead of scrolling through weeks of notes. Trello also lets us set dates for tasks and record progress as it happened for tasks using checklists for subsections of tasks, useful for getting an overview on how far along other team members are with their own tasks.

Structure

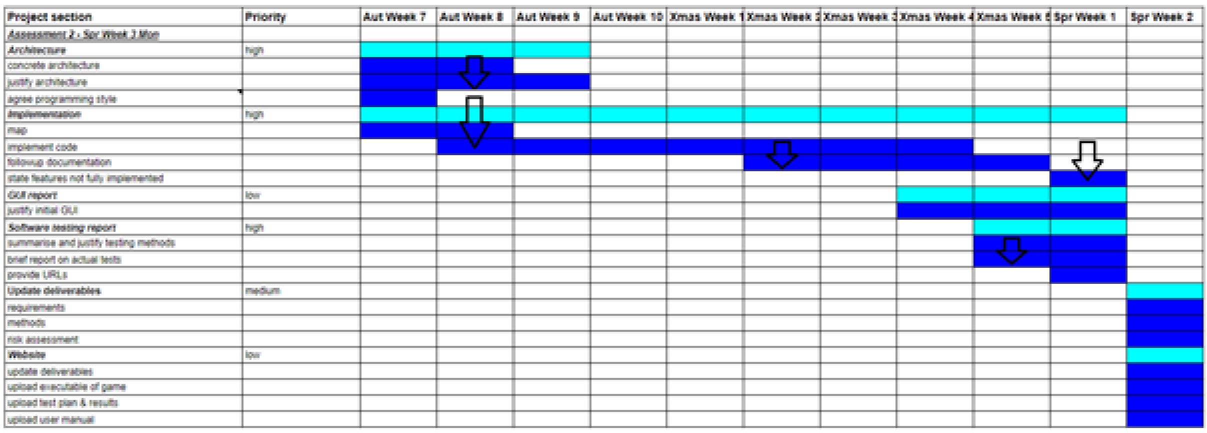
With our project following the Scrum development method, how our team is organised should also follow this method, forming a Scrum team [1]. The development team is self-organising, adhering to the agile methods principles [2]. Being a software engineering project, all team members will play some part on this team, members may be more experienced or capable than others but these members should help those who need it to ensure every member has a good understanding of the software being developed.

Continuing the adoption of the scrum team structure, the team requires a ScrumMaster [1] to be in charge of each Scrum. They check that the scrum process runs smoothly and guides the team in Scrum specific tasks, such as the daily scrum meeting. The ScrumMaster is not a project manager [1] but will be in charge of each scrum. We will be alternating which members are the ScrumMaster throughout the project to give everyone a chance at being in control.

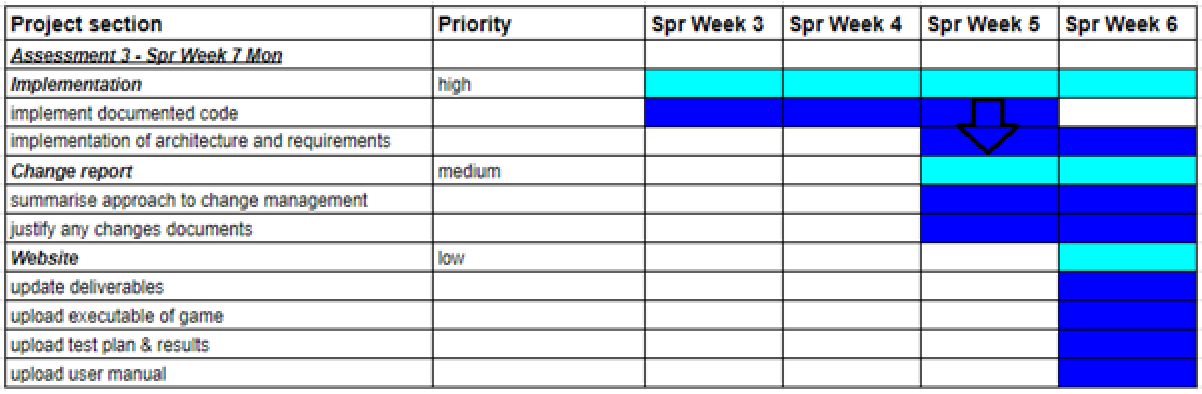
As for assigning roles in the team for a more long-term basis, we thought it best if we decide on specific team roles later into the project where everyone knows each other better so roles can be assigned based on individuals skills and how they work. Giving a role to a person who is not suitable for such role is counterproductive and wastes time when we have to adjust the team and relevant documents to reflect any changes made to the team structure. Assigning team member roles well after we have started the project does pose some risks. With no team roles assigned, members could quite easily lose track of what they are supposed to be doing. Tasks may also go uncompleted as team members may be unaware that it was assigned to them. This should be mitigated by the use of our Trello board, to keep members focused on their task allocated there for the scrum. This should not be as much of an issue when team members are assigned roles as they will be in charge of a single area of the project.

Project plan

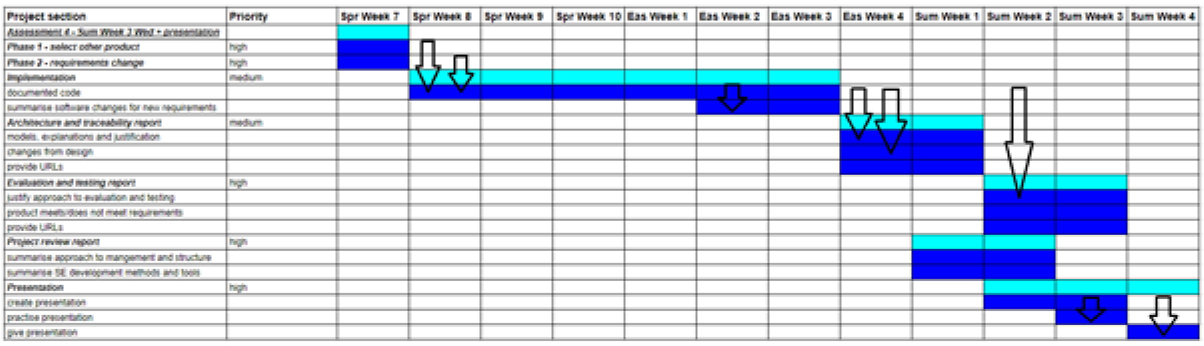
Similar to how sprints will have prioritised tasks, overall tasks regarding deliverables for the project will be given priorities too, largely based around the number of marks available for the tasks and any crucial tasks that must be done to ensure the continuation of the project. Each task has been broken down into sections, such as the document the work will appear in to add an element of organisation to our planning. We will be making use of Gantt charts, which can be seen below, as they make identifying what task should be done at a particular time very easy because of their clear layout. Each tasks should ideally be finished when indicated on the Gantt chart, the end of the week for the box highlighted. We have tried to plan around exams to avoid affecting results. We will likely try to finish large tasks, especially implementation as soon as possible to avoid document work piling up towards the end.



Project section	Priority	Aut Week 7	Aut Week 8	Aut Week 9	Aut Week 10	Xmas Week 1	Xmas Week 2	Xmas Week 3	Xmas Week 4	Xmas Week 5	Spr Week 1	Spr Week 2
Assessment 2 - Spr Week 3 Mon												
Architecture	high	▨	▨	▨								
concrete architecture		■	■									
justify architecture		■	■	■								
agree programming style		■										
Implementation	high	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	
map		■	■									
implement code			■	■	■	■	■	■	■			
followup documentation							■	■	■	■		
state features not fully implemented											■	
GUI report	low								▨	▨	▨	
justify initial GUI									■	■	■	
Software testing report	high									▨	▨	
summarise and justify testing methods										■	■	
brief report on actual tests										■	■	
provide URLs											■	
Update deliverables	medium											▨
requirements												■
methods												■
risk assessment												■
Website	low											▨
update deliverables												■
upload executable of game												■
upload test plan & results												■
upload user manual												■



Project section	Priority	Spr Week 3	Spr Week 4	Spr Week 5	Spr Week 6
Assessment 3 - Spr Week 7 Mon					
Implementation	high	▨	▨	▨	▨
implement documented code		■	■	■	
implementation of architecture and requirements				■	■
Change report	medium			▨	▨
summarise approach to change management				■	■
justify any changes documents				■	■
Website	low				▨
update deliverables					■
upload executable of game					■
upload test plan & results					■
upload user manual					■



Project section	Priority	Spr Week 7	Spr Week 8	Spr Week 9	Spr Week 10	Eas Week 1	Eas Week 2	Eas Week 3	Eas Week 4	Sum Week 1	Sum Week 2	Sum Week 3	Sum Week 4
Assessment 4 - Sum Week 3 Wed + presentation		▨											
Phase 1 - select other product	high	■											
Phase 2 - requirements change	high	■											
Implementation	medium		▨	▨	▨	▨	▨	▨					
documented code			■	■	■	■	■	■					
summarise software changes for new requirements							■	■					
Architecture and traceability report	medium								▨	▨			
models, explanations and justification									■	■			
changes from design									■	■			
provide URLs									■	■			
Evaluation and testing report	high										▨	▨	
justify approach to evaluation and testing											■	■	
product meets/does not meet requirements											■	■	
provide URLs											■	■	
Project review report	high									▨	▨		
summarise approach to mangement and structure										■	■		
summarise SE development methods and tools										■	■		
Presentation	high										▨	▨	▨
create presentation											■		
practise presentation												■	
give presentation													■

Task dependencies are shown by the arrows, linking work that must happen before the end of the arrow. Critical path is shown cyan, multiple sub teams will be working on different tasks concurrently.

References:

[1] I. Sommerville, Software Engineering, Pearson, 2016.

[2] J. G. R. C. M. B. J. H. S. M. A. v. B. A. H. K. S. A. C. R. J. J. S. W. C. J. K. D. T. M. F. B. M. Kent Beck, "Principles behind the Agile Manifesto," [Online]. Available: http://agilemanifesto.org/iso/en/principles.html. [Accessed 26 10 17].

[3] Rational Software, "Rational Unified Process," 1998. [Online]. Available: https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf. [Accessed 27 10 2017].

[4] S. a. ScrumInc, "Scrum Guide | Scrum Guides," ScrumInc, [Online]. Available: http://www.scrumguides.org/scrum-guide.html. [Accessed 27 10 2017].