Top Right Corner

Software engineering methods, development and collaboration tools

We shall be following a Scrum like approach during the development of our system. Scrum is an Agile methodology[1,2] meaning that the planning, development and testing phases are interweaved[1]. We opted for an Agile methodology as we are inexperienced software developers and therefore there is no way that we could completely design a system prior to commencing development (as a plan-driven methodology would have us do[1]). We would almost definitely have to replan the entire process and redesign the entire system multiple times, which would take up too much of our time meaning that we spent less time actually developing the system (and due to other deadlines and obligations we can't justify the overhead involved in such a process). Also, developing the entire system prior to testing would mean that we didn't discover errors until far too late in the process, which could result in a lot of work having to be done to redesign the system and refactor our code. Incrementally designing and developing our system allows us to adapt to changes in requirements more easily (such changes will be issued as part of the SEPR project) and account for the fact that our understanding of the system and time estimates for work that is to be completed will change. For example, we cannot say at this stage how much a roboticon or one unit of food should cost or how much time each player should have to purchase roboticons and won't know until we are at a point in development where we can test these things to find out what works best. Scrum is appropriate due to the size of team being between 3 and 9 people[3] and as we working to strict deadlines on non-safety critical software.

Scrum breaks down the software engineering process into a series of fixed lengths cycles (1-4 weeks in length) called sprints[1,3]. At the start of each sprint there is a sprint planning session[3]. During which the team decides what work is to be completed. The highest priority items that they believe can be implemented during the sprint are selected from the product backlog (an ordered list of everything that may be needed and is to be done). The team also defines a sprint backlog that contains only the work to be completed during the current sprint and a fairly detailed plan that details how the work is to be carried out. The sprint backlog can also be used to track the progress of the team throughout the sprint, which will allow us to be sure that everyone is pulling their weight. At the end of each sprint the development team carries out a sprint review along with the stakeholders, where they talk about what was done and what changes have been made to the product backlog during the sprint (or that should now be made at the end of it). The stakeholders are invited so that they can provide feedback on what has been done and the progress of the product and suggest changes/features that should be included in the product backlog. A sprint retrospective is also carried out around this time which the team uses to identify what went well in the last sprint and what improvements could be made with regards to team working, productivity, communication and alignment with the Scrum process. This is done so that team can learn from their mistakes and increase their productivity whilst making the development process more enjoyable (an extremely useful meeting for us, as we are new to Scrum). We feel that the sprint retrospective and sprint reviews are key to us increasing our productivity and ensuring that we deliver high value products. A daily Scrum is also usually carried out at the start of the day where each team member states what they did yesterday, what they are going to do today and if they can see anything that may impede the progress of themselves or the group.

We aren't going to practice the exact the Scrum methodology exactly. We are not going to have daily Scrums and we are going to combine the other 3 meeting types into a single meeting that won't have quite as much stakeholder involvement as Richard is the only stakeholder and has a commitment to other groups and other things outside SEPR. We aren't having daily Scrums as we likely won't be working on SEPR every day (due to some days being predominantly filled with lectures and other commitments that we have to keep), nor will we all be in the same place at the same time everyday as some of use choose to work from home. Instead we are utilising Trello (see below) so that everyone can see what everyone else is doing and has done. In addition, any potential problems shall be brought up in the group chat and dealt with immediately. We are also going to combine the other 3 meeting types into one as the only time that we can guarantee we'll all be in the same place at the same time is during the SEPR practical sessions so it makes sense to get all of the meetings done at that time. If necessary we may choose to extend the meeting beyond the end of the practical session.

A Scrum board is a way to visualise the current state of the sprint backlog[4]. There are columns for: requirements to be implemented, tasks that have not yet been started, tasks that are in progress and tasks that are completed. We are using Trello (https://trello.com/) to create and maintain a virtual Scrum board (and we plan to also use it for our product backlog). A physical board would be impractical as we would have nowhere to keep it and we all from work different places and thus wouldn't all be able to access or maintain it.

As well as Scrum we also considered Kanban and XP. In essence Kanban involves breaking down the project into a series of small discrete tasks and carrying out no more than a set number of these tasks at once as opposed to defining a set period of time in which to carry out a number of tasks[5]. We felt that the lack of self imposed deadlines may hinder our productivity and make it more difficult for us to tell whether we are on track. We also won't be able to identify how we could improve our productivity as easily due to the lack of sprints and sprint retrospective meetings. XP is very similar to Scrum but allows for teams to change their minds mid-iteration about what work is to be completed by the end of the iteration[1, 6]. We feel that this flexibility may be a disadvantage as it could allow us to become lazy or spend too much time changing the plan and not enough time doing real work. XP also mandates a series of engineering practices that followers of XP must adhere to[1]. Two of which we agree with and have chosen to implement: test-driven development and constantly refactoring code as soon as improvements are found. However XP also dictates that pair programming must be utilised, which is not something we are able to do as we don't all always work at the same time in the same places. Finally XP dictates that user requirements are recorded on story cards that describe a task that the user may wish to carry out using the system. Are requirements are not written in this format and we do not wish to express them in this way.

As stated we are going to perform test-driven development, we shall use the JUnit automated testing framework (http://junit.org/junit4/) which is supported by Eclipse[7] (the IDE we are using) and JMockit (http://jmockit.org/) to mock out dependencies. We shall write unit tests that specify what our program should do before we start coding a specific feature and all unit tests must pass by the end of each Sprint. Automating much of our testing should save us a lot of time especially when it comes to refactoring and retesting our code.

We are using to Git to manage our version control (with our repository hosted on GitHub), Google Drive and Google docs to share and work on our documentation and a Facebook group chat to handle communications outside of meetings. We're also using Gradle(https://gradle.org/) to handle build automation and testing (making it easier for others to build and test our code).

It's worth noting that each sprint will last 1 week, and over the christmas and easter breaks we likely won't be able to have face to face meeting and therefore will have to resort to using the group chat when planning sprints.

Team organisation

As we are following a Scrum like methodology we self organise, we do not have “a manager” everyone is involved in the sprint planning session and will have a say as to what work they would like to compete and input to all decisions that need to be made. We feel that this is best as none of us have management experience or know the strengths and weaknesses of all the others and therefore no one knows how best to delegate tasks or how much work each member is capable of completing in a sprint. Our approach falls in line with Sommerville’s view point as he states that software developers (even very experienced ones) often make poor managers and that any good manager must be aware of the strengths and weaknesses of all team members in order to allocate them tasks that motivate them[1].

As stated we are using Trello for our Scrum board and product backlog. The board contains several lists: one for the product backlog containing all features and requirements that need to be implemented, one for the work that is yet to be completed in the current sprint, one that shows which tasks are in progress and who is completing them and one to show the tasks from the current sprint that have been completed. All team members may access and update the Scrum board and it will be updated at every meeting. This way of working will allow us to fully organise the next sprint, creating a detailed plan for the next week and ensuring that we can track everyone’s progress throughout the Sprint (allowing us to ensure that everyone is pulling their weight). Even once the implementation process is over and we are just working on documentation for the assessment we are still going to operate in this manner.

The only formal roles that we have are a Scrum master (not to be confused with any kind of manager, they serve and work with the team not above them[3]) and a meeting notes keeper.

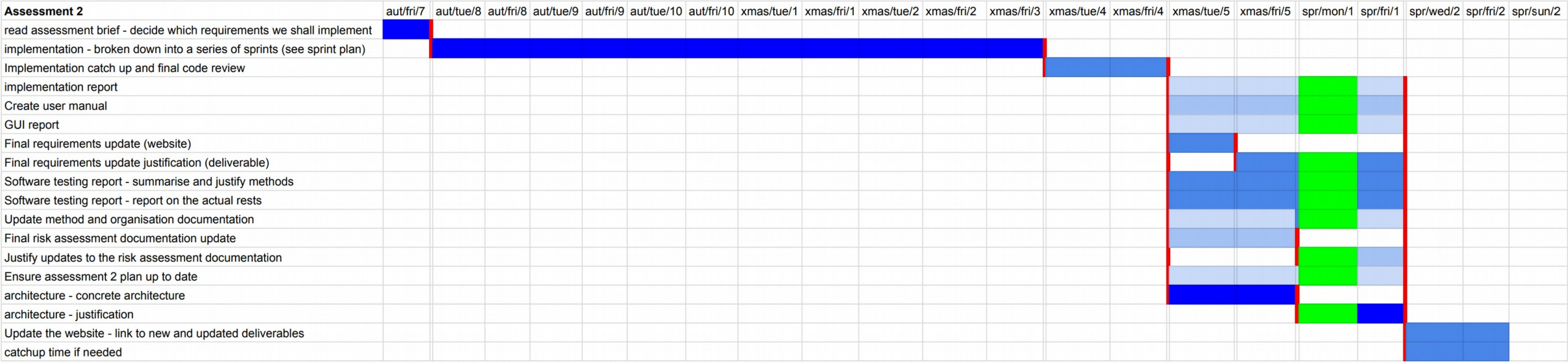
The Scrum master is there to help to[3]:

- Help deal with any problems that arise that may impede progress
- Help ensure that the Scrum process is followed
- Facilitate meetings
- Ensure that the team is self-organising effectively
- Help to ensure that high-value products are produced

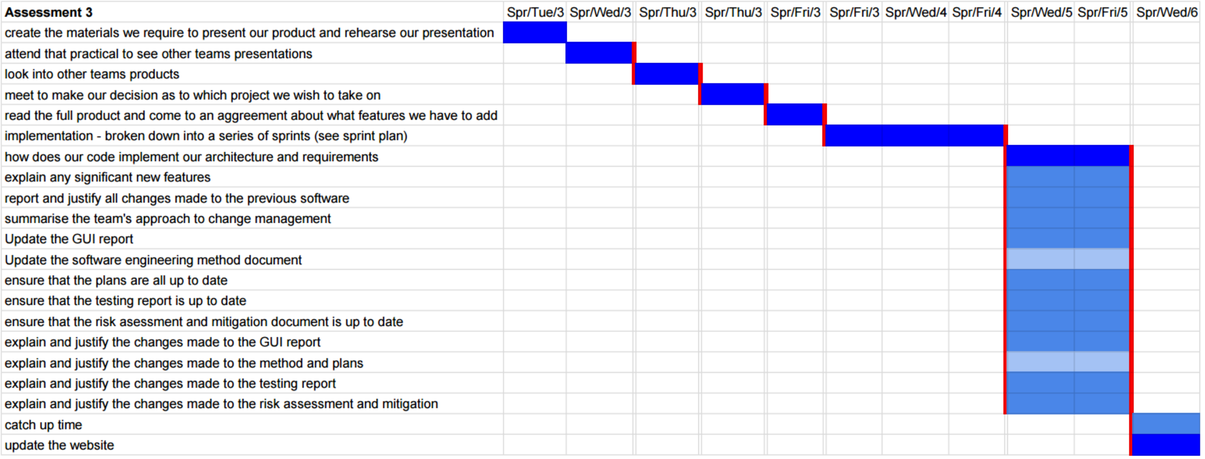
With regard to the overall project plan details about who exactly is going to perform what task and the order in which tasks that may be completed in parallel are to be completed has been omitted. This is due to what has been stated above, we won’t decide who is doing what and exactly when things shall be done until we have the appropriate sprint meeting. Due to this layout of the plan (blocks of time in which many tasks can be carried out in parallel and at any time) it is not possible for us to define a critical path as if any one of the tasks that can be completed in parallel is not finished in time the whole project will be delayed. The plan is broken down into half week increments (apart from in some places where it was necessary to deviate from this due to assessment deadlines etc). Each task is to supposed to be started at the time above the first coloured in block relating to that task and to be finished by the time stated above the block that is after the final coloured in block relating to that task. Priorities have been assigned to tasks based on the number of marks available for the task, the estimated amount of effort and the starting times of other tasks that are dependant upon their completion.

This approach to organization and planning is appropriate as many aspects of the system have not been fully determined for example we don’t know what random events we wish to support (see the requirements document) and we already know that a requirements change will be issued. Meaning that we will have to come together as a team to discuss these issues and can only plan in the short term.

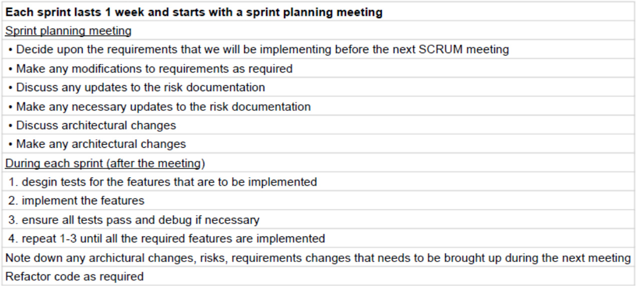
SEPR project plan (larger versions: https://sepr-topright.github.io/SEPR/documentation/assessment2/plans.pdf)



Assessment 2	aut/fri/7	aut/tue/8	aut/fri/8	aut/tue/9	aut/fri/9	aut/tue/10	aut/fri/10	xmas/tue/1	xmas/fri/1	xmas/tue/2	xmas/fri/2	xmas/fri/3	xmas/tue/4	xmas/fri/4	xmas/tue/5	xmas/fri/5	spr/mon/1	spr/fri/1	spr/wed/2	spr/fri/2	spr/sun/2
read assessment brief - decide which requirements we shall implement	🟦																				
implementation - broken down into a series of sprints (see sprint plan)		🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦									
Implementation catch up and final code review													🟦	🟦							
implementation report															🟦	🟦	🟩	🟦			
Create user manual															🟦	🟦	🟩	🟦			
GUI report															🟦	🟦	🟩	🟦			
Final requirements update (website)															🟦						
Final requirements update justification (deliverable)																🟦	🟩	🟦			
Software testing report - summarise and justify methods															🟦	🟦	🟩	🟦			
Software testing report - report on the actual rests															🟦	🟦	🟩	🟦			
Update method and organisation documentation															🟦	🟦	🟩	🟦			
Final risk assessment documentation update															🟦	🟦					
Justify updates to the risk assessment documentation																	🟩	🟦			
Ensure assessment 2 plan up to date															🟦	🟦	🟩	🟦			
architecture - concrete architecture															🟦	🟦					
architecture - justification																	🟩	🟦			
Update the website - link to new and updated deliverables																			🟦	🟦	
catchup time if needed																			🟦	🟦	



Assessment 3	Spr/Tue/3	Spr/Wed/3	Spr/Thu/3	Spr/Thu/3	Spr/Fri/3	Spr/Fri/3	Spr/Wed/4	Spr/Fri/4	Spr/Wed/5	Spr/Fri/5	Spr/Wed/6
create the materials we require to present our product and rehearse our presentation	🟦										
attend that practical to see other teams presentations		🟦									
look into other teams products			🟦								
meet to make our decision as to which project we wish to take on				🟦							
read the full product and come to an aggreement about what features we have to add					🟦						
implementation - broken down into a series of sprints (see sprint plan)						🟦	🟦	🟦			
how does our code implement our architecture and requirements									🟦	🟦	
explain any significant new features									🟦	🟦	
report and justify all changes made to the previous software									🟦	🟦	
summarise the team's approach to change management									🟦	🟦	
Update the GUI report									🟦	🟦	
Update the software engineering method document									🟦	🟦	
ensure that the plans are all up to date									🟦	🟦	
ensure that the testing report is up to date									🟦	🟦	
ensure that the risk assessment and mitigation document is up to date									🟦	🟦	
explain and justify the changes made to the GUI report									🟦	🟦	
explain and justify the changes made to the method and plans									🟦	🟦	
explain and justify the changes made to the testing report									🟦	🟦	
explain and justify the changes made to the risk assessment and mitigation									🟦	🟦	
catch up time											🟦
update the website											🟦



Each sprint lasts 1 week and starts with a sprint planning meeting

Sprint planning meeting

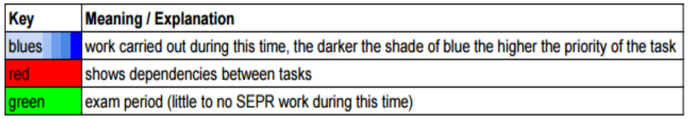
- Decide upon the requirements that we will be implementing before the next SCRUM meeting
- Make any modifications to requirements as required
- Discuss any updates to the risk documentation
- Make any necessary updates to the risk documentation
- Discuss architectural changes
- Make any architectural changes

During each sprint (after the meeting)

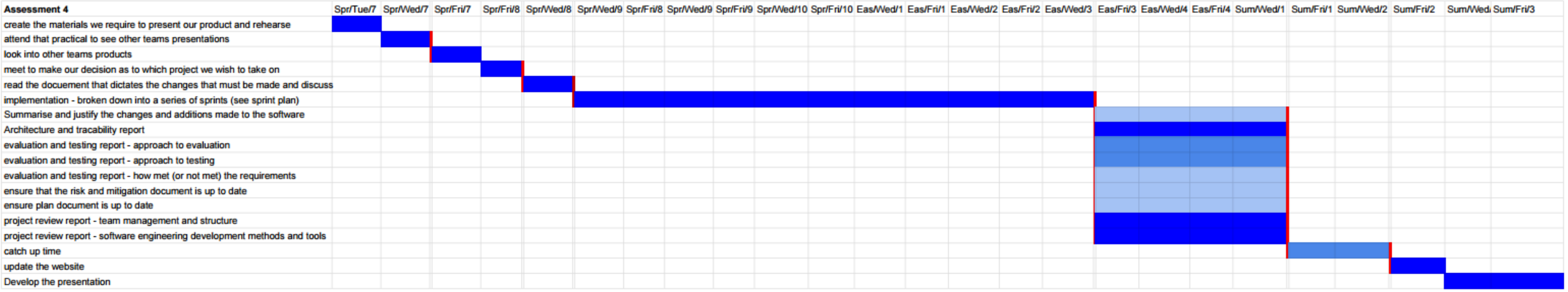
1. desgin tests for the features that are to be implemented
2. implement the features
3. ensure all tests pass and debug if necessary
4. repeat 1-3 until all the required features are implemented

Note down any archictural changes, risks, requirements changes that needs to be brought up during the next meeting

Refactor code as required



Key	Meaning / Explanation
blues 🟦	work carried out during this time, the darker the shade of blue the higher the priority of the task
red 🟥	shows dependencies between tasks
green 🟩	exam period (little to no SEPR work during this time)



Assessment 4	Spr/Tue/7	Spr/Wed/7	Spr/Fri/7	Spr/Fri/8	Spr/Wed/8	Spr/Wed/9	Spr/Fri/8	Spr/Wed/9	Spr/Fri/9	Spr/Wed/10	Spr/Fri/10	Eas/Wed/1	Eas/Fri/1	Eas/Wed/2	Eas/Fri/2	Eas/Wed/3	Eas/Fri/3	Eas/Wed/4	Eas/Fri/4	Sum/Wed/1	Sum/Fri/1	Sum/Wed/2	Sum/Fri/2	Sum/Wed	Sum/Fri/3
create the materials we require to present our product and rehearse	🟦																								
attend that practical to see other teams presentations		🟦																							
look into other teams products			🟦																						
meet to make our decision as to which project we wish to take on				🟦																					
read the docuement that dictates the changes that must be made and discuss					🟦																				
implementation - broken down into a series of sprints (see sprint plan)						🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦									
Summarise and justify the changes and additions made to the software																	🟦	🟦	🟦	🟦					
Architecture and tracability report																	🟦	🟦	🟦	🟦					
evaluation and testing report - approach to evaluation																	🟦	🟦	🟦	🟦					
evaluation and testing report - approach to testing																	🟦	🟦	🟦	🟦					
evaluation and testing report - how met (or not met) the requirements																	🟦	🟦	🟦	🟦					
ensure that the risk and mitigation document is up to date																	🟦	🟦	🟦	🟦					
ensure plan document is up to date																	🟦	🟦	🟦	🟦					
project review report - team management and structure																	🟦	🟦	🟦	🟦					
project review report - software engineering development methods and tools																	🟦	🟦	🟦	🟦					
catch up time																					🟦	🟦			
update the website																							🟦		
Develop the presentation																								🟦	🟦

Bibliography

[1] I. Sommerville, *Software engineering*, 10th ed. Boston: Pearson, 2016, pp. 72-88, 652.

[2] M. James, "An Empirical Framework For Learning (Not a Methodology)", *Scrummethodology.com*, 2016. [Online]. Available: http://scrummethodology.com/. [Accessed: 31- Oct- 2016].

[3] "Scrum Guide", *Scrumguides.org*, 2016. [Online]. Available: http://www.scrumguides.org/scrum-guide.html. [Accessed: 31- Oct- 2016].

[4] J. Bowes, "Agile concepts: the Scrum Task Board - Manifesto", *Manifesto*, 2014. [Online]. Available: https://manifesto.co.uk/agile-concepts-scrum-task-board/. [Accessed: 31- Oct- 2016].

[5] J. Bowes, "Kanban vs Scrum vs XP – an Agile comparison - Manifesto", *Manifesto*, 2015. [Online]. Available: https://manifesto.co.uk/kanban-vs-scrum-vs-xp-an-agile-comparison/. [Accessed: 31- Oct- 2016].

[6] M. Cohn, "Differences Between Scrum and Extreme Programming", *Mountain Goat Software*, 2007. [Online]. Available: https://www.mountaingoatsoftware.com/blog/differences-between-scrum-and-extreme-programming. [Accessed: 31- Oct- 2016].

[7] "Help - Eclipse Platform", *Help.eclipse.org*. [Online]. Available: http://help.eclipse.org/neon/index.jsp?topic=%2Forg.eclipse.jdt.doc.user%2FgettingStarted%2Fqs-junit.htm. [Accessed: 31- Oct- 2016].