Method Selection and Planning

Our software engineering methods

When we came to decide on the methodology that we would use as a team we were aware that the vast majority of other groups in previous years used an agile methodology. We also knew that historically methodologies such as waterfall have been widely used. For that reason we focused most of our research on agile and waterfall. We then focused on agile only after establishing early on that Waterfall and other similar methodologies are unfit for our purposes, especially since we knew we would have our requirements changed halfway through. We decided that Waterfall was not for us since it is infamously inflexible. Once you have moved past a step it is very hard to change what you have done.

Since Waterfall is inflexible and we knew that as a team we would be forced to be flexible, we started to look agile. Agile in not so much a methodology, but a frame of mind that many methodologies have sprung from. It emphasises small, incremental changes. The Agile Manifesto, the document which sets out the "frame of mind", says "Responding to change over following a plan." This is exactly what we will need to do as a team since we will need to respond to a changing product, new requirements, and later, a new product again. Agile also seems to encourage a close and casual relationship with the client. Some methodologies such as Scrum go as far as to blurring the line between development team and client by introducing a product owner as a role in the team. This also seems to be encouraged by the structure of our project, with weekly practicals that both we and the lecturers (acting as our client) will be attending. Finally an agile approach aims to take away the red tape in software development, with mantras such as "Working software over comprehensive documentation" and "Customer collaboration over contract negotiation." This makes agile suited to us since a big factor to consider is that we all have many other responsibilities. SEPR is not our 9-5 jobs.

Clearly agile is exactly the sort of approach we should take. The next question is which methodology we should use within the umbrella of agile. The two biggest methodologies seem to be Scrum and Kanban. Having seen on an online poll that very few teams actually use an "off the shelf" methodology and prefer to use set methodologies only as a guide, we have decided to use a fusion of the two. We have gone for a methodology based mostly off of Scrum and taking elements of Kanban where needed.

From Scrum we are taking the concept of user stories, although we are structuring them more as descriptions of features, rather than stories of a feature being used. While this seems to be a standard for Scrum, we believe this would be more practical for our group. ~~We are also using the concept of daily stand-up meetings, but since we have other responsibilities besides SEPR and it is hard to organise for everyone to be free for a meeting every day, these are now "frequent informal meetings", in person when possible, over discord when not. These will allow us to check in and ensure nobody is getting behind~~. We had already implemented twice weekly meetings that worked better for our group than the Pi-rates daily meetings as it gave team members more time to get work done as some people cannot work every single day, thus making that days meeting futile for them.They will also mean that each individual member should have a better picture of what is going on in the overall project, something which should help prevent clashes in the form of people working to different ideas in their heads. We will not be using burndown charts. With the small-scale amount of tasks, time and team members we have to manage we should be able to keep track of progress as we go, making a burndown chart redundant and overly bureaucratic. We will use relatively short sprints of around a week, but use them in a more informal style than some variations of Scrum lay out. We will make use of retrospectives at the end of each sprint. We will likely not use releases since we don't really have enough features to make it valuable and at our scale each assessment is basically a release. Within each sprint we will use a Kanban board. We will also use another Kanban-esque board for features in the product backlog and sprint backlog. Our

product owner will be the lecturers, and will have a more backseat and less integrated role than most versions of scrum since the lecturers have many projects to oversee. We will have a scrum master who will do almost exactly as in most variations of scrum. They will be the same person as the “Organiser” described in the team organisation section. We will have twice weekly main meetings which allow us to help each other with things we are struggling on and resynchronise so that we work as a team rather than individuals.

So in summary we will use the framework that scrum sets out, but in an altogether more informal and casual way, similar to the more freeform approach of Kanban. We will use kanban within sprints.

We decided that it would probably be best not to plan the Kanban too formally since we could decide better what we really needed as we got to it. We could afford to do this since Kanban isn’t too formally structured so we can change what we are doing as we go without causing problems. As a starting point however, we will use a board with five columns: To-Do, In Progress, Needs Testing, Testing, Finished. The idea is that at the start of a sprint, all of our tasks for that sprint will go into the To-Do column and everything will shift right as the sprint goes on until everything is in the finished column.

Our Tools

Tool	**Purpose**	**Justification**
Facebook messenger	Communication (Quick, informal messages.)	• We all already use it • Pushes messages to phones, seen quickly.
ProjectLibre	General project/task management	• Gantt style timeline • Task dependencies • Start and end dates
Google Drive	Non-Coding related file storage (E.g. Assessment documents)	• Shared file space • Many collaboration tools
Github	Coding related file storage / Version control	• Very powerful version control tool • Allows us to work on different parts of the project independently
Star UML	UML and flowcharts	• Very little learning curve • Integration with google drive • Clear and nice looking flowcharts and UML • Flowcharts and UML allow the whole team to quickly and easily get on the same page in terms of how algorithms work and how our architecture is structured.
IntelliJ	IDE	• Code completion • Refactoring • Debugging tools
Photoshop/ GIMP	Photo-editing tool for front end development	• Range of useful tools • Reliable

Our team organisation

When researching team structures for general project management and teams, both within and beyond software engineering, we found that most structures seem to fall in to one of two categories: Mechanistic (or tall) and Organic (of flat).

A tall team structure is one with a clear chain of command and influence is projected from the top of the hierarchy, down through the ranks. ~~We very quickly dismissed a tall structure. While we all have different strengths and weaknesses, none of us really has any more experience than any other. Essentially we had no qualifiers to put one person higher in the chain of command than anyone else. Besides that, our team is too small to really use an effective chain of command~~. We have decided to keep a hierarchy in order to keep decision making concise to avoid elongated discussions over issues.

We are now left with a flat structure. Flat structures are not as commonly used in business since they are more suited to smaller teams. However we are a smaller team and a flat structure should mean that we all are “in the loop” on which direction our project is headed in at any given time. A flat structure focuses more on collaboration and cohesion than management and control. We felt that this serves our purposes well since, as mentioned earlier, nobody has any more experience or stake in the project than anyone else. Other advantages of a flat structure include the fact that they are more adaptable and flexible. None of us have had real exposure to how teams are operated in a workplace yet so it is inevitable that we will make some missteps in our approach to team working. Having a less rigid structure will more easily allow us to adapt our style and move past these issues.

One thing that we found when researching was the concept of the “whole team” approach. It is a concept taken from extreme programming and lean production and it emphasises the team dynamic. The key idea is that every member of the team is aware of and appreciates every other members strengths.The concept focuses on the fact that individuals, while having their own tasks, do not operate in isolation and have shared responsibility for the quality of the final project. . It takes emphasis from individuals and frames the team as one unit. Any team member should be be willing to switch roles and help the team reach its goal by doing whatever is needed, rather than just doing what they were assigned.

For the reasons above, we decided to structure our team as follows: Every team member is equal and in disagreements a decision should be come to by compromise and discussion. If a solution is still not decided on, majority rules. ~~We do not have any formal roles in our group. This allows people to be dynamically assigned to tasks depending on their individual strengths and preferences. This is something that not all teams can do, but we can due to the small size of our team and project. The only exception is that there is one designated organiser, Scott, who the team elected to act as a kind of leader.~~ After losing two team members we decided that it would be beneficial to allocate formal roles to distribute responsibility among the remainder of the group. The roles are as follows: ‘Test Manager’, ‘Front-End Developer’, ‘Back-End Developer’, ‘Documentation Manager’ and ‘Organiser’. The organiser can always be overruled by the team and will act more as a representative of the team, in the sense that decisions are made by the team and sent up to him as opposed to him making decisions and imposing them upon the team. The final defining feature of our team is that we have come up with a buddy system wherein each individual in the team is paired with someone else. ~~Since we have an odd number of team members, we will have a pair and a three.~~ Each member of a pair of buddies will make it their business to know what the other is working on and how it works. This way, each member of the group has at least one other person who has a decent knowledge of what they are working on and how it works. For every person their buddy will be their first port of call with technical and workload problems and if a person is unable to complete their work for some reason (e.g. medical leave) it is their buddy who will take that work from them.

Our plan for the remainder of SEPR

Please see “Planning Sheet.pdf” to find our plan for assessments 1 through 4. As our team approached Assessment 3 we added extra details to this plan [12].

References

When this section was written, rather than researching and writing up as the research was done, I read/watched through all of these sources to learn about the areas over a few days and then a few days later used that knowledge, in conjunction with prior knowledge from experience and lectures, to write up this document. As a result there are very few parts of this document that are taken directly from any one source, but rather the information was adsorbed and all used together to create the end result.

In light of this, we felt that the standard IEEE referencing that we have tried to follow everywhere else was not suitable here, since the information came from my understanding of all of the sources combined and to attribute each point to a single source would be inaccurate. Instead please find below each of the sources and what I feel that I learned from them.

General idea of Agile and some quotes: [1]
Setup of a scrum team: [2, 3, 7, 8]
Roles within a scrum team: [2, 3, 7, 8]
Features of scrum: [3, 7, 8]
General idea of Kanban: [8, 9]
Some general team structures (e.g. flat vs tall): [4, 5]
Whole Team approach: [6, 10]

[1]"Manifesto for Agile Software Development", *Agilemanifesto.org*, 2018. [Online]. Available: http://agilemanifesto.org/iso/en/manifesto.html. [Accessed: 06- Nov- 2018].

[2]"Roles on Agile Teams: From Small to Large Teams", *Ambysoft.com*, 2018. [Online]. Available: http://www.ambysoft.com/essays/agileRoles.html. [Accessed: 06- Nov- 2018].

[3]"Scrum: What It Is, What It’s Not, & Why It’s Awesome - Atlassian", *Atlassian*, 2018. [Online]. Available: https://www.atlassian.com/agile/scrum. [Accessed: 06- Nov- 2018].

[4]"Common Organizational Structures | Boundless Management", *Courses.lumenlearning.com*, 2018. [Online]. Available: https://courses.lumenlearning.com/boundless-management/chapter/common-organizational-structures/. [Accessed: 06- Nov- 2018].

[5]E. Devaney, "The Pros & Cons of 7 Popular Organizational Structures [Diagrams]", *Blog.hubspot.com*, 2018. [Online]. Available:

https://blog.hubspot.com/marketing/team-structure-diagrams. [Accessed: 06- Nov- 2018].

[6]"Extreme Programming: Whole Team", *Coding Journeyman*, 2018. [Online]. Available: https://codingjourneyman.com/2015/04/13/extreme-programming-whole-team/. [Accessed: 06- Nov- 2018].

[7]"Intro to Scrum in Under 10 Minutes", *YouTube*, 2018. [Online]. Available: https://youtu.be/XU0llRltyFM. [Accessed: 06- Nov- 2018].

[8]"Scrum vs Kanban - What's the Difference? + FREE CHEAT SHEET", *YouTube*, 2018. [Online]. Available: https://www.youtube.com/watch?v=rIaz-l1Kf8w. [Accessed: 06- Nov- 2018].

[9]"Kanban 101 - What is Kanban?", *YouTube*, 2018. [Online]. Available: https://www.youtube.com/watch?v=jf0tlbt9lx0. [Accessed: 06- Nov- 2018].

[10]"What is whole-team approach (team-based approach)? - Definition from WhatIs.com", *SearchSoftwareQuality*, 2018. [Online]. Available: https://searchsoftwarequality.techtarget.com/definition/Whole-team-approach. [Accessed: 06- Nov- 2018].

[11]S. Langridge, "The case for Asana", *Pi-Rates Team Drive*, 2018. [Online]. Available: https://drive.google.com/open?id=1C0spu32TZPG0YseA32pROJpjbtJZvcW6F7TqwVktcWc. [Accessed: 06- Nov- 2018].

[12] S.Langridge, “Planning Sheet”, *Pi-Rates Team Drive*, 2018. [Online]. Available: https://docs.google.com/spreadsheets/d/18a2nhmZNz2kR_czBymUDFA9f7OT1HThpxKhHB6XRSTM/edit#gid=1957747775