First Reflection Report Hakau Ballard – 1181917

**Introduction**

As part of the team Cool\_Name, I have been working on the ‘GitHub-Shepherd’ project for MetOcean. ‘GitHub-Shepherd’ is best described as a Git-based server deployment system. This project is quite large and complex at this point, which has required the utilisation of more formal organisation systems and development approaches then I have experienced before. One that has been of particular use is SCRUM, which I will provide some detail on. Additionally, I will give a description and some opinions on Git’s role in the project, which has been a key focus for me. The focus will tend to be on small parts of the project and development that have made me think a little and learn something a little bigger.

**Scrum** *as a team*

This project marked my first encounter with Scrum, which we elected to adopt under the recommendation of MetOcean. The divide and conquer aspect has been hugely useful to us as a team in terms of allowing us to break up the complexities of our system into more easily understandable sub-problems. Early in the project we focused on identifying and outlining these sub-problems with our client, which has allowed us a better understanding of how we can best the tackle the whole project without feeling overwhelmed. Additionally, they would regularly specify how important they considered something, allowing us to maintain a notion of what they think is important that the project does overall as a whole. We could also use this is a basis to organise our sprints. As a team, I think we have been able to effectively break down problems as our clients add them, which has been hugely useful to staying on track. The short time frames of sprints also suits us as students as we can focus on more difficult tasks when we know we have more free time and it also allows us to schedule around timetabling issues.

Additionally, Scrum style meetings are very useful. We can’t follow the typical template exactly due to timetabling differences and such but we follow a similar approach. This is to have short, regular meetings focusing on progress since the last meeting, tasks before the next meeting and possible issues. Structuring our meetings this way has meant we usually know where one another are at and can help each other with problems.

**Scrum** *for me*

The ‘sprint’ is where I personally found scrum most useful. The ability to tunnel in on one aspect of the project for a short period was a great way to prevent myself from getting distracted or losing motivation. This became evident very early on, the first two-three weeks in fact. Early on I focused heavily on getting a small but critical portion of our Git backend to work, which was to hunt for local repositories and pull them. This goal had a small enough scope over a short timeframe that I could see clearly how I would achieve it, allowing me to focus on actual implementation. This was very useful for me as I have a tendency to try and do too much at once, which often results in me getting confused, hitting a blank wall and losing interest out of frustration. As I slowly wore out on the Git backend I was able to move onto something different for my next sprint, which was to build a rough layout for web app interface. People tend to get ladled and stuck with a specific aspect of a project, i.e. “The backend guy”, so having the option to migrate keeps things interesting.

**Scrum** *overall*

Scrum tends to be one of the most common Agile frameworks and I can now see why. It has been critically useful to allowing us to manage with an unfamiliarly large project. It’s not only allowed us to effectively rationalise the project and formulate ways to tackle it iteratively but has also kept personal interest. Its usefulness was unexpected due to the simplicity of the framework, but in retrospect I feel like I’ve got a reasonable idea as to why and how it works. If I feel I have learned one thing in the paper so far it would be that correctly choosing a development framework is hugely important in working groups on larger projects.

**Git** *Pulling*

A large portion of my work in the project has been on the Git backend. I have found this to be a useful learning experience in terms of understanding the inner workings of a tool I tend to use frequently. I feel like it is a topic that I now know substantially more about after having to do some research into it, as it was mostly ‘black magic’ going into the project. Here I hope to give an example of things I have had to consider.

The three features of Git most important to our project are the pull command, tag retrieval and commit hashes, which we use to safely distinguish between repositories. The role of pulling is one that I have contemplated over the project. The main reason for this has been due to how pull deals with branches, which has been important as our clients quite often use branches. First, it fetches the remote of the branch and gets the new commits. It then performs a merge with remote branch and the current local branch. This is where possible problems arise. In performing the merge you can produce a git ‘tree’, which represents the relationship of branches, which is not what you intend to produce, based on how git merge works. This was somewhat unexpected as we went into adopting the Apple slogan of “It just works!”, so we had to find solutions. A solution we devised was to simply confine the user to pulling from specified tags. Additionally, I have been working on a fetch/merge approach, which may allow us to actually pull from individual branches safely. This new approach has forced me to consider a new aspect of the project, which is that of the user’s experience. This fetch/merge pattern tends to take a long time to complete, which can leave the user waiting. Now the user may not necessarily want to get at particular branches and would rather just opt for pulling ‘masters’. Here we use a strategic pattern to determine which command to use, the pull or the fetch/merge combination. We base the strategy switch on whether or not the user has selected a non-master branch in branch column of the table. We decided that it would be best to inform the user that it may take a while to complete and ask if they wished to go ahead. I found considerations like this interesting, as I don’t have much experience in HCI. It served as a good reminder to always be mindful of the end-user and their interaction with the system, whether it be by trying to optimise certain things for them or simply providing feedback.

**Conclusion**

I have found this project to be useful as it has required me take a new approach to software development, which provided a useful structure to progressing through our project. Additionally, I have learnt some new practical concepts, primarily relating to Git.