Final Report - Assignment 5

*Daniel Zambetto -* 26895730

# Agile software development practices

During this assignment James and I incorporated many agile practices into our work. We followed a sprint approach, as the assignment was broken into two iterations each sprint lasted roughly a week. With a basic program implemented, based on the first iterations user stories, the deliverable still had basic functionality and it achieved the goals. The second iterations deliverable included more features as we built upon the existing system. Using a scrum approach, with each iteration seen as a sprint.

We *planned for each sprint* as soon as we received the assignment brief. A proper sprint planning meeting would include the product owner, who is the person that the system is being built for. Along with the ScrumMaster, the person responsible with ensuring the team continue to work and pushes them to achieve to a high standard by applying scrum practices. The team is also present during this first meeting. As James and I were the only members involved in this project we may be seen as just the development team with Arvind (tutor) and Nawful (lecturer) could potentially be seen as product owners. Sprint planning was done by reviewing the given user stories and breaking them down. User stories allowed the team to implement specific functionality and ensure all needs are meet for the product owner. We did not implement a sprint goal as we felt it was not needed, the goals were already well defined in the assignment specification and we knew what needed to be met to complete the assignment.

We were not able to implement a daily scrum as we did not have time to meet up, we also didn’t communicate very much and relied on accomplishing jobs set from the sprint planning session. As we did not have a chance to review and set up the day it made accomplishing work sometimes troublesome , as we did not communicate our work completed to each other. This caused myself to sometimes lose focus and miss some goals but we eventually accomplished what was needed.

Throughout the first and second iteration of the project, time was being tracked with a timesheet as well as on trello. The project was small, so instead of implementing charts and timelines such as a Sprint Burndown Chart we split user stories and allocated sizes to them. This was similar but instead of estimating time to complete we used the size estimate of the task, this way things would be prioritised based on size and what needed preliminary user stories.

In terms of Sprint Review and Sprint Retrospective, we adapted our own similar method and reviewed our own work and looked at what was done well. As a small team we were aware of everything and just discussed what had been accomplished and worked on starting the next sprint from there. There was no sprint backlog as all tasks were finished from the first and second iteration.

# Team work

James and I were able to accomplish all tasks that were set and were able to coordinate ourselves quite well. At times however communication was lacking but considering we were both very busy we managed to work together effectively.

In each iteration we would do the planning in class we would set tasks then as it was the only opportunity that we had to do so in person. Firstly we would ensure we have read the specification, so when we met we would be able to clear up anything that we were unsure of. James was very helpful in ensuring I understood what was needed and would encourage questions to invoke more conversation about the task. We split the user stories up by discussing what needs to be done, as this was the best way for us to do it so we both were aware of all user stories and their sizes so we could estimate how long it would take to implement. Once everything was on trello we would choose which ones we were to implement.

Communication was mainly through WhatsApp, as we both had access to it. We would usually text, though when things needed to be clarified we would ring each other. We both found this to be suitable as we both would reply in a very short amount of time. Tasks were accomplished at home and we would update our timesheets on what we have done so the other member was able to see, we would also push work to github. Github was also a very helpful avenue to see what James had completed as it allowed me see changes on tasks, and then I would ask him directly if anything needed clarifying.

This worked well as the tasks were relatively small and quite easy to implement for two people, however this approach would not scale. Instant messaging would not be suitable for anymore than two people working in a group, we were able to make it work as we were the only ones involved. If we were to involve more people I would recommend incorporating daily scrum, being able to communicate with everyone involved would have helped in keeping focused on the project as well as help in time tracking. Trello was a very suitable way to keep track of tasks and it helped in organising tasks and reminded us to accomplish things and what we needed to do.

# Design

In terms of design, we specified our architecture in a prior assignment. We used UML diagrams such as activity and sequence diagrams. Both these documents helped us flesh out our design, and split functionality depending on what suited the client side and server side the most.

Our program handles time calculations of motions on the client side, after a motion end has been received from the server. Then at the client side the time is calculated which would decode into the type of motion (L for long and S for short), which is sent back to the server. This could potentially cause problems by using too many resources, as this could be all accomplished at the client side. However our system allows the user to see any motion detected so this is not a major problem. On a large scale system and with many users this may not scale and it may be better to have the server to handle short and long motion identification. It is however quite easy to add more functionality. With the exception of a few lines of code scattered on their own which may reduce readability.

Most of the functions in our back and front end are modular, with low coupling. Functions that deal with board communication are all grouped, and the function that handles decoding is designed so it handles error such as when motion combinations don’t correlate to anything. This low coupling allows for easy debugging for developers and comments in the code as well as a comprehensive README that follows the product.

Our UI is based on a Material Design Lite template. This is an extremely clean and recognisable design used in Android, making it really easy to use. It has clear feedback for the user and responses from the server as easily registered. This allows for anybody to easily use the interface as it is is very transparent to users.

The board communication is handled using Johnny-five, which allows for a level abstraction when communicating with the board allowing for its libraries to handle communication directly with the board. This is a solid framework and makes things easier and fast, allowing for an overall clean back-end. It meets all of our needs in a reliable fashion. We use soket.io in communicating with the client, which again allows for a level of abstraction when sending motions to the client and sending type of motion back to the server.