**Final report**

**Agile software development practices:**

The way I approached my project is mainly following the Agile approach, therefore I followed many Agile practices, but there are also a few practices that I did not adopt.

The project has a total of 2 iterations, before each iteration, my team will have a sprint planning meeting. For the first part, we meet with the product owner(tutor) and confirm on unclear requirements; then the second part the team will discuss about the requirements. The team will then come up with a product backlog in the form of user stories, prioritise them and estimate how long each user story will take and when can it be finished. By following the plans made, our team’s progress has always been in schedule, therefore the Agile sprint planning method helped our team to make steady steps in the development of the project. I also adopted the Agile design practice, as my team produced a sequence diagram and an activity at the start, as the project is being developed, we always revise and make change to these diagrams. As our design is an ongoing process, our team is able to implement new requirements whenever they arise. My project’s development was also test-driven, using the Mocha and Chai testing framework, I was able to pick up and fix bugs whinin my project very quickly along the development. In addition, refactoring was also adopted in the my approach to the project, and I found it really helpful because as we reduce/change redundant code, my project was always easy to maintain and extend. Another Agile practice I adopted was continuous integration, and I could always deliver working program, this practice not only made it easy to show product owner my progress, but also kept my testing simple to be done.

The biggest impact the academic environment brought to my software development practices is that the product owner which is I could easily contact tutor/lecturer. This reduced so much difficulties compared to the real world Agile approach, because to arrange a meeting between development team and the product owner is one of the biggest challenge in Agile approach in real world.

There are also a few Agile practices I did not adopt, such as scrum daily meeting. The reason behind this was because it’s hard to get the team together everyday, and the team found it enough having 3 meetings per week, additional meeting would become a waste of all members team. Another big thing my team did not adopt was having a proper scrum master, as the team is only made by two people, both members were at the same level and will discuss when a big decisions needed to be made and luckily we can always come to agreement.

When I think back to my project development, I think the only thing I might change is to select a scrum master for the team, therefore if any conflict or disagreement occurs, the scrum master can resolve it or make the final decision. Other than this, I think my approach to this project was good and the development of the project a success.

**Working in teams**

The project team is made by two people, so in order to develop this project, me and my partner had to work together toward the same goal. Although we faced challenges, our teamwork has becoming more effective and efficient over the course of the project development.

As there are only two people in the team, and none of us had a leader’s characteristic, we did not select a group leader. Therefore, all the decisions are made by both of us together, as we did not have many disagreements, there were no conflict which was important and valuable for the team. Me and my partner always work on understand the requirements in the tutorial class, therefore we can correct any misunderstanding or disagreements immediately because the tutor is there to answer our questions. Overall, I think I coordinated with my partner quite well, and we are both easy going people so if one suggests something the other would agree if we both think it’s correct or reasonable.

The communication between me and my partner was done in meetings, tutorial class and Facebook. None of us has ever missed a single tutorial class, because we both think this is the most important time for us to communicate every week, so we did most of our communication in the tutorial classes. In the class, not only we allocate tasks, we also talk about what we did poorly and what we did well last week, so we can try to make improvements in the next week. Since the interviews are also during the tutorial class, we also conduct our sprint review in the class, so our performance is continuously improving. The most frequent communication happens on Facebook, because I can message my partner anytime anywhere if I need to. One thing our team did well at was whenever one of us has made some good amount of progress, we let each other know through Facebook message, even when we get stuck. By doing this, we were always well aware of each other’s progress so we can make quick schedule adjustments. Additional meetings only happens if we need to do testing together or there is big issue with our development, because Facebook and tutorial classes are enough for the team.

The task allocation is also done in the tutorial class, since our class is on Thursday, so we can then both start working on our own parts in the weekend. Most of the time, we divide all tasks into half in both coding and documenting, and each will take one half. I will end up doing more documentation or reporting and my partner will code more because he has the device so it is easier for him to perform testing on the program. We are always satisfied with the task allocation since we always make the allocation together.

If I was programming in a small company, I would definitely select a leader within the team, a leader will can inspire and motivate the team and resolve conflict because in the real world, conflicts occurs more. I would also arrange more meetings with both the team and the product owner, to ensure we are making steady and correct progress, because a real company would have less error tolerant.

**Design**

The development team implemented the Agile approach in this project, so maintaining, expanding and changing the project are all relatively easy to do.

If the client is willing to support variations on the Morse Code App, it would firstly depend on what variation the client wants. If the client wants to change the way the app functions, it would be hard to do because I need to redesign this app. However, if the client wants some functionalities in the app, it is quite easy to accomplish, because the app has almost all functions as individual methods, modify one function would hardly have any impact on other methods. This is because the team aimed for low cohesion when developing the project, this reduced the difficulty of testing and maintaining the project. Therefore, if a change is requested from the client, the I only need to design a new function or change an existing function and add it to the app, it is much easier than redesigning the whole project, as the new function would have much effect on the other ones.

As stated above, a low cohesion program makes debugging simple. A bug is most likely to occur inside a particular function, because bugs in the main function is normally trivial also easy find and fix. If one functionality is not performing correctly, instead of testing the whole program, I just need to test that function. The comments in the code can also become handy when finding and fixing bugs, every single method has comments describing its functionality, input and output, so even people with no knowledge about the code can easily find out where a bug is most likely to be when something goes wrong. Despite the advantages of my project’s code, there are still something I could do better. Firstly, I would use less variables, this has been a problem throughout the course of the development for myself, my code if full of variables and they make reviewing the code a pain for me. Therefore, I should redesign the code when I get stuck rather than just add a new variable and make my code messy. Another improvement I would like to make is focus a bit more on simplicity rather than functionality in my design, since functionality is the most important part, my code designing ignored simplicity, as a result although I always have a working program, bugs might be harder to find and fix. Other than these two areas, I think debugging is not an over complicated process for my project.

I am quite happy about my app’s user interface, although it is not visually stunning, it is simple and clean. It is self explanatory and all user interactable parts such as buttons have clear meaning, will not cause misunderstanding or mislead the user. Plus for such simple program, a fancy looking user interface is not necessary. However, if this program is used for kids I would make all the test and buttons bigger for higher visibility, and modify some texts to make them easier to understand. The reason I did not do these is because I do not think kids are unlikely to use a morse code app. Adding new control or improve my user interface should be simple process and they barely have any affect to each other, as long as we keep the related controls close, there is no problem with changing them.