Advanced Programming

FIT3140

Assignment 5 Final Report

Team 29 – Li Cheng

Name	Student ID
Li Cheng	24864099

Table of contents

Topic 1. Agile software development practices

Adopted agile practices

We adopt the following three agile practices in this development, they are dest-driven development, project backlog and refactoring.

For this project, it is crucial to have two selected functionalities unit tested to ensure program's correctness. For the first iteration, two unit tests were written after the basic implementation is done. However, it is not test driven development because the test cases were developed after major works are done. In the second iteration, we realized our mistakes, wrote some test cases first and developed the program to pass the test specifications. This is far closer to test driven development. To summarise, we benefit from test-driven development in iteration 2 where the program is able to function correctly with minimum code written.

Project backlog is also used for this development. Start with a list of high-value user stories from the project specification, we converted each user story into a Trello board task card. Fully expand each user story, carefully prioritize and estimate the execution time of each task card. The entire project is then splitted into two iterations where each iteration is granted one week to complete. This has allowed us to fairly divide the workload among two team members. Trello is useful in keeping each task on track and ensures each iteration submission can be made on time.

Refactoring is the last but crucial step to improve the nonfunctional aspects of our software product. We use refactoring most throughout the development process. Refactoring includes changing the design of a software artefact without changing what it does, cleaning up the code to make it more comprehensible, extensible, maintainable. In the development process, we find that `morseCodeInterpreter` class has the responsibility to distinguish long and short signals, and then interpret encoded messages into English letters. Rather than making this class too large, the team splits this large class into `morseInterpreter class` and `longShortInterpreter` class where each class is responsible for a particular task. There are a lot more and general examples of refactoring in our development process. For example, having a helper method for a long method to make it less complex and more readable, making repeated code units into a single method to present a clear, well-defined, simple-to-use class interface. In summary, refactoring helps the team to improve the code maintainability and create a more expressive internal architecture to improve software extensibility.

None-adopted agile practices

We did not adopt the following two agile practices in this development, they are continuous integration and scrum daily meeting.

Continuous integration is not heavily emphasized in this project. Continuous integration is the practice of merging all team member's working copies to a shared mainline several times a day - perhaps as many as 10 times a day. It is not widely used given that the size of project is not large enough to make this approach feasible. The entire project, consists of iteration 1 and iteration 2, has an implementation time of two weeks. And the total commits of the project is around 50 times. Given the infrequent updates over the lifetime of the project - two weeks, there are not so many merges required to be done. Secondly, continuous integration requires team members to pass automated unit tests in their local environment and verifying they all passed before committing to the mainline. The entire project only requires two unit tests which are done in an ordered manner, therefore it is not worth the effort to adopt continuous integration. In summary, the team does not think continuous integration is that necessary.

Scrum daily meeting is not emphasized in this project. Daily scrum requires a 15-minute meeting at the start of each day of all team members. Given that the size of the team is as small as two, it is very flexible and effective to facilitate communications. In summary, the team does not think scrum daily meeting is that necessary.

Agile software development improvement

As mentioned before, the implementation of scrum daily meeting really depends on the size of the project and the team size. If the project requires, I will adopt scrum daily meeting to meet the best standard. Secondly, I realized the mistake of writing test cases after the basic implementation is done. In future development, I will write the test cases first, ensure minimum code to pass the test cases, and then refactor the code to acceptable standards. The final thing I can improve is to have a retrospective immediately after finishing iteration 1. This will give the team a better feeling of where we are at, and how to complete iteration 2 in a efficient manner.

Topic 2. Working in teams

Coordination

Our team approaches this project by facilitating effective communication, task allocation, difference resolution by aligning with users' requirements. The team is satisfied with their teamwork, I think the team did a great job in coordination.

Communication

Communication is done in two forms, offline and online. Offline communication takes place in weekly workshop. Online communication takes place in multiple forms, Facebook messenger, Google docs, Github repository and Trello board. In general, offline communication is not as flexible as online communication, but brings the benefit of more efficient teamwork. In future project, I will tend to have face-to-face teamwork to improve team's efficiency.

Task allocation

The task allocation is done on Trello board. Overall, I am satisfied with the task arrangement.

Small company hypothesis

In a small company, employees are very likely to work in a team of two to a team of twenty. The communication will be made easier in workplace but the complexity of task will generally increase. All of the above agile practices are likely to be applied in a small company environment, they will involve planning game, design, test-driven development, refactoring, continuous integration and scrum daily meeting. The most likely change in a small company environment will be longer development lifecycles. In other words, each agile practice will be in practice in detail to overcome any potential project pitfalls. Pitfalls like team member missing, redesign legacy business systems, user changes system requirements after the project's implementation, the choice between short-term corporate profits versus sustainable while expensive development cost.

Topic 3. Design

Extensibility

Depend on what support the client wants to add into the existing Morse Code App, the difficulty of adding in such new support varies. Currently, there are six source files in use. We have server and client files in the root directory, which reads signals, updates firebase and controls web application panel respectively. In the library directory, we have arduino hardware file and virtual arduino hardware file, which interacts with physical arduino device or stimulating arduino device respectively. And two other library files morse interpreter and long short interpreter, which translates motion start, end events into English letters and into short, long signals.

Here are several examples explains the difficulty as to how the client would like to support variations on the morse code app. If the client wants to add variants in how motion start and motion end signals are interpreted the `longShortInterpreter` is the only class needs to be changed. If the client wants to support more character interpretation for example Chinese characters, simply add support in the morse interpreter file. If the client wants to replace the arduino device with another different hardware, only the server file will be modified and some other hardware interfaces will be added into the library folder. If the client wants to add some other features on the web application control panel. The client source file and index html file are the files to be modified. Overall, I think this software implementation keeps high cohesion where each object responsibilities are focussed, therefore supporting variations only changes one to two source files.

Maintainability

The object responsibilities in the software implementation is highly focussed. Once an error occurs, it is fairly easy to deduce what class is responsible for producing that error. To verify the deduction, one only needs to write some test cases which reproduce this type of error. Finally make sure one can pass the new error cases as well as the test cases developed by us. Secondly, if one were to redesign the code to make it more maintainable, one must make sure redesign the program will not fail the unit cases developed earlier. In summary, I think this project is easy to maintain because classes' responsibilities are focussed and automatic unit tests are provided for maintenance purposes.

Application user interface

The visual design of this app is great. The user interface is clear, simple to navigate, one click to use. The visual design is suitable for any group who gets used to general smartphone operations. With respect to adding new controls to the user interface, one can modify the client file, index html file to add new control elements, and modify index scss file in sass folder change web page fonts, background, ...etc