Development Principles and Practices

NJRB2 | MVS9 | FA296

This document will detail the real development principles we have chosen to use, as well as the modifications we have made to these principles to better fit them to our project. We will also justify our use of these principles.

# Which principles have we chosen to use?

We have decided to use elements of Scrum and Extreme Programming for the development of our project.

Scrum is an agile development framework where work is organised into small items to be completed within a set period of time, called a ‘sprint’. Frequent ‘stand-up’ meetings occur to share progress and hold-ups in the work items. At the end of each sprint a retrospective meeting covers what went well and what went poorly.

Extreme programming is an agile development methodology which takes traditional good practice of software development to the extreme. This could involve activities like pair programming (‘extreme’ code review), complete testing, and strict code simplicity.

## Scrum principles

The principles we have taken from Scrum are:

* Having a retrospective
* Having stand-up meetings
* Performing sprints and sprint planning meetings
* Assigning point/time values to work items

The retrospective is where we will describe what works well in our development process and team work, as well as what isn’t working well – this activity takes place after every sprint. Stand-up meetings to inform the group of progress and delays will take place every three days, although they may be changed to be every two days if we find that the meetings are too infrequent. This is because we all have other coursework to be doing alongside the project, and so there may be some days where not everyone works on the project, making it useless to have a daily stand-up.

We will also use sprints to plan and develop work packages, and have sprint planning meetings in order to decide these. Work packages will be assigned point/time values to help organise them, and we will also assess the backlog of work we have, and assign priorities to the backlog. The most major change between true Scrum and our versions of the various meetings is that we do not have anyone assigned as Scrum Master or Product Owner. The former was deemed unnecessary due to the small team size and familiarity of the team with each other; the latter was deemed unnecessary as there s no user/customer who must be represented in this project.

During the first retrospective meeting on Tuesday 16th October 2018 we decided that we would also bend the rules to allow new work items to be added to a sprint between sprint planning meetings, in order to allow flexibility in the work allocated for weeks with little coursework, or when new information meant that new tasks should be created; this rule also means that the total amount of work for a group member may increase or decrease between sprint planning meetings.

## XP Principles

The principles we have taken from Extreme Programming are:

* Pair programming
* YAGNI
* Code review
* Unit test all code

As with the Scrum principles, the XP principles have been modified to better suit the fact that we are not going to be working on this project as a full-time job, having university timetable commitments and other coursework to do as well. As such, we have set pair programming times where at least two of us will be free to work, and which will give us a clear time to do project work amongst our other pieces of coursework. Code review will take place during each GitHub pull request and at set meetings where we demonstrate and explain the work we’ve done so far. These are done to maintain code quality, as well as to keep other group members up to date on work which they may not have seen or understood completely during the pair programming sessions and pull request reviews.

We will be unit testing all code in order to ensure code does not break as changes are made, which is particularly important with regards to the emergency shutdown command for the drone. The ‘You Aren’t Gonna Need It’ principle is simple, but should help to keep code as readable and short as possible, as well as making commits for stages of work shorter.

## Other principles

We have specifically chosen not to use:

* Test-driven development
* Security by design

The first of these is due to the time-consuming nature of TDD – tests must be thought of, written and run before the functionality is implemented, after which the tests must be run again, and the code rewritten again to add optimisations and improvements (if strictly following TDD). Due to the limited development time we will have for the project, as well as the fact that we are not writing a safety-critical system (with the exception of the drone emergency shutdown, which will be well-tested) we decided we should not follow TDD. For the same reasons given for TDD, we will also not be considering security as an important aspect of this project.