Garrett Jennings, A01491831

Design Insights

I designed the Drone Dispatcher program in several stages, initially I was studying examples of UDP socket communication in python, and trying to get the simulator running. After I was confident in my understanding of UDP, the Tello API, and PlantUML syntax, I went on to create a basic UML design but I struggled to visualize a good design as I haven't done a lot of design historically. I decided to start building my design even though I felt it wasn't adequate. Initially I was attempting to do test driven development but as I am inexperienced with it and I found it difficult to know what direction to take, especially with my shaky designs. So I started to build what I had drawn out in UML, but as quickly as I began to build I realized what changes I need to make to my design, so I went back and changed it, went back and forth a few times between the designs and the code.

When I checked out the drone I had very limited time, I was between work and my early morning class and tried to as quickly as possible change the code so I could successfully connect the socket to the drone. I was eventually able to successfully connect and fly a mission a couple times. After this I went back and did my unit tests. I was able to use my original experiment code that I did pull heavily from examples for the test server for the unit testing.

Something that I realized that I could have done better was focus on test driven development but I found it was hard to do starting from nothing at all. Perhaps when I get more comfortable with design it will be easier. I felt that as I built things up, I did a good job of diagnosing when classes were getting too many responsibilities and I was able to continue breaking things up and spreading out functionality.

My use of strategy pattern was initially kind of contrived but then something clicked and then I was able to see that what I was doing was actually close to being really useful. With some adjustments I was able to find a useful way to implement strategy pattern that allows me to have missions types be in control of how they execute themselves rather than something else managing them. It shouldn't be too difficult in the future to make it so that we could make any mission that will execute itself in any way.

One problem I noticed I had when testing in is that it is hard to test top layer things without testing all the things beneath it simultaneously. This is likely due to a problem in my design and I would like to find a way to fix this in the future. I think it would possibly require some small rearrangements to my code but it would likely improve my designs significantly.