Garrett Jennings, A01491831

Design Insights

Grader Notes

During this phase I have tried to implemented the requirements 1, 2, 3, 5, 7, 8, 10, 11 While I have worked very hard to get to the state I am at and have made alot of progress toward each point. I'm going to be as honest and specific as possible to help grades be as easy as possible

- 1.1 Drone simulator writes the drone status to port 8890 at 10 hz in the background via thread, currently doesn't start once in command mode for simulator
- 1.2 DroneMonitor is successfully reading status
- 2.1 Currently not implemented, but as all the status is there It wouldn't be hard to add. All the ground work is there.
- 2.2 Currently not implemented, but as all the status is there It wouldn't be hard to add. All the ground work is there.
- 2.3 Currently not implemented, but as all the status is there It wouldn't be hard to add. All the ground work is there.
- 3.1 Currently not implemented, but would be a simple change to the DroneStatusStore. All the ground work is there.
- 3.2 Currently not implemented, but would be a simple change to the DroneStatusStore. All the ground work is there.
- 5.1 Yes can load from a file, this is currently managed in the MissionFactory class
- 5.2 Supported file types are .json .csv and .py
- 7 None of the specifics of this requirement have been implemented, but I have implemented Mission Type classes, you could have any mission type to support these features or any sort of behavior, miss types could be created to support any arbitrary behavior
- 8 This one is really close, I would argue that is done enough to pass the requirement. The drone simulator handles commands based on a dictionary of lambdas, you could pass it any dictionary with behavior for any command, as long as the function manages its own parameters. I just want to pull the action handling functions out of the DroneSimulator class to call this one totally complete
- 10 If I understand this one correctly, I have the DroneStatusStore which handles message serialization and validation this is use by both sides of communication
- 10.1 This one is a little abstract, not sure if I understand it, DroneStatusStore handles status messages, has functions for making changing status easier.

- 11 Again this is the DroneStatusStore, responsible for status messages and storing status
- 11.1 Can be used by both the DroneDispatcher and DroneSimulator

Experience Summary

I first started fixing the things that I feel I did poorly during the homework 1 phase, by completely overhauling my Menu class. Adding a Mission Flyer that is a better implementation of strategy pattern and changing my Mission types to be a proper template pattern rather than the wierd pseudo strategy pattern they were before. I spent alot of time trying to break up my design into components that could be easily tested in a vaccum apart from other components, I felt I made significant improvements from the previous homework. Implementing a Mission Library that allowed my missions to be stored and validated apart from the DroneDispatcher class which lead to a chain reaction of design changes that really flattened out my class hierarchy and allowed me to write more meaningful tests.

With a better foundation to work from I went on to improve how my mission factory would work. As well as making it possible to import missions from the different file formats. I then started started to design some of the new components I would build in UML. I started to get the hang of things and could focus on designing the software rather than worrying to much about the UML. I had a really hard time trying to conceptualize the ideas I need to build could be best implemented. I had no idea how what came to be know as my DroneStatusStore, DroneMonitor, and DroneSimulator would work but doing the UML Diagrams and unit tests before building them really helped me get going. I almost felt like I had a sort of writers block but this helped me break through it. I added multithreading to my DroneSimulator class so it would listen for commands in the background as well as report statuses, on two separate threads. The drone monitor Also uses a thread to listen and update status 10 times a second. As the deadline approached and I realized how far behind I was I began to hurry myself and I could feel my code design and quality start to slip which was a big realization for me about how quality control in the real world works.

In hindsight I felt I learned alot on this assignment, Seeing as how I failed to finish everything up in time I think I spent too much time trying to improve the foundation of HW1 rather than simply moving on the new functionality. I feel this is a delicate balance of improving existing code and adding functionality which will likely be a huge part of my life long career. As the deadline approached I mentioned how I noticed my code quality was beginning to fall apart. It is hard to say if me trying to design, test, and code as fast as possible really saved me that much time. I feel if I had stepped back and calmed down I think I would have been able to tie all the loose strings together in a more elegant way it would have probably taken just as long but in the end the resulting product would have been superior.

Another insight I had while building this assignment was that my school computer completely died on me. It won't even power on. So I had to buy a new computer and reinstall all of my tools. Luckily as I had been making frequent commits and all my UML diagrams have source code that was also in version control I was able to restore almost all of my work except for a few things.

Though it did take me a minute to get back into what I was doing. During this ordeal I saw a real benefit in choosing good tools like PlantUML git and github that really saved me a lot of time getting my environment set up again and preventing data loss. I cant imagine what would have happened if I had lost all my UML diagrams or something terrible to that effect.