4.1 - Identifying gaps and omission in the testing process

Evaluating my testing process, I've identified 4 key gaps I would like to cover:

1. Input space coverage and confidence
2. Performance / sustainability / scalability testing
3. Limited observability of internals
4. Environmental assumptions

Input space coverage and confidence

Throughout my testing process, I use randomly generated points to specify destinations for the drones. As with any random selection, this cannot possibly cover the entire input space for every pipeline or test run, this limits the amount of coverage I can safely say I have for any given test, and limits the confidence I can have that a push 100% works.

This approach could lead to rare edge cases slipping through many rounds of testing, however this is also the nature of the input space.

In an attempt to mitigate this, tests are run 10 times each by default, reducing the impact of the randomness by sampling a larger sample of the input space in a given test run. The amount of repetitions is easily configurable if a higher confidence is needed.

To further mitigate these issues in the future, a handful of harder tests could be constructed by hand, directly targeting potential edge-cases. Weighted random point selection could also be used to prefer points in trickier situations closer to no-fly zones, increasing the likelihood an edge-case is found. Finally, running the tests maybe 100 or 1000 times each for push request merges into master would help increase confidence that the master branch was working as expected.

Performance/sustainability/scalability testing

Currently I have entirely ignored tests for larger numbers of med dispatches, concurrent requests, and sustained high request rates.

This means that my response time claims and proposed scalability (due to being a stateless REST service) are not validated. Response times can vary under load, and without explicitly testing multiple instances running at once, I can't prove scalability.

I made these decisions because load testing required more infrastructure than I was ready to develop for this course, and because this kind of requirement was out of scope for the ILP coursework, and so was not something I focused on at all during it's development. This is also largely out of scope for a CI pipeline, which is an integral part of this coursework.

If I had to improve on this in the future, I would develop load-testing frameworks, attempting to find the tipping point for the performance of a single instance of my service, and then testing load balancing and automated scaling using a framework like Kubernetes.

Limited observability of internals

When developing my tests for this coursework, I wrote instrumentation to assist my tests, however I didn't add any to the internals of my existing functionality.

This makes failures somewhat harder to diagnose, as tests only observe and analyse the output of these functions, not the internal behaviour. This is somewhat mitigated because I can perform tests between functionality (like the unit testing I did directly on the AStar pathfinding), however there is still a lot of un-observable functionality.

However, as my focus for this coursework was on functional correctness, I didn't need to analyse the internals of anything, just the outputs. For a more extensive test suite, more instrumentation inside key functional code would be required, and the entire codebase could probably use with a refactor along with that to increase the modularity and testability, aligning more with the Single Responsibility Principle.

Environmental assumptions

Throughout my testing, I generally assume that the external environment is perfect, as I don't perform any testing on the inputs from the ILP REST service, and also haven't tested across the network, ignoring the potential issues an unstable network could cause.

For a production environment, these kinds of tests are crucial. Real-world environments are much noisier and imperfect than in development or testing, so ensuring a system thrives in that environment is essential.

Overall

Overall, there are some significant gaps in my testing if this were to be analysed as a production process. However, this level of testing is appropriate for a coursework, and the sections I've chosen to focus on have few enough gaps that I am happy with the overall level of testing.