# 4.3 - How the testing carried out compares with the target levels

## 100% Coverage of my stated functional requirements from LO1

In my testing suite, each of my stated requirements has at least one positive test (a test that is expected to pass), with the final requirement only having more than one because it was decomposed into multiple unit-level tests.

Whilst this counts as 100% coverage, in a production environment you would want a wider range of tests for any given functional requirement. This level of coverage is acceptable for this coursework, but falls short of a professional level of coverage for such core components of the system.

This shallow testing is due to the nature of the coursework, as the majority of the work is towards reflection and analysis rather than raw test coverage, and so more time was dedicated to the understanding of the material and writing of the portfolio than was spent developing tests. In a professional environment, the focus would be on the implementation of a deeper array of tests covering core functionality more completely.

## High repetition targets for random tests (>50 in CI)

As of now, every test is repeated 10 times in the CI pipeline. This was to ensure that pipelines were fast enough for quick iteration during development of both the pipeline itself, and the tests for this portfolio.

In a production setting, this could be upped, however given the current pipeline takes ~2 minutes, the targeted number of test repetitions would take this up to 10 minutes at the minimum, with a more likely target of 100 repetitions taking 20 minutes.

As the ILP coursework tested for correctness not performance, my implementation attempts to compute the most efficient possible delivery route. Whilst ensuring that this works in the setting of the ILP coursework, this comes with a heavy performance cost once you get to 10 or more deliveries, as the computation time scales with the number of permutations of the delivery set.

This low performance reduces the ability of the pipeline to run in a reasonable amount of time with higher repetition counts. To increase the feasible repetition count for the pipeline, either the performance of the delivery path calculation needs improved for larger delivery counts, or tests need to be performed on only a couple of deliveries - which may impair the effectiveness of the testing.

## Higher confidence in my non-functional performance (tests on larger amounts of deliveries, 10-15+)

As stated above, this system was not designed to handle more deliveries well, as it was stated during ILP that we would have to handle at most a few deliveries. Currently it takes almost 40 seconds to compute the drone paths for 10 deliveries, and this gets dramatically worse the more deliveries there are. In it's current state, my testing suite does not meet the target set out in 4.2.

In a production setting, this poor performance wouldn't be manageable, and would limit the applicability in real world use, however is acceptable at a coursework level. My system should be able to handle and split many deliveries at once, even at the expense of perfect solutions.

## Negative tests for stated functional requirements (tests that should intentionally fail - negative coverage essentially)

Due to the time constraints of this coursework, I focused on only positive test coverage, as this is easiest to analyse. Currently my testing suite doesn't contain any negative testing at all, and so falls completely short of this goal.

This is partly because we were explicitly told that there would be no marking for negative tests for the ILP coursework, and so none of the functionality is designed to recognise these failure cases.

In a production setting, the recognition of impossible tasks or ones that shouldn't be completed is crucial to ensure the safety and robustness of an implementation. The complete absence of testing present in my current suite wouldn't be suitable and would likely hold the system back from real-world deployment.