Test Implementation

This document contains the testing techniques, results and evaluation for the ILP Pizza Dronz project. The three requirements previously used in the Test Plan will be used again, to demonstrate the understanding of this learning objective. These requirements are:

1. The drone should be considered to be close to a location if it is within 0.00015 degrees to it.

2. The orders must be validated correctly. *(The system must ensure that the orders coming in are valid and if they are not, they will not be carried out. The system will go through each order when it is placed and check if all aspects of the order are valid such as an acceptable order date and of the restaurant is available on the day of the order. If the order is invalid, it will indicate this with an error message and this order will not be placed.)*

3. The system must not allow for the drone's flight path to enter any no fly zones.

Testing techniques

1. **The drone should be considered to be close to a location if it is within 0.00015 degrees to it.**

This is a Unit level requirement and therefore will not need a complex testing technique to verify it. A unit test will be the way that most of the unit level requirements will be tested. This is the process of testing individual components of a software in isolation and can be written without knowing the context or infrastructure of the rest of the code. Therefore we can take a Test driven approach in these cases, and write theses black box tests before development begins.

We will also use boundary value analysis which will test the scenarios at the upper and lower bounds of their accepted ranges. This will eliminate the need for hundreds of tests in a test suite. In this case a unit test can be written using JUnit that will see whether or not the function works for a value high than 0.00015, a value lower than 0.00015 and a value of 0.00015.

2. The orders must be validated correctly.

This is an integration level requirement and we will be taking a bottom-up integration approach. This will allow us to test all of the individual validations first before iterating and testing them together. We will use equivalence partitioning to split up this requirement into smaller unit level validation requirements. This will guarantee that all different possible inputs will be tested.

We can use category partitions testing for the unit level tests which will help to whittle down any unnecessary combinations of inputs for testing. Each unit level validation will be tested using boundary value analysis as a JUnit test, and then the tests will be integrated together and tested as one. For this we can use pairwise combinatorial testing, which will massively reduce to amount of tests that need to be done whilst still checking every possible combination of correct or incorrect inputs.

Finally, some functional testing will be used to test the connection between the order validator, the REST server and the rest of the system. As each of the separate components has already been tested with unit tests and then has been tested as a whole system, just some black box functional tests will be enough to fully validate this requirement. Structural testing should be carried out to evaluate the coverage of the tests.

3. The system must not allow for the drone's flight path to enter any no fly zones.

This is a system level requirement and therefore systemic functional testing techniques will be used to create multiple test cases. The requirement will once again be broken down by partitioning to allow for more smaller tasks to be tested. This will make the testing more reliable and more handleable.

Much scaffolding will be needed to test this requirement, it will require a simulation of the entire routing system.We will need to synthesise the different scenarios the flightpath could be calculated in, including different no-fly zones and different start and end points. Additionally, actual data from the rest server will need to be tested on, so that the integration is proven to work. This allows for testing of the integration between the drone routing system and the REST system.

The outputs of these simulations will need to be observed and put into GeoJson for manual visual checks. This testing will allow the developer to manually see whether are not there are problems with the detection of being in a no-fly zone, and can fix this error if it occurs. Finally, structural testing should be carried out to evaluate the coverage of the tests.

Evaluation criteria

Test coverage is the percentage of the system that undergoes testing. By guaranteeing that the test suite comprehensively covers a large enough portion of the system, it confirms that every component of the system has been tested, which ensures that no components weren't tested.

1. **The drone should be considered to be close to a location if it is within 0.00015 degrees to it.**

As this is a Unit level requirement, the tests can be evaluated on a pass-or-fail basis.

2. The orders must be validated correctly.

As this is a high priority requirement, the testing approach to it should be pessimistic. This involves actively looking for and testing potential failure scenarios and edge cases in a system. All of the Unit tests and the pairwise combinatorial tests will be done on a pass-or-fail basis. The functional, structural testing should require 80% code coverage.

3. The system must not allow for the drone's flight path to enter any no fly zones.

As this is a another high priority requirement, the testing approach to it should be pessimistic as well. This involves actively looking for and testing potential failure scenarios and edge cases in a system. Any Unit tests tests will be done on a pass-or-fail basis. The functional, structural testing should require 80% code coverage. If any of the manual visual checks are not 100 percent accurate, this will be a failed test and the code will have to be reviewed.

Results of testing & Evaluation of results

The full list of tests and results can be found in the tests folder of my ILP project. Below the summarised results.

1. **The drone should be considered to be close to a location if it is within 0.00015 degrees to it.**

This test passed with every piece of synthetic data it was given. The edge cases were considered and were successfully detected by the test implementation. There were no issues with this unit test.

2. The orders must be validated correctly.

Most of the Unit tests passed correctly with every piece of synthetic data give apart from the following:

- A date that had a month over "12" was not being flagged as an error.

- A CVV being characters that weren't numbers was not being flagged as an error.

Most of the pairwise combinatorial tests passed apart from the following:

- An order with too many pizzas was flagged with the incorrect error status, it instead had the status that the pizzas were from multiple restaurants which was not the case

These issues were all fixed relatively easily once flagged by the tests, and were either the result of mistypes or logical mistakes. This gave me confidence that the tests were picking

up on any mistakes correctly and that I could then iron out any bugs with the order validation as they arose. The structural tests found that there was a method coverage of 91% and a branch coverage of 86%. This is above the 80% goal and so I am happy with this. These test results indicate a good coverage of any potential problems with the order validation.

3. The system must not allow for the drone's flight path to enter any no fly zones.

While the drone's flight path had issues with testing elsewhere, it passed every test relating to avoiding no-fly zones. It also avoided them on every manual visual test that was done on different scenarios and simulations. This gives a lot of confidence that the flightpath of a drone will never enter no-fly zones during its routing. The method and branch coverage was high for any of the routing areas of the code. It does not need coverage in the order validation area as that is not needed for this requirement, therefore I am happy with the level of coverage reached by the tests.