Learning Outcome 2

Dan Lindsay

Outline of Test Plan Approach

The requirements listed in the `RequirementsDoc.pdf` file provide a framework for the following test plans. Of the requirements listed in the file, a majority were deemed crucial to the system's functionality and are tested in the following test suites. These requirements were prioritised due to the limited time and manpower available to create tests.

The requirements can be grouped into three test suites, those being:

1. Tests related to the correct functionality of the Order Validation system
2. Tests related to the correct functionality of the Coordinate Management system
3. Tests related to the correct functionality of the Command Line Argument Validation system

In order to test these system-level requirements, unit tests have been written, which when run in series will simulate each system in its working state. As such, each test suite is designed to replicate a given system present in the overall PizzaDronz system.

1 Order Validation Test Plan

As mentioned in the LO1 section of the portfolio, the test suites of the PizzaDronz system will feature Equivalence partitioning tests derived from the requirements listed in `RequirementsDocument.pdf`. The test suite listed below relates to the validation of orders received from the RESTful API. Each test was repeated 100 times, and inputs were randomised to provide a range of values to validate.

1. **The bank number of an order's credit card must be a 16 digit number, with no characters other than integers. The partitions of this requirement are as follow:**
   (a) The case where the credit card number is more than 16 digits
   (b) The case where the credit card number is 16 digits
      i. The case where the credit card number contains letters
      ii. The case where the credit card number contains no letters
   (c) The case where the credit card number is less than 16 digits
      i. The case where the credit card number is between 1-15 digits
      ii. The case where the credit card number is 0 digits
      iii. The case where the credit card number is Null
2. **The CVV of an order's credit card must be a 3 digit number, with no characters other than integers. The partitions of this requirement are as follows:**
   (a) The case where the CVV is more than 3 digits
   (b) The case where the CVV is 3 digits
      i. The case where the CVV contains letters
      ii. The case where the CVV contains no letters
   (c) The case where the CVV is less than 3 digits
      i. The case where the CVV is 1-2 digits

   ii. The case where the CVV is 0 digits
   iii. The case where the CVV is Null

3. **The expiry date of an order's credit card must be in the future and must be 5 characters long in the form of MM/YY. The partitions of this requirement are as follows:**
   (a) The case where the expiry date is in the correct format
      i. The case where the expiry date is in the past
      ii. The case where the expiry date is in the present month or in the future
      iii. The case where the expiry date is not a real date

   (b) The case where the expiry date is in the wrong format

4. **The number of pizzas in an order must be between 1 and 4. The partitions of this requirement are as follows:**
   (a) The case where there are more than 4 pizzas in an order
   (b) The case where there are 1-4 pizzas in an order
   (c) The case where there are 0 pizzas in an order

5. **The total price of the order must be the delivery cost, £1, + the prices of all pizzas in the order. The partitions of this requirement are as follows:**
   (a) The case where the specified total price of the order is more than the actual total cost
   (b) The case where the specified total price of the order is the same as the actual total cost
   (c) The case where the specified total price of the order is less than the actual total cost

6. **The pizzas in the order must come from the same restaurant, and that restaurant must be open on the day the order is placed. The partitions of this requirement are as follows:**
   (a) The case where the pizzas in the order come from the same restaurant.
      i. The case where the restaurant is open
      ii. The case where the restaurant is not open

   (b) The case where the pizzas in the order come from different restaurants

For the sake of repeatability, mock orders were created for each test case, as the RESTful API may not always be available.

2 Coordinate Management Test Plan

This test suite, like the Order Validation test suite, will be made up of Equivalence partitioning tests derived from the requirements listed in `RequirementsDoc.pdf`. The test suite below relates to the coordinate management system, which is used by the PizzaDronz system to plan its flight paths. Incorrect functionality of this system could result in a flightpath which does not adhere to key requirements of PizzaDronz. Certain tests were repeated with randomised values to provide a wider range of values to test.

1. **The drone must move to a point which is 0.00015 longitudinal/latitudinal degrees or less away from it. The partitions of this requirement are as follows:**
   (a) The case where the next position is 0.00015 degrees or less away from the drone
   (b) The case where the next position is more than 0.00015 degrees away from the drone

2. **The drone must stay in the central area until it has delivered a pizza. In order to do this, the drone must recognise when it has entered the central area. The partitions of this requirement are as follows:**

   (a) The case where the drone is inside the central area
   (b) The case where the drone is outside the central area

3. **The drone must avoid No Fly Zones. In order to do this, the drone must recognise when its next position is inside a defined polygon. There was one partition of this requirement:**
   (a) The case where the next position is inside a No Fly Zone

4. **The drone must move in the eight compass directions, when the angle is set to 999 the drone will register the move as a hover. The partitions of this requirement are as follows:**
   (a) The case where the angle is a multiple of 45, up to 315 degrees
   (b) The case where the angle is 999

For the sake of repeatability, a mock central area was created and mock points were generated which imitate those which could be taken from the API.

3 Command Line Argument Validation Test Plan

This test suite is more simple than the previous test suites, as there are only three possible partitions. These partitions are:

1. The case where the date and the URL are correct
2. The case where the date is in the incorrect format
3. The case where the URL is incorrect

To test this requirement, arguments were constructed which were then used to access the RESTful API. As such, the date chosen for testing is past the due date for this report, so that the tests may be run successfully.

4 Running in under 60 seconds

One of the most crucial system requirements, beyond the application running successfully and correctly which was tested above, is the application's ability to run in under 60 seconds. To test this, the application was fed orders in large quantities to all possible restaurants. A timer was implemented to measure the elapsed time of the application, and to ensure that the application runs in under 60 seconds.

Test Plan Quality

The tests included in the above test plan are all unit and system level tests, which provide a close-up and birds-eye view of how the application functions respectively. The test plan laid out above provides a good range of unit tests for crucial parts of the system, ensuring that the order validation, coordinate management, and command line validation systems function correctly. These tests cover most, if not all, possible issues which could come from incorrect data being fed into the application from the RESTful API. The system level test, wherein the system must run in under 60 seconds, shows that the PizzaDronz system runs correctly when given correct inputs (being a date for which orders were placed, and a URL to access a RESTful API). This system test also proves that the PizzaDronz system meets one of its key requirements, being that the system can run fully in under 60 seconds.

Given more time and manpower, however, integration level tests would be implemented as the current test plan does not include any. The main part of the system which would benefit from integration testing would be the JSON Handling system. Data corruption which occurs during the JSON deserialising stage could render the system unable to produce a correct flightpath, even when the data on the REST service is correct. This would be a failure to meet the specification.

Evaluation of Instrumentation

The instrumentation for each part of the system is detailed in their respective sections of this supplemental document. The mock data created is done so with elements of randomisation, which contributes to a wider range of values being tested. Testing on a wider range of values allows for a wider net to be cast, and prevents tests from being too specific to the values they were written with. The mock data takes the form of Java Objects, whereas in actuality the values being fed through the application are decoded from JSONs. The decoding of values from JSON files introduces the risk of data corruption, which the mock values do not capture. This means that the testing suites do not guarantee the applications performance on JSON data.