Learning Outcome 4

Dan Lindsay

Gaps and Omissions in the Testing Process

The largest gap/omission in the testing process is the lack of tests related to the JSON handling sections of the PizzaDronz system. Discussion of this omission, as well as the omissions present in the implemented test suites, is included in this section.

1. **Lack of Documentation**
   A noticeable omission from the test suites created for the PizzaDronz system is the lack of documentation for any test suites. The reason behind this omission is simply a lack of time. The development and implementation of the test suites took a substantial amount of time, and coincided with other coursework deadlines. Documentation would provide context for and reasoning behind each test, which could aid developers who aim to fill in the omissions stated in this supplemental document. Given more time, relevant documentation would be produced, as it would provide an easier jumping-off point for other testers (or even myself at a later date).

2. **JSON Handling System**
   Another large omission from the test plans stated in `LearningOutcome2.pdf` is the lack of tests written for the JSON Handling sections of the PizzaDronz system. As shown in `LearningOutcome3.pdf`, the Line Coverage for the Input Validation section of the syste, was around 67%. A third of the lines not being reached is quite a staggering number, as it seems to imply that the Input Validation System is not adequately tested. This is not the case, however, as the Input Validation System's tests are all related to the Command Line Arguments, which was the focus of the test suite. The lines missing from the Line Coverage metric are actually those related to the JSON Handling sections of the system. This section has two different types of error associated with it:

   - The RESTful API does not have the appropriate JSON files
   - The URL provided is malformed

   In the first case, tests were not created as it would require complex mock data to be created. The RESTful API provided for the PizzaDronz system is always alive, and always contains the correct data to be accessed, meaning the system could not be tested with simply the RESTful API. Again, the development and implementation of software tests for the PizzaDronz system was majorly constrained by time and manpower. In future, the JSON handling system could be tested in the same way as the other systems present in PizzaDronz, using Equivalence Paritioning unit tests.
   In the second case, the URL being malformed is actually caught and handled in multiple places throughout the Input Validation system, and so very specific edge-cases are required in order to access all of these error cases. The solution to this lack of coverage is quite simple, these edge-cases could be tested individually by other testers in order to increase the Line Coverage metric.

3. **Order Validation System**
   As evidenced by the 100% Line Coverage in Table 2 of `LearningOutcome3.pdf`, there were no gaps in the actual line coverage of the system. Additionally, all partitions identified in the test plan were covered in the implemented test suite, suggesting that the Order Validation system is well-tested. The mock data generated for the Order Validation System are all generated as Java Objects, which is not the case for the actual deployment of the system. More integration tests which involved the Order Validation System could prove fruitful for testing the system in a way which is closer to the actual function of the app.

4. **Coordinate Management System**
   All identified partitions of the Coordinate Management System are implemented in the test suite, and so the desired coverage of 100% is met. There are, however, two lines which are not reached, resulting in a Line Coverage of 93%. Upon inspection, these two lines are related to the name of the central area, which is provided by the RESTful API. A test could have been written to supply the Coordinate Management system with an area that does not have the "central" name, but these lines were overlooked during development of the tests.

5. **Command Line Argument Input Validation System**
   As stated in the JSON Handling System portion of this section, the Line Coverage for the Input Validation System was 67%, meaning a third of the lines related to the Input Validation System were not reached by the test suite. One can see, however, that all identified partitions were covered in the implementation. Due to the Command Line Arguments being validated in the same class as the JSON Handling system, the Line Coverage has been rendered misleading and does not serve as a proper metric for how well-tested the Command Line Argument Validation system is. In terms of actual partition coverage, the system is adequately tested.

6. **Running Under 60 Seconds**
   This requirement was tested adequately, by providing the system a large number of orders for it to route to. It was easy to see the results, as the biggest metric for success was whether or not the system could run in under 60 seconds. The PizzaDronz system also achieved a Line Coverage of 88%, which was impressive considering the lack of a test suite for JSON Handling. Another possible avenue for stress-testing the entire PizzaDronz system could be introducing new restaurants which are all over the globe, forcing the routing algorithm to generate flightpaths to places thousands of times further than it would ever reasonably have to route to. These tests do focus on the most intensive part of the system, being the routing algorithm, so further tests could also be written to stress-test other parts of the system to better model user interactions with the system.

Target Coverage/Performance Levels

For any further testing of this software, testers should aim for coverage of 100% in their Primary metric (whether this be partition coverage, boundary coverage, or otherwise). Testers should also aim for a coverage of 100% in Secondary metrics, such as Class, Method, and Line coverage. In terms of improving the tests already implemented, testers should focus on increasing the Line coverage metric of the Coordinate Management and Command Line Validation systems.

Current Coverage/Performance Levels

The test suites for each system do indeed reach the target Primary coverage levels of 100%, as all identified partitions were implemented. 100% Class and Method coverage are also achieved, implying that the test suites will access all methods present in the systems being tested. Line coverage, however, does not meet the ideal coverage of 100%, due to the lack of JSON Handling tests. Given more time and manpower, this integral part of the system would be tested, and would lead to the Line coverage target of 100% being met for the entire PizzaDronz system.