LO4 - Software Testing

B196275

January 23, 2024

1 Gaps and Omission

Due to the time constraints, it was infeasible to satisfy/test all requirements, and this could be considered as a large "omission", however for this portfolio, I will discuss gaps and omissions within the testing that I carried out.

- **No Condition Coverage Testing**: Condition coverage helps ensure that all possible decision outcomes within the code are tested, thus providing a more comprehensive evaluation of my code in comparison to branch testing (condition coverage implies branch coverage). However, intellij doesn't offer a feature to monitor condition coverage, thus preventing me from carrying it out.
- **Insufficient Test Data**: Almost all the test data is synthetic and thus may not be representative of the data that the application could encounter whilst in use. However, due to the nature of the software, the synthetic data is likely adequate, but a more diverse set of inputs may instill more confidence in our test results.
- **Insufficient Test Documentation**: Given more time, the tests could have been better documented for the software's maintainability. This would make it easier for future developers to adjust and add more tests.
- **Rest Server Testing May be Inadequate**: Due to a lack of time and not being in control of the REST server, I was unable to perform reliability or performance testing on the REST server (perhaps through some form of stress testing), thus
- **No Mutation Testing Was Carried Out**: Due to a lack of time I couldn't implement any mutation tests on my code. Mutation testing is where systematic alterations are created in the code to create mutants. Tests that do not detect these mutants are deemed week, however, if I could see that my test-suite eliminates these it helps me verify the strength of my tests.

2 Target Coverage/Performance Levels

Functional Testing:
Target: All tests (100%) must pass, as these are primarily based off the specification in the test plan, thus are well-suited to evaluating the functionality of the system.
Actual: All tests passed as shown in Section 4 of `LO3.pdf`, thus we can be confident that the system functions as expected.
Next-Steps: No further action is required to meet the target.

Structural Testing:
Target: 100% method, line and branch coverage across all tested classes.
Actual: This criterion was not fully met. It was not possible to reach 100% branch coverage for both LngLatHandler and PathFindingAlgorithm. This is because the cases that would have to be tested for those branches to be covered are extremely rare, and thus it was difficult to craft test cases such that those branches were entered.
Next-Steps: Given a longer time-resource I would be able to generate the cases that would allow me to for example obtain full branch coverage with the PathFindingAlgorithm tests. There is a case that is invoked only when we need to backtrack which is an extremely rare occurrence and thus is not

worth the time required to create the case. I also feel upon reflection that 100% branch, line, and method coverage is unrealistic across all the classes I am testing, and a target value of 85% might be more realistic in the future.

Model Based Testing:
Target: 100% path coverage of the Decision Flow Graph, and the paths generated by the PathFindingAlgorithm are valid.
Actual: This criteria was met, I only had to write 2 test cases, due to the nature of the algorithm I was testing. I could see the branches associated with the node selection were exhausted after analysing the branches that were entered. The paths created were valid according to our definition of a valid path, thus I can be confident that the algorithm filters out invalid moves as expected.
Next-Steps: No further action is required to meet the target.

Performance Testing (Statistical):
Target: All 100 executions are each completed in under 60 seconds, and the average time of each execution should be below 60 seconds.
Actual: This criterion was met, as shown in the results in `L03.pdf`, the average was well below 60 seconds (3.11989 seconds), and the test written to assert the runtime for each execution passed, thus we can say that our software has achieved its performance quality that the stakeholder insisted on being prioritised.
Next-Steps: No further action is required to meet the target.

Fault Level (Statistical):
Target: By the end of development the software should have 0 faults (defined as a test case that fails).
Actual: This criterion was met as shown in Figure 1, the number of faults progressively declines as we approach the final major build, and after build 6 we can see that the number of faults in our software is 0.
Next Steps: No further action is required to meet the target.

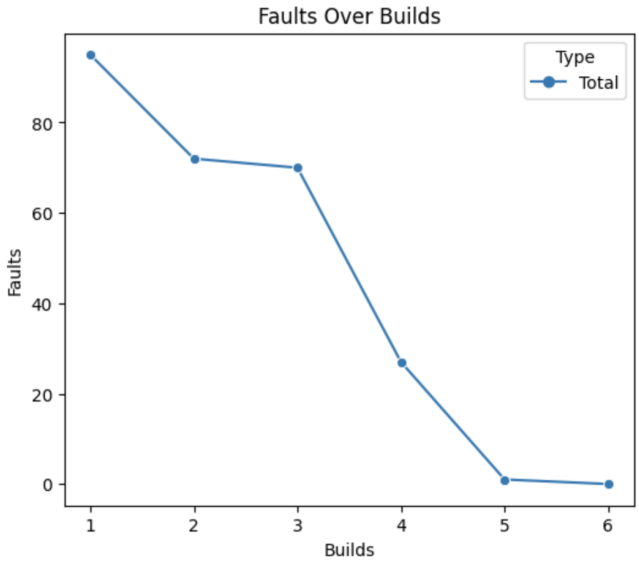

Figure 1: Graph showing faults being eliminated throughout the project