SWEN301-P3 ramireana 300365371 Ana Ramirez

The Testing Process

Since the purpose of the assignment was to test the provided program, it came as a surprise to me when I found quite a few bugs in the code while conducting my own tests. The specification itself was also flawed because of the numerous contradictions within it, as well as the ambiguity of it. I found that I had to make a lot of assumptions and to follow my own instincts, and as a result of this I could never be really sure if what I was doing was correct or not. Because of these factors, the testing process was not as smooth as I expected it to me and unfortunately meant that I was not able to complete as much of the assignment as initially anticipated.

One of the first challenges that I came across when starting the assignment was the challenge of processing the assignment specification itself. Vague statements such as “mail originating from overseas for New Zealand destinations is delivered at no cost” threw me off as I was left with questions such as “is that no cost for KPS? Or for the customer?”. It meant that I spent a lot of time stressing over whether what I was doing was correct, rather than moving on and creating more tests if I had a more specific specification. Further parts of the specification that I found perplexing were statements such as “Domestic air priority and domestic standard priority are the same”. This brief statement meant that I had to draw my own conclusion that this was referring to the fact that the price for charging the customer was the same for both domestic standard and domestic air. This further confused me since it opposed the statement about how a higher priority meant a larger cost. A lot of the time I drew my own conclusions but I was always left with a of feeling of unease as I never felt on the right track and questioned if what I was doing was right at all. As a result of this, around half of my tests fail, and half pass, because of the contradicting requirements within the specification. I was not sure if I was meant to be writing tests that conformed to the data.xml, since there were also many issues within this file. I added a few data points to the xml file in order to help with my testing, but am still unsure if I was allowed to do this since the specification made no comments on this. Within some of my scenarios I gave brief descriptions on any assumptions I made and why I was doing certain things. Because of the conflicting specifications and the failing tests, if I had more time, I think I would have written more tests to cover all these failing tests as well, by which I mean possibly write tests that contradict each other in order conform to the specification and the xml file.

Design and Quality Analysis

Kelburn Postal Service has its own software tool, KPSmart, that allows it to monitor all its mail as well as track its progress between its distribution centres. In this report I will give a brief overview of KPSmart’s design and quality as well as make comments and draw conclusions on the assignment provided to us, which was to test the KPSmart program given a specification and the KPSmart source code.

To a fundamental level, the KPSmart system works, but during my time creating tests for it, I have found several flaws that hindered my testing process and made it a lot more difficult than I believe it needed to be.
Delving into a few serious flaws, there are a few instances of completely non-functional code - for example, the getRoutes method in the TransportMap class is hardcoded to return null, and is thus completely useless and unusable. Since we were not permitted to change the source code of the project, I had to ignore this method altogether in my testing process.

I found consistency to be a common issue in the design of the program also. For example, an issue that appeared a few times was when the origin location of a parcel is unknown, a NullPointerException is thrown, whereas if the location was an **unknown** location, a RouteNotFoundException would be thrown. In good design, both of these issues should probably throw a NullPointerException in order to uphold consistency and promote ease of readability.

A quality factor that I found to be lacking was that of a lack of documentation throughout the source code. Because my task was to test the program, I had to understand the code at least to a basic level, and found that the poor amount of documentation made this a lot more drawn out than it needed to be. Because of this, this is a factor that contributes to the programs overall lack of quality.

Finally, the issue that had the greatest negative impact on my ability to test the program was the xml files. The data.xml file provided contained the data that I was meant to use to conduct my tests was flawed in many respects – prices were sometimes wrong, there were inconsistencies, and was not very extensive in its content. There was a lot of unrelated data which made it hard to make sense of things a lot of the time, and was frustrating to work with.

The overall functionality of the KPSmart system works but once you delve deeper, there are certainly some areas where there is a lot to be desired, and the amount of errors, inconsistencies, and non-functional code gives off an impression of a lack of code quality and poor design.

Overall thoughts:
To conclude, I would say that the KPS system is mostly functional, and its major flaw is in its data.xml file. The xml had multiple issues with incorrect data and I believe it may need to be scrapped and re-written in order for a person to test the system fluently. I believe that instead of xml files, a database could be used instead to make the data a lot easier to read and to group all related data together nicely. The system itself is operational and holds up while still having room for improvement; however this is not the major flaw and can be improved without too much difficulty.