CIS*3190 Software for Legacy Systems

Assignment 3 - Cobol Re-engineering

Maneesh Wijewardhana (1125828)

- The biggest challenge when re-engineering the Cobol program was trying to understand the flow of the program in general. Since Cobol requires you to structure your program in a specific way, it was difficult to know which sections did what such as “procedure division”, “working storage section”, etc. However, once I learned most of the syntax and program structure, it started to make more sense and easier to trace through.
- I did not find it challenging to find and fix the bugs in the code due to the compiler spitting out errors one after the other with details on what went wrong. For example, in the beginning, there was a simple syntax error that was caught by the compiler which was an easy fix. After this, a variety of file I/O warnings popped which hinted at the fact that I need to define file I/O in order to get the program to compile.
- The part of the Cobol program that was the most challenging to comprehend would be the working storage section where variables were being declared and defined. It was difficult to know at a first glance, what each of the numbers that prefixed the name of the variable meant. Along with this, the idea of filler variables and the pic statement took a long time for me to wrap my head around. Since the languages I’m used to always follow a similar procedure of data type and name to declare/define variables, looking at the Cobol way seemed really verbose and hard to follow. I also think that the concept of tables and records was confusing at the start but after some more research on how Cobol handles data structures, it made more sense to me.
- I could not imagine re-engineering a 10,000 line program consisting of Cobol. The verbose nature of the language would put me off, and I feel as though it would take a significant amount of time to accomplish the task. If the code is poorly organized, lacks proper documentation, or has been modified by multiple developers over time, the task would most definitely be challenging. Also, if the task was similar to this assignment where re-engineering meant removing any legacy features and modernizing the program, having to do this for a 10,000 line program would take a significant amount of time.
- I think Cobol has survived as long as it has because of early institutions that acquired the language to build products as we know it today. Organizations such as banks and government have existed since the beginning and since Cobol is an early language, these organizations were built on top of it. Due to this, the migration to modern languages for these organizations would cost more than to keep paying existing developers to write and maintain Cobol codebases. In many cases, it makes more sense to continue using Cobol and modernizing it where possible rather than starting from scratch with a new system. Also, it seems as though Cobol is a highly scalable language that can handle large amounts of data and complex business logic. This would make it ideal for large organizations as mentioned earlier that need to process massive amounts of data quickly and efficiently.