Erica Muschelli

COSC 603 APG

21 FEB 2016

Software Testing and Maintenance

Spring 2016

Project #1 - Reengineering Legacy Systems

**Task 5 – Program Modification/ Reengineering**

Of all the tasks that I needed to complete, I found this task to be the most time consuming and difficult to accomplish. Normally if I was to re-engineer a legacy system, I would first read the developers documentation to help build an understanding of the software application. This application lacked the necessary background information. Much of the information provided throughout the source code was high level (general) with no background information provided. As a result, I struggled with determining what the correct output would be when running this program. Not being able to actually run the code in it’s original environment to see how the input was processed left me nothing to compare my results to since the comments within the code didn’t explain well how the output was being generated.

I found the application to lack proper and complete documentation. Also, the lack of cohesion (how the different parts work together or not) in addition to the age of the language greatly impacted my ability to properly reengineer this program from Fortran77 to Java. Since I was not an “expert” or even familiar with “Fire Danger” I wasn’t able to fully understand the execution behavior of this application. Understanding the inputs and outputs in addition to the expected data flow, the control flow, and the core algorithms used added to my frustration in attempting to re-engineer this system. Having knowledge of these factors would have made the re-engineering of this application much easier and reducing and/or prevented the potential for software bugs that could have been introduced due to the lack of knowledge.

When reviewing the program code, the presentation of the code was not good. I found it extremely easy to become lost in the application flow. The lack of indentation made it difficult to understand what lines of code was included/executed by the DO loops. The code was not modular resulting in duplicate code being used throughout the application. For example, “Percip” calculation (Rain) on lines 1 and 12, is found on two locations within the source code of the program. The lack of commenting within the program made it difficult understand what each of the conditional logic was performing as well as the expected output. This experience has left me with a solid understanding as how a program should not be written. As difficult as I have found Task 5 of this project to be, the lessons I have learned in the process are extremely valuable. I have learned the importance of incorporating those elements that increase “Human Readability” when developing software. Human understanding is essential when it comes to software maintenance and maintainability. How is one able to maintain, modify, and enhance an application that is not fully understood? The re-engineering of any legacy system cannot be successfully performed without “understanding”.