# 

# 

# 

# 

# **Lab. Report #5 – Software Reliability Assessment**

# **Group #:** 17

**Student Names:** Matteo Morrone, Adeshpal Virk, Sam Farzamfar, Taimoor Abrar

# 

# 

# 

# **Table of Contents**

# Introduction

# Assessment Using Reliability Growth Testing

# Assessment Using Reliability Demonstration Chart

# Comparison of Results

# Discussion on Similarity and Differences of the Two Techniques

# How the team work/effort was divided and managed

# Difficulties encountered, challenges overcome, and lessons learned

# Comments/feedback on the lab itself

# **Introduction**

In this lab we were tasked with using reliability assessment tools. By looking at it using reliable growth testing and using a reliability Demonstration Chart. By using these tools we can find the failure count and time interval to figure out MTTF, failure rate, and reliability.

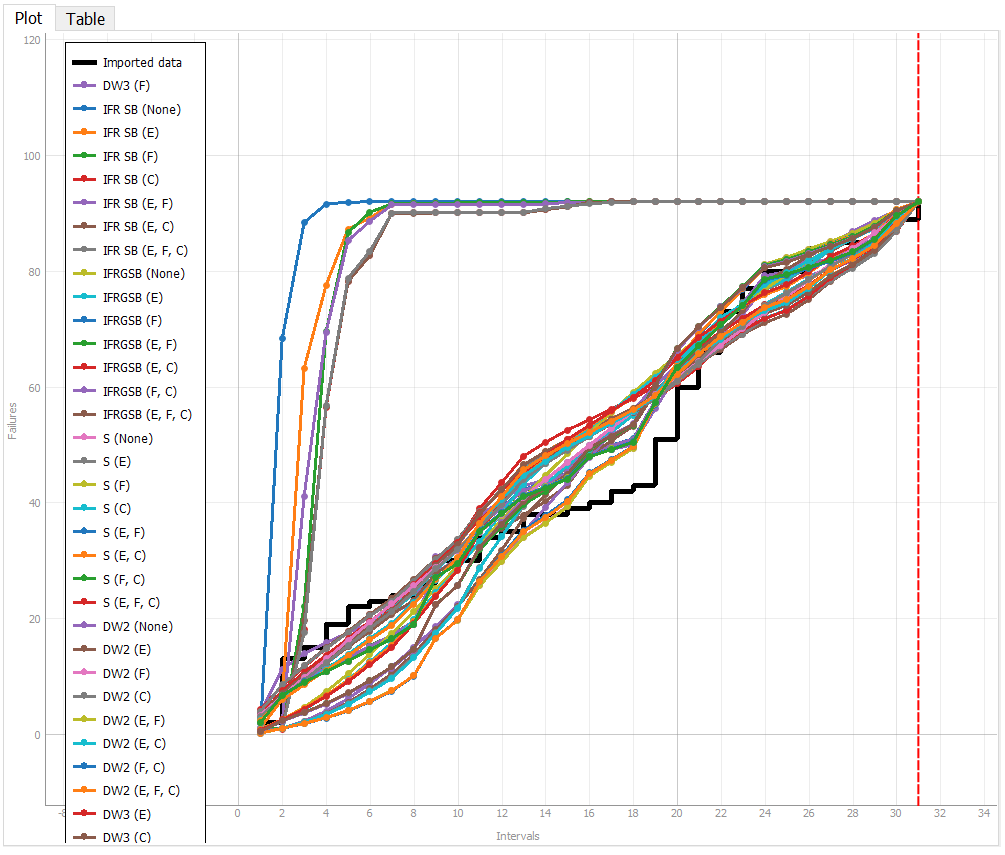
# 

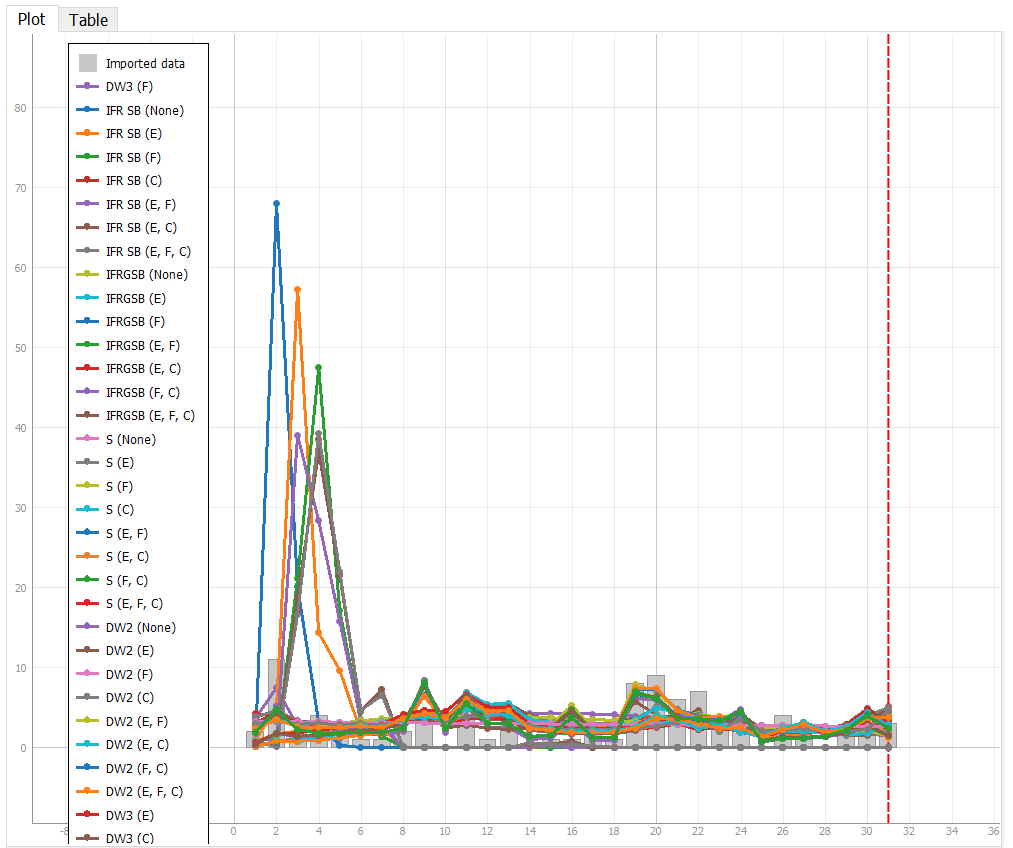
# 

# 

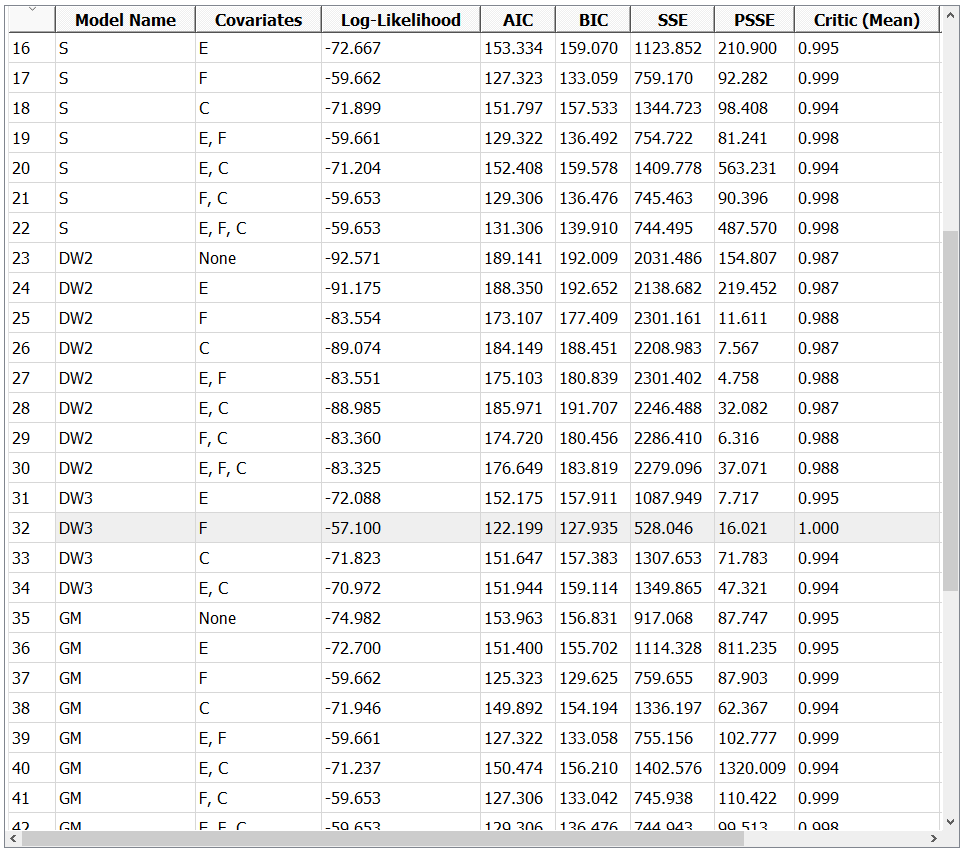
# **Assessment Using Reliability Growth Testing**

First we decide that we would plot every possible graph using C-SFRAT which gave us the following plots

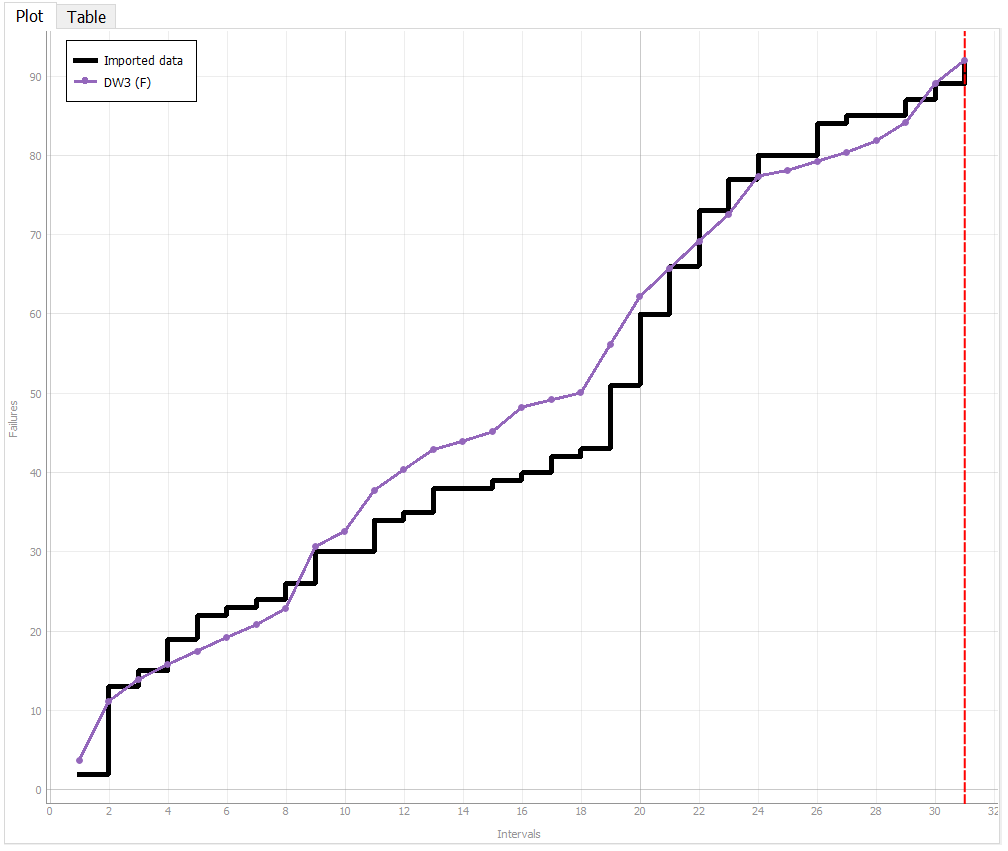


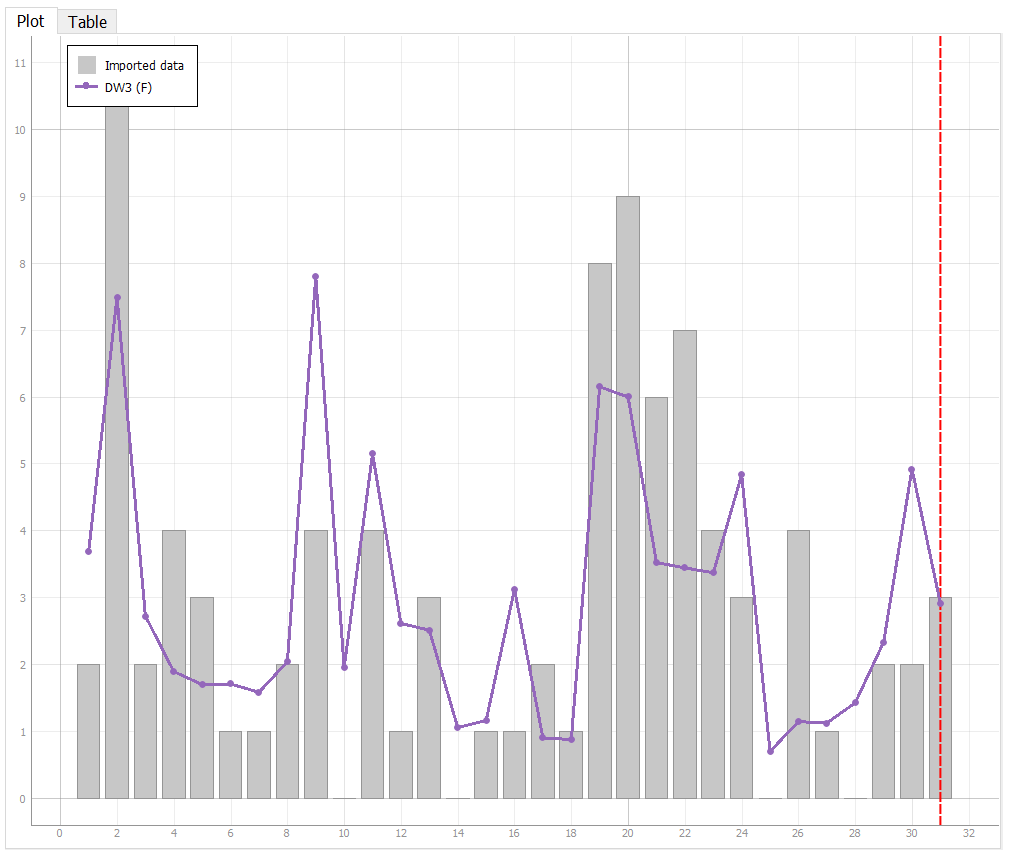


Then we looked at the following table to compare our graphs which told us that the DW3 graph with covariate F was our best plot.

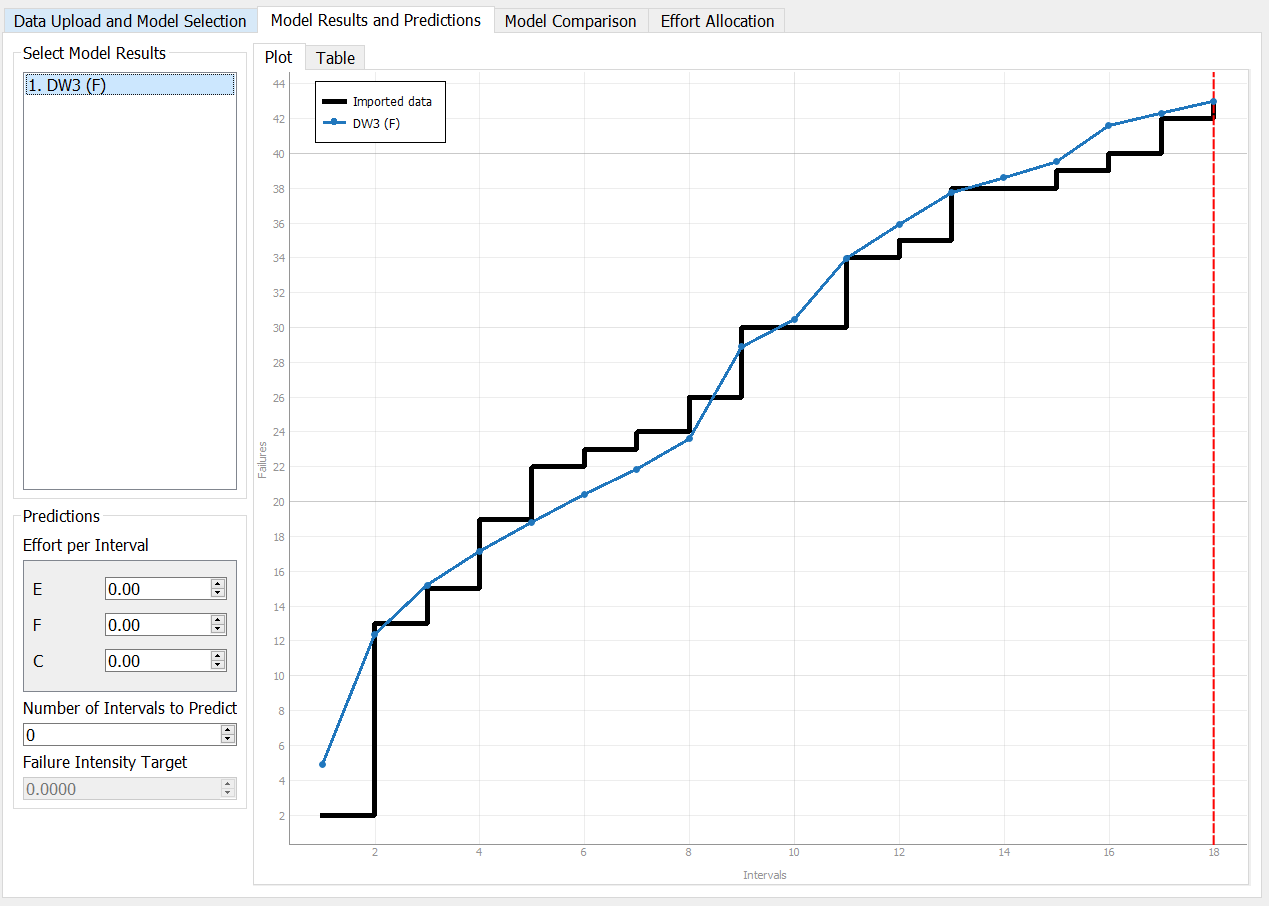


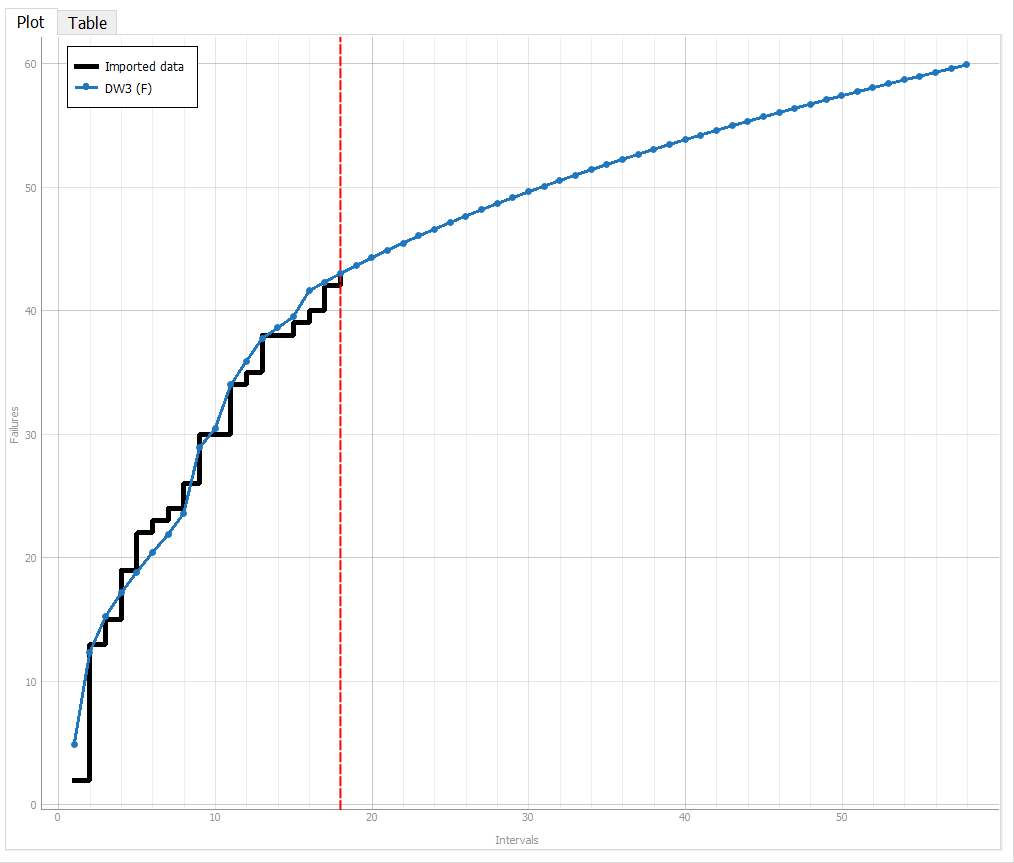
With DW3 and covariate F we got the following best fit graphs.





After looking at the above graph and calculating the Laplace we decided that our data should be limited to 18 intervals.



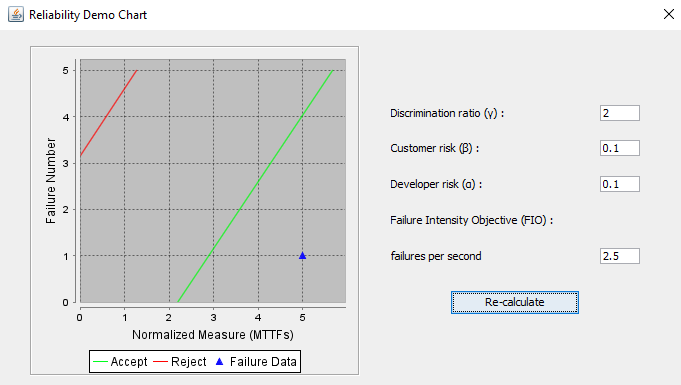
Then we predicted 40 Intervals and saw that it is steadily declining.

Calculating MTTF will allow us to get failures per time interval.

MTTF = (Failures at interval 18) / ( Interval 18 )

= (43.00) / (18) = 2.39

# **Assessment Using Reliability Demonstration Chart**

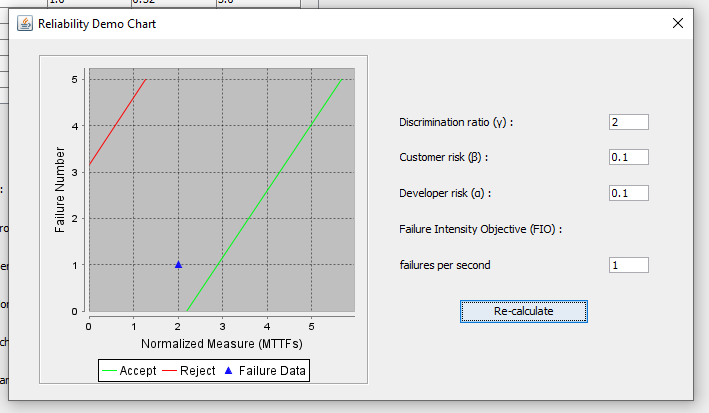


If we have a failure tolerance of 2.5 failures per second then our failure data is accepted. With a Discrimination ratio of 2, Customer risk of 0.1 and Developer risk of 0.1.

We can see that when we change the doscrimination ratio:



Our program becomes more accepting with a larder Discrimination ratio and less accepting with a smaller discrimination ratio.



We can also see that if we change our acceptance range to 1 failure per second then our data results in ambiguity and we need to do further research. This shift of the graph to the left is a result of requiring better test data to determine if the program should be accepted or rejected.

# **Comparison of Results**

After looking at the results of both methods it can be concluded that the program is reliable. This is due to the fact that the C-STRAT program provided us with the trend in failures per second and failures at any point in the program's run time. Finding the MTTF can be found with further inference of the graphs.

# **Discussion on Similarity and Differences of the Two Techniques**

The primary difference that can be found in the two methods is what the techniques are based on. In Reliability Growth Testing takes into account failure count and MTTF.

In the Reliability Demonstration Chart it takes into account inter failure times and MTTF. Both graphs create a visual for the data included over a specific range of time. In the Reliability Demonstration Chart the failure data is analyzed with the accepting and rejecting conditions.

# **How the team work/effort was divided and managed**

Teamwork is a fundamental part of the software testing process. This class has enabled us to hone our interpersonal skills and in turn allow us to collaborate and in turn split the work evenly. The lab comprised of 2 components the first being reliability growth testing and the second being the reliability demonstration chart. Our team decided that the work would be split evenly and we split both the sections amongst two people each. After each section was completed by the designated pair, the alternate section was peer reviewed and tested. At the end our group had a meeting where we went over each component of the lab step by step to ensure everything was correct and the work was divided evenly.

# **Difficulties encountered, challenges overcome, and lessons learned**

Similar to the last lab, our first difficulty some of our team members encountered was with the software itself. The download itself took upwards of two hours on some machines as our computers refused to believe that it was not malicious. We could not get STRAT to work in the slightest due to a lack of instruction so our group had to end up resorting to the C-SFRAT software. After switching softwares to remedy this issue, we found it rather difficult to work in the second software as well, also due to a lack of instruction. The data had to be manually manipulated in order to be used as well. After a significant amount of time we were finally able to get enough of the software working to complete our lab assignment.

# **Comments/feedback on the lab itself**

The lab itself was very challenging with a clear lack of explanation in the documentation. The documentation provided very minimal instruction and only at a very basic level, throwing us to the metaphorical wolves for the bulk of the assignment. However, the lab was interesting as it gave us an exciting glimpse into some tools which presumably will be used in the field. The introduction of tools such as C-SFRAT that we could potentially be using in our careers in the future was quite beneficial. Though there was a helpful introduction preceding the document explaining useful terms and concepts, the lab seemed to have much less instruction than the previous 4 labs, and difficulties with the software itself were rather difficult to troubleshoot.