Introduction

I set out to do some dynamic analysis on the org.apache.commons.net project, focusing more around the FTP class. I used Aspect Oriented Programming using a tool called AspectJ to inspect method / class executions and calls. I explain my approach to analysing the FTP class and what technologies I used. Afterwards I give some thoughts around the analysis of the system.

Implementation Details

For the dynamic analysis, I created a tool that could inspect source code during runtime execution to monitor what the system is doing by the user while recording important information regarding method / class executions and calls. I used a methodology called Aspect Oriented Programming and a tool called AspectJ that adheres to the Aspect Oriented Programming. The dynamic tool uses Java, that reads in the AspectJ logs, it then sorts out data and converts it the appropriate data structure to be used for Visualization graphs.

We inspect all methods that belong to the org.commons.apache.net that was executed from the following unit tests –

* FTPClientConfigFunctionalTest
* FTPClientConfigConfigTest
* FTPClientTest
* FTPCommandTest
* ListingFunctionalTest
* TestConnectTimeout

Just a side note the FTPClientConfigFunctionalTest and ListingFunctionTest unit tests did fail when I ran them. So the results that I got might not be 100% accurate when compared to a successful pass on those unit tests.

Those unit tests were recorded with our aspects that inserted the recorded data into a log file tmdcTrace.log which is all trace method calls and tmteTrace.log for all methods executed. They are appended with a numeric value that indicates the unit test that was run i.e. FTPClientConfigFunctionTest = 1. There is a total of 12 log files. 6 of which is tmdcTrace and 6 is tmteTrace.

The dynamic tool then merges all tmdcTrace.log files into one file called “totalTmdcTrace.log” and all tmteTrace.log files into “totalTmteTrace.log”. The totalTmteTrace.log is then read in by the tool, it then gets the average execution times for all methods, it also gets the average execution times for all classes, and it then gets converted into two csv files, one for method execution times and one for class execution times. The totalTmdcTrace.log is read in and all duplicate relationships are removed and then it gets converted into a digraph format ready for Graphviz to use.

The following tools were used for visualising the data we prepared using the dynamic tool.

* Excel 2013 – Bar chart. This was used for the “Which classes required most of the time to execute and “Which methods required most of the time to execute”
* Graphviz – dot digraph. This was used for “Which methods called each other most frequently”