

System Re-engineering  
Assignment 2:  
Dynamic Code Analysis

Name: Shashidar Ette

User-id: se146

**NOTE:**

## Table of Contents

Dynamic Code Analyzer .....	3
Design.....	3
Trace logging .....	4
Trace log Analysis.....	4
Process .....	4
Output generation .....	6
Classes and Methods CSV .....	6
Class Diagram.....	6
Time Series and phase execution analysis.....	7
Final Overview.....	8
Figure 1 Dynamic Code Analyzer class design .....	3
Figure 2 Part of the class diagram generated by the tool showing XYPlot.....	7
Figure 3 XYAreaChartTest time-series plot .....	7
Figure 4 XYBarChartTest time-series plot .....	7
Figure 5 XYLineChartTest time-series plot.....	8
Figure 6 XYStepAreaChartTest time-series plot.....	8
Figure 7 XYStepChartTest time-series plot .....	8

# Dynamic Code Analyzer

Dynamic code analyzer (DynamicCodeAnalyser.java) is implemented as a console application. This application interacts with different elements created as part of the assignment 2 course work.

The application and the solution are generic and can be applied for dynamic analysis of any library with minor changes.

The basic design and implementation details are in following sections.

## Design

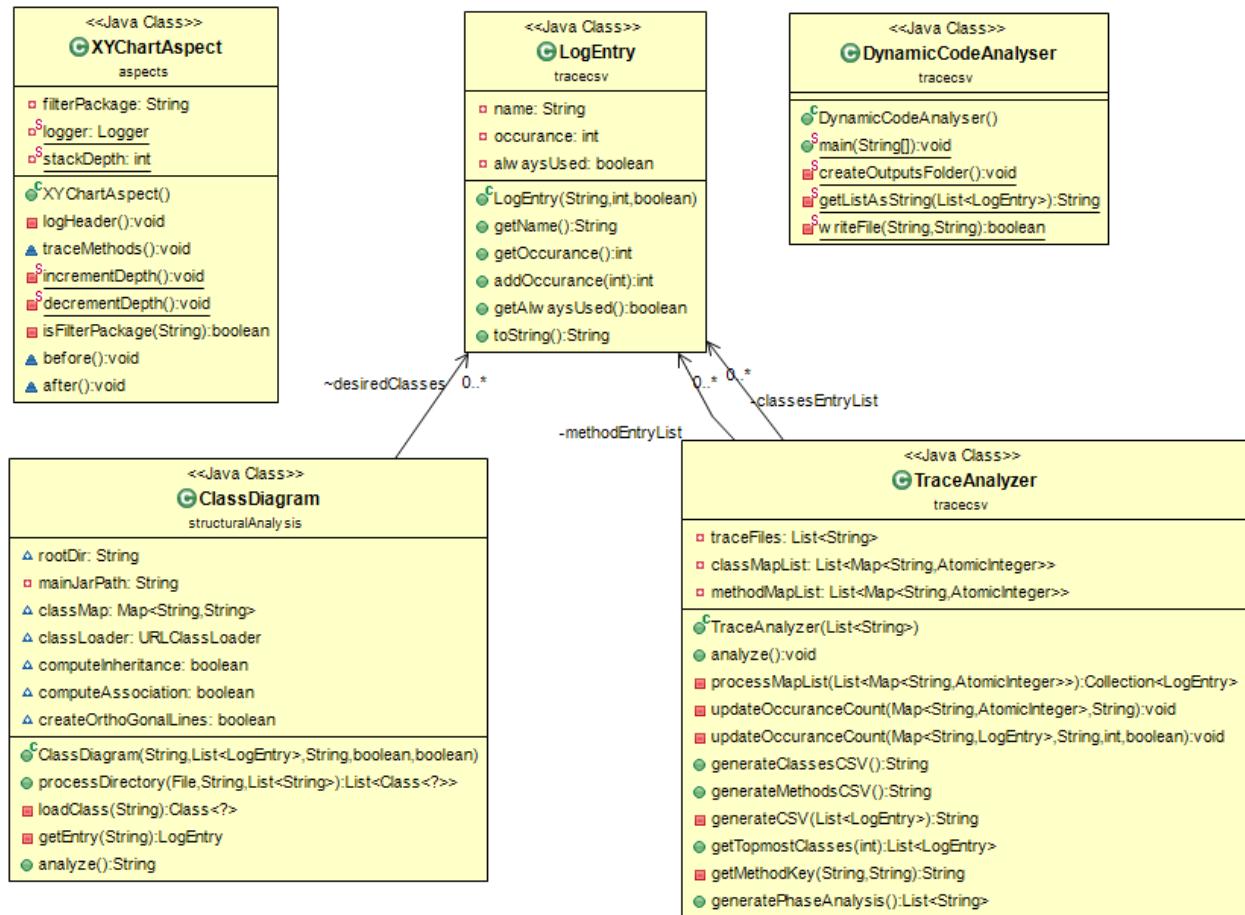


Figure 1 Dynamic Code Analyzer class design

- **XYChartAspect**
  - o This is the aspect class responsible for capturing the function execution trace of the tests within the JFreeChart library.
  - o Essentially it has `before()` pointcut to consider execution trace before getting into a method call and `after()` pointcut to consider execution trace return from a method call.
  - o It uses both the pointcuts to capture `StackDepth` at any given instance of the method call.
  - o In addition, it uses `filterPackage` parameter as a filter to capture only the required trace. In case of the coursework, this parameter is set to "org.jfree" for analysis of JFreechart library.
- **TraceAnalyzer**
  - o Core class responsible for analyzing the traces generated using the aspect.
  - o Generate various outputs required as part of the course work. Each of the output can be generated through relevant public function.

## - ClassDiagram

- This class was originally created by neilwalkinshaw on 24/10/2017 for Reverse-Engineering a Class Diagram using Reflection lab session
- It has been extended or modified for System Re-engineering assignment 2 - Dynamic analysis to generate class diagram
- This has been made generic in terms of receiving the classes and main jar information from the client so that it only deals with the processing and generation of class relationship and association diagram.
- It also accepts inheritance and association as parameters in the constructor and is used as settings for processing the class diagram as dot files.

## Trace logging

AspectJ compiler is used to generate the XYChartAspect as aspect.jar.

```
ajc -cp C:\aspectj1.8\lib\aspectjrt.jar -outxml -outjar aspects.jar XyChartAspect.aj
```

This jar along with Junit and hamcrest-core-1.3.jar on top of classes and the main jar of JFreeChart library with dependencies is executed on each of the tests.

```
java -javaagent:C:\aspectj-1.8.12\lib\aspectjweaver.jar -cp aspects.jar; jfreechart\target\jfreechart-1.5.0-SNAPSHOT-jar-with-dependencies.jar; jfreechart\target\test-classes;..\Assignment2\Libs\junit-4.12.jar;..\Assignment2\Libs\hamcrest-core-1.3.jar org.junit.runner.JUnitCore  
org.jfree.chart.AreaChartTest
```

In general, this can be executed on any test case and the output is traces.csv file capturing

DateTime, ClassName, FunctionName, Parameters, Line, StackDepth

## Trace log Analysis

Once the all the traces are generated, the traces can be moved to a common folder. With the use of Dynamic Code Analyzer application. The traces can be processed and required outputs can be generated.

## Process

As part of the process and using the console application, the user needs to provide following information highlighted in the console output below. Once the valid data is provided, the application generates the data in [Outputs] folder.

```
[ Welcome to Dynamic Code Analyzer ]  
This application will process the trace files generated by aspects.jar and process them to  
create several outputs.  
1. Generate the summary of the process as Classes.csv, Methods.csv2. list of Topmost Classes  
based on the total occurrence  
3. Class relationship dot file to Dot application to generate the class diagram4. Generate the  
time series data to analyzed using the PhaseAnalysis.R script.  
  
Please select the folder with trace files (generated using the aspects.jar):  
C:\Users\Shashi\Documents\MS\SEM2\SRE\Assignment2\assignment2-  
shashidarette\dataFiles\outputs\TraceFiles  
Please select the folder with classes (aka .class) files:  
C:\Users\Shashi\Documents\MS\SEM2\SRE\Assignment2\assignment2-  
shashidarette\jfreechart\target\classes  
Please select the location of the main jar (used to generate trace files) required to generated  
class relationship dot file:  
C:\Users\Shashi\Documents\MS\SEM2\SRE\Assignment2\assignment2-  
shashidarette\jfreechart\target\jfreechart-1.5.0-SNAPSHOT-jar-with-dependencies.jar  
Please select the number of classes to be considered to generate class relationship dot file  
(based on total number of occurrences) for ex: 25:  
50  
Classes CSV  
Generated classes.csv file.
```

Methods CSV  
Generated methods.csv file.

Top 50 classes are:

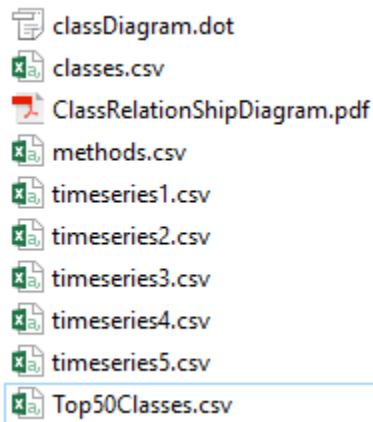
org.jfree.chart.axis.TickUnit 4780  
org.jfree.chart.util.Args 2156  
org.jfree.chart.plot.XYPlot 1712  
org.jfree.chart.ui.RectangleInsets 1690  
org.jfree.data.xy.XYSeries 1312  
org.jfree.data.xy.XYSeriesCollection 1198  
org.jfree.chart.plot.Plot 1133  
org.jfree.chart.axis.ValueAxis 1038  
org.jfree.chart.JFreeChart 957  
org.jfree.chart.axis.DateTickUnitType 882  
org.jfree.data.Range 864  
org.jfree.chart.axis.Axis 752  
org.jfree.chart.axis.NumberAxis 685  
org.jfree.data.xy.XYDataItem 672  
org.jfree.chart.renderer.AbstractRenderer 478  
org.jfree.chart.event.ChartChangeEvent 426  
org.jfree.chart.block.AbstractBlock 370  
org.jfree.data.xy.AbstractXYDataset 306  
org.jfree.data.general.Series 286  
org.jfree.chart.renderer.xy.AbstractXYItemRenderer 272  
org.jfree.data.xy.XYBarDataset 249  
org.jfree.data.general.DatasetUtils 233  
org.jfree.chart.ui.Size2D 225  
org.jfree.chart.title.Title 190  
org.jfree.data.xy.IntervalXYDelegate 186  
org.jfree.chart.axis.NumberTickUnitSource 177  
org.jfree.chart.block.BlockBorder 160  
org.jfree.chart.util.AbstractObjectList 144  
org.jfree.chart.text.TextUtils 137  
org.jfree.chart.StandardChartTheme 135  
org.jfree.chart.axis.DateTickUnit 132  
org.jfree.chart.block.RectangleConstraint 125  
org.jfree.chart.LegendItem 125  
org.jfree.chart.title.LegendTitle 125  
org.jfree.chart.ui.RectangleEdge 124  
org.jfree.chart.axis.TickUnits 123  
org.jfree.chart.ui.TextAnchor 90  
org.jfree.chart.block.BlockContainer 90  
org.jfree.chart.title.LegendGraphic 82  
org.jfree.chart.plot.DefaultDrawingSupplier 80  
org.jfree.chart.util.BooleanList 76  
org.jfree.chart.text.TextBlock 75  
org.jfree.chart.axis.DateAxis 74  
org.jfree.chart.text.TextFragment 70  
org.jfree.chart.axis.DateAxis\$DefaultTimeline 61  
org.jfree.chart.text.TextLine 60  
org.jfree.data.general.AbstractSeriesDataset 58  
org.jfree.chart.title.TextTitle 50  
org.jfree.data.general.AbstractDataset 48  
org.jfree.chart.axis.NumberTickUnit 45  
Generated Top50Classes.csv file.

Class Relationship and Association diagram (.dot format):  
Generated classDiagram.dot file (use dot to generate the class diagram).

Time Series

Processing trace file: C:\Users\Shashi\Documents\MS\SEM2\SRE\Assignment2\assignment2-shashidarette\dataFiles\outputs\TraceFiles\XYAreaChartTest\_Trace.csv  
Processing trace file: C:\Users\Shashi\Documents\MS\SEM2\SRE\Assignment2\assignment2-shashidarette\dataFiles\outputs\TraceFiles\XYBarChartTest\_Trace.csv  
Processing trace file: C:\Users\Shashi\Documents\MS\SEM2\SRE\Assignment2\assignment2-shashidarette\dataFiles\outputs\TraceFiles\XYLineChartTest\_Trace.csv  
Processing trace file: C:\Users\Shashi\Documents\MS\SEM2\SRE\Assignment2\assignment2-shashidarette\dataFiles\outputs\TraceFiles\XYStepAreaChartTest\_Trace.csv  
Processing trace file: C:\Users\Shashi\Documents\MS\SEM2\SRE\Assignment2\assignment2-shashidarette\dataFiles\outputs\TraceFiles\XYStepChartTest\_Trace.csv  
Generated timeseries1.csv file.  
Generated timeseries2.csv file.  
Generated timeseries3.csv file.  
Generated timeseries4.csv file.  
Generated timeseries5.csv file.  
All the outputs are generated in [Outputs] folder.

The outputs are:



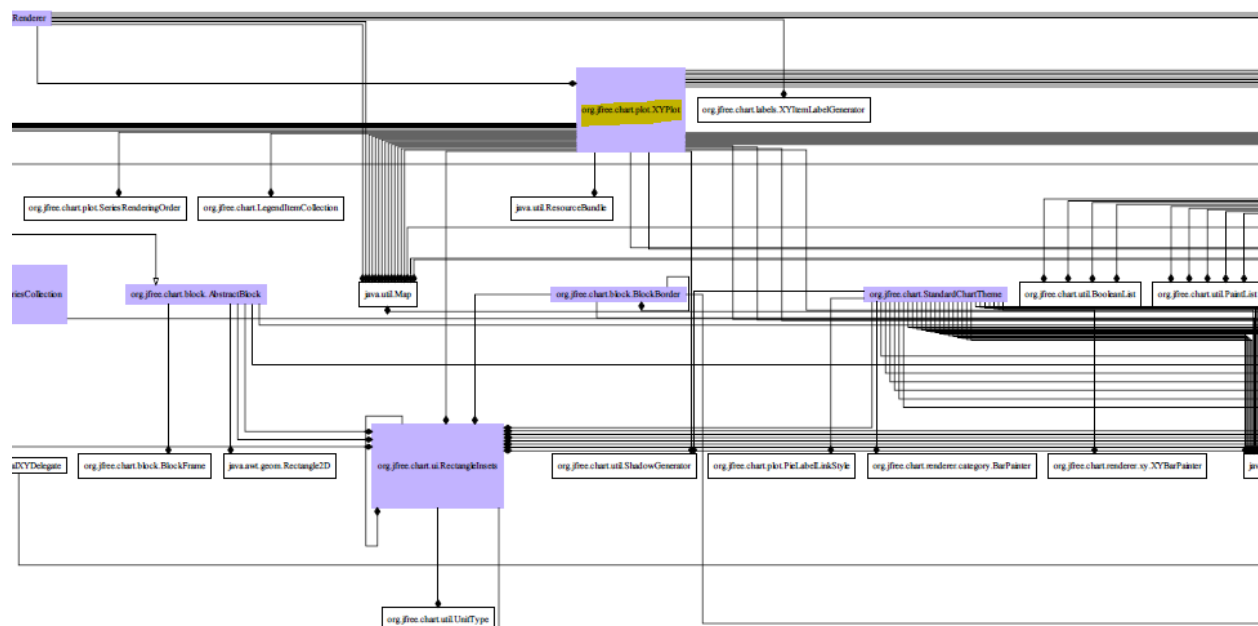
## Output generation

## Classes and Methods CSV

These CSV files contain the list of classes and methods present in all the given trace files as part of the course work. There are 62 out of 87 classes (from `classes.csv`) & 398 of 556 methods (from `methods.csv`) are used in all the traces.

## Class Diagram and Insights

The class diagram generated essentially shows the classes hierarchy and association. Specifically, for XYPlot and JFreeChart classes. Both form part of core classes and are essentially involved in all the traces marked in blue although other utility classes as axis.TickUnit, plot.Plot, xy.XYSeries, ui.RectangleInsets form a major part of the execution and hence are an essential part of the library.



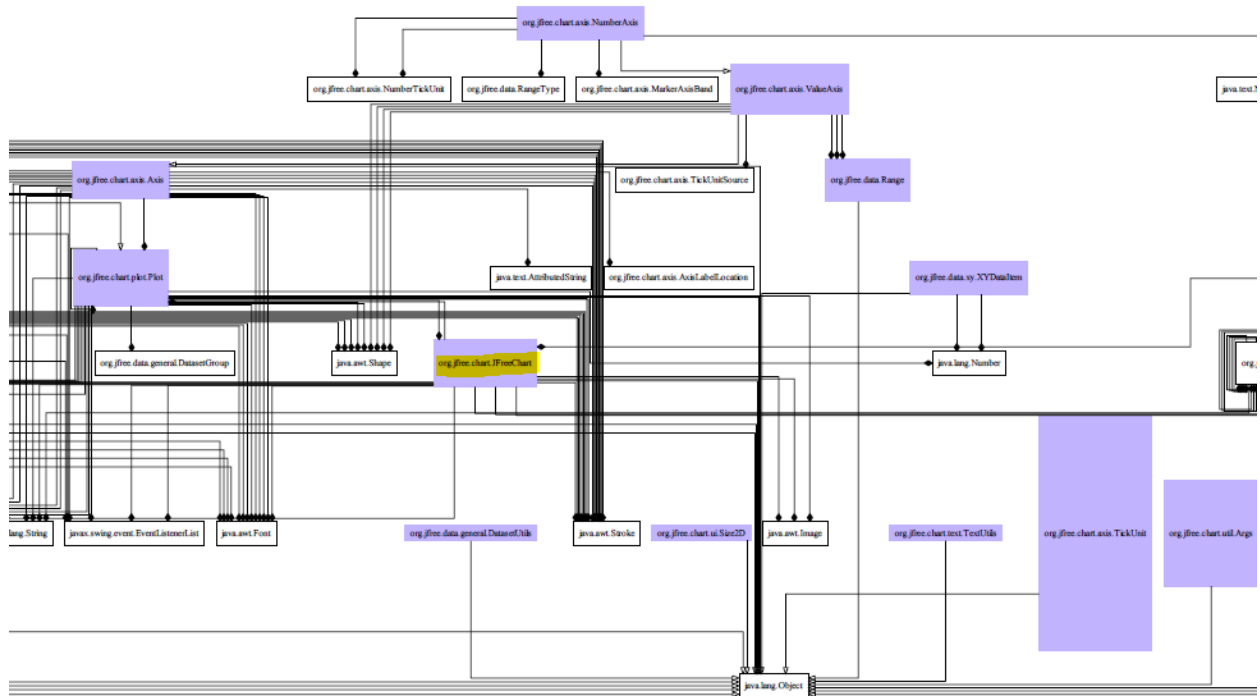


Figure 2 Part of the class diagram generated by the tool showing XYPlot

## Time Series, phase execution analysis & insights

From the TraceFiles, TraceAnalyzer also generates time series data for each of the trace files individually.

They also show where functions are called in all the traces with a light blue dot (•) and else with an orangish dot (•).

The plots are represented in the images below.



Figure 3 XYAreaChartTest time-series plot



Figure 4 XYBarChartTest time-series plot



Figure 5 XYLineChartTest time-series plot



Figure 6 XYStepAreaChartTest time-series plot

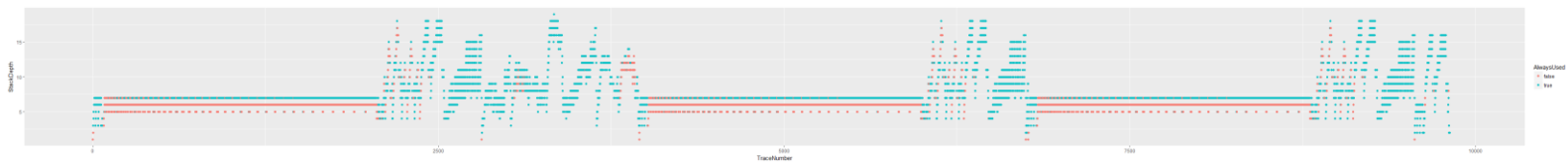


Figure 7 XYStepChartTest time-series plot

Few of the valuable insights are as below:

- All the traces show majority methods which are used in all implying the code and design developed highly cohesive.
- All the traces show maximum Stack Depth of around 20-25. This means the number of calls being made to various aspects of JFreeChart is uniform. This implies low cyclometric complexity and high maintainability.
- From XYStepChart time series plot its evident that specific functions are called repeatedly to perform required to create step chart.

## Final Overview

From the outputs generated, there are 62 out of 87 classes (from classes.csv) & 398 of 556 methods (from methods.csv) are used in all the traces. The class diagram shows high reliability and association amongst classes. All the traces time series plot show majority methods which are used in all. These insights can be used to imply that JFreeChart and per say XYPlot is very cohesive. Further analysis can be done to look for instances of re-engineering aspects, otherwise JFreeChart is well designed in general from an engineering perspective.