**Data Pruning**

The SVN commit history for Apache Ant is where the information about revision ID, name of a person, date, number of lines and changed path of every commit is stored. The information in this file is vital to take our project one step further. Fortunately, we had an access to this file. The file has 149034 lines and 12958 commits in total. The first and last commit took place on January 13, 2000 and January 27, 2014 respectively. The date of commit information in the file helped us to associate which commit belongs to which version of Apache ant. There are five releases associated with Apache Ant were considered. The month and year of release for each version is as follows:

|  |  |  |
| --- | --- | --- |
| **Version Number** | **Month** | **Year** |
| 1.5 | July | 2002 |
| 1.6 | December | 2003 |
| 1.7 | December | 2006 |
| 1.8 | February | 2010 |
| 1.9 | March | 2013 |

***Fig 3.1: Apache Ant versions***

The first four releases are used for analysis and the last one is used for validation/prediction. The data from SVN are parsed, cleaned and filtered using Java. During this process, we found that some of the commits were problematic in the sense that they are not related to classes E.g commits which changed the license, and removed the authors from code etc which are eventually removed. The number of commits between one version to another and the number of classes impacted in each commit are collected. This information is then classified and assigned to each of 3 buckets. The buckets are termed as High Volatility, Medium Volatility and Low Volatility. The High Volatility bucket contains the classes which got changed in all three times, the Medium Volatility buckets contains the classes which got changed two times, and the Low Volatility buckets contains the classes which got changed only once.

|  |  |  |
| --- | --- | --- |
| Version to Version | Number of Commit | Java Classes |
| 15\_to\_16 | 434 | 788 |
| 16\_to\_17 | 630 | 1089 |
| 17\_to\_18 | 296 | 565 |
| 18\_to\_19 | 101 | <669> |

***Fig 3.2: Number of commit and impacted classes between versions***

There are 294, 503, and 554 classes are classified as High, Medium and Low Volatility respectively. This is written into three text files based on Volatility.

The above information are now should be used to calculate metrics. We have adopted “chidamber and kemerer” definition for calculating class level metrics and MOOD for calculating System level metrics. The class level metrics – LCOM, LCOM Henderson, Cohesion, RFC, and Size/LOC – and the system level metrics – Method Hiding Factor (MHF), and Attribute Hiding factor (AHF) – are calculated.

The class for each metrics is designed. The design reveals that all the metric classes exhibit some common behavior so we have designed an abstract parent class to take care of common tasks such as loading the volatility buckets and writing the results to the database. We have developed a standard template pattern so that each child class would need to only implement one method, and write to a collection defined in the parent. Each child class receive the system object from abstract parent class to process the classes, methods, and attributes of the system and generate the value of above mentioned metrics for each buckets. E.g. LCOM metrics will be calculated for all the classes in High Volatility, Medium Volatility and Low Volatility buckets. For system level metrics, there will be one value generated for each bucket since each bucket is considered as a system. The most interesting is that the abstract parent class can be applied not only for Apache Ant but for all the Java projects.

The summary metrics collector which only interacts with the abstract is designed to perform additional data check to find if the metrics were themselves correlated or not.