International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-4 Issue-2, July 2014

Code Quality Evaluation for Open-Source Software System

Ravinder Kaur, Meenakshi Sharma, Hardeep Singh

***Abstract*— Conventionally, the explore on quality attributes was set aside confidential inside the association that perform it by means of constricted black-box techniques. The appearance of open source software has transformed this image by permitting us to estimate both software products and procedures that yield them. This paper represent the results of a pilot case study intend to understand open source code evaluation and quality analysis by using statistical description of different source code metrics. Towards this finish we have measured quality characteristic of different versions of an application written in java.**

***Index Terms*— metrics, open source software, software measurement, statistical code evaluation and code quality**

I. INTRODUCTION

Open source software's are the programs or software whose source code is freely accessible to the universal public from its original design without any kind of charges. These software not only permit users for accession of its source code but also give privileges to alter the code accordant to necessitate and to reinvent the new code to heighten the utilization. Open source code is usually produced as a combined attempt in which programmers make better the code and distribute the changes inside the society [1]. The programmers concerned in open source projects are highly encouraged because they made softwares mostly for own satisfaction and therefore too expected to be extremely creative software. Open source software development is basically relies on a quite straightforward scheme i.e. the central part of the software is formulated nearby particular programmer or a group of programmers. These softwares have to be strictly modular, independent and self explanatory to permit development to volunteers. Now a day, a lot of IT companies are taking a rigid stare at Open Source and free software through gaze at sinking expenses and rising production. At face value, switch to liberated software seems to be fond of an effortless choice. Obvious it is beneficial if it is adequate to contain organizational desires, then why should not use it? These needs lead to hastily development of open source software systems and are becoming very bulky in software market. As the open source software's are mostly developed at remote sites so it is necessary to investigate the quality and reliability of the source code.

Manuscript published on 30 July 2014.
*****Correspondence Author(s)
Ravinder Kaur, Department of CSE, Sri Sai College of Engineering & Technology, Badhani, India.
Prof. Meenakshi Sharma, Department of CSE, Sri Sai College of Engineering & Technology, Badhani, India.
Dr. Hardeep Singh, Department of CSE, Guru Nanak Dev University, Amritsar, India.

The nucleus of open source deeds happens only at the code level. So it is sensible to focus there, measure and assess the consequential product i.e. the source code. The purposes of this paper is to tale and talk about the grades of a case study that observe the quality of the source code take away from open source repository i.e. http://sourceforge.net/projects/jstock/. One more reason of our study is to classify structural metrics that could facilitate to distinct more than one aspirant version of a software component when shaping the contents of an open source release, in exacting metrics linked to component size. The paper is organized as follows. Section 2 presents a concise overview of software metrics used in this study .Section 3 gives an introduction to open source software. Section 4 presents the study of software evaluation. Section 5 represents the analysis of statistical description for various metrics and the last section presents conclusions and future scope.

II. METRICS USED IN THIS STUDY

This section represents the metric used in this study for software evaluation and statistical description. The details of metrics that we investigated are the following:

1) Lines or LOC (Lines of Code):- It is the oldest metric which is used to measure the module size. The main concern was developed in line of code is "what to include in measurement". It basically represents the number of every nonempty, non comment lines of the body of the class and every single one method.
2) Avg Complexity: - It is useful for ascertaining how nasty the code is? As classify by McCabe (1976), this is a metric which relies on a graph hypothesis that represents the total number of linearly free path in a related graph [3]. The basic goal of this metric is to evaluate the maintainability and testability of the software module. Mc.Cabe describe complexity as
   V(G)= e-n+2
   Where V(G) is cyclomatic complexity of particular flow graph G.
   e= total number of edges in graph G
   n= total number of nodes in graph G.
3) Avg Depth or DIT-- This metric is basically represents the distance of the utmost path from the root node to the end node of the tree [2]. A deeper a class hierarchy means more methods were used or inherited.
4) Classes--This metric represent the all classes presents in a project. A class is an elementary element of object-oriented designs which consists of data members and methods that illustrate the deeds of a class. [7]

5) Class size-- Mark Lorenz and Jeff Kidd in 1994 introduce a metrics to quantify software quality evaluation which was applicable to class diagrams [7]. It is the proportion of the number of lines of code divided by the total number of classes. Class size metric is strongly related to complexity metric as both have true positive correlation between them
6) Methods per Class-- It is the reckon of methods implemented inside a class. Classes having large number of methods are supposed to be extra application specific, restrictive to the probability of reuse [2].
7) Statement per Method-- It is useful for a broad feel of how large every method is. It is mostly used to calculate the average method size in terms of the number of lines or statement per method in a project.
8) Statements-- it counts the standard number of executable statements per component.
9) Files-- this metric counts the total number of source code files in a system.

III. CASE STUDY

This section presents a concise introduction to the software used in this study namely called "JStock- Free Stock Market Software". According to cook the software can be categorized as E-type [4][5]. The used software has been written in Java and is an open source [5]. All the data associated to versions of software is available on internet. The selection of software is essentially guided by the limitations of metrics measuring tools and the accessibility of version's particulars and source code.

A. JStock- Free Stock Market Software

JStock is free stock market software for many countries. It provides Stock watchlist, intraday stock price snapshot, Stock indicator editor, Stock indicator scanner and Portfolio management etc. Free SMS/email alert supported. We download multiple versions for our case study from repository http://sourceforge.net/projects/jstock/ and perform software evaluation and statistical analysis among different metrics of each version.

IV. ANALYSIS OF EVALUATION OF JSTOCK

Software is not inclined to tear and wear but at a standstill it may possibly turn into worthless if it is not revised in reply to continually varying user's requirements. Software desired to grow in command to be used for large era. Lehman et al has done wide-ranging study on progression of outsized and extensive lived software [6]. Lehman's laws of software evolution states that incessant transformation and development is mandatory for maintaining the software long-lasting. The laws also suggest that as the time passes away, it become further complicated to add new functionalities to software system as a result of changes and growth. Therefore it becomes necessity to test whether the open source code allows rapid evolution or not. As a result of which it became easy enough to allocate numerous modifications and extensions. Undoubtedly, it ought to be legible and self-expressive to make possible these activities. The accessibility of source code and alteration particulars of open source software has provided a sustain to the swot of software evolution. In this section we analyze the Lehman's laws of software evolution [5] in the light of measures of several metrics, computed for different versions of JStock on the rampage through evolution tradition. The figure 1 shows the growth curve of different metrics for the different versions of open source software JStock. In the above graph the data point for metrics lines and statements are shifted to secondary y-axis due to high range of their numeric values. Lehman's law of continuing growth [4] states that the functionality provided via the software ought to repetitively grow up so as to grant user contentment over longer era. Now the thing is that the growth can be interpreted as raise in the extent of code or raise in the utility being provided via the software. The software has grow up in size can be resolute by observing the variation of size metrics over consequent releases. Lehman used the parallel approach. We calculated and examine the LOC, number of classes, method per class, average statement per method, class size, files, statements, average depth etc. metrics for different releases of JStock. Figure 1 represents the linear growth curve for each metric for different versions. Growth of evolving software, canister to calculated by observing variance in the total number of classes, number of methods per class and by number of statements per method, in terms of functionality. Law of continuing change by Lehman [4] is also satisfied by the calculated data. It has been observed that the complexity of software tend to rise over a numerous releases except various measures are engaged to keep make sure for complexity. It is usually supposed that development adds to rise in complexity even if the appropriate changes are made; the evolution process may not be manifest for the attributes that conforms to boost in complexity. In case of object oriented systems, the complexity of the software system or subsystem canister to method per class, class size and average depth. We computed these metrics as a complexity measure for JStock.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

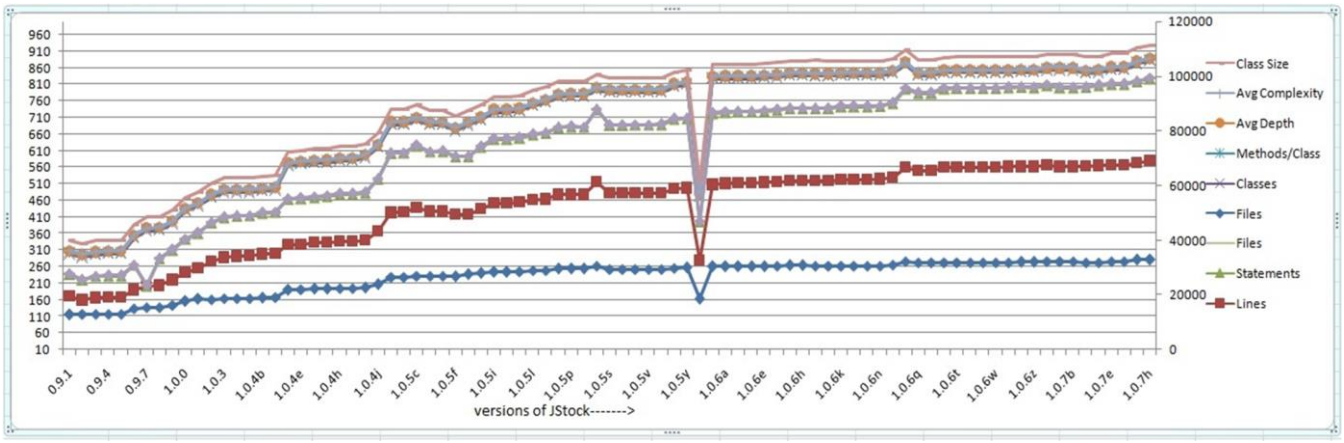

Fig. 1 Growth Curve of Different Metrics for Different Versions of JStock

V. MEASUREMENT AND CATEGORIZATION OF OPEN SOURCE CODE QUALITY

Users always need software with high quality. There is always a fashion of exploitation components in software application development, due to its apparent advantage of little cost, high quality and fewer times to develop applications along with several others. So it is necessary to measure the quality of open source code. The quality of code is resolute by a lot of elements, and measuring it is far from trivial [1]; For the purpose of the study we chose to examine source code metrics. We choose that metric because we could easily measure these metrics by using metric tools. The table 1 represents the relationship between quality criterion and corresponding metrics. Table 1 clearly represents that the functionality the system can be calculated by metrics classes, statement per method and method per class. Deployablity of the system can be check by measuring class size and complexity metrics. The metrics lines, statements and files are used to measure the scalability characteristic of quality. And depth of inheritance tree metric is used to measure the changeable nature of the system.

Table 1. Relationship between Quality Criteria and Metrics

Criterion	Associated metric
Functionality	Classes, statement per method, method per class
Deployablity	Class size, complexity
Scalability	Lines, statements, files
Changeability	Depth of inheritance tree

Table 2 presents descriptive statistics for software metrics calculated for an open source software. Descriptive Statistic is the discipline of quantitatively describing the main features of a collection of information or quantitative description itself. In our research work, source code or quality metrics are calculated for huge no. of multiple versions of Jstock and arranged in tabular form. For the statistical analysis, each metric was considered as a random numerical variable. Statistical analysis is a science of collecting, exploring and presenting large amount of data to discover underlying patterns and trends. Each module of an application has been measured and the mean value of each metric has been computed crosswise an application. Descriptive statics across all application is given in table 2. For each metric, the minimum, maximum, mean, standard deviation and median values are calculated. In some belonging, awfully varying values have been discovered, but this is normal given the wealth of open source examined and the large number of peoples that have been concerned in the development of the software. For LOC the maximum value is generally high. Standard deviation for classes and statements are extremely high and alternatively it leads to high value for method per class, statement per method and source code files. The achieved result for statements are seemed to be generally far-away from the formal choice, but this is most likely be as a effect of poor classification and understanding of the impact of metric on a quality of code. The below table 2 give the complete representation of all the factors of statistical analysis i.e. minimum, maximum, mean, standard deviation and media for all the source code or quality metrics.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

	Minimum	Maximum	Mean	Standard Deviation	Median
LOC	18040	69002	5.143	1.5196	57254
Avg Complexity	1.92	2.41	2.15	0.1455	2.19
Avg Depth	2.7	3.49	3.210	0.1502	3.23
Classes	172	600	467.5	127.21	530
Class Size	31.006	38.7	35.36	2.012	35.02
Method per Class	3.94	433	18.67	75.85	4.12
Statements per Method	6.67	884	18.78	94.40	8.69
Statements	9.838	30095	2.155	7609.72	24769
Files	115	281	229.1	49.979	252

Table 2. Statistical Analysis of different Metrics

VI. CONCLUSION AND FUTURE SCOPE

In this paper, we have tried to present experimental data to add up quantitatively to the ongoing symposium relating to the genuine power of open source code style of software development. Our aim was actually to explore the benefits that source code evaluation and statistical analysis of metrics can make available to open source and provide clues for further empirical research. This study has opened up more opportunities for research in the field of software evolution and quality measurement.

VII. ACKNOWLEDGMENT

I would thank to Prof. Meenakshi Sharma for her guidance, support and encouragement during this study period. I would like to gratefully and sincerely thank to Dr. Hardeep Singh for giving his valuable time for guidance in research work and support during the research period. Finally, I honestly thank to my parents, family, and friends, who provide the advice and financial support. The product of this research paper would not be possible without all of them.

REFERENCES

1. Ioannis Stamelos , Lefteris Angelis, Apostolos Oikonomou & Georgios L. Bleris "*Code quality analysis in open source software development"*, Info Systems J (2002) 12, 43–60, published in Wiley Online Library
2. S. R. Chidamber and C. F. Kemerer, "*A metrics suite for object oriented design,"* IEEE Trans. Software Eng., vol. 20, no. 6, pp. 476–493, 1994.
3. T.J. McCabe. "*A complexity measure''. IEEE Transactions on Software Engineering*, SE-2(4), December 1976.
4. Lehman, M.M. 1980 "*On understanding laws, evolution, and conservation in the large-program life cycle"*. Journal of Systems and Software, Elesvire, 213-221.
5. http://sourceforge.net/projects/jstock/
6. Belady, L. A. and Lehman, M.M. 1976. "*A model of large program development*'. IBM Syst. J. 15, 225–252.
7. M. Lorenz, J. Kidd, *"Object Oriented Software Metrics",* Prentice Hall, NJ, IEEE Transactions (1994).Y. Yorozu, M. Hirano, K. Oka, and Y.
8. Morisio, M. & Tsoukiàs, A. (1997) IusWare, "*A methodology for the evaluation and selection of software products." IEEE Proceedings on Software Engineering*, **144**, 162–174.
9. O'Reilly, T. (1999) *"Lessons from open source software development*". *Communications of the ACM*, **42** (4), 33–37.
10. Pighin, M. & Zamolo, R. (1997) "A predictive metric based on discriminant statistical analysis" *Proceedings ACM ICSE ¢97*, 262–269.]

AUTHOR PROFILE

Ravinder Kaur, is pursuing M.Tech in Computer Science and Engineering at SSCET Badhani, Punjab, India under Punjab Technical University Jalandhar. Her ongoing research area is on quality metrics and open source software evaluation. Her area of Interest is Software Engineering.

Meenakshi Sharma, is working as Associate professor in department of Computer Science and Engineering at SSCET Badhani, Punjab, India. She has more than 16 years of teaching experience. Her area of interest are Parallel Computing, Cloud Computing and Network Security

Dr. Hardeep Singh, is working as a Professor in department of Computer Science & Engineering in Guru Nanak Dev University Amritsar, India his area of specialization is Software Engineering .His numbers of publications are more than 90.His area of interest is Information System.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)