Using Metrics To Measure The Complexity, Understandability And Readability Of Open Source Software Over Multiple Releases

Uttamjit Kaur

Department of Computer Science, GIME, Amritsar, India
Email: er.uttamjitkaur@gmail.com

ABSTRACT : Open-Source Software (OSS) is becoming very popular in today's software development environment. In OSS, the development and maintenance of the software is decentralized due to which maintainability is core issue in OSS development. The maintenance of OSS is a never ending task.Software metrics help to control the uality of OSS. This paper reviews popular object-oriented metrics to analyze LOC, CC RCC and CLC of the JACOB Open Source Software (OSS) of Twenty-two releases over a successive versions.The terms LOC, CC, RCC and CLC was computed for all the Twenty-Two successive versions of OSS. The software metrics wre calculated using Understand4Java tool.

Keywords : Open-Source Software,Software Metrics,Versions and Software Complexcity**.**

1 INTRODUCTION

IN a changing environment, all software's must change with time to meet the need of its users "[2]. Therefore it is important for software development and software developers to keep software operational and to enhance the functionality of the software "[2] in such a way that the problems arising from changes are reduced. Open Source Software (OSS) has been getting more interest in the last few years ".Open Source Software (OSS) is usually developed by volunteers from all over the world working co-operatively "[2]. In Open Source Software (OSS) development scenario, software is available on internet and it allows developers to contribute to the new functionalities, improvement of existing software version and submitting bug fixes to the current release "[2]. In the OSS, the development as well as maintenance of the software is decentrized due to which maintainability is core issue in OSS development "[2].In such a software development scenario the maintenance of the open source software is a never ending task.Software metrics help to control the quality of Open Source Software.

2 LITERATURE REVIEW

Jubair J. Al-Ja'afer and Khair Eddin M. Sabri King Abdullah [1] presented how Chidamber and Kemerer (CK) introduce the concept of metrics suits that encapsulate with the Method, coupling, cohesion and inheritance. The CK and LK metrics are used to evaluate the design of object oriented programs. Author wants to define the strength and weaknesses of a java programs. Mrinal Singh Rawat, Arpita Mittal and Sanjay Kumar Dubey [7] in this paper, author describe about the software quality can be viewed differently according to factors i.e.: User view, Manufacturing view, product view, value based view. Author defines the software metrics as a valuable entity and used to measure the standard that helps to evaluate the quality, efficiency, design and product. It describes the merits and demerits of object oriented metrics. Hayes et al. [9] derived a model for estimating adaptive maintenance effort. It was concluded that the number of LOC changed and the number of operators changed are strongly correlated with the maintenance effort. Denis et al. [2] investigated the relationships between software maintainability and other internal software quality attributes. The source code characterstics of five java-based open-source software products wre analyzed.

3 OBJECTIVE OF THE STUDY

The objective of this study is to analyze Line of Code (LOC), Cyclomatic Complexcity (CC), Count Line Code (CLC) and Ratio Comment to Code (RCC) for Twenty-Two versions of the Open Source Software.

4 RESEARCH METHODOLOGY

The Source code of Open Source Software (OSS) i.e JACOB was used in the study as the data source.JACOB is a JAVA-COM Bridge that allows you to call COM Automation components from java.It uses JNI to make native calls to the COM libraries.JACOB runs on x86 and x64 environments supporting 32 bit and 64 bit JVMs.As of versions 1.8,the following things are true about JACOB:

- The project license changes from the LGPL to BSD.
- JACOB is now complied with java 1.4.2.
- The project is hosted on SourceForge.

Understand 4 java tool provides information regarding code.All information on functions, classes, variables, etc, how they are used, called, modified and interacted with.Understand is very efficient at collecting metrics about the code and providing different ways for yo view it.There is a substantial collection of standard metrics quickly available as well as options for writing metrics cover exactly what you need.

TABLE 1

VALUES OF METRICS FOR DIFFERENT RELEASES OF JACOB

Versions	Av. CC	LOC	Count Line Code	Ratio comment To code
1.9	1.46	4484	1483	1.71
1.9.1	1.61	5005	1704	1.64
1.10.0	1.7	5422	1856	1.65
1.10.1	1.71	5466	1874	1.64

1.11	1.66	6244	2220	1.55
1.11.1	1.69	6279	2250	1.54
1.12	1.74	6686	2432	1.51
1.13	1.76	6845	2506	1.49
1.14	1.81	6845	2942	1.63
1.14.1	1.81	8396	2947	1.63
1.14.3	1.8	8420	2949	1.64
1.15	1.92	7927	2806	1.61
1.15-M4	1.92	7927	2806	1.61
1.16	1.93	7932	2810	1.61
1.16-M1	1.92	7928	2806	1.61
1.16-M2	1.93	7932	2810	1.61
1.17	1.93	7948	2818	1.6
1.17-M2	1.93	7945	2820	1.6
1.17-M3	1.93	7942	2820	1.61
1.17-M4	1.93	7948	2818	1.6
1.18-M1	1.93	7948	2818	1.6
1.18-M2	1.93	7948	2818	1.6

5 ANALYSIS

This section presents various object-oriented metrics and their relationship with the quality of the Open Source Software (OSS).A comparison of the measured values of the metrics is made with an increase in the value of Cyclomatic Complexcity (CC) and Lines of Code (LOC) represent it is harder to understand and maintain the software. It clearly represents a decrease in the quality of the software.There are many metrics that are traditional functional development. The metrics that are applicable to object oriented development: Complexcity, Size and readiability.To measure the complexcity, the Cyclomatic complexcity is used.

5.1 METRIC 1: Cyclomatic Complexcity (CC)

Cyclomatic Complexity (CC) metrics [6] used to identify the complexity of a software program which is based on a directed graph. The formula for calculating the cc is the Number of edges minus the number of nodes plus 2. A high value of Cyclomatic Complexity indicates high complexity of a program which is harder to test, understand, modify and maintain [9]. To reduce the complexcity of software development, it is suggested to limit the Cyclomatic Complexcity to 10[11]. From Table1 it is clear that cyclomatic Complexcity increases as releases over its lifecycle.Figure1 shows the Cyclomatic Complexcity of JACOB.

Fig: *Cyclomatic Complexity*

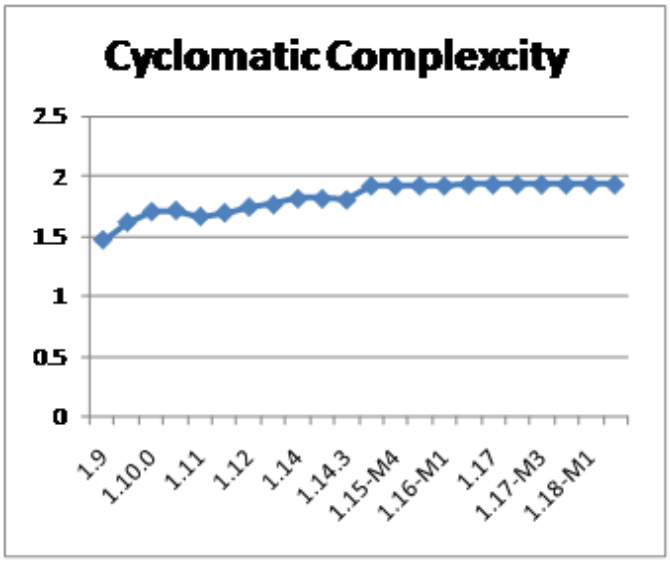

Fig 1.*Cyclomatic Complexcity of JACOB*

5.2 METRIC 2: Lines of Code (LOC)

Lines of Code represent the size of a class that is used to evaluate the ease of understanding of code by developers and maintainers. LOC measures vary depending on the coding language used and the complexity of the method. However, since size affects ease of understanding by the developers and maintainers, classes and methods of large size will always pose a higher risk. High value of LOC indicates less understandability of the software and also affected the quality of the software [9].From Table1 it is clear that LOC increases as releases over its lifecycle.Figure2 shows the LOC of JACOB.

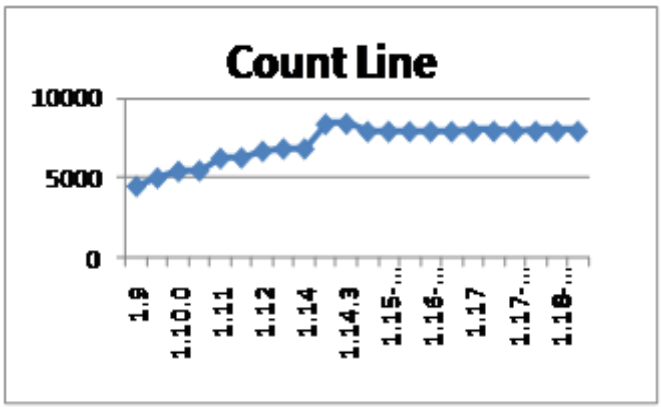

FIG 2: *LOC OF JACOB*

5.3 METRIC 3: Count Line Code (CLC)

In many programming language, Count Line Code (CLC) counts blank lines, comment lines and physical lines of-source code. CLC is known to run on many Windows as Linux, FreeBSD, OpenBSD, Mac OS X or Higher.From Table1 it is clear that CLC value increases over the successive versions.Figure3 shows the Count Line Code of JACOB.

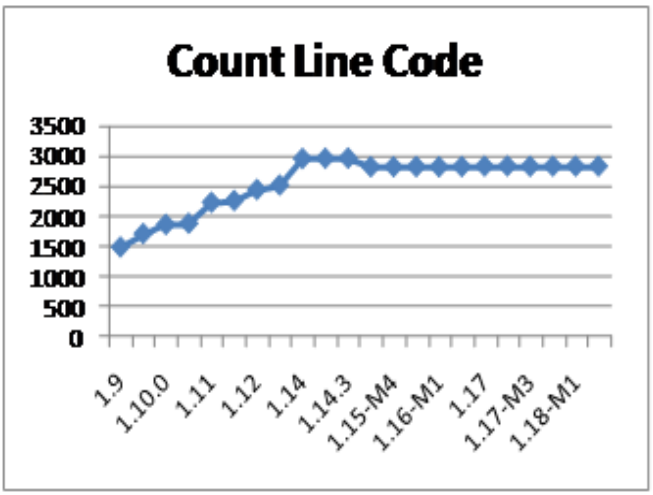

Fig 3: *Count Line Code of JACOB*

5.4 METRIC 4: Ratio Comment to Code (RCC)

Comment Percentage is defined as a ratio of the Number of comment lines to the Number of Non-Blank LOC [9]. In any stage of the life cycle, comments will help developers and maintainers to better understand the programs. Since comments assist developers and maintainers, higher comment percentage increases understandability and maintainability.From Table1 it is clearly defined that RCC decreases. So readability of code is difficult to underatand by software developers and development. Figure4 Shows the RCC of JACOB.

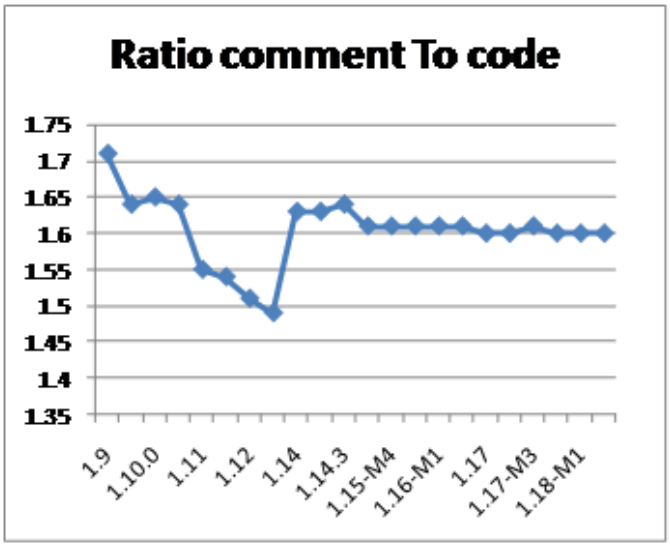

Fig 4: *Ratio Comment to Code of JACOB*

6 CONCLUSION

The Lines of Code, Cyclomatic Complexcity, Ratio Comment to Code and Count Line Code of the JACOB software was observed over Twenty-Two Successive versions.From the result it was observed that JACOB has the highesr Cyclomatic Comlexcity and Lines of Code value. The lower values of Ratio Comment to Code in case of JACOB indicates that the versions of this software are harder to understand.Increases in the vaues of Cyclomatic Comlecity,Lines of Code and decreses in the value of Ratio comment to Code indicates that JACOB software must contain complex class,packages and events that harder to understand and reuse.

REFERENCES

[1] CHIDAMBER-KEMERER (CK) AND LORENZE-KIDD (LK) METRICS TO ASSESS JAVA PROGRAMS Jubair J. Al-Ja'afer and Khair Eddin M. Sabri King Abdullah II School for Information Technology, University of Jordan, Jordan.

[2] Coleman, D. 1992 Assessing maintainability. Proceeding of the 1002 Software Engineering Productivity Conference, pp.525-532, San Jose, CA.

[3] Denis Kozlov, Jussi Koskinen, Markku Sakkinen and JJouni Markkula, “Assessing Maintainability hange Ovr Multiple Software Release”, Journel of software maintenance and evolution: Research and Practice, 20(1), pages 1-58, 2008.
http://sourceforge.net/projects/jacob-project/
http://sourceforge.net/projects/jacob-project/files/?source=navbar
https://scitools.com/

[4] International Journal of Advanced Computer Science and Applications, Vol. 3, No. 1, 2012 “Survey on Impact of Software Metrics on Software Quality” Mrinal Singh Rawat1, Arpita Mittal2 Sanjay Kumar Dubey3.

[5] Jacobson, Ivar, Object oriented Software Engineering,A Use Case Driven Approach,Addison-Wesley Publishing Company,1993.

[6] J. Hayes, S. atel, L. Zhao, “A MtricsBased Software Maintenance Effort Model”. Proceedings 8th European Conference on Software Maintenance and Reengineering, IEEE Computer Society Press, Pages 254-260, Los Alamitos, Calfornia, 2004.

[7] McCabe, T.J. 1`976. A Complexity Measure, IEEE Transaction on Software Engineering, vol.2, No.4, pp.308-320.

[8] Watson, A.H, McCabe, T.J and Wallace, D.R.1996.Structured testing: A testing methodology using the Cyclomatic Complexity metrics.National Institute of Standards and Technology Special Publication 500-235.