ijates
ISSN 2348 - 7550

OPEN SOURCE SOFTWARE MAINTENANCE COST EVALUATION USING MAINTAINABILITY INDEX AND CODE METRICS

Reenu Rani[1], Dipen Saini[2], Amandeep Kaur Cheema[3]

[1,2,3]*Dept. of CSE, Guru Nank Dev University, Amritsar Pb. (India)*

ABSTRACT

To cope up with today's changing environment, software maintainability cost evaluation becomes more necessary. Software maintenance cost is the overall cost or effort required to maintain the software after its initial deployment. The main objective behind this paper is to evaluate the maintenance effort because in certain cases it has been found that the maintenance cost is found to more than developing the software again. In this paper, a software maintainability cost prediction method has been used .The maintainability index and code metrics are used to evaluate the cost of the software maintenance. An open source software VLC player has been used as a case study to evaluate the maintenance cost on the bases of maintainability characteristics which has been used as evaluation criteria.

Keywords: Software Maintenance, Open Source Software, Software Evolution, Code Metrics, Software Reuse.

I. INTRODUCTION

Software maintenance plays an important role in software development. In software development life cycle there are different phases like requirement gathering, designing, implementation; testing and maintenance .Software maintenance is to make the software adaptable with changing environment, corrective, and perfective after adding some more functionalities and preventive from future change after delivery. Software maintenance is important part for an accurate performance of software. Pigoski [1] define that the part of industry's expenditures used for maintenance was 40% in the early 1970s, 55% in the early 1980s, 75% in the late 1980s, and 80% in the 1990s onwards. According to IEEE, the yearly cost of the product maintenance in United States exceeds $70 billion [2] Software evolution is incremental process from version to version. New versions are developed to overcome the limitations of existing version and to add some more functionalities to make it adaptable to new environment. Software modification is unavoidable because of changing environment when the software is used new requirements are materialize into it, The business atmosphere changes day by day, Bugs must be repaired for accurate performance, New computers and tools are added to the system to enhance the performance of existing system, The presentation or dependability of the structure may have to be enhanced [3] .Evolution makes software structure complex. To evaluate maintenance cost in software evolution process can be difficult. Bennett [4] illustrate that evolution have no ordinary definition but it is preferable substitute for maintenance. The phenomenon of evolution is closely related to Open Source Software (OSS) as there is a frequent release of versions [5]. Figure 1 shows the model of our research.

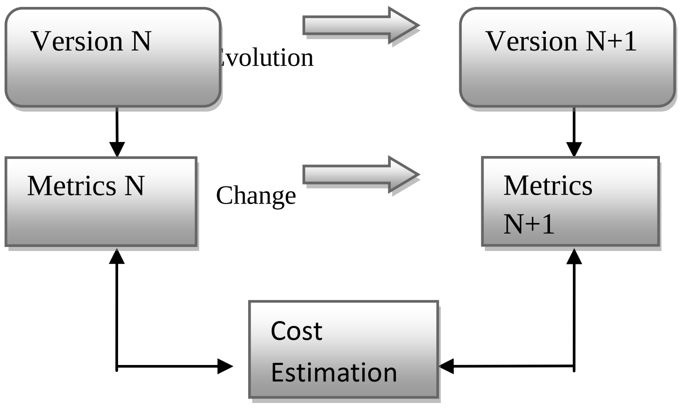

Figure-1 Research Methodology

Pino[6] describe that software maintenance phase is very costly phase throughout the development process. Cost is main reported issue in maintenance from customer and developers point of view. [7] Maintenance cost evaluation gives the benefits to organization. When the software is developed then there is need to modify it again and again to make it adaptable. Software maintenance cost is high as compare to development cost. It varies from system to system can be shown by figure-2. In this paper, we purpose a method to evaluate maintenance cost of open source software using maintainability index and code metrics with quality measurement. Currently, the evaluation is an crucial in software systems procedure[8] .Most of the companies switched from closed source software to open source software to succeed market distribute, to increase product expansion, and develop market dispersion code metrics [9] Table -1 shows the software maintainability code metrics that we examined

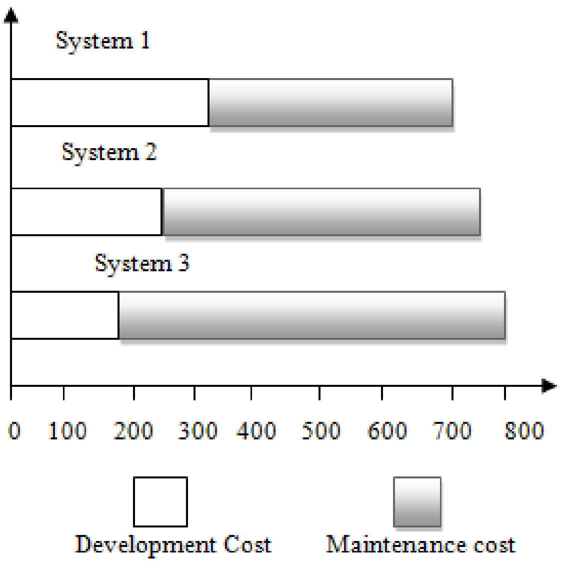

Figure-2 Development and Maintenance Cost from System to System.

II. RELATED WORK

Liang [12] describe that due to volatility of changes amount in a product evolution process, the cost and risk of software maintenance are difficult to forecast . Software maintainability is significant feature in the estimation of product change of a software product. In this paper threshold is establish as quantitative criteria of product maintainability and a hidden markov model (HMM) is used to predict maintenance behavior of software. In this software metrics are also used to determine the initial specification of software at the time of delivery.Alireza [13] describe that software complexity manipulation is one of the demanding trouble in software engineering. Complexity of system makes reliability of system evaluating cost of development and maintenance complicated. This paper evaluates a method to measure complexity of product using metrics statistical model. This method is applied on 'Nasa Software Engineering laboratory ' samples used as case study and its position results are also presented in this paper which shows the comprehensive complexity measure.Teresa[14] has describe that STPEGS has developed new preventive maintenance optimization process based on mathematical model which is develop in university of Texas at Austin. Existing maintenance data of university is uses as dataset for the process. This paper calculated that the preventive maintenance optimization process has the prospective to save thousands of dollars in yearly collective maintenance cost and it also enhance the overall reliability of product. M. Abdellatief[15] shows that component based development used to increase the productivity and quality of the system. It is also reduce the cost of development and time. This paper create the path between component developers and component users. For this component information flow(CIF) measurement and multi dimentional approaches are used. It is easy to locate the faults errors in the system by measuring the Design of component based software system. This paper purposes CIF that based on inter component flow and intra components and set of metrics based on it are develop results shows that purposed metrics are valid size measures. Application software's are more useful nowadays . Measuring and evaluated this type of software's are more complex. Harry M. Sneed [16] purposed a method for better prediction of maintenance effort by measuring and evaluating a dot net application system and how the metrics were comprehensive to evaluate the static quality criteria of a large Dot Net application. The ISO standards 9126 are used as quality criteria in this paper. Lawler and Kitchenham have purposed such a procedure using a quantity structure support based on their Tycho Metric tool [17]. Garcia et al. have illustrate a related approach [18]. Open source software are the free software for which there is freedom to use it. These type of softwares can be downloaded from websites like sourceforge and can be easily used for our project , we are free to modify source code of the product. Swapna[19] describe that OSS can present widespread opportunities for full of meaning participation with the code. Author's description on their experiences and training in integrating open source software into an preliminary sophomore/junior SE course,. There are multiple OSS are used for the students practice to become familiar with product maintenance. to figure out another's software, to adapt it for the improved, and to realize a logical approach. Anas[20] describe that software maintenance is main requirement in software development process because it requires more assets and labors than the other phases of the development process. Author describe the software maintenance process model that significance the collision of the software feature on the repairs process. Cost estimation is the main issue in the software maintenance process the growing of the software size and complexity caused serious problems in maintenance. Most methodologies were purposed to estimate cost and effort but the results has not be acceptable addressed till now. This paper gives the theoretical analysis of maintenance process model using evaluatiocriteria.

Levels	Code metrics Index
Method Level	Cyclometric Complexity, No. of unused parameter, No. of unused variables, No. of comment lines
Class level	Class between objects, Depth of inheritance
File level	Maintainability index, Halsted volume
Package level	No. of lines of code

Table-1

III. ATTRIBUTES AS EVALUATION CRITERIA

1. **Documentation:** Documentation is inclusive, clear and short information about the software. Software documentation plays an important role in software maintenance. For software maintenance it is important to understand the existing software. After understandability it is easy to maintain that software. The feature of the product documentation affects directly the software understanding. With cyclometric complexity and depth of inheritance code metrics documentation of system can be identified.
2. **Conciseness:** Conciseness means a software provides only the information needed to comprehensive the task. Conciseness is a software property provides a task completion with a least amount of code. If the code is concise in nature then maintainability can be cost effective and with less effort. With identified the lines of code , conciseness can be measure.
3. **Legibility:** Legibility consider as a base for software maintenance. If there is legibility of software then it is easy to simplify the required modification, so it distinct that the information should be simple to understand. For maintainability there is need of legibility to understand the software quality.
4. **Modularity:** Modularity is useful to solve big size and complicated problems. In which one large object is decomposed between different objects. Problem can be solved one by one in different objects and system can be understood with different objects not as one object. Modularity of one software can be predicted with coupling between the objects and cohesion. For maintenance coupling between objects should be low and cohesion should be high. Modularity allows find the errors or the functions that required adaption simply and proficiently. This allows to evaluates every fraction of the software individually and then evaluate the combination among these fractions [20].
5. **Reusability:** Reusability is the property to use the existing code to new software implementation with adding some more functionality and make it corrective. Reusability increases the software productivity. If the software has great maintainability then it can be reuse. Reusability can be identified with metrics maintainability. Reusability can speed up the software growth procedure. [23]

IV. QUALITY ANLYSIS BASED ON CODE METRICS

Code metrics are the software measures through which developers can estimate which method should be modified Development teams can recognize potential risks, understand the recent position of a project, and follow evolution throughout software improvement. From the recent studies it is collected that the metrics used for software quality measurement and maintainability have reveal that software properties, personality, record are useful to evaluate the maintainability and quality of that particular software[10].The different kinds of code metrics are given below: We are analyzing the quality of open source software in our purposed work for maintainability prediction to evaluate the maintenance cost of that particular software. Nowadays estimation is crucial in software structure practice [20] .We are using vlc player as our open source software with its evolution process. Vlc player is most popular software used by all the computer users. Vlc player have 82 versions till now from size 6.64MB to 23.9 MB. In our paper we are using 5 most useful versions for maintainability cost evaluation. We are using analyst 4j as my tool for quality measurement. Analyst4j offers an atmosphere to evaluate and visualize code feature with help of programmed software metrics and charts/graphs. Software metrics representation various quality attributes of software/code, which are of great use to understand code quality. Through quality measurement, maintenance cost can be evaluated.

V. EXPERIMENT RESULTS

5.1 Maintainability Index: It finds a key value between 0 and 100 that shows the comparative relieve of software maintenance code. Higher a value better will be maintainability cost. [21]

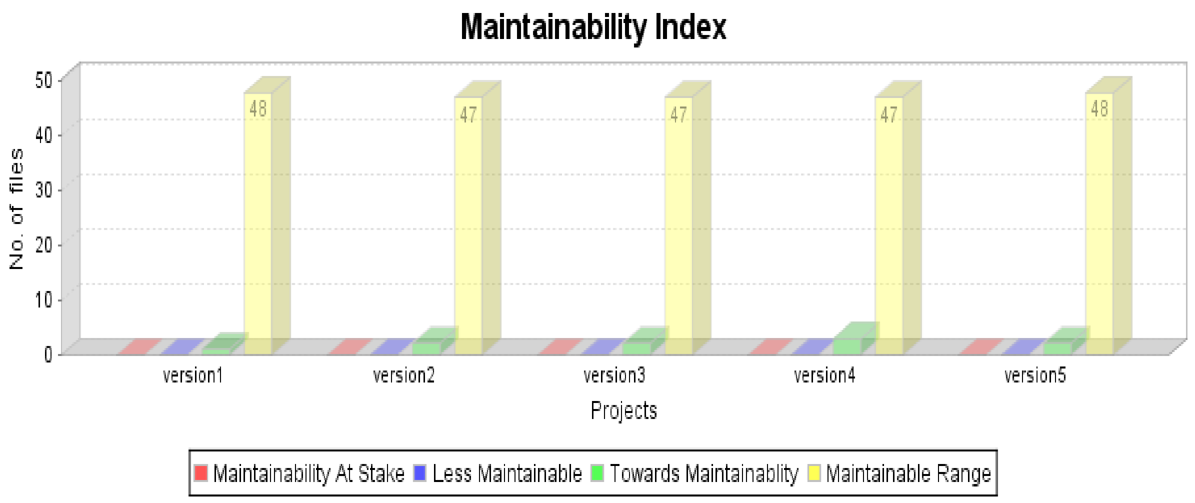

Figure-3 Quality from Version to Version Based on Maintainability Index

In this figure there is maintainability index value for different versions from 1 to 5 are 48,47,47,47 and 48 respectively which shows that version1 and version5 have high maintainability index value and has good maintainability.

5.1.1 Line of code: Indicates the estimated number of lines in the code. The count depends on the IL code so it's not the exact number of lines in the source code file. Higher calculation strength indicates that a kind or scheme is demanding to do extra effort and should be separation. It might also specify that the kind or technique might be tough to maintain.[22]

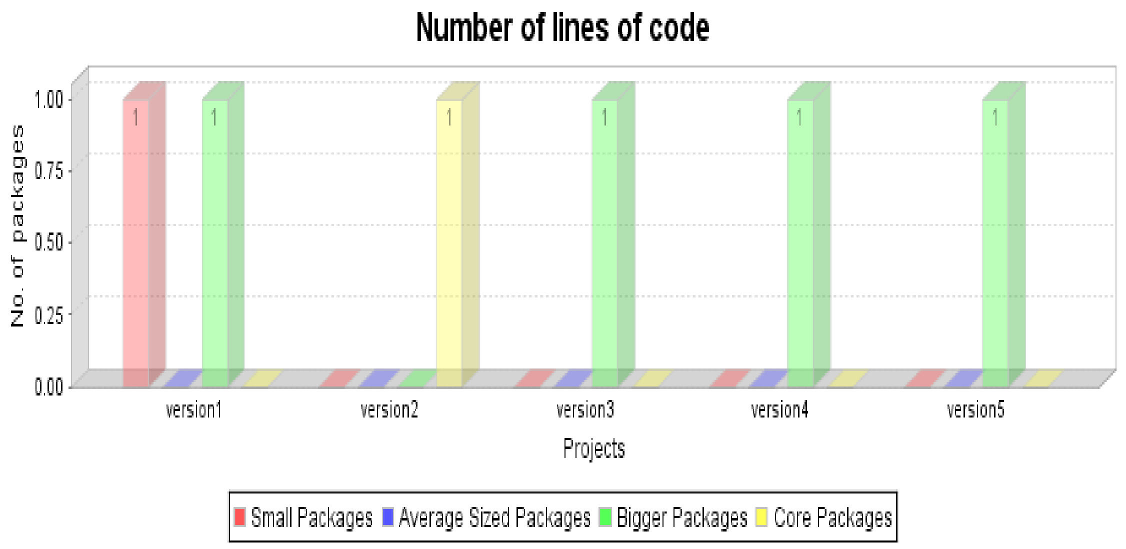

Figure-4Based on Lines of Code

The product which have smaller lines of code are more highly maintable. From this figure it is clear that version 1 have smaller package of code which is more maintable then others.

5.1.2 Cyclometric Complexity: – It calculates the structural complication of the code. It is calculating the amount of dissimilar code paths in the pour of the program. A code that has complicated control flow requires more effort to test and acquire good coverage code and will be less maintainable. [22]

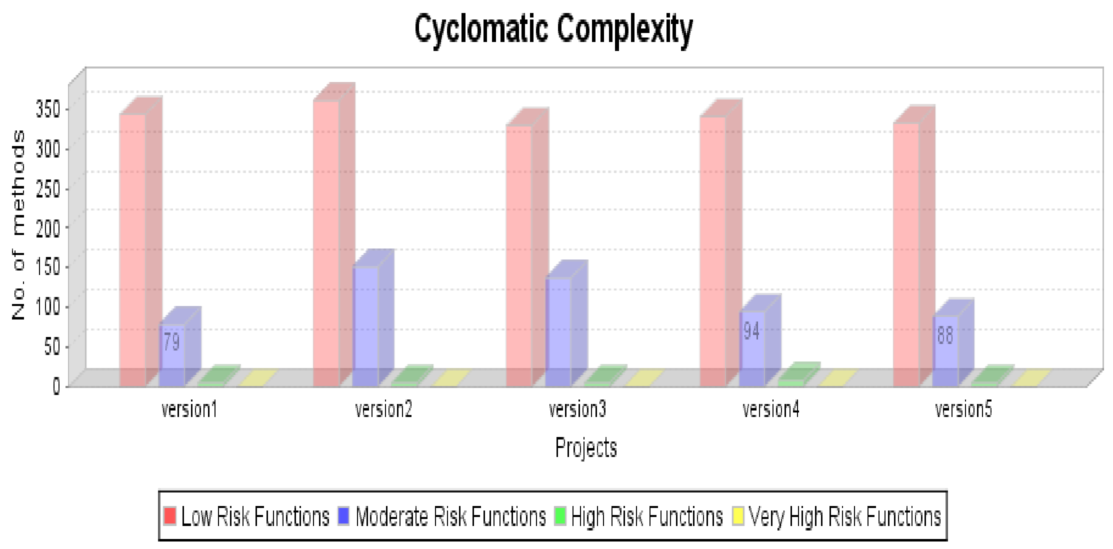

Figure-5 Based on Cyclometric Complexity

Lower complexity indicates higher maintainability. In this figure version 1 have lower cyclometric complexity so it have higher maintainability.

5.2 Depth of Inheritance: The hierarchy from the least child node to the root node is known as depth of inheritance. Shows that the total number of class definitions that expand the path of the class hierarchy. Deeper the hierarchy more effort will be need to understand the code and maintenance will be high.

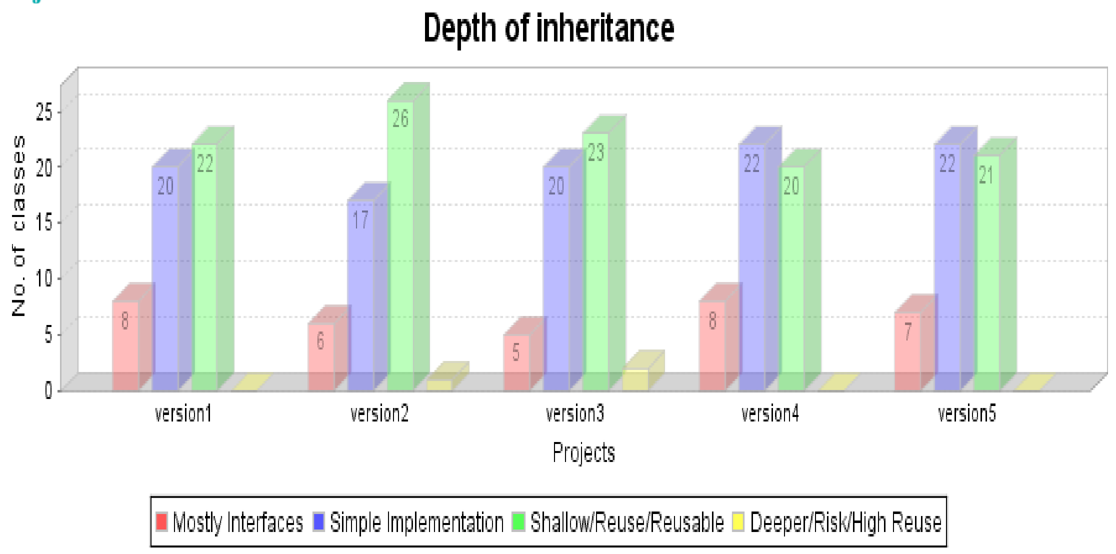

Figure-6 Based on the Depth of Inheritance

Figure shows that version1 have more interfaces and simplicity which shows that less depth of inheritance in the code so version 1 have higher maintenance than others.

5.3 Coupling Between Objects: Measures the coupling between objects through local variables, parameters, return types, generic or template instantiations, base classes, method class, fields defined on external types, and attribute embellishment. Coupling should be low and cohesion should be high for good product design. Low coupling indicates a design that is easy to maintain and reuse because of its less interdependencies on other types.[22]

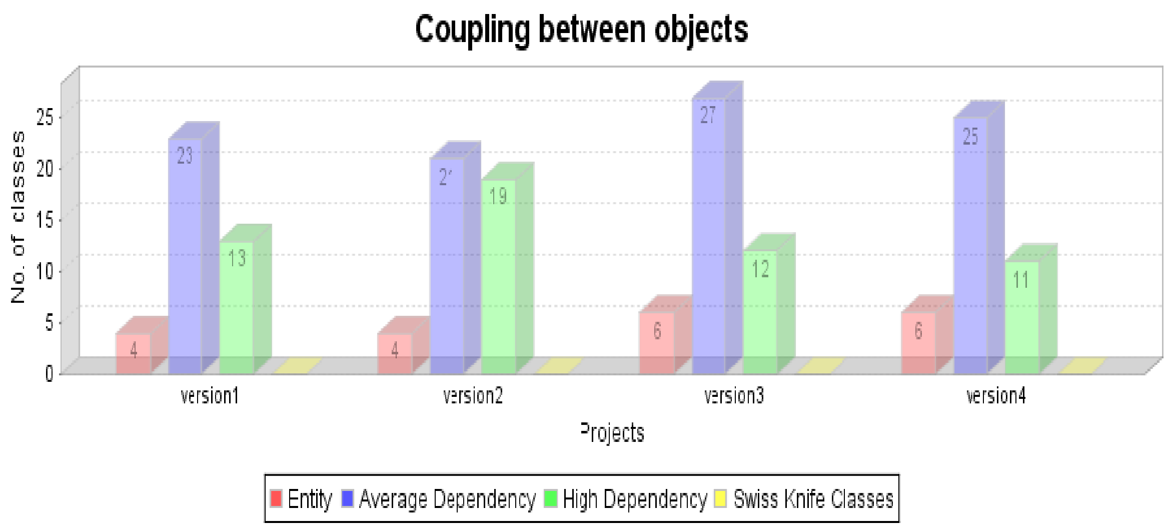

Figure-7 Quality Analysis Based on Coupling Between Objects

If the objects are independent from other objects in the source code then it illustrate coupling between objects are low it shows high maintainability. This figure indicate that version 5 have more value of entity and average dependency which describe that version 5 have low coupling and high maintainability.

5.4 Number of Unused Variables: The variables which are less useful and increase the maintenance

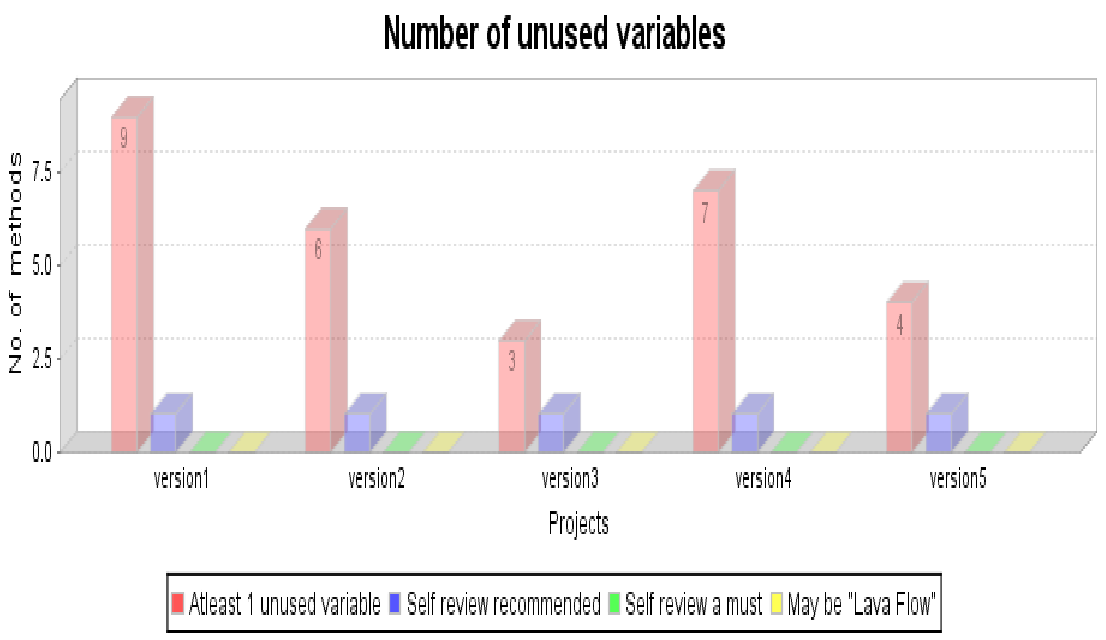

Figure-7 Based on no. of Unused Variables

The codes which have lower amount of unused variables are more maintainable. Figure indicates that version 3 and version 5 have lesser number of unused variables as compare to others so version 3 is more maintainable as compare to others.

5.5 Number of Unused Parameters: The parameters which are not useful in code and make the software complex which effect to maintenance

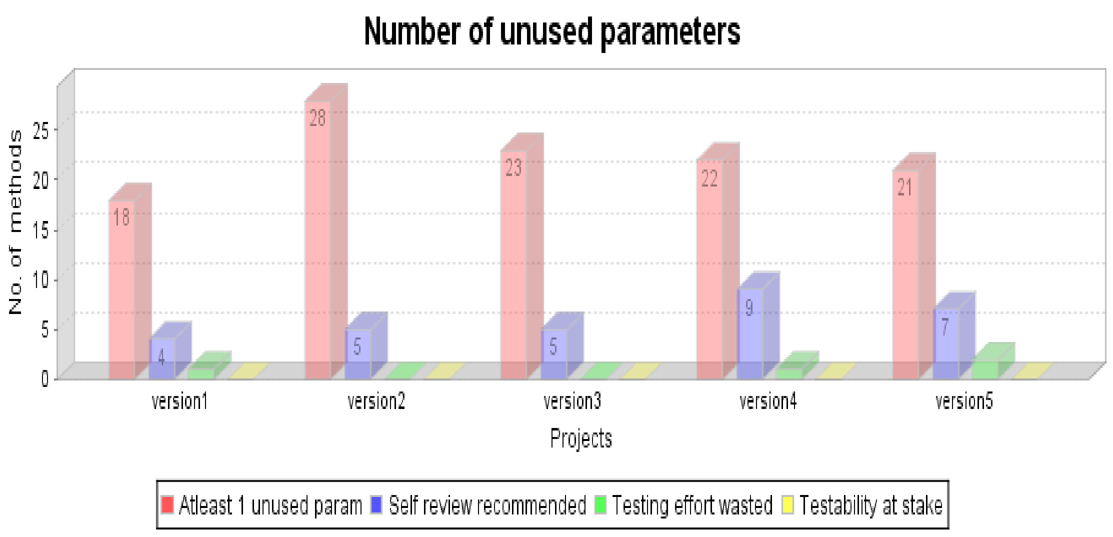

Figure-8 Based on Unused Parameters

It is the same as variables if code have lower number of unused parameters then it will be more maintainable. Figure shows that version1 , version 5 , version4 respectively have lesser number of unused parameters but in theses versions tested effort is wasting so version 3 has less testing effort require and have less unused variable than version 2 so version 3 is more maintainable than others. Table -2 gives the framework for average values of all the versions and are used to evaluate optimal version having better maintainability, cost effective and reusability on the bases of attributes using code metrics.

Table-2

Attributes	Code metrics	V1 (avg value)	V2 (avg value)	V3 (avg value)	V4 (avg value)	V5 (avg value)
Documentation	Cyclometric Complexity	1.77	2.07	2.06	2.02	1.85
	Depth of inheritance	1.42	1.76	1.76	1.46	1.48
Conciseness	Lines of Code	3853	5267	4633	4959	4165
Legibility	No. of unused variables	0.02	0.02	0.01	0.02	0.01
	No. of unused Parameters	0.08	0.08	0.07	0.11	0.11
Modularity	Coupling of Objects	8.63	10.05	8.13	8.45	7.28
Reusebility	Maintanibility index	116.67	105.34	109.62	111.39	116.37

VI. CONCLUSION AND FUTURE WORK

Software maintenance has become an important part of software development. It is the process to make the software corrective, perfective and adaptive by altering the already running software. Researches has described that the software maintenance cost increases day by day because of the change of demand of that particular software like ant viruses .Nowadays it is about 80% of the development cost. So evaluation of software maintenance cost has found to be essential in software practice. Maintainability of the product is the cost required to modify and maintain that software so in this paper a method for maintainability prediction to evaluate the maintenance cost has been used. The software evolution process has been used to find the optimal version which is highly maintainable It has also described that the version which is highly maintainable have high reusability. As a result, from comparative analysis it is concluded that version1 have higher maintainability and have more reusability among others. In near future we will propose new software metric based upon artificial neural network to trace the maintenance cost in more efficient way.

REFERENCES

[1] Pigoski, T. Practical Software Maintenance. Wiley computer publishing, 1997.

[2] AL-Badareen, Anas Bassam, et al. "The impact of software quality on maintenance process." ACM International Journal Of Computers 2 (2010).

[3] Ian Somerville ,"Software Engineering" Chapter 21 ,"Evolution Process" 7th edition published by dorling Kindersley(India) pvt. ltd. copyright © 2004.

[4] Bennett, Keith H., and Václav T. Rajlich. "Software maintenance and evolution: a roadmap." Proceedings of the Conference on the Future of Software Engineering. ACM, 2000.

[5] Fazal-e-Amin, Ahmad Kamil Mahmood, Alan Oxley," An Evolutionary Study of Reusability in OpenSource Software", International Conference on Computer & Information Science (ICCIS) 978-1-4673-1938-6/12/$31.00©2012 IEEE

[6] Pino, Francisco J., Francisco Ruiz, Felix Garcia, and Mario Piattini. "A software maintenance methodology for small organizations: Agile_MANTEMA." Journal of Software: Evolution and Process 24, no. 8 (2012): 851-876.

[7] April, Alain, and Alain Abran. Software maintenance management: evaluation and continuous improvement. Vol. 67. John Wiley & Sons, 2012.

[8] James W. Paulson, Member, IEEE, Giancarlo Succi, Member, IEEE Computer Society, and Armin Eberlein, Member, IEEE Computer Society," An Empirical Study of Open-Source and Closed-Source Software Products", IEEE transactions on software engineering, vol. 30, no. 4, april 2004.

[9] P. F. tiako, "maintenance in joint software development," in computer software and applications conference, 2002. compsac 2002. proceedings. 26th annual international, 2002, PP. 1077-1080.

[10] Don Coleman and Dan Ash, Hewlett-Packard Bruce Lowther, Paul Oman," Using Metrics to Evaluate Software System Maintainability", NlX-9162194/$4.00 © IEEE

[11] Lincke, Rüdiger, Jonas Lundberg, and Welf Löwe. "Comparing software metrics tools." Proceedings of the 2008 international symposium on Software testing and analysis. ACM, 2008.

[12] Ping, Liang. "A Quantitative Approach to Software Maintainability Prediction." In Information Technology and Applications (IFITA), 2010 International Forum on, vol. 1, pp. 105-108. IEEE, 2010.

[13] Sepasmoghaddam, Alireza, and Hassan Rashidi. "A novel method to measure comprehensive complexity of software based on the metrics statistical model." Computer Modeling and Simulation (EMS), 2010 Fourth UKSim European Symposium on. IEEE, 2010.

[14] Tutt, T. E., et al. "Risk-informed Preventive Maintenance optimization." Reliability and Maintainability Symposium (RAMS), 2012 Proceedings-Annual. IEEE, 2012.

[15] Abdellatief, Majdi, Abu Bakar Md Sultan, A. A. Ghani, and Marzanah A. Jabar. "Multidimentional size measure for design of component-based software system." Software, IET 6, no. 4 (2012): 350-357.

[16] Sneed, Harry M., and Katalin Erdos. "Measuring and Evaluating a DotNet Application System to Better Predict Maintenance Effort." Software Measurement and the 2012 Seventh International Conference on Software Process and Product Measurement (IWSM-MENSURA), 2012 Joint Conference of the 22nd International Workshop on. IEEE, 2012.

[17] Lawler, J./ Kitchenham, B.: “Measurement modeling technology”, IEEE Software, Vol. 20, Nr. 3, p. 68

[18] Garcia, F./ Serrano, M./ Ruiz, F./ Piattini, M.: “Managing a software process measurement – a meta-model based approach”,Information Sciences, Vol. 177, Nr. 2, p. 2570

[19] Gokhale, Swapna, Therese Smith, and Robert McCartney. "Teaching software maintenance with open source software: Experiences and lessons." Frontiers in Education Conference, 2013 IEEE. IEEE, 2013.

[20] AL-Badareen, Anas Bassam, Mohd Hasan Selamat, Marzanah A. Jabar, Jamilah Din, and Sherzod Turaev. "The impact of software quality on maintenance process." ACM International Journal Of Computers 2 (2010).

[21] http://blogs.msdn.com/b/zainnab/archive/2011/05/26/code-metrics-maintainability-index.aspx?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+zainnab+%28Visual+Studio+Tips+and+Tricks%29.

[22] http://msdn.microsoft.com/en-us/library/bb385914.aspx.

[23] Neha Budhija and Satinder Pal Ahuja," Review of Software Reusability" .International Conference on Computer Science and Information Technology (ICCSIT'2011) Pattaya Dec. 2011.