International Journal of Scientific Engineering and Technology (ISSN : 2277-1581)
Volume No.3 Issue No.6, pp : 738-742 1 June 2014

Improvement and Implementation of Software Quality by Using Software Metrics

Dileram Bansal, Ajit Saxena, Gajendra Singh

Department of Computer Science and Engineering,
Sri Satya Sai Institute of Science and Technology, Sehore (M.P) India
dileram81@gmail, saxena.ajit2007@gmail.com

Abstract: **Without the software development and software product knowledge it's very complicated to understand, keep away from improvement in the quality of software. There should be some dimension process to forecast the software development, and to appraise software products and its quality. In This paper provides a brief view on Software Metrics, Software Quality and Software Metrics techniques that will forecast and evaluate the specified superiority factors of software which will relate to quality. It additional discusses regarding the Quality as given through the principles like ISO, principal elements necessary for the Software Metrics and Software Quality as the measurement method to forecast the Quality in the Software. Java source code evolution are using for Software Metrics, like Defect Metrics, Size Metrics, and Complexity Metrics. Presented experiments are proving that, the software quality can be analyzed, observed, and enhanced through software metrics usage.**

Keywords*: Software Product Metrics, Software Metrics, Quality, Software Code, Software Product Metrics, Software Quality Metrics, , Software Metrics Usability*

Introduction: As we know that Quality in the software is prime issues to development of any software product and all most all company are working toward in this era. Software market are increasing rapidly and software users are demanding good quality product for this they are ready to pay higher cost With this amplify in prospect and ramble in the software field, countries and multinational companies are ongoing to spend great deal of currency, effort and time to improving the software quality [4]. The number of errors and bugs occurred throughout the process of software development which has to be found in the early on of development stage for good quality. If the errors or bug are found behind schedule, then the corrective accomplishment will be very costly [6, 9]. Software companies will be significantly benefited if there is method to plan and forecast the software development scheduled. The process or method of measuring the software is called as software metrics. Software metrics can be defined as, "Mathematical measure of software that is responsive to differences in software characteristics[10]. It gives us a quantitative measure of a characteristic which the body of Software exhibits.Its aspire is to development process or method of software by controlling the various aspects[1,2] .So it can be said that metrics are used to enhance the ability to recognize , manage and quantify the essential constraint during its development or it can also be said that dimension of software product and the method or process by which it is being developed.

The information gained from software metric can be helped to control and manage the process of development, which will escort to enhancement in the outcome of the software product. Good software metrics should have the capability to forecast the software development method or process. The outcome obtained from the software metrics can be helped to point out, which portions of software code have to modified or changed [12]. Software metrics are considered as a approach to access the quality of big system [13] and have been functional to object oriented systems as well [7, 11, 15]. IEEE prints some standard for the software quality metrics [17], which is help to the development.Main Motive was to provide a methodical approach for the establishment of quality metrics in software through implementing, analyzing and identifying with validating the software quality metrics of a system. The development of metrics as given by IEEE [17] is given are as follow.

- Software Quality metrics Identification
- Software Quality metrics Implementation
- Software Quality Activity
- Analyzing the results of metrics
- Software Quality Requirements Establishment

Software Quality: It's very difficult to define the word of quality not for the reason that of the difficulty to achieve, but because of the difficulty to explain the term.Quality has various meanings for various users. For example if we yet to be paid a vehicle, then will describe the quality as ruggedness of the vehicle, or the fastness of the vehicle, or the looks of the vehicle. So definition of quality varies with the views of the user using or it can also be said as the views of the beholder. When it comes to software, the beholder is, the person using the software, or the person interacting with the software, when it is executed .That is, the person will be satisfied when the software does what he or she wants to do, when it is purchased. The software purchased includes the code, but the users will only be interested in the working and service offered such as the user manual, help and support .In case of software developed for the internal use in company or in an organization, the quality is about the performance of the software whenever the user asks

the development organization to produce it. Quality cannot be defined by the technical excellence alone [18].

Software Quality Standards: There are few set of quality standards pertinent for the companies which involved in development of software. The defined standards should be met by the companies involved in development of software [8]. Few standard Companies responsible for giving the standards are [15].

- ANSI: American National Standards Institute.
- ISO: International Organization for Standardization
- IEEE: Institute of Electrical and Electronics Engineers
- EIA: Electronic Industries Association.
- IEC: International Electro technical Commission
- AIAA: American Institute of Aeronautics and Astronautics.

Various standard organizations have various definition for their quality standards. ANSI is the simply organization that does not build standards but it approves the standards.The usage of software metrics will decrease the subjectivity during the evaluation of software quality and it gives decisions making ability about the software quality [12, 13].

Software Metric: Software Metrics can also be helped to distinguish the duplicated code which can afterward be uninvolved by applying suitable refactoring [11]. As we have discussed previously, software metrics is alienated in to two parts: software process metrics and software product metrics. Software product metrics is used to assess the final products of the software i.e design documentation or software code. Software process metrics is used to assess the software development process, ie. Overall development time and Methodology Type.Software quality requirements establishment is the primary level of software metrics and its begins with the all the attributes that describe the software quality requirements should be settled through the user-oriented views and management are then assigned to the attributes [17].

Figure 1: Software Metrics Model

Software metrics measuring software products is various for various paradigms. Figure 1 is showing the general view of software metrics model.

Product Metric: Product metrics are usually derivative from the system itself [14]. The metrics information of this type can be composed after exact time intervals. The preliminary work in product metrics deals with the attributes of the source code.

It's forever better to have metric information in the preliminary stages of software development because it will raise the probability of controlling the Software development process with results. Figure 2 is showing the general view of software product metrics model.

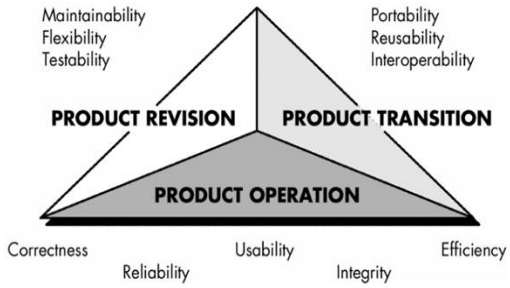

Figure 2: Software Product Metric Model

Size: Size software metric is controversial metric for software but for the most part its used [14, 15]. It becomes controversial for the reason that there is no ideal assessment for size, which everybody agrees on. The size metrics is an stab to measure the software size, and the broadly used size metrics is Lines of Code (LOC). The size metrics has some deficiency for the reason that it cannot be measured until the method or process of software development is completed. Some Halstead's metrics are also used to assess the size metrics, but they are not discussed hear. Lines of Code (LOC) are one of the mainly used metrics for the program size [11]. LOC is calculated through the total number of lines of code in a function. The total number lines can be with or without the blank and comment lines [30]. The decision to include the blank and comment lines will be of the developers. The size metrics can be extended to measure the size of a system by summing all the LOC metric values of all the functions in the system. The calculated values of lines of code metrics is shown in results section (Table 2 to Table 6).

Complexity: Complexity software metrics is measured as the measure of control flow in a function. The complexity software metrics is used to measure the relation between the multifarious codes and its failures. The best example of Complexity software Metrics is Cyclomatic Complexity Metrics. Cyclomatic Complexity metrics was proposed by McCabe in the year 1976. It is a measure derived from the product itself [17]. It is helped to assess the control flow complexity in a function. It is also considered as one of the internal metrics, as it built early warning from the collection of the collection of internal metrics [17]. The measured values of cyclomatic complexity metrics can be calculated numerically or can be represented in figures. There are tools for representing the cyclomatic complexity in figures. The calculated cyclomatic complexity is shown in results section

Defect: It is an external measure of the system derived from the external assessment of the behavior of the system [16]. It is used to measure the number of defects in a software product and the data required for the metrics is collected from the product itself. So, it can be said that it quantifies the product metrics. There has been no particular procedure for the

measurement of number of defects. One of the alternative methods for the defects metrics is to find number of errors during code inspection.

Method: The Method used in this paper consists of both investigation and practical approach. The investigation on Software Quality Software Quality through case study of different journals, text books, research papers, online material and by the usage of standards such as ISO. Some of the Software Quality Metrics, such as Product metrics through journals, text books, research papers, online material and by the usage of standards such as IEEE. The practical approach is on Software Product Metrics, such as: Lines of Code: The Lines of Code metrics can be found by using the integrated development environment (for example: eclipse) or by running code in compiler, which gives the total number of lines and in case of any error in the code, it also gives the lines of errors. Cyclomatic Complexity Metrics: It can be found by the using the software Cyvis[37], in which the metrics to find the complexity, total number of methods and the number of lines in each method is predefined. Defect Metrics: It is the total number of errors found during the execution of program.

RESULTS

Total number of line which is executable in the code can be calculated with the counting of the line in a code. i.e., not including the commentary lines. The results are further discussed in discussions section.

Table 2: Summary of Lines of Code Metrics

Number of Classes	Class name	Lines of Codes	Number of methods
1	Class 1	435	14
2	Class 2	423	14
3	Class 3	67	4
4	Class 4	47	4

Table 3: Lines of Code Metrics for Class 1 (Summary)

Number of Methods	Method	Lines of Codes
1	Run Server	93
2	Get Stream	16
3	<init >	61
4	Process Connection	63
5	Display Image	6
6	Send Data	21
7	Set text field editable	4
8	Wait for Connection	45
9	Access $000	8
10	Access $100	3
11	Access $200	3
12	Close Connection	15

Table 4: Lines of code Metrics for Class 2 (Summary)

Number of methods	Methods	Lines of Codes
1	Wait for Connection	24
2	Process Connection	33
3	Run Client	47
4	Send Data	27
5	<init >	43
6	Display Image	6
7	Close Connection	23
8	Get Stream	29
9	Access $000	7
10	Access $100	3
11	Access $200	3
12	Set text field editable	6

Table 5: Lines of code Metrics for Class 3 (Summary)

Number of Methods	Methods	Lines of Codes
1	<init>	5
2	Main	37

Table 6: Lines of code Metrics for Class 4 (Summary)

Number of Methods	Methods	Lines of Codes
1	<init>	3
2	Main	18

Cyclomatic Complexity: With the help counting the total number of methods we can calculate to the cyclomatic complexity of every class, and the complexity implicated throughout its control flow. Cyclomatic complexity experiments results are shown in Figure 3, and Figure 5 - 8 are showing each class cyclomatic complexity.

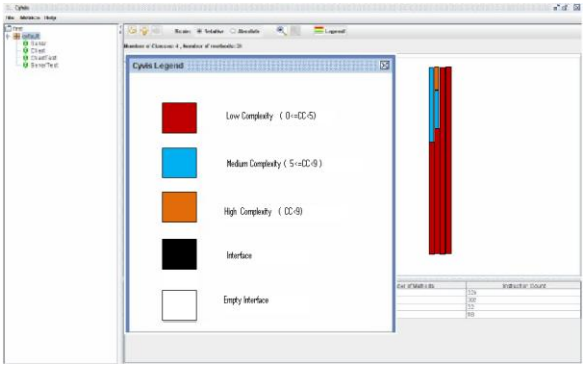

Figure 3: Cyclomatic Complexity

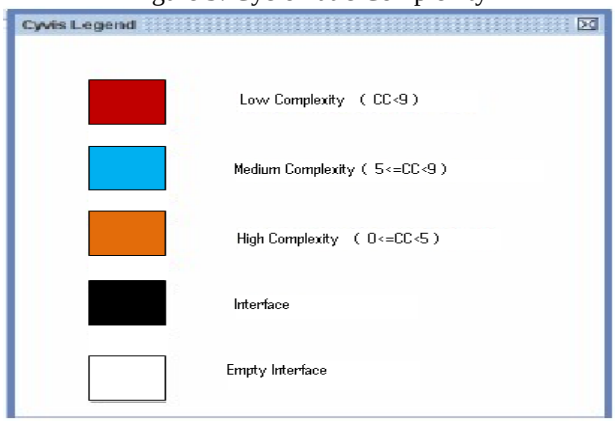

Figure 4: Color Coding for Cyclomatic Complexity Metrics [37]

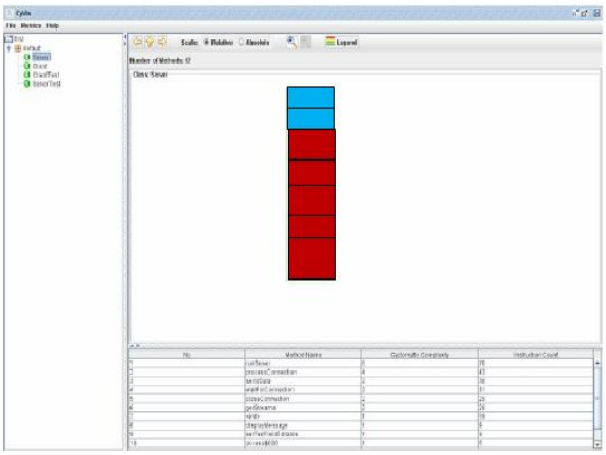

Figure 5: Cyclomatic Complexity for Class 1

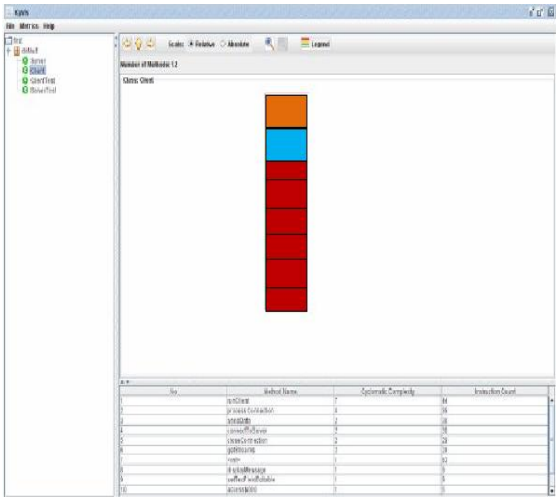

Figure 6: Cyclomatic Complexity for Class 2

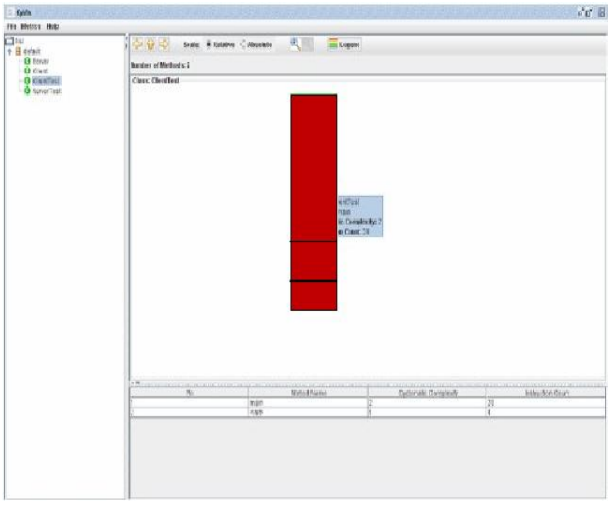

Figure 7: Cyclomatic Complexity for Class 3

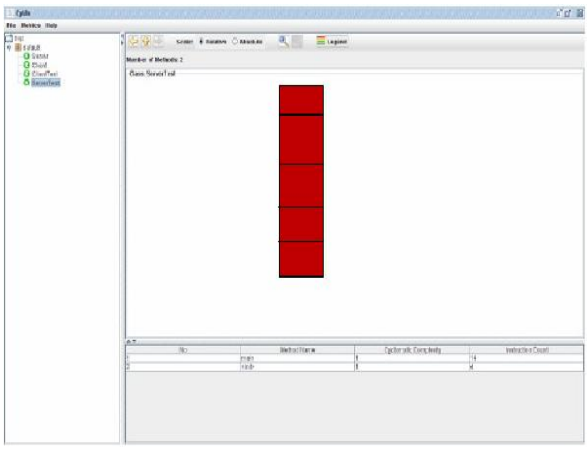

Figure 8: Cyclomatic Complexity for Class 4

Summary: Any metrics cannot describe to the quality or can be correlated to the quality. However, they can be use to improvement of the quality. They can be used to describe the parameters that influence the quality and also the changes that can be prepared to get better the quality. The other major advantages of these metrics are that, they can be used to generate the test or experiment cases for software testing. They also supply us with the data like the total number of lines not comment line in the code, the most multifarious piece of code and also the total number of process or methods contain in the code. Every metrics provides us with exact code information. The lines of code metrics signify the size or length of program and also the total number of methods or process implicated in the program. The results (Experiment) of the Lines of Code Metrics are discussed below: summary with related to source code of the java in table 3.

It consists of 4 classes and the metrics for lines of code for each is calculated. The number of methods involved in each class is also calculated. By summing the total number of lines for each code, the total size of the system can be found. Table 3 to table 6 gives the summary for Class 1, class 2, class 3 and class 4. The usage of this metrics will reduce the subjectivity by providing the total number of lines in each class and the number methods present in it. It makes the software more clear and visible. The lines of code for each class can be cross checked by comparing it with summary of classes given in Table 3, Table 4, Table 5, and Table 6. Cyclomatic complexity, apart from providing us with the complexity in each and every method involved in the code, also provides us with the flow of complexity i.e., structural complexity. It also indicates how complicated the flow is in a function and also indicates how many test cases are needed to perform the basis path testing on the function. The results of Cyclomatic Complexity Metrics are discussed below: Figure 4 shows the cyclomatic complexity for java source code. As discussed in lines of code metrics, it consists of 4 classes and flow complexity is shown in Figure 4. The vertical bars represent the classes, and it is from left to right. The horizontal bars represent the method involved in each class, and it is from top to bottom. The colors shaded in

each method represent the cyclomatic complexity of that method. The meaning of the color and its level of complexity is shown in Figure 4. The method with blaze orange color will have the highest cyclomatic complexity, and its value will be greater than or equal to 7. Blue color represents medium cyclomatic complexity with its value ranging between 5 and 9. Red color is for low cyclomatic complexity and it will range between 0 and 5. As there has been no interfacing in the java source code, interfacing is not being discussed. But its color representation is shown Figure 4. Figure 5 shows the cyclomatic complexity for class 1. From Figure 5, it can be said that the first two methods divided by horizontal lines have the moderate complexity, and the ten methods below it have the low level of complexity. Figure 6 shows the cyclomatic complexity for class 2. From Figure 6, it can be said that the first method in this class has the highest complexity, followed by the second method with medium complexity, and the remaining ten methods have the low complexity. Figure 7 shows the cyclomatic complexity for class 3. From Figure 7, it can be said that the two methods in it have the low level of complexity. The value of the complexity of particular method can be viewed at the bottom of Figure 5, Figure 6, Figure 7, and Figure 8. The value of complexity can also be found by placing the arrow over the particular method as shown in Figure 7. Defect metrics does not have particular procedure to measure the total number of defects in the system. The alternative method is to calculate some of the characteristics of the code. As the java source code has been provided after its development, only one characteristic of it has been calculated i.e., the total number of errors during code inspection. The java source code has been inspected and the total number of errors during inspection has been found.

CONCLUSION

In this Paper we have discussed about quality of the software, metrics of software and various application and standard for those.

For the results point of view we have used java code and its evaluated on the pre-defined or pre-selected metrics and noted down the value of different metrics. From the presented values of metrics i.e., number of errors, cyclomatic complexity and lines of code. From these metric it's clear that we can improve the quality of the software and easily identify area where need to concentrate during development of the software.

REFERENCES:

i. *Konstantinos Stroggylos, Diomidis Spinellis: "Refactoring – Does it improve software quality?". IEEE publications, Fifth International Workshop on Software Quality 2007.*

ii. *software quality?". IEEE publications, Fifth International Workshop on Software Quality 2007.*

iii. *Dr. James A. Bednar and Dr. David Robertson: "Software Quality and Standards". SEOC2 Spring 2005, Quality/Standards.*

iv. *Demeyer and S. Dueasse. Mettles: "Do they really help? In Proc. Languages at Modules and Objects". Hermes Science Publication, pages 69-82 2004*

v. *Evans, Isabel: "Achieving Software Quality Through Teamwork".Norwood, MA, USA: Artech House, Incorporated, 2004.*

vi. *Kitchenham, B: " The Failure of Quality, Proceedings of the Second Workshop on Software Quality". ICSE 2004.*

vii. *Chulani, S, Ray, B., Santhanam, P. and Leszkowicz, R.:"Metrics for Managing Customer View of Quality", IEEE Metrics conference, Sep. 2003*

viii. *Tom Mens and Serge Demeyer: " Future Trends in Software Evolution Metrics". ACM publications 2002.*

ix. *O'Regan and Gerard: "Practical Approach to Software Quality". Secaucus, NJ, USA: Springer-Verlag New York, Incorporated, 2002*

x. *Jeffrey Voas: " A New Generation of Software Quality Conferences", IEEE publications, IEEE Software January/February 2000.*

xi. *Wei li: " Software Product Metrics – Using them to Quantify Design and Code Quality". IEEE publications, December 1999/ January 2000.*

xii. *N. Fenton and S. L. Pfleeger: "Software Metrics: A Rigorous and Practical Approach". International Thomson Computer Press, London, UK, second edition, 1997.*

xiii. *Wei Li and Harry Delugach: "Software Metrics and Application Domain Complexity". Computer Science Department The University of Alabama in Huntsville Huntsville, AL 35899, IEEE publications 1997.*

xiv. *B. Lagufi, D. Proulx, E. M. Merlo, J. Mayrand, and J. Hudepohl: "Assessing the benefits of incorporating function clone detection in a development process". IEEE Computer Society Press, 1997.*

xv. *ISO/IEC: "DIS 14598-1 Information Technology – Software Product Evaluation". ISO 1996.*

xvi. *McCabe, Thomas J., and Schulmeyer, G. Gordon: "The Pareto Principle Applied to Software Quality Assurance, in The Handbook of Software Quality Assurance", Schulmeyer, G. Gordon and McManus, James I., eds., New York: Van Nostrand Reinhold Company, Inc., 2nd ed., 1992.*

xvii. *McCabe, T. J.: "A Complexity Measure". IEEE Transactions on Software Engineering, Vol. 2, No. 4, pp. 308-320, 1976.*