ANALYSIS OF SOFTWARE QUALITY USING SOFTWARE METRICS

Ermiyas Birihanu Belachew[1], Feidu Akmel Gobena[2] and Shumet Tadesse Nigatu[2]

[1]Dept. of Software Engineering and Computer Science, Wolkite University, Ethiopia
[2]Dept. of Service and Information System Engineering, Polytechnic University of Catalonia, Spain

ABSTRACT

Software metrics have a direct link with measurement in software engineering. Correct measurement is the prior condition in any engineering fields, and software engineering is not an exception, as the size and complexity of software increases, manual inspection of software becomes a harder task. Most Software Engineers worry about the quality of software, how to measure and enhance its quality. The overall objective of this study was to asses and analysis's software metrics used to measure the software product and process.

In this Study, the researcher used a collection of literatures from various electronic databases, available since 2008 to understand and know the software metrics. Finally, in this study, the researcher has been identified software quality is a means of measuring how software is designed and how well the software conforms to that design. Some of the variables that we are looking for software quality are Correctness, Product quality, Scalability, Completeness and Absence of bugs, However the quality standard that was used from one organization is different from others for this reason it is better to apply the software metrics to measure the quality of software and the current most common software metrics tools to reduce the subjectivity of faults during the assessment of software quality. The central contribution of this study is an overview about software metrics that can illustrate us the development in this area, and a critical analysis about the main metrics founded on the various literatures.

KEYWORDS

Software Metrics, Software Quality, Software Testing, Software Faults, Software Engineering

1. INTRODUCTION

Correct measurement is the prior condition in any engineering fields, and software engineering is not an exception. Software metrics have a direct link with measurement in software engineering. According to De Marco; "You can't manage what you can't measure!" [1] And Campbell also gives an emphasized the importance of measurement in software engineering by setting "If you aren't measuring, you aren't managing" — you're only along for the ride [2].

Software metrics will reduce the subjectivity of faults during the assessment of software quality and it provides a quantitative basis for making decisions about the software quality. Metrics are the numerical value of software and it is used to predict the fault [3]. Software metrics occur file-level, class-level, component-level, method-level, process-level and quantitative values-level metrics [4], This helps the project manager and software engineers to find defects and making the prevention method for the defect.

Software metrics can be applied to each software development phase. During requirement analysis software metrics can be developed, for instance, in order to determine cost estimation and resource needed. At the time of system design we also develop metrics in order to count

function point. Metrics applied at implementation phase are also used to measure software size [5]. According to Vikas Verma , having software metrics have a number of benefits such as provide a basis for estimation and facilitates planning ,a means for controlling status reporting, identifying risk areas and effectiveness and efficiency of testing[6]**.**

Measuring the software project has a number of benefits for company it saves development effort, time and money. In addition to this for complex projects using metrics have easy to understand, identify common problems early, and manage resources [7].As mentioned above even if it has the benefit there is also drawbacks that are better understanding (knowledge) and need a lot of effort and time. Software metrics enable software developers to analyze their code and make improvements if required. Metrics can be developed for software size, cost estimation, software quality, maintainability, defect analysis and software testing [5]. In this paper, we studied the state of art of software metrics; specifically focus on the quality metrics to measure software's both product and process.

2. STATEMENT OF PROBLEMS

Software defect is not only the quality of software, but also increase costs and suspend the development schedule. In addition to above, there could be many reasons for the system to be faulty; most of them are because of the human factor, mistake and errors made in designing or coding by the peoples, data entry, documentation and communication failures. In order to solve the most common human factors we shall use software fault prediction before us taking the software systems to test and maintains.

Software testing is one of the most critical and costly phases of software development. Project managers need to know ''when to stop testing?" and ''which parts of the code to test?". The answers to these questions would directly affect defect rates and product quality as well as resource allocation (i.e. Experience of test staff, how many people to allocate for testing) and the cost.

As the size and complexity of software increases, manual inspection of software becomes a harder task. In this context, alternative methods that are used to predict potential effects clearly are software defect prediction. Since testing typically consumes 40 - 50% of development efforts, and consumes more effort for systems that require higher levels of reliability, it is a significant part of the software engineering [8-10]. Software fault predictions used to solve the problem that reduces the qualities software product. In this paper, we studied the software tester to predict and fix the bug before it is delivered to customers that assures the quality of software and evaluate the state of art of software metrics; specifically focus on the quality metrics to measure software's product and process.

3. OBJECTIVE

The primary objective of this study was to asses and analysis's software metrics used to measure software quality particularly software product and process

.

4. LITERATURE REVIEW

The first survey on software metrics was done by Kafura in 1985 and he suggests existing code metrics, complexity metrics and validation metrics. Generally , in this survey work have shown that the major relationships exist between the software metrics and quality attributes such as

comprehensibility of code, error characteristics, length of coding time, and structural soundness [11].

According to Ming Chang et al [12] discussed the role of software metrics and software measurement for software quality. Authors also classified the software metrics according to various manners which are commercial, important, observation, measurement and software development. In addition to this, the author also discussed various methodologies which are around 15 measurement methodologies and 24 types of testing with their definitions, formula and effects.

Poornima Gupta et al [13] presented the software fault prediction using artificial intelligence methods and this research work focused on related work on software metrics particularly on AI approaches and software metrics.

Kunal Chopra et al [14] discussed about software metrics complexity using Ndepend to measure software product such as size metrics, control flow metrics and data flow metrics. The final contribution of the researcher was introducing the most common known and used software metrics proposed and evaluation their use in constructing models of the software development process.

5. CLASSIFICATION OF SOFTWARE METRICS

In order to measure the software from the requirement up to generating of source codes the software metrics can be grouped as follows:

- Product metrics (code metrics)
- Process metrics
- Resource metrics

5.1 PRODUCT METRICS (CODE METRICS)

This is one of the software metrics that we are applied to measure the quality of the software systems; mostly it measures the system final product such as software code or design documentation. It can be size metrics, the complexity metrics (Cyclomatic and Halsted). It can also be internal or external attribute measurement of the products [5]**.**
The external attributes include the parameter that is going to measure: software usability and re-usability, portability and efficiency and the internal attribute includes the size of the software, correctness, complexity, bugs and testability.

5.1.1 SIZE METRICS

The size metrics is an attempt to quantify the 'Size' of software, and the widely used metrics is a Line of Code (LOC). The size metrics have some limitation because of it cannot be measured by the process of development is completed. It measures the implementation of the project only. The size metrics are used to measure "volume, length, quantity, and overall magnitude of software products" [15]. The main strength of line of code metrics is easy to count extensively automated for counting and requires various software estimating tools while the drawbacks are may contain dead code, blank and comments, mixed language projects and ambiguous software reuse [16].

5.1.2 CYCLOMATIC COMPLEXITY METRICS

The word Cyclomatic comes from a number of fundamental cycles in connected, undirected graphs [17]**.** This was proposed by McCabe for measuring the software especially for Design phase. McCabe described by viewing the program graph, and finding the number of paths to reach entire graph. If the complexity of the program become long, it is difficult to easily count the number of paths. Due to the above reason McCabe suggests counting the number of basic paths, which is Cyclomatic number**.** Cyclomatic complexity has two basic benefits which giving it the number of recommended tests for software and it is used in all phases of software development cycles, beginning for the design phase in order to achieve software quality parameters .

Cyclomatic complexity can be expressed as the following equation 1:

$$V(G) = E - N + 2P \text{ or } V(G) = E - N + 2 \quad \text{---------- (1)}$$

Where:

V (G) = Cyclomatic Complexity
E =Number of edges
N =Number of nodes
P =Number of connected components or parts

McCabe rule to count edge and nodes [18]:

To count the number of nodes and edges for graph McCabe sets the rule these are as follow:

- **If / while statements:** This rule is applied to binary counting mainly for and/or operations in our programming code. For instance 'if (count < m || count = MAX) then...' in this example we have seen two binary decision nodes.
- **Do / for statements: "**iteration statements count as one binary decision node**" [18].**
- **Case / switch statement:** There is an option to be n or n-1(where n is the number of iterations). For different programming language, there may be different alternative iteration.

5.1.3 HALSTEAD METRICS

Halsted suggested that measuring the software program by counting the number of operator and operands in program. In order to measure operands and operator he developed the basic formula that includes vocabulary, length and Volumes of the software the program [19] **.**

- **Program Vocabulary**: This includes the total number of operands and operator. Program vocabulary can be expressed in equation 2 :

$$n = n1 + n2 \quad \text{---------- (2)}$$

Where:

n =Program vocabulary
n1 =Number of unique operators
n2 =Number of unique operands

Program Length*:* it refers to the total usage of 'all' operators appearing in the implementation plus the total usage of 'all' operands appearing in the Implementation. Program length can be expressed in equation 3

$$N=N1+N2 \quad \text{---------- (3)}$$

Where

N = N o program length
N1 = 'all' operators appearing in coding
N2 = 'all' operands appearing in coding

Program Volume: it refers to the size of the program. Program volume can be expressed in equation 3.4

$$V=N \log 2\, n \quad \text{---------- (4)}$$

Where

V= program volume
N =program length (from Equation3.3)
n= program Vocabulary (from Equation 3.2)

5.2 MATHEMATICALLY EXTRACTING ATTRIBUTES

The existing software metrics tools and how they internally measures the given source code have discussed previously. Here, we focus on doing the measurement practically with sample code the product metrics (McCabe metrics, Halsted metrics and Line of source code). The programming language that we selected to show was C++, but it's also possible to choose other programming Language. Sample code to show how we can measure C++ code using software metrics tools is depicted in algorithm 1 below:

Table 1: Measuring source code via Halsted Metrics

Algorithm 1 Sample source code for Halsted Metrics

```
1: void sort(int*a, int n){
2: inti,j,t;
3: if (n <2) return;
4: for(i=0;i<n-1i++){
5: for(j=i+1;j<n;j++){
6: a([i]) > a([j]){
7: t = a[i];
8: a[i] = a[j];
9: a[j] = t; }}}
```

Measuring source code via Halsted Metrics

Count	Operands	Count	Operators	Count	Operators
3	<	3	{	1	0
5	=	3	}	2	1
1	>	1	+	1	2
1	-	2	++	6	a
2	,	2	for	8	i
9	;	2	if	7	j
4	(	1	int	3	n
4	)	1	return	3	t
6	[]				

Further Halsted Metrics measures from the sample code

Total number of operands(N1)=50
Total number of operator(N2)=31
Unique number of operator(mu1)=17
Unique number of operands(mu2)=7
Length(N)=N1+N2=50+31=81
Vocabulary(V)=n1+n2=17+7=24
Difficulty(D)=$(mu1/2) * (N2/mu2) = 17/2 * 30/7 \approx 36$
Volume(V)=Nlog2n=81log224=371
Effort(E)=V*D=371*36=13356
Effort(E)=V*D=371*36=13356
Time to understood(T)=E/18

Cyclomatic Complexity measure from the sample code

Count if=2
Count Loops=2
Control flow graph(V[G])=count if+Count Loop +1=5

5.3 PROCESS METRICS

This is software metric that is used to measure the quality of the software system. It measures the software development life cycles such as type of methodology, the staff status and the required time to finalize the system. This is mainly measured in software development life cycles and it measures the parameter such as duration estimate, cost assessment, effort required, process quality and effectiveness/efficiency of the development process [5]**.** Prone

5.4 OBJECT ORIENTED METRICS

Since object oriented programming becomes popular now a time; as it increases its popularity most companies are going to use it, the complexity and fault proneness become a problem so that there is software metrics to resolve this problem before delivered to the customers. The most known object oriented metrics are Chidamber and Kemerer (CK) metrics [20]. Chidamber and Kemerer (CK) metrics were designed to measure the object oriented having the feature such as inheritance, coupling, and cohesion[20]. The basic difference between OO metrics and Procedural

one is quickly led to new kinds of software metrics aimed exclusively at OO projects [16] **.** The main advantage OO metrics are psychologically attractive and simple from complex OO projects [16]**.** The drawback OO metrics are do not support studies outside of the OO paradigm, full life-cycle issues, have not yet been applied to testing, maintenance and lack of automation [16] **.** In Table 2 below object oriented metrics and its description was discussed.

Table 2: Chidamber and Kemerer (CK) metrics and their descriptions

No	Name	Description
1	WMC	Weighted methods per class
2	DIT	Depth of inheritance tree
3	NOC	Number of children
4	CBO	Couplinng between object classes
5	RFC	Response for a class
6	LCOM	Lack of cohesion in methods

6. CURRENTLY EXISTING SOFTWARE METRICS TOOLS

There is a number of software metrics tool in the market. The basic goal of these metrics is to upgrade the process of developing, maintaining and managing software's. Some of them are open sourced while others are proprietary tools. The main differences that we were looking in each are language support, platform support, licensing prices, supporting metrics, availability and license type. Simply we can conclude that metrics based assessment of a software program and measures taken to improve its design differ considerably from tool to tool. For the purpose of determining the suitable metrics the researcher passes through free searching on internet and finally they have got around 46 metrics [21]. Some of them are mentioned in Table 3 below.

Table 3: Summary

No	Name	Link to home page	License type	Programming Support	Operating System	Supported metrics
1	Analyst4j [22, 23]	www.codeswat.com	Commercial	Java	Windows	WMC, RFC, CBO,DIT, McCabe, Halstead Effort,Volume
2	CCCC [24]	http://www.sourceforge.net/projects/cccc/	Open source	C++ and Java file	Solaris	-lines of code, McCabe's complexity and metrics
3	Chidamber & Kemerer Java Metrics [25]	http://www.spinellis.gr/sw/ckjm/	open source command-line tool	Java	Windows 9x/ME/NT/2000 /XP, Unix and Linux	C&K object-oriented metrics
4	Dependency Finder [21]	http://depfind.sourceforge.net/	open source	Java	Windows 9x/ME/NT/2000 /XP, Unix and Linux	object-oriented software metrics
5	Eclipse Metrics Plug-in 1.3.6 [26]	http://sourceforge.net/projects/metrics/	Open source	Java	Windows 9x/ME/NT/2000 /XP, Unix and Linux	cycles in package and type dependencies
6	Eclipse Metrics Plug-in 3.4 [27]	http://eclipsemetrics.sourceforge.net/	Open source	Java	Windows 9x/ME/NT/2000 /XP, Unix and Linux	metrics during build cycles through the Problems view, of 'range violations' for each metric
7	OOMeter [21, 28]	Research papers	Commercially available	Java, C# source code and UML models	Windows 9x/ME/NT/2000 /XP, Unix and Linux	Measure a number of quality attributes (coupling, cohesion and complexity and code metrics such as lines of code (LOC).)
8	Semmle [29]	http://semmle.com and https://en.wikipedia.org/wiki/Semmle	Commercially available	object-oriented codes		To search for bugs, measure code metrics
9	Understand [30]	Research paper	commercially available	14 languages (Java, C,C++,fortan ,PHP .....)	all major operating systems including Solaris	maintaining, measuring and analyzing source code
10	QA-C[5]	Research paper	commercially available	C	Windows, Solaris (sparc) and RedHat Linux	Halstead metrics and cyclomatic complexity metrics
11	Testwell CMT++ [5]	Research paper	commercially available	C/C++	Windows and Unix only	complexity measurement tool

7. CONCLUSION AND FUTURE WORK

In software development, software testing is highly desirable to assure the quality of the software product. Software testing performed via manual and software metrics, the former one (manual) is costly and it required high time interval to perform it because of it now a day software engineer moves to systematic measurement method which is software metrics. This study conducted to

reveal to asses and analysis's software metrics used to measure software quality particularly software product and process. Software metrics used to measure the software product and process. The researcher used a collection of literatures from various electronic databases which available since 2008 to understand and know the software metrics; the researcher has been identified software quality is a means of measuring how software is designed and how well the software conforms to that design. Some of the variables that we are looking for software quality are Correctness, Product quality, Scalability, Completeness and Absence of bugs, However the quality standard that was used from one organization is different from others for this reason it is better to apply the software metrics to measure the quality of software and the current most common software metrics tools. In the future the researcher recommends the specific application area of each software metrics and how can perform by the researcher to enhance the quality of software applications.

REFERENCES

[1] D. a. Tom, Controlling Software Projects. New York: Yourdon Press, 1986.

[2] Campbell , Luke, and B. K, "Software Metrics: Adding Engineering Rigor to a Currently Ephemeral Process " 1995.

[3] Pooja P and D. A. Phalke, "Survey on Software Defect Prediction Using Machine Learning Techniques," International Journal of Science and Research, vol. 3, December 2014

[4] Malkit S and D. S, "Software Defect Prediction Tool based on Neural Network " International Journal of Computer Applications, vol. 70 2013.

[5] H. R. Bhatti, "Automatic Measurement of Source Code Complexity," MASTER'S THESIS Computer Science and Engineering Luleå University of Technology.

[6] Vikas V and S. M, "Applications of Software Testing Metrics In Constructing Models Of The Software Development Process " Journal of Global Research in Computer Science, vol. 2 pp. 96-98 May 2011.

[7] Fernando, Wijayarathne, M. D. Fernando, and I. Guruge, "The Importance of Software Metrics: Perspective of A Software Development Projects In Sri Lanka," SAITM Research Symposium on Engineering Advancements, 2014.

[8] C. Z. a. Q. SHICHAOZHANG, "Data Preparation for Data Mining " Applied Artificial Intelligence vol. 17, pp. 375–381 2003.

[9] S. Shivaji, "Efficient Bug Prediction and Fix Suggestions " PhD A dissertation UNIVERSITY OF CALIFORNIA 2013.

[10] Chayanika S, Sangeeta S, and R. S, "A Survey on Software Testing Techniques using Genetic Algorithm," International Journal of Computer Science vol. 10, pp. 381-393, January 2013.

[11] D. Kafura, "A survey of software metrics " presented at the ACM annual conference on The range of computing New York 1985

[12] Ming Chang Lee and T. Chang, "Software Measurement and Software Metrics in Software Quality," International Journal of Software Engineering and Its Applications, vol. 7, 2013

[13] Poornima Gupta and P. Sahai, "A Review on Artificial intelligence Approach on Prediction of Software Defects," International Journal of Research and Development in Applied Science and Engineering vol. 9, February 2016.

[14] Kunal Chopra and M. Sachdeva, "EVALUATION OF SOFTWARE METRICS FOR SOFTWARE PROJECTS " INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY vol. 14, April 2015

[15] S. D. Conte , a. V. Y. Shen, and W. M. Zage, "A Software Metrics Survey " in Technical Reports P. U. C. Science, Ed., ed, 1987.

[16] C. J, "Strengths and Weaknesses of Software Metrics " Software Productivity Research LLC, 2006

[17] A. H. and and T. J, "Structured Testing: A Testing Methodology using the Cyclomatic Complexity Metric " National Institute of Standards and Technology Gaithersburg, September 1996

[18] H. B. KLASKY, "A Study of Software Metrics " The State University of New Jersey New Brunswick2003

[19] J. Cahill, J. M. Hogan, and R. Thomas, "Predicting Fault-Prone Software Modules with Rank Sum Classification," pp. 211-219, 2013.

[20] R. C. and and F.Kemerer, "A Metrics Suite for Object Oriented Design " IEEE Transactions on Software Enginneering vol. 20, 1994.

[21] Rüdiger L, Jonas L, and W. L, "Comparing Software Metrics Tools," ACM, July 2008.

[22] (Monday, August 01, 2016). Analyst4j Find Using Metrics Available: www.codeswat.com

[23] Syntegrico. Analyst4j Find Using Metric.

[24] (Friday October 30, 2015). CCCC Software metrics Available: http://www.sourceforge.net/projects/cccc/

[25] (Friday October 30, 2015). Chidamber and Kemerer Java Metrics Available: http://www.spinellis.gr/sw/ckjm/

[26] (Friday, October 30, 2015). Eclipse Metrics Plug-in 1.3.6. Available: http://sourceforge.net/projects/metrics/

[27] (Friday, October 30, 2015). Eclipse Metrics Plug-in 3.4 Available: : http://eclipse- metrics.sourceforge.net/

[28] Jarallah S, Raimi A, and Rufai and Sohel M, "OOMeter: A Software Quality Assurance Tool " Proceedings of the Ninth European Conference on Software Maintenance and Reengineering 2005

[29] (Friday, October 30, 2015). Semmle Available: http://semmle.com

[30] (Monday, August 01, 2016). SciTools Source Code Analysis and Metrics, Understand for Java. Available: www.scitools.com