iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 4 **Issue:** XI **Month of publication:** November 2016

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)

A Literature Review on Various Software Metrics

Mohamad Akram[1], Srujana Mandala[2], Manjula R[3]
School of Computer Science and Engineering, VIT University, Vellore

Abstract: ***A quantitative basis is provided by software metrics for prediction and planning of the development process of the software. Therefore software qualify is improved and can be controlled without difficulty. An overview of different software quality metrics that are found in the software engineering literature are discussed in this paper, moreover the paper focuses on the need of software metrics and its contribution towards the software quality. It is necessary to control the software complexity to improvise the product quality. In different phases of the software development life cycle the software metrics can be used. The strengths and weakness are of different software metrics described are discussed in the paper. Classification of metrics like software quality metrics and the object-oriented metrics are the highlights of the paper.***
Keywords: ***complexity metrics, object oriented metrics, size metrics, software metrics, software quality.***

I. INTRODUCTION

In the software development object oriented is more useful and the object oriented metrics used are vital for measuring the software quality .object-oriented design contains the quality of the software and all the properties related to any large scale or small scale project.[5] it's a degree where a particular attribute or characteristics is holded by a system object .Object oriented is basically a classifying approach that can classify a problem in terms of object and there are possibilities that it also provides many paybacks on reusability , reliability , decomposition and adaptability of the problem into effortless understood objects and also provides future modifications.[13] Project work product and the necessary software resources are measured and predicted by the software engineer with the help of software metrics. In additional to the above mentioned it provides a quantitative procedure for accessing the quality of the internal attributes of product , therefore enabling to access quality before the completion of the project .

For considering the design of good software metrics are very important source they are the tools of measurement. The word metrics is adequately used to mean specific measurement took for particular process .As per IEEE standards the term metrics is defined as "a quantitative measure of degree to which a component, system, or a given attributes" [12].

The features of the quality metrics are: [15]

A. Orthogonality

The ability to represent different ways of system through measurement. [6]

B. Minimality

The ability of using the minimum number of metrics.

C. Accuracy

The ability to measure the magnitude of errors through the quantitative measures. [11]

D. Reliability

Portability of failure of free software operations for certain durations

E. Formality

It is the ability for different peoples to get similar values for similar system.

F. Implement ability

Ability to perform the task independently. [7]

©IJRASET: All Rights are Reserved

Volume 4 Issue XI, November 2016
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

II. CLASSIFICATION OF THE SOFTWARE METRICS

Fig 1: classification of the software metrics. [7]

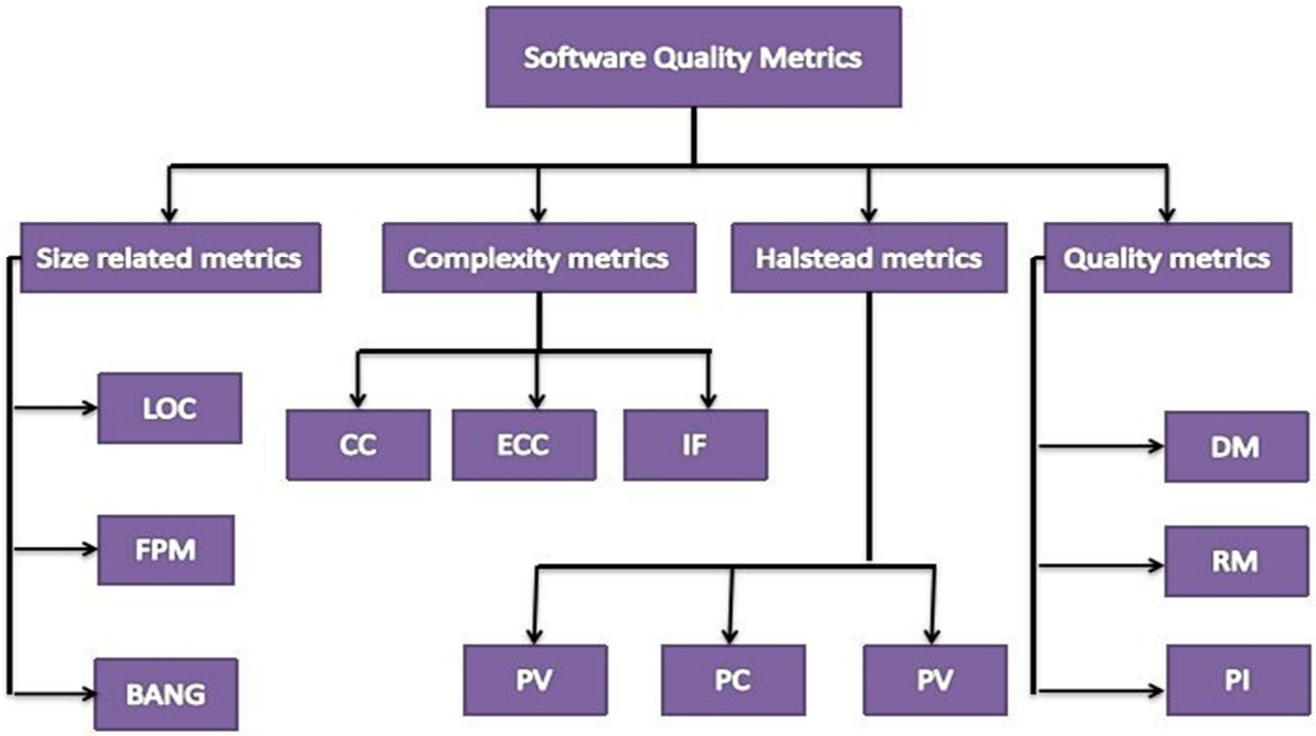


Fig 2: classification of the Quality metrics. [7]

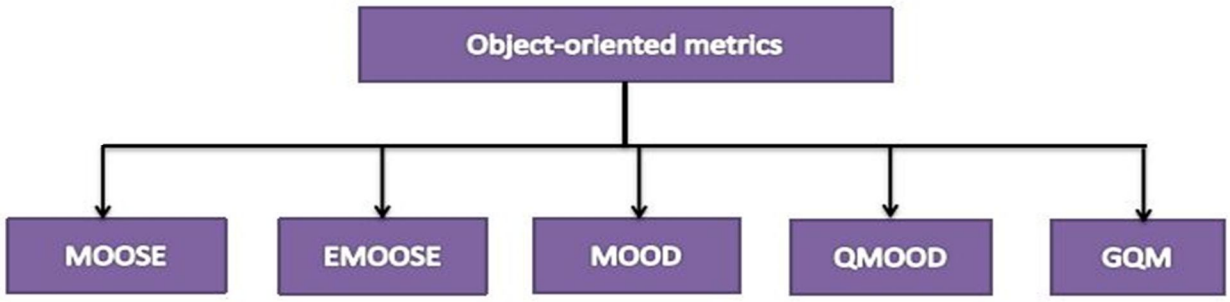


Fig 3: classification of the Object-oriented metrics. [7]

©IJRASET: All Rights are Reserved

III. LITERATURE SURVEY

A. *Software Quality Metrics*

Here below are some of the software quality metrics that are used in the software development process

1) *Size related metrics*: Size related metrics are the type of metrics which helps to compute the software size. To evaluate the size of the software there are three types of software metrics.

a) *Line of code (LOC):* LOC is one of the oldest type of metrics which is used to evaluate the size of the module but the main problem which was developed in LOC is "In evaluation what is to be included". [4]

b) *Function point metrics:* Function point metrics is the type of metrics which is used to evaluate the line of code only when the accessibility of code is present and so that it cannot be used in early stage. There is a method to resolve the evaluation of software size early in the enlargement of life cycle which was proposed by Albrecht. This mainly depends on inquiries, input of the user, output of the user and the values expected to measure the value in evaluating the size of the program and thus intention which is needed for the development. [4]

c) *Bang:* Bang is defined as a function metrics by Dc. Marco .This function metrics can be calculated by using data primitive's applicable form the set of formal specification for software and some algorithms. This gives the measure of total performance and transported to the user. [7]

2) *Complexity metrics:* In 1976, T.J.Mc. Cabe explained in detail about the design of metrics for finding the metrics complexity and these can be defined as

a) *Cyclomatic complexity (CC):* it is said to be module logical complexity which was proposed by T.J.Mc.Cabe in 1976.To calculate the desirability and maintainability of the software module is the basic goal of this CC metrics. In software system, this metrics can be used as a mark of reliability. Mc.Cabe define as, [9]

$$V(G) = e-n+2$$

Where,
V(G) = CC of flow graph G of process in which interested,
e= number of edges in G,
n= number of nodes in G.
Alternate way to calculate:

$$V(G)=P+1$$

Where,
P=no of predicate nodes which represents statements of Boolean in code.[4]

b) *Extended cyclic complexity (ECC):* Mc.Cabe evaluates the complexity of the program using ECC. But it fails to describe the differentiation in the complexity cases involving the conditional statement with single condition. Myers suggest ECC that may be explained as,

$$ECC=e\ V(G)=Pe+1$$

Where,
Pe=no of predicate nodes in flow graph G weighted by number of compound statements.[7]

c) *Information flow:* Information flow with in the structure of a program as a metrics for complexity of the program proposed by Kafura and Henry. Here the metrics can be found by the no of local information flows output (fan-out) and information flows input (fan-in).The process may be defined as,[7]

$$C=[\text{Procedure length}]*[(\text{fan-in})*(\text{fan-out})]2$$

3) *Halstead metrics:* It was proposed by Halstead in 1976, the software science theory. The main aim of this theory is to find totally the software production effort which consists of some length(N),volume(V) and vocabulary(n).It defines,[9]

a) *Program vocabulary (n):* Generally in programming languages, the program can be viewed as some set of tokens and these tokens which refers as operators and operands. Halstead described vocabulary (n) as,

$$n=n1+n2$$

Where,

©IJRASET: All Rights are Reserved

n1= the no of unique operators in the program.
n2= the no of unique operands in the program.

b) Program length (N): This length may be defined as the count of the overall number of operators and operands. It can be represented as,

$$N=N1+N2$$

Where,
N1= the no of operators in the program.
N2= the no of operands in the program.

c) Program volume (V): This program volume (V) can be evaluated as the storage volume required being there in the program.[9]

$$V=Nlog2n$$

4) Quality metrics: In this metrics, some type of quality metrics occurs. They are,

a) Defect metrics: There is no effectual procedure for the count of the total number of errors, the design change number, program with number of intended errors, the errors encountered by the code inspections and the program test number which may be considered as a substitute measures to the defects.
b) Reliability metrics: In reliability metrics, the quality of internal product is measured generally using the no of bugs present in the software or how long the software runs before the crash encountered.[7]
c) Maintainability index: prediction of software maintainability is explained by many functions which was proposed by Dr. Paul W. Omen. The index of the software maintainability is measured as follows,

$$MI=171-5.2*In(ave\ V)-0.23*ave\ V(g)-16.2*In(ave\ LOC)$$

Where,
Ave V=average halstead volume per module.
Ave V (g) =average extended cyclomatic complexity per module.
Ave LOC=average line of code per module.[14]

metrics	Advantages	Disadvantages
LOC	Dependent on language Easy computation Size oriented metrics	Counting standards are unavailable. Varies from person to person Useless for comparing hand written code and auto generated code
FP	Size oriented metrics Dependent on language Estimation of resources and cost required for software development.	Counting process is manual. Experience required.
CC	Computed early in life cycle Minimum effort can be measured. Provides complexity of various designs	Doesn't measure the data complexity but measures the program's control complexity. Size of the program is ignored. Nested conditional structures are harder to perceive.

TABLE 1: STRENGTHS AND WEAKNESS OF SIZE RELATED METRICS.[4]

©IJRASET: All Rights are Reserved

5) Object oriented metrics

a) Metrics for object-oriented software engineering (MOOSE):Some metrics which inaugurated expressive amount of interest and are presently the most prominent object-oriented suite for the evaluation of object-oriented software proposed by Chidamber and Kemerer(CK).This CK metrics suite contains 6 metrics which estimates variant characteristics of the object-oriented design. They are:

i) Weighted methods per class (WMC): WMC represents the evaluation of the complexity sum of various methods present in a class. Here usage of number of methods and complexity of the methods takes place which were maintained by the effort and predictor of time. Large number of methods occurs in a class which has great impact on the children of a class as the methods of the parent in the class will be inherited by child. By using cyclomatic complexity of the methods, can find the class complexity. If WMC is higher, it says that the considered class is more complex as compared to the low values.[9]

ii) Depth of inheritance tree (DIT): Here a tree exists with root node and leaf nodes. This DIT metric helps in finding the longest path from root to the end of the tree. In general, DIT indicates complexity, class behavior and the design complexity of potential reuse and class. Thus it says that it is very hard to understand the system with many inheritance layers. On other side it defines that larger DIT value can be reused. If there exists a deeper class hierarchy, where it is very difficult to predict the behavior of class and many methods occurs in class in which the deeper tree indicates the complexity level of design because all the facts consists of many methods and class are involved. Thus the tree with deeper hierarchy defines the reuse of inherited methods.[17]

iii) Number of children (NOC): Chidamber and Kemerer proposed the number of children (NOC) metric can be explained for the sub-class of the immediate one which is coordinated by the class which is of the form class hierarchy. From this points it describes, NOC is used to evaluate “How many of the immediate subclasses are going to inherit the parent class methods”. It says if the number of children is greater the improper abstraction of parent class likelihood. Even the number of children is greater the reuse of potential is greater, since inheritance is one of the types of reuse.[17]

iv) Coupling between objects (CBO): This type is used for counting the class number where certain class is specifically coupled. we can say if there is existence of any rich coupling then it results in decrease of characteristic like class modularity where makes it less attractive for the class reuse and the class of highly coupled will be more sensitive for the change in any other part of the design which becomes very difficult for the maintenance of the coupling of classes. CBO (coupling between object classes) metric is known as “it defines for the given classes the count of the total number of non-inheritance related type couples with given classes. It explains that the “class” unit which is used in this metric is so difficult to justify and defined the forms of coupling of classes: message passing, inheritance and abstract data type which generally will be available in concept like object-oriented programming.

v) Response for class (RFC): The response for class (RFC) defines that it is a response set which evaluates as a set of methods which will be executed in response and where the messages receive from the object of the class a message. The value which is large also complicates debugging of the object and even testing where it needs the particular tester to be have the functionality knowledge .If the RFC value is very large it will be taking complex class which is a worst case scenario, and the RFC value will be helped in assuming the time needed for the time which is needed for testing the class.[17]

vi) Lack of cohesion methods (LOCM): LOCM is the type of metric which is used in counting the pairs of the disjoint methods number minus the pairs of the similar methods number used. The disjoint method refers to no common instance variable whereas the similar method refers to at least occurrence of one common instance variable. By using the type of instance variable which is same defines measurement of the pair of methods. As we know cohesiveness which raises encapsulation in certain class is very desirable and due to low cohesion property may lead to the splitting of the class in to two or more sub-classes. The complexity increases due to low cohesion in the methods which leads finally to the decrease of proneness of errors during the development.[9]

6) Extended metrics for object-oriented software engineering (EMOOSE): EMOOSE metrics was proposed by W.LI et al. It may be defined as,

a) Message pass coupling (MPC): It is said to be the total number of messages that will be sent by the operation of the considered class.

b) Data abstraction coupling (DAC): It is said to be the count of the no of classes which will be an aggregate to the current class and also describes the data abstraction coupling (DAC).

c) Number of methods (NOM): It will be used in counting the no of operations that takes place are local to the given class (i.e. only the type of operations of the class which defines the methods number for its measure.)

©IJRASET: All Rights are Reserved

d) Size-1: Size-1 is used in finding the total number of line of code.
e) Size-2: Size-2 is used in counting the local attributes number and the operations number which is described in the class.[7]

7) Metrics for object-oriented design (MOOD): metrics for object oriented design metrics was defined by F.B Abreu et al. Metrics for Object oriented design refers a structural model of the objected oriented paradigm like inheritance, message passing, and polymorphism and encapsulation .Each of the object oriented design metrics were expressed to measure. The actual use any feature of a particular design is defined by the numerator. Attributes and methods are the two main features of the MOOD metrics. The status of an object in the system is represented by attributes whereas methods are used to modify or maintain various kinds of status of the objects.[9] Metrics of object-oriented design are defined as follows:

a) Attribute hiding factor (AHF): Attribute hiding factor is described as the ratio of the sum of the attribute invisibilities that described in all classes to the number of attributed described totally in the system under the consideration.[7]
b) Method inheritance factor (MHF): Method inheritance factor describes the ratio of the total number of all inherited methods of the considered classes of the system which is under the consideration to the number of the available methods for all type of classes.[2]
c) Attribute inheritance factor (AIF): Attribute inheritance factor is described as the ratio of the total sum of attributes inherited in all type classes of the system which is under consideration to the number of attributes available.[9]
d) Polymorphism factor (PF): This factor describes the ratio of the definite number of available different polymorphic situation. Here we may say AIF and MIF are used to evaluate the class inheritance and to give the property of similarity in to the classes. And CF is used to evaluate the coupling characteristic in the classes. We can say coupling will be of two types:

i) Static
ii) Dynamic coupling.

This results in widening the class complexity and will reduce the reuse of potential and encapsulation that best maintainability. The software developers of the object-oriented design will always try to cut the high factor of coupling. In a class, the polymorphism factor is used to define the polymorphism in a class of particular one and also emerge from inheritance.[10]

8) Goal Question metrics (GQM): GQM design was proposed by V.L.Basili. This design was generally defined for describing the defects for a firm of projects like in the environment of NASA Goddard space flight center. He also contributed the sequence set which will be helpful for the designer's team. The main aim of goal question metrics is too explicit the templates meaning which will cover the environment, purpose and the perspective. Some firm of guidelines also given for driving nature of the question and the metrics. It generally presented a framework which includes three steps:

a) List the project maintenance and the dominant goals of the development.
b) Derive the questions from each goal that can be answered to define whether the particular goals are met or not.
c) Decide which one must be evaluated in order to answer all the questions abundantly.[7]
d) Goal (Conceptual level): For a particular object a goal is defined, for a variety of reasons with respect to the various number of quality models from different points of objective related to certain environment.[16]
e) Question (Operational level): Here a firm of questions will be used to represent the achievement of definite goal is going to be performed based on some represented model type.
f) Metric (Quantitative level): Here firm of data occurs with all questions in order to answer them in a perceptible way. Here the data takes place can be of subjective and objectives, if they all are object dependent that may be evaluated and not on the viewport from where they all have been taken. Consider an example, document versions, on certain task the staff hours dissipated, and the program size.[16]
The goal question metrics describes some goals, process those all goals in to certain type of questions, and the questions finally restrained in to the metrics type. Let G1 and G2 be two types of the goals, and for these two types of goals commonly there occurs Q2, and there will be occurrence of M2 metric which may be answered by all the questions. The main aim of this process is that every metric which is analyzed will be placed in the context, so that M1 metric will be possessed in order to answer any type of the

©IJRASET: All Rights are Reserved

question Q1 which helps in leading to achieve the G1 goal.[16]

9) Quality model for object-oriented design (QMOOD :It is one of the type of quality model which is comprehensive and inaugurates a precisely defined and imperiously a validated model to determine the quality attributes of object-oriented design such as reusable method and understandable method, and itemize it through some mathematical formulas with some structural Object-oriented design characteristics like coupling technique and encapsulation method.[15] The quality model contains of six equations which originate a relationship between the six quality attributes of the object-oriented design and the properties are: Functionality, effectiveness, extendibility, reusability, understand ability, flexibility and eleven design type characteristics.

The QMOOD total evaluation may be defined from the Bansyia's thesis. QMOOD types metrics are explained in two ways are,

a) System measure: The system measure defines the type of metrics like DSC, NOH, NMI, NSI, NIC, NLC, NNC, NAC, AWI, ADI, ANA[7].

b) Class measure: The class measure metrics defines some type of metrics like MAA, MFA, MAT, MFM, MOS, MOA, DAM, MRM, MAM, DOI, NOC, NOA, CIS, NPT, NOO, NOP, NOA, NPM, NPT, NAD, NRA, CSM, MCC, DAC, MAC, CCN, CCP and CCM.[7]

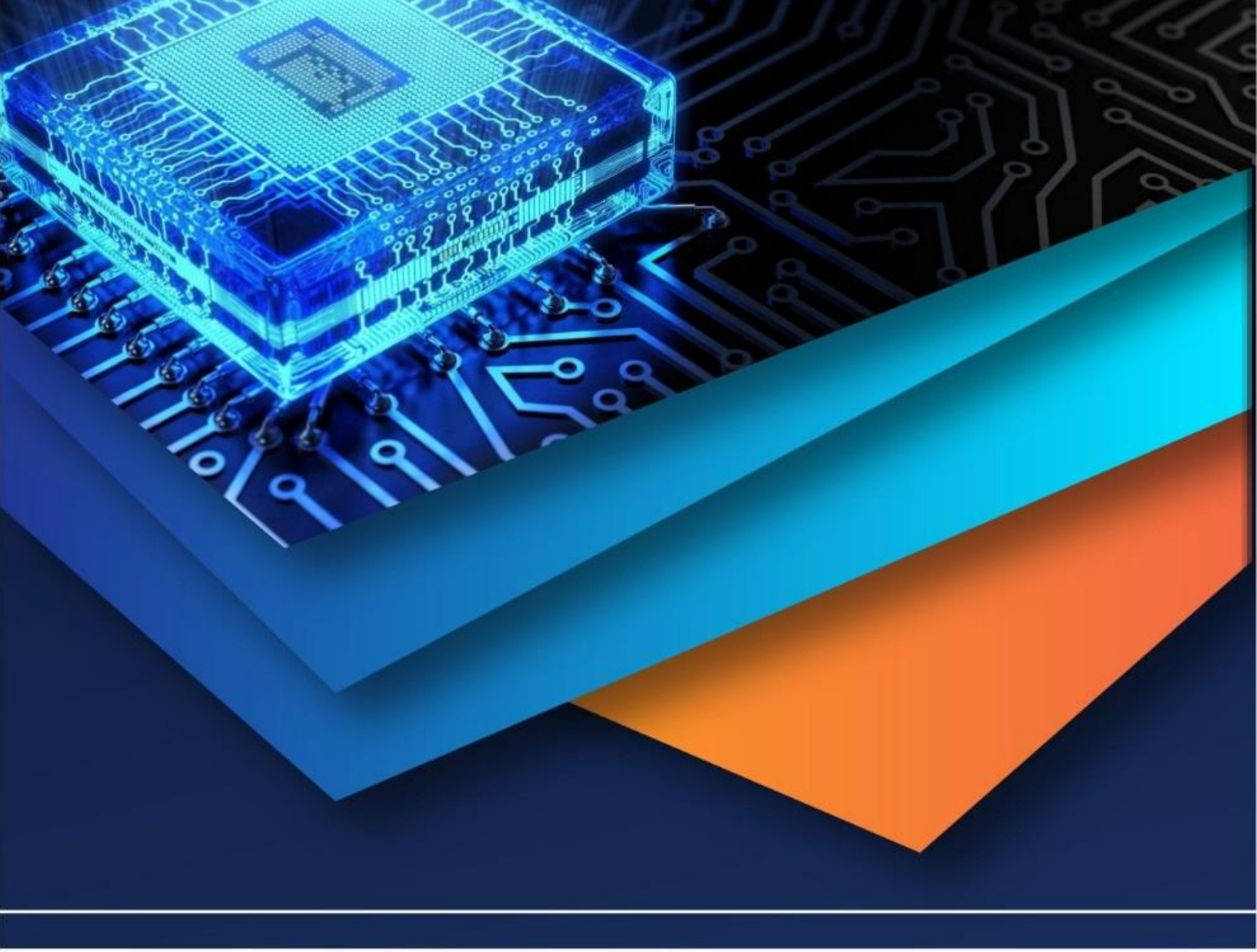
IV. CONCLUSION

With the rapid development in software industry, the measurement of the software product becomes more complex thus increasing the need of better software metrics has gone up over the time. There are various numbers of software metrics available, all these metrics mainly focus on the development, management and maintenance of the software, therefore in order to correct and detect various bugs these metrics must be used in the early stages of the software development life cycles, thus preventing the lateral errors .each one of these metrics has important features and their working. This paper describes the various software metrics available like sixe metrics, complexity metrics, halstead metrics, quality metrics and object-oriented metrics. The strengths and weakness of the size metrics are discussed in this paper .study and survey has to be done for selecting the better metrics. This paper helps the researcher for better selection of the software metric based on the product. These metrics increase the reusability of the software and reducing the software maintenance cost.

REFERENCES

[1] R. S. Pressman, "Software Engineering, A Practitioner's Approach", Sixth Edition, Mc Graw. Hill, 2005. .
[2] Deepak Arora, Pooja Khanna and Alpika Tripathi, Shipra Sharma and Sanchika Shukla, "Software Quality Estimation through Object Oriented Design Metrics"IJCSNS International Journal of Computer Science and Network Security, VOL.11 No.4, April 2011.
[3] J. Bansiya, C. G. Davis, "A Hierarchical Model for Object-Oriented Design Quality Assessment", IEEE Transactions on Software Engineering, 28, (1), (2002), 4–17.
[4] Dr. Sanjeev Dhawan, Kiran,"Software Metrics – A Tool for Measuring Complexity", International Journal of Software and Web Sciences (IJSWS), ISSN (online): 2279-0071.
[5] C. Neelamegam, M. Punithavali, "A survey on object oriented quality metrics", Global journal of computer science and technologies, pp. 183-186, 2011.
[6] V.L &C.C., Manager, SATC "Principal Components of orthogonal Object-oriented metrics (323-08-14)", white paper analyzing the result of NASA object-oriented data, May 29, 2003.
[7] Amit Sharma, Sanjay Kumar Dubey, "Comparison of Software Quality Metrics for Object-Oriented System", IJCSMS International Journal of Computer Science & Management Studies, Special Issue of Vol. 12, June 2012.
[8] Mrinal Singh Rawat , Arpita Mittal and sanjay kumar Dubey "Survey on Impact of Software Metrics on Software Quality" .(IJACSA) International Journal of Advanced Computer Science and Applications,Vol. 3, No. 1, 2012.
[9] Dr Sonal Chawla and Gagandeep Kaur. "Comparative Study of the Software Metrics for the complexity and Maintainability of Software Development" ,(IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 4, No. 9, 2013.
[10] F.Brito, E. Abreu and W. Melo, 1996, "Evaluating the Impact of Object-Oriented Design on Software Quality", 3rd Int'l S/W Metrics Symposium, March 1996, Berlin,Germany.
[11] Fairy Richard, "Software engineering Concepts", Tata Mc Graw Hill, 2003.
[12] R.S.Pressman, "Software Engineering-A practitioner's Approach" Eight Edition, Mc. Graw Hill International Edition 2014
[13] B. Henderson, seller, "object oriented metrics: measure of complexity", Prentice Hall, 1996.
[14] http://www.projectcodemeter.com/cost_estimation/help/GL_maintainability.htm
[15] J. Bansiya, C. G. Davis, "A Hierarchical Model for Object-Oriented Design Quality Assessment", IEEE Transactions on Software Engineering, 4-17,2002
[16] Victor R. Basili-Gianluigi Caldiera and H. Dieter Rombach ,"Goal question metric paradigm" , Encyclopedia of Software Engineering-2, ISBN #1-5400-8
[17] http://www.aivosto.com/project/help/pm-oo-ck.html

©IJRASET: All Rights are Reserved



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)