IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-5, May 2018 E-ISSN: 2347-2693

Metric Based Automatic Quality Analysis of Object Oriented Systems

A. Jatain

Department of Computer Science, Amity University, Manesar, Gurugram, India

*Corresponding Author: amanjatainsingh@gmail.com, Tel.: +91-9711050859

Available online at: www.ijcseonline.org

Accepted: 11/May/2018, Published: 31/May/2018

***Abstract*—** Today most of the languages support objects oriented architecture to develop an organized software system and customer base for these system is increasing because incentives in object oriented paradigm is huge. At the same time, drawback is that object oriented systems are complex and difficult to understand. So quality of these system must be monitored and maintained which require lots of money, time and efforts. In software community when prediction is made based on metrics, software quality is one of the most discussed term. Software quality can be estimated by analyzing the key quality attributes of the software system. The objective of this paper is to define a metric suite and designing of automatic tool to evaluate the object oriented software system under complexity analysis in order to analyze the software criticality. **Methodology:** This paper focusses on automatic quality analysis of object oriented system. For this different metrics and metric suites are analyzed and a tool is developed in .net environment that accept software program as input and perform a metric based analysis. To validate the tool a case study (LMS) is considered as input.

***Keywords*—** Component, ,Learning Management, Metrics, Object Oriented, Quality, Software.

1. INTRODUCTION

Software engineering is one of the most important division of computer science that itself controls the software development process so that a quality product can be produced. Software development and software quality analysis are the parallel process defined to improve software effectiveness. In the last decades quality in software products has been deliberately emphasized. But quality can be controlled by making it quantifiable, i.e. by measuring it [4]. To perform software quality analysis measurement plays a vital role. Measurement in software industry is described in terms of metrics. Many metrics related to software quality have been developed and used in literature. Software metrics are best described as the measurement components, units, processes or the activities to analyse the software system under different aspect. Metrics are the imperative factor to measure software quality. Software quality is the extent to which a deliverable product conforms to its specification. In software engineering evaluating the quality of object oriented software always has always been a fundamental task [3].

Object oriented system development is buzzword in system development since last two decades and there is substantial information that object oriented metrics plays an important role in quality management decisions [1]. As object oriented paradigm gaining popularity, it would be better to analyse object oriented metrics with respect to software quality. This migration to object oriented paradigm has originated the demand of finding effective means of measurement to quantify its encapsulation and polymorphism mechanism and this has raised need of definition of a huge numbers of object oriented metrics [5] [6]. Designing software metrics as an aid to assess the quality of object oriented software system has had deep and useful impact on the overall system. Various metrics are defined in literature to realize its importance in evaluating the software quality of object oriented system [2]. They assess software quality on the basis of different software attributes. However, it is difficult decision for a software engineer to select those metrics which are more useful. No one provided a suit which is capable of collecting essential metrics to perform automated quality analysis of software system. In this paper a tool is designed by collecting some major metrics to assess quality of object orientation in the software system. It is widely recognized that epidemic use of object- oriented paradigm can only arise when there are tool systems that impart development support along with visualizing the code [7]. This paper is organized as follows: Section 2 discussed the related work. Section 3 presents the list of selected metric and proposed metrics for quality analysis, section 4 discusses the proposed system and last section provides the conclusion and insight for future work.

2. RELATED WORK

A lot of work is performed by different researchers in the area of software quality estimation based on software metrics. There are number of reliability models used by the

researchers to estimate the software quality. The software quality and reliability are estimated to analyse the software system effectively. The software estimation based analysis is performed by the researchers using metrics. These metrics give the analysis in quantitative form so that the easy decision can be taken for software quality. The quality metrics are defined under the component and process based estimation to schedule the software system for future projects. Various metrics have been proposed in literature [8]. The six most widely used design measures are proposed by Chidamber et al. [9] emphasising on class hierarchy. After CK suite various class level new design metrics were introduced by many. To maintain the object oriented system Henary et al. proposed a set of metrics in [11]. A systematic review of existing object oriented measures is provided in [12]. Hong et al. in [13] proposed a model based object oriented software metrics tool, JBOOMT with an objective to guide software engineer to measure object oriented products. Kahn et al. in [14] described Weighted Class Complexity (WCC) an integrated class based metric. This metric was proposed with an intention to measure the quality factors like efficiency, complexity, understandability, maintainability and reusability.

Christopher et al. designed a tool that automatically gather set of object oriented product metrics for software development process and this tool provides capability to estimate future development effort using past experience [15]. Mohab et. al measured the software quality of object oriented system by assessing reusability of system using proposed metric suite [16]. The metric suite includes three metrics: inheritance metric, interaction metric and structural metric. Paramveer et. al [3] proposed DynaMetrics: a runtime tool to analyse the object oriented systems. This tool is capable of estimating the metric values for C++ and Java programmes. Huang et al. [17] described some concepts and techniques to enhance the quality of object oriented systems and compared the traditional metrics with specific metrics of object oriented system with the intention to find the suitability of these metric while assessing the quality of system. Zhiyi Ma [19] analysed the quality of object oriented models by measuring the level of defects in different phase of project and also suggested the solutions for the defects.

We analysed various proposed metrics to evaluate the quality of object oriented systems. Different metrics are defined according to diverse requirement and usage. Almost each of the metric targets a particular dimension of a project or some of the metric captures the same dimension again and again. There is no concrete set of metrics, which is capable to collect domain specific useful metrics to perform an automated OO software measurement.

3. SELECTED METRICS TO ANALYSE THE QUALITY OF SOFTWARE SYSTEM UNDER OBJECT ORIENTATION

Today most of the languages support object oriented architecture to develop an organized software system where it is divided into interconnected modules and interconnectivity between these modules is defined in terms of methods and variable calls which can be API or non API function call. Under the rules of the object orientation the software system is defined in different ways. But while analysing a software system, the important factor that needs to be measured is interconnectivity between the software modules and how the structure of these software modules is clearly evaluated using object oriented metrics. To design the automatic quality evaluation system for OOS, we have considered size, complexity, coupling, cohesion and interference based metrics due to their major significance. Desirable value for these metrics is categorized either as low or high as shown in table 3.1.

Table 3.1: Metrics Analysis

Type of Metric	**Desirable Value**
Size Metric	Low
Complexity Metrics	Low
Information Hiding Metrics	High
Coupling Metrics	Low
Cohesion Metrics	High
Inheritance Metrics	High(Trade off)

Line of code (LOC) and Function points (FP) are considered as the commonly used size metrics for software systems. LOC can be calculated by counting lines of code that may or may not include blank line and comments. It is suitable for both procedural and object oriented system. Function point metrics represents the software size analysis under class level estimation which is parametric and performed by evaluating five main component class: User input type, output type, inquiry type and external file specification to the system. The most suitable and widely used metric suite for OO software system is CK metrics suite defined by Chidmber and Kemerer [21]. CK proposed following classic set of six metrics:

I. Weighted Methods Per Class (WMC)

$$WMC = \sum_{i=1}^{n} c_i \quad (1)$$

This metric analyse the complexity of methods and the weighted sum of these complexities is the WMC.

II. Depth of Inheritance Tree (DIT)

DIT metric for the class is the depth of inheritance of the class and in case of multiple inheritance, the values of DIT will be the maximum length from the node to the root of the tree. It is based on flow path analysis and it represents the particular node distance from the root.

III. Number of Children (NOC)

It measures the number of subclasses that are going to inherit the methods of the parent class and the number of child classes to a main class represents the number of children.

IV. Coupling between object classes (CBO)

If the members or methods of a class are accessed in other class, it is called as coupling and for a class values of CBO is the count of the number of other classes to which it is coupled.

V. Response for a class (RFC)

$$RFC = |RS| \quad (2)$$

RS is the response set for the class and

$$RS = \{M\} \cup_{all\ i} \{R_i\} \quad (3)$$

Here $\{R_i\}$ = set of methods called by *i* and $\{M\}$ = set of all methods in the class

RFC is the set of methods that are defined in response to a message received by an object of that class.

VI. Lack of Cohesion in Methods (LCOM)

It measures the cohesiveness of a class. To promote encapsulation, cohesiveness of methods is desirable.

3.1 Proposed Extended Metrics

Considering CK suite and taking LOC and FP as base metrics, following four metrics are defined to assess the quality of OO software systems:

I. API Function Usage Index

The library function used in a software program is represented by an API. This metrics is used to represent the number of API functions which are being accessed in a software system. These API functions are the embedded software modules that boost up the development process.

II. Module Interaction Analysis

This metrics represents the interaction between different software modules. It is the sum of all the components procedure accessed by a particular module. The interaction is defined in terms of method and variable access of a software module in other software module.

III. External Call Analysis

It performs the interaction analysis a software component with other components present within the system. This interaction analysis can be in the form of object analysis or in the form of class level analysis. The library function call is not included in this analysis. Lower dependence on library functions ensures better reusability.

VI. Interface and System Analysis

These metrics correspond to the interaction of resources with the modules and the system. Lower value of these metrics determine the lower interaction of existing resources with the system modules thus reducing the complexity. Less complexity will increase the reusability of the system.

4. Proposed System to evaluate OOSS

Today most of the languages support objects oriented architecture to develop an organized software system. Almost all the software systems use the same paradigm. In OOS interconnectivity between modules is defined in terms of methods and variable calls which also includes API and API function calls. Proposed approach analyses the OOS under some important metrics defined by research communities and three metrics defined in section 3.1. The research design of the proposed system is defined under four steps:

I. Analysing object oriented metric suits and other basic object oriented metrics
II. Selection of the significant object oriented metrics to evaluate quality of OOS.
III. Defining four object oriented metrics and feeded into system to analyse the OOP.
IV. Designing of user friendly tool that accept the OOP as input, calculate the defined metrics and automate the process of evaluation.

We have designed a software tool that will accept the OO software program as input and perform a metrics based analysis to evaluate its quality. Initially, the analysis involves the basic metrics calculations viz. to calculate the number of modules, the number of methods in a module etc. Once this information is collected, the major defined metrics set is evaluated. The presented system is divided into two stages: 1) Basic Analysis and 2) Object Oriented Analysis. The basic analysis consists of the evaluation of basic size and functional metrics in the system. These metrics are base to the object oriented metrics. The system accepts the OO source code or the software as input and perform the

component search over it. The components are defined in terms of methods, class and interface. On the basis of these detailed information the metrics are estimated. The basic flow of the systems is depicted in figure 4.1.

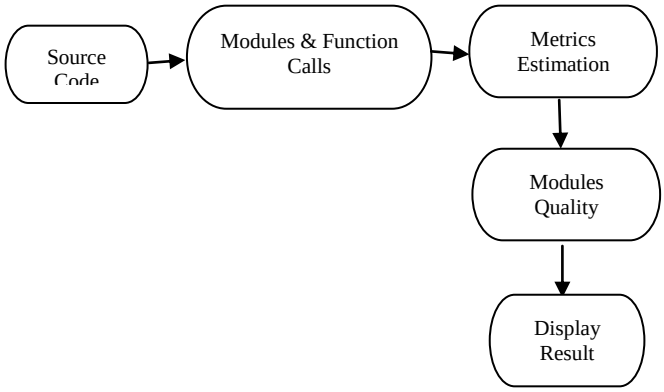

Figure 4.1 Basic Process Flow of the System

As OOP is given as input in the proposed system, it counts the number of classes, interaction between classes, interfaces and number of methods in each module. After collecting these parameters, the software metrics are evaluated. This evaluation consists of the major object oriented metrics viz. cohesion, coupling and proposed metrics viz. API interaction and the module interaction metrics. All discussed metrics are calculated using proposed tool automatically and then software quality analysis is performed. This automation saves the significant efforts and ease the process of analysing the quality of a particular system developed using OPP paradigm. The snapshot of the tool is depicted in figure 4.2.

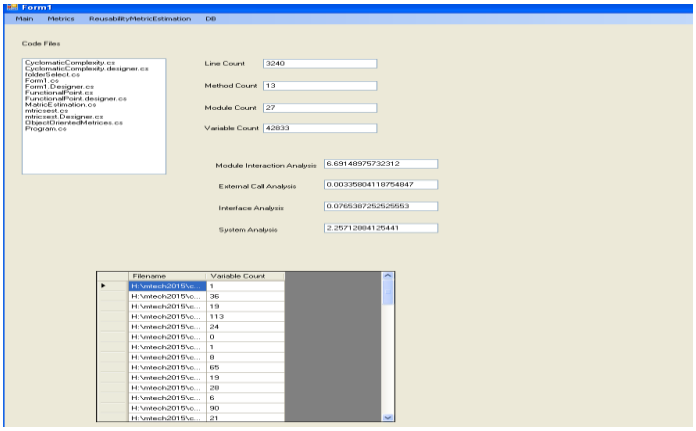

Figure 4.2 Graphical interface of the proposed tool

Figure 4.2 shows the various metrics results, when an OOS (Learning Management System (LMS) is an OO software system developed by final year students for the administration, tracking and delivery for educational courses. Its size is 3546 LOC) is loaded in the proposed tool. Characteristics of the LMS is shown in table 4.1. All the project files are fetched, loaded in the list box. After loading the content of the folder filtration is applied at entry level and only the code files are retained and unnecessary files are removed. Tool shows the result of Method Count Metrics, Module Count Metrics, Variable Count Metrics, Variable Count Metrics, ModuleInt Metric, ExternalCall Metric, and Integrated Component Metric.

Table 4.1 Statistics for LMS

Properties	Values
Number of Files	18
Code Files	12
Use of System Library	Yes
Module Interaction	Yes
Component based Use	Yes

LMS is having five different modules, where each module is having various variables and associated methods. The interaction between these modules is defined based on the variables and the function call and is in both direction. Table 4.2 is showing the number of variables and methods in each module. The interaction between the modules is also defined based on the class level or object level access and each module can use some library functions from other environment. Table 4.3 shows the variable and method access with each module.

Table 4.2 Module Statistics

Module	Method	Variable
Student Catalogue	4	6
Course Management and Certification	7	10
Administration	3	2
Resource	2	5
Setting	6	4

Table 4.3 Module Level Variable/Method Accessibility Metrix LMS to measure system metrics

Modules	Student Catalogue	Course Management and Certification	Administration	Resource	Setting	API Call
Student Catalogue	0/0	4/3	2/3	3/5	2/2	3
Course Management and Certification	3/4	0/0	3/2	4/4	5/2	2
Administration	2/1	2/2	0/0	3/2	1/1	3
Resource	1/3	2/2	2/2	0/0	2/3	4
Setting	4/2	3/2	3/2	3/3	0/0	2

4.1 Metrics Based Analysis

We analysed the LMS system by estimating metric for each of the module under complexity analysis so that the software criticality will be analysed. Higher the software complexity or criticality more fault prone the system will be.

1. Module Interaction Analysis

Table 4.4 represents the interaction between different software modules within system and outside the system (may be library functions). The total value of metric shows the components procedure accessed by a particular module.

Table: 4.4 Module-wise interaction Analysis

Module	Called	Call	Metric
Student Catalogue	0	20	0
Course Management and Certification	7	20	0.35
Administration	5	19	0.263158
Resource	8	27	0.296296
Setting	4	18	0.222222
Total			1.131676

2. External Call Analysis

Table 4.5 shows the interaction analysis of a software component with other components present within the system. Lower dependence on outside functions ensures better reusability.

3. Interface and System Analysis for Integrated Component

Table 4.6 depicts the interaction of resources with the modules and the system. Less value of the metric determine the lower interaction between the modules.

Table 4.5 Estimation of External Call Analysis

Module	External	Call Other	Metric
Student Catalogue	3	20	0.15
Course Management and Certification	2	20	0.1
Administration	3	19	0.157895
Resource	4	27	0.148148
Setting	2	18	0.111111
Total			0.667154

Table 4.6 Interface and System Analysis Metric

Module			
	7	24	0.291667
Student Catalogue	6	28	0.214286
Course Management and Certification	6	28	0.214286
Administration	7	14	0.5
Resource	8	17	0.470588
Setting	6	22	0.272727
Total			1.749268

5. Conclusion and Future Work

To analyse the quality of a software system, foremost requirement is to quantify the various attributes of the system and software metrics provide a mathematical approach to measure these quality attributes. In this paper, we have studied some relevant metric and also proposes a set of four metrics. To automate the process of estimating these metrics, a tool is developed in Asp .Net that will accept the project code from the user and calculate the metrics automatically. The tool accepts the path of software code as input and fetches all the software components from different modules. The work analyse the separate module of the software system as well as the whole system and also generate system metric based on all software modules analysis. Proposed work analyses the software system for cohesion, coupling and interface matrices. After fetching the components the next task is to estimate the basic metrics over the system such as LOC, function count, module count etc. The automatic evaluation of the major metrics ease the process to evaluate the quality of an OOS and criticality of the system can be observed. Once we have available such kind of data easily about a particular system, it plays an important role to judge the quality of the system. In future, we can extend the tool to accept the software system developed in other paradigm viz. CBS or aspect oriented software system and judge their quality.

References:

[1] H. Linda and Lawrence E. Hyatt, "Software Quality Metrics for Object-Oriented Environments", Software assurance technology centre, Greenbelt, USA, pp. 1-7, 1996.

[2] S. R. Chidamber and C.F. Kemerer, " Authors Reply", IEEE Transaction on Software Engineering, Vol. 21, No. 3, pp. 265, 1995.

[3] Paramvir Singh and Hardeep singh, "DynaMetrics: A runtime metric based analysis tool for object oriented software system", SIGSOFT software engineering Notes, Vol. 33, No. 6, pp. 1-6, 2008.

[4] T. DeMarco., "Controlling Software Projects, Management, Measurement and Estimation", 1st Edition, Prentice Hall, 1982.

[5] M. Lorenz and J. Kidd. Object –Oriented Software Metrics, 1st Edition, Prentice-Hall, Englewood Cliffs, NY, 1994.

[6] I. S. Organization, "ISO 9126-Quality Characteristics and Guidelines for their use", Brussels, 1991.

[7] I. Morschel and Ch. Ebert, "Metrics for quality analysis and improvement of object oriented software", Information and Software Technology, Elsevier, Vol. 39, No. 7, pp: 291-302, 1996.

[8] M. Xenos and D. Stavrinoudis and K. Zikouli and D. Christodoulakis, "Object oriented Metrics- A survey", Proceedings of the FESMA 2000, Federation of European Software Measurement Associations, pp. 1-10, Spain, 2000.

[9] S. R. Chidamber and C. F. Kemerer, "Towards a metrics Suite for object oriented design", Proceedings of International Conference on Object Oriented Programming: System, Language and Applications, 197-211, 1991.

[10] Chidmber S. and Chris Kemerer F., "A metric suite for object oriented Design, IEEE transaction on software engineering, Vol. -20, no.6, pp. 476-493, 1994.

[11] Wei Li and Sallie Henry, "Object Oriented Metrics that Predict Maintainability, Journal of Systems and Software, Elsevier, Vol. 23, pp. 111-122, 1993.

[12] Abreu, F. B., Esteves, R. and Goulao, M., The Design of Eiffel Programs: Quantitative evaluation using the MOOD metrics, Proceedings of TOOLS'96, USA, pp. 1-18, 1996.

[13] Hong, Hei,Tao Xie and Fuquing Yang, "A model based approach to object oriented software metrics", Journal of Computer Science and Technology, Vol. 17, No. 6, pp. 757-769, 2002.

[14] R. A. Khan, K. Mustfa and S. I. Ahson, "An empirical validation of object oriented design quality metrics", Journal of Kind Saud Uiversity, Elsevier, Vol. 19, pp: 1-16, 2007.

[15] Christopher L. Brooks and Christopher G. Buell, "A tool for automatically gathering object oriented metric", Proceedings of Aerospace and Electronics Conference, IEEE, pp: 835-838,1994.

[16] Brij mohan Goel and Pardeep Kumar Bhatia, "Analysis of reusability of object oriented system using object oriented metric", ACM Sigsoft Software Engineering Notes, Vol.38, No. 4, pp. 1-5, 2013.

[17] Rui Huang, Mingyu Li and Zhang Li, "International Journal of Information and Education Technology, Vol. 3, No. 4, pp. 433-436, 2013.

[18] A. Jatain and Y. Mehta, "Metrics and models for software reliability, "Proceeding sof International Conference on Issues and Challenge in Intelligent Computing Techniques, pp. 210-214, 2014.

[19] Zhiyi Ma, "An approach to improve the quality of object oriented models from novice modelers through project practice" Frontier of Computer Science, Vol. 11, No. 3, pp. 485-198, 2017.

Authors Profile

Dr. Aman Jatain pursed B.Tech from MDU, University, M.tech from Thapar university and Ph.d from NCU (Formely ITM) University in computer Science. She is currently working as Assistant Professor in Department of Computer Sciences at Amity University, Gurgaon. She is a member of CSE & IETE. She has published more than 40 research papers in reputed international journals including Thomson Reuters (SCI & Web of Science) and conferences including IEEE and it's also available online. Her main research work focuses on Software Engineering, Data Mining, IoT and Networking. She has 9 years of teaching experience and research experience.

