ISSN:0975-9646

Object Oriented Metrics: Revisited

Aman Jatain
Assistant Professor, Amity University
Gurgaon, India.

Jyoti Kataria
Assistant Professor, Amity University
Gurgaon, India.

Abstract - Software Metrics are developed and used by various Organizations for estimating and assuring software code quality, size estimation, complexity, maintenance and operation while generating software application development and software design. In the field of software's, accuracy is a major concern and every individual would like to get as much accuracy as they could. Accomplishment of system is based on the result of every distinct stage of development with proper evaluating techniques. Metrics are measures of product, process and people who are involved in the area of development process and acts as quality pointers. In this paper comparative study of object oriented software metrics is provided with the help of components which are an essential as well as the backbone to the functionalities of the software metrics.

Keywords – Class, Component, Depth of inheritance tree, Metrics, Object Oriented system.

I. INTRODUCTION

The quality of software systems became a significant issue in the continuation of software business i.e. the huge amount of software used in the markets and their role in handling precise and risky tasks. Engineers have been improving the software quality with their design process and proposing new methodologies for all software development steps, from requirements specification to testing. The size and complexity of software continue to grow throughout the development life cycle of a software system which leads to an increase in the effort and cost of software maintenance. Even after the development of software system, the software system needs to evolve continually to satisfy the user requirements by addition of new features, as per the business needs, improving the quality of the software systems, etc. Overall success of the software begins with properly understanding the requirements, planning and scheduling of the project, development of the process, expertise people's contribution, SQA activities with precise set of metrics, documentation and toolset.

Metrics are measures of product under development, activities, people involved, gives a vision on their quality to make whole process successful. Numerous metrics are defined and reformed in software industries to address problems with different levels of complexity as well as for measuring various properties of the software systems. Also software metrics used provides information about the resources, processes and products evolve during the software development. Software metrics provide factual and quantitative information. From the practical point of view, within an organization the metrics tools defined by engineers and the managers are system based. A large amount of research has been done over the past decades on the concept that how to measure the various aspects of software use and development, starting from the production of coding by programmers to the satisfaction of the end customers with using the software systems to their business.

Rest of the paper is structured as follows. Section 2 discusses about relevant research work in the subject. Section 3 gives a brief explanation of metrics categorization along with a table containing comparative study about metrics. Concluding remarks are given in Section 4.

II. RELATED WORK

Abreu *et al.* provides a classification structure for the TAPROOT. This structure was defined with two different vectors, which are granularity and category. Six different categories of Object-Oriented metrics are defined, they are complexity metrics, design metrics, quality metrics, size metrics, reuse metrics and productivity metrics and also proposed three different levels of granularity that are methods, software and class. M. Alshayeb *et al.* has given two iterative techniques for the pragmatic study of object oriented metrics. It includes one short-cycled agile method and other long-cycled structure evolution process. In the short-cycled agile process, the outcomes observed were that the line of code and the design efforts were added, deleted and changed with the prediction of object oriented metrics whereas the same points were not predicted by the long-cycled structure process.

R.D.Neal *et al.* gives the study for the validation of object-oriented software metrics and then found that a few of the proposed metrics could not be considered as the valid measure. R.Harrison et al. suggested a statistical model which is obtained from logistic regression from identifying threshold values for Chidamber and Kemerer metrics. H.Lieu et al. has given a perspective that the quality of software also plays a significant role in terms of financial and safety aspects. They also bridged the gap between design and

quality measurement of the metrics. M.Subramanyan et al. proposed some metric suites and recommended that for the developers it is very important to recognize various design aspects of the software and also different methods to enhance the quality of the software.

Racheal Harrison et al. discussed about the six properties for the object oriented design metrics and also measured the object-oriented features like polymorphism, inheritance, coupling and encapsulation. C.Shyam et al. suggests some software metrics through which we can calculate the quality of modularization of the object oriented software. Y.Zhou et al. considered the fault severity using the machine learning methods and using the experimental assessment of fault proneness which predict the capability of the object oriented design metrics.

J.Xu et al. have proposed an object-oriented metrics which describes the fault estimation using empirical analysis and also uses the CK metrics to apprise the number of faults in a particular program.

C.Neelamegan et al. surveyed four object oriented metrics and mostly focused on the measurements that are totally applied on the various design and class characteristics. Dr.B.R.Sastry et al. trying to implement the graphics user interaction with the aid of software metrics and also tried to the quality and the quantity of the of object oriented software development cycle.

III. METRICS CATEGORIZATION

Metrics can be categorized into three different components which are kinds, size and measures. Also two different kinds of software metrics are process metrics and product metrics. Process metrics quantify the process which is used to develop the software and then to evaluate the efficiency of fault detection. Product metrics quantify the characteristics and features of the product being developed to determine the reliability and size. Measures are also divided into two different types which are direct and indirect measures. Direct measure are used to measure the line of code, effort, cost, memory, speed etc. while the indirect measures are used for complexity, quality, functionality, efficiency, maintainability, reliability etc. Size oriented metrics can be categorized as LOC-Lines of Code, KLOC-1000 lines of code etc.

- ***Component***

Components are categorized as the collection of various pre-programmed tools which are used as the add-on page. There are also various tools present to measure Java source code. These tools will measure various different parameters in Java program. One of the major benefit of the component based tool is that the user can also select the tool of their own choice to measure the program according to the requirements. In this the user can also know more details about the tools and can find the links from where the tools can be downloaded. Some tools can also show warning messages and charts if the program is not structured appropriately or is not having proper format. All the components would not meet the user requirement criteria. Some can be used for measuring the program while some may be used for generating the report.

A software element is a coherent package of software implementation that presents published and well-defined interfaces are reusable. They can be individually developed and delivered such components together to form an application. The significant and relevant metrics relevant for the component quality during the execution of design phase are:

Component Size Metrics (CSM): CSM should be created on the concept of total number of sub-components such as use-cases or classes.

Weighted Methods per Class (WMC): WMC is based on the number of local methods which are defined in the component. It is basically related to size complexity.

Depth of Inheritance Tree (DIT): DIT refers to the maximum depth of the element in the inheritance tree. The deepness of the element hierarchy is directly proportional to larger the number of methods it is likely to inherit, which is making it further complex to predict the behavior of component.

Number of Children (NOC): NOC represents the number of immediate sub-components of a particular component. It actually measures inheritance complexity.

Count of Base Components (CBC): CBC is also based on the numbers of base components like NOC.

Response set for a class (RFC): RFC is the set of methods that can be potentially implemented in response to the received message by the component's object. It can be stated as the number of methods in a particular set.

Characteristics of components

Some significant characteristics of software components in usage perspective are as follows:

- Assumption of architectural embedding
- Presentation of each functionality via definite "incoming" or "provides" interfaces
- Presentation of parametric dependencies via specific "outgoing" or "requires" interfaces
- Static dependencies
- Targeting individual component platform
- Collaboration of other components

Requiring per-instance context
In brief, Ian Sommerville tabularizes the primary characteristics of components as shown below in the tabular form.

Metrics for Object oriented Designs
We have also provided a comparative study of the object oriented software metrics. These metrics are:

a) Morris Metrics:
Morris et al. suggested a metrics suite for object oriented metrics systems, it defines the system in the shape of tree structure. Morris defined the complexity of the object-oriented system in the shape of the depth of tree. This depth of tree evaluates the number of sub nodes of tree, large number of sub nodes of tree shows more complexity in the system. Therefore, complexity of an object is equivalent to total number of sub nodes or depth of tree.

b) Goal Question Metrics (GQM):
GQM approach is developed by V.L.Basili. This approach was initially defined for the evaluation of defects in various project sets of NASA Goddard Space Flight Center environment. He also delivered the set of categorization which are valuable for the programmers. The objective of the GQM is to express the significance of templates which covers purpose and prospective for driving metrics and questions. It delivers framework comprising three steps:

i) List most important goals of development or maintenance part of the project.
ii) Derive questions from each goal which must be answered to conclude that whether the results are being met or not.
iii) Decide which all parameters must be measured in proper order to answer all the questions satisfactorily.

Goal (Conceptual Level): A goal is specified for an object, for different variety of reasons and with respect to different models of quality with numerous different points of view. Objects of measurement are processes, products and resources.
Question (Operational Level): A set of questions are recognized to characterize the method of achievement for a particular goal which is going to accomplish using some specific characterizing model.
Metric (Quantitative Level): A defined set of data is combined with every single question in order to get a quantitative answer. This data can be subjective and objective, if they show dependencies on the objects only which they can be evaluated and not related to the viewport from which they might have taken. For example, size of a program, staff hours spent on a task, number of versions of a document.

c) Lorenz & Kidd Metrics
Lorenz & Kidd proposed a set of metrics which can be categorized in four categories that are internal, external, inheritance and size. Metrics defined for the class intervals are completely oriented towards the cohesion whereas the external metrics were utilized to reuse and examine. Inheritance based metrics are thoroughly concentrated on those concepts in which procedures are reused through the class hierarchy method. Size oriented metrics for the object oriented class can be concentrated on the average value of the object-oriented software, operations and attributes of an individual class and the count of the metrics as a whole.

d) Extended Metrics for Object-oriented Software Engineering:
W.Li et al. proposed this metrics of the MOOSE (Metrics for Object-oriented Software engineering) model. They may be defined as-

i) Message Pass Coupling (MPC): MPC stands for the number of message which can be replied by the class operations.
ii) Data Abstraction Coupling (DAC): DAC is used to evaluate the total number of classes which are combined to the current class and are also showing data abstraction coupling.
iii) Number of Methods (NOM): NOM is used to calculate the number of operations which are local to the class i.e. individual those class operations who can provide the number of techniques to measure it.
iv) Size1: Size1 is used to identify the count of line of code.
v) Size2: Size2 is used to compute the total number of Operations and local attributes defined in the class.

Table 1 provides a comparative study of metrics. These metrics can help to measure the size, complexity and efforts.

TABLE 1: METRIC COMPARISON

Source → / Metrics ↓	Morris	GQM	Lorenz & Kidd	EMOOS
DIT	Y			
LCOM	Y			
CBO	Y			
CS			Y	
NOA			Y	
NOO			Y	
SI			Y	
OS			Y	
OC			Y	
NP			Y	
MPC				Y
DAC				Y
NOM				Y

CONCLUSION AND FUTURE SCOPE

The concepts and features presented in this paper are mostly conceptual in nature but they also have a robust influence in software development processes. These metrics can be used by the software developers in order to develop and check the quality of the system software. A frame work is created which helps to pull together the concepts quality metric, component and their characteristics. Future research is needed for measuring the quality of the metrics with accurate measurement on the real word projects and then checking the quality of metrics for similar and different projects. We can also check the quality with the view point of the Developers and the Clients.

APPENDIX

DIT: Depth of inheritance tree
LCOM: Lack of cohesion in methods
CBO: Coupling between objects
CS: Class size
NOA: Number of operation added by some class
NOO: Number of operation overridden by subclass
SI: Specialization Index
OS: Average operation size
OC: Operation complexity
NP: Average number of parameter per operation
MPC: Message pass coupling
DAC: Data abstraction coupling
NOM: Number of methods

REFERENCES

[1] R.Harrison, S.J.Counsell and R.V.Nithi, "An Evaluation of the MOOD set of object oriented metrics", *IEEE transaction*, 24(6), 1998.

[2] J.Xu, H.Danny, L.Fernado Capretz, "An Empirical validation of object oriented design metrics for fault prediction", *Journal of computer science*, 4(7), 2008.

[3] Y.Zhou, H.Leung, "Empirical Analysis of Object Oriented design metrics for predicting high and low severity faults", *Journal of software engineering and application*, 2012.

[4] C.Neelamegan, Dr.M.Punithavalli, "A survey of Object Oriented Quality Metrics", *Global journal of computer science and technology*, 9(4), 2009.

[5] R.Harrison,Smaraweera,L.G.Dobie and Lewis, "Comparing Programming paradigm: An Evaluation of Functional and Object-oriented programs, *Software Engineering Journal*, 1996.

[6] Rodiger Lincke, Jones Lundberg, Wilf Lowe, "Comparing Software Metrics Tools", *ACM ISSTA*, Seattle, 2008.

[7] V.L.Basili, L.Briand and W.L.Melo, "A Validation of object-oriented metrics as Quality Indicators", *IEEE transaction software engineering*, 8(11), 2013.

[8] R.D.Neal: "The validation by measurement theory of proposed Object-oriented Software Metrics", Virginia, Commonwealth University, 1996.

[9] B.F.Abreu, "Design metrics for OO software system", ECOOP, Quantative Methods Workshop, 1995.

[10] Ian Sommerville, "*Software Engineering*", Pearson Education, 8th edition, 2008.

[11] M.Lorenz and J.Kidd, "*Object-Oriented Software Metrics*", Prentice Hall, 1996.