Comparative Study and Review on Object Oriented Design Metrics

Sonam Gupta
Shri Shankaracharya Group of Institution
Dept. of Information & Technology
Bhilai, Chhattisgarh, India
sonamgupta.jbj@gmail.com

Anish Lazrus
Shri Shankaracharya Group of Institution
Dept. of Information & Technology
Bhilai, Chhattisgarh, India
anishlazrus@gmail.com

Abstract

The best illustrations to programming advancement issues are frequently touted as object oriented procedures. Software complexity, computation time, project efforts are the essential metrics which are measured by most popular technique, Object Oriented Design. There are different methodologies through which we can discover the product cost estimation and predicates on different sorts of deliverable things. Question arranged measurements guarantees to diminish cost and the upkeep exertion by serving as early indicators to gauge programming flaws. Such an early evaluation enlarges the nature of the final programming. This paper reviews some of the metrics of Object Oriented metrics. A correlation table is kept up through which we can break down the contrast between all the object oriented measurements viably and effectively.

Keywords— *Software Engineering Metrics, Object Oriented Models, Inheritance Tree.*

I. INTRODUCTION

Various programming metrics identified with software quality confirmation have been proposed in the past are as yet being proposed. A few books showing such metrics exist, for example, Fenton's [1], Sheppard's [2] and others. The vast majority of these metrics are available to all programming languages, a few metrics apply to a particular arrangement of programming language. Among metrics of this kind, are those that have been proposed for object–oriented programming language.

These days, a quality designer can look over a massive amount of object–oriented metrics. The question posed is not the absence of metrics but rather the choice of those metrics which meet the particular requirement of every software project. A quality architect needs to confront the issue of selecting the fitting arrangement of metrics for his product estimations. Various object–oriented metrics exploits the information picked up from metrics utilized as a part of organized programming and adjust such estimations in order to fulfill the requirements of object–oriented programming. Then again, other object–oriented metrics have been created particularly for object–oriented programming and it is inconsequential to apply them to organized programming. The above figure demonstrates the various leveled structure of the metrics.

II. CK METRICS MODEL

Chidamber and Kemerer characterize the purported CK metric suite [3]. CK metrics have produced a lot of intrigue and are right now the most understood well suite of estimations for OO software [4]. Chidamber and Kemerer proposed six metrics; the accompanying exchange demonstrates their metrics.

A. Weighted Method per Class (WMC)

WMC measures the intricacy of a class. Many-sided quality of a class can for instance be figured by the cyclomatic complexities of its techniques. High estimation of WMC demonstrates the class is more mind boggling than that of low values.

B. Depth of Inheritance Tree (DIT)

DIT metric is the length of the most extreme way from the hub to the base of the tree. So this metric figures how far down a class is announced in the legacy chain of importance. The accompanying figure demonstrates the estimation of DIT for a basic class progression. DIT speaks to the intricacy of the conduct of a class, the multifaceted nature of plan of a class and potential reuse.

Subsequently it can be difficult to comprehend a framework with numerous legacy layers. Then again, a huge DIT esteem demonstrates that numerous strategies may be reused.

C. Number of Children (NOC)

This metric measures what number of sub-classes will acquire the techniques for the parent class. As appeared in above figure, class C1 has three kids, subclasses C11, C12, and C13. The measure of NOC around shows the level of reuse in an application. On the off chance that NOC develops it implies reuse increments. Then again, as NOC expands, the

measure of testing will likewise increment since more youngsters in a class demonstrate more duty. Thus, NOC speaks to the exertion required to test the class and reuse.

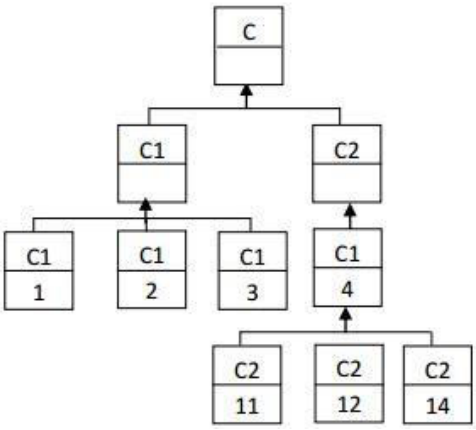

Fig.1. value of DIT in class hierarchy

D. Coupling between articles (CBO)

The possibility of this metrics is that a question is coupled to another protest if two question follow up on each other. A class is combined with another if the techniques for one class utilize the strategies or characteristics of alternate class. An expansion of CBO shows the reusability of a class will diminish. Consequently, the CBO values for every class ought to be kept as low as could be allowed.

E. Response for a Class

RFC is the quantity of strategies that can be summoned in light of a message in a class. Pressman [5] States, since RFC expands, the exertion required for testing likewise increments in light of the fact that the test grouping develops. On the off chance that RFC builds, the general plan multifaceted nature of the class increments and turns out to be difficult to get it. Then again bring down qualities demonstrate more noteworthy polymorphism. The estimation of RFC can be from 0 to 50 for a class12, a few cases the higher esteem can be 100-it relies on upon venture to extend.

F. Lack of Cohesion in Methods (LCOM)

This metric uses the thought of level of closeness of strategies. LCOM measures the measure of cohesiveness present, how well a framework has been planned and how complex a class is [6]. LCOM is a check of the quantity of strategy combines whose comparability is zero, short the tally of technique matches whose similitude is not zero.

Raymond [6] examined for instance, a class C with 3 techniques M1, M2, and M3. Let I1= {a, b, c, d, e}, I2= {a, b, e}, and I3= {x, y, z}, where I1 is the arrangement of example factors utilized by technique M1. So two disjoint set can be found: I1 Ç I2 (= {a, b, e}) and I3. Here, one sets of strategies who share no less than one example variable (I1 and I2). So LCOM = 2-1 =1. [13] States "Most of the strategies characterized on a class ought to utilize a large portion of the information individuals the vast majority of the time".

On the off chance that LCOM is high, techniques might be coupled to each other through qualities and after that class configuration will be intricate. Thus, planners ought to keep union high, that is, keep LCOM low.

III. LITERATURE SURVEY

Kumar Rajnish et al. [7]. The inheritance metrics give us information about the inheritance tree of the system. Inheritance is a key feature of the Object-Oriented (OO) paradigm. This mechanism supports the class hierarchy design and captures the IS-A relationship between a super class and its subclass. Several OO inheritance metrics have been proposed and their reviews are available in the literature. In doing so, an attempt has been made to define empirical relationship between the proposed inheritance metric suites with considered existing inheritance metrics and the focus was on which how the inheritance metric suites were correlated with the existing ones. Data for several C++ classes has been collected from various sources.

Kailash Patidar et al. [8]. Software engineering aims at development of high-quality software and tools to promote quality software that is stable and easy to maintain and use. In order to assess and improve software quality during the development process, developers and managers use, among other means, ways to automatically measure the software design of object oriented programming. Cohesion, coupling, and complexity are common types of such metrics. The cohesion of a module indicates the extent to which the components of the module are related. A highly cohesive module performs a set of closely related actions and cannot be split into separate modules.

K.K.Aggarwal et al. [9]. The increasing importance of software measurement has led to development of new software measures. Many metrics have been proposed related to various constructs like class, coupling, cohesion, inheritance, information hiding and polymorphism. But there is a little understanding of the empirical hypotheses and application of many of these measures. It is often difficult to determine which metric is more useful in which area. As a consequence, it is very difficult for project managers and practitioners to select measures for object-oriented systems. A key element of any engineering process is measurement. Measures are used to better understand the attributes of the model that we create. But, most important, we use measurements to assess the quality of the engineered product or the process used to build it.

Gopal Goyal et al. [10]. A large numbers of metrics have been proposed for measuring properties of object-oriented software such as size, inheritance, cohesion and coupling. The coupling metrics presented in this paper exploring the difference between inheritance and interface programming. Object-oriented design and programming is the dominant development paradigm for software systems today. Recently so many languages are object-oriented (OO) programming languages. In object oriented programming we provide abstraction by classes and interfaces.

Dr. K.P.Yadav et al. [11]. The increasing importance of software measurement has led to development of new software measures. Many metrics have been proposed related to various constructs like class, coupling, cohesion, inheritance, information hiding and polymorphism. The central role that software development plays in the delivery and application of information technology, managers are increasingly focusing on process improvement in the software development area. It is very difficult for project managers and practitioners to select measures for object-oriented system. This demand has spurred the provision of a number of new and/or improved approaches to software development, with perhaps the most prominent being object-orientation (OO).

IV. REVIEW OF METRICS

Two important of the OO metrics for object oriented software development are.

a. Chen Metrics
b. Lorenz and Kidd Metrics

A. *Chen Metrics*

Chen et al. [12] proposed some major metrics for measurement performance, throgh which it can define "What is the behavior of the metrics in object-oriented design".
They describe the behavoirs like:

- Class Coupling Metric
- attribute Complexity Metric
- Cohesion Metric
- Reuse Metric

B. *Lorenz and Kidd Metrics*

Lorenz and Kidd et al. [13] proposed set of metrics that can be grouped in 4 category.

- Size
- Inheritance
- Internal
- External.

Size metrics based on the count of various attributes. Whereas Inheritance based on operation to be reused. External metrics are used to examine and reuse.
Various categories of metrics are:

- Class Size
- Number of Operation
- Average Operation Size
- Operation Complexity

TABLE I*. shows various source construct and its metrics*

SNO	Source Construct	Metrics	Object Oriented Structure
1	Traditional Metrics	Cyclomatic Complexity	Function
		Lines of Codes	function
		Comment Percentage	function
2	New Object Oriented Metrics	Weight Method per class	function and class
		Response for a class	class and message
		Coupling between objects	coupling
		Depth of Inheritance	Inheritance

V. CONCLUSION

This review paper presented some of the metrics used for evaluation of object oriented system. Metrics are the indicators of the performance of the system. Paper has analyzed and compare various software metrics. As a number of software processes are increasing, the complicity also increasing. There will always be need of software metrics to test system performance. In this paper, we have demonstrated all the software metrics for object oriented systems. They are basics for measuring various characteristics like size, time, complexity, performance etc. Paper intended to provide reviews and survey of software metrics to help researchers and practitioners for better understanding of Object Oriented metrics in short and selective way.

REFERENCES

[1] N. Fenton et al, “Software metrices: a rigorous and practical approach”, International Thomson Computer Press 1996
[2] M.J. Sheppard & D. Ince, ―Derivation and Validation of Software Metrics, Clarendon Press, Oxford, UK, 1993

[3] Chidamber, Shyam, Kemerer, Chris F. "A Metrics Suite for Object- Oriented Design" M.I.T. Sloan School of Management E53-315, 1993
[4] C. Jones, “Estimating Software Costs: Bringing Realism to Estimating”, 2nd Edition, Mc Graw Hill, New York, 2007
[5] Roger S. Pressman: ―Software Engineering‖, Fifth edition, ISBN 0077096770.
[6] Raymond, J. A, Alex, D.L: ―A data model for object oriented design metrics‖, Technical Report 1997, ISBN 0836 0227.
[7] Kumar Rajnish , Arbind Kumar Choudhary, Anand Mohan Agrawal. INHERITANCE METRICS FOR OBJECT-ORIENTED DESIGN,International Journal of Computer Science & Information Technology (IJCSIT), Vol 2, No 6, December 2010
[8] Kailash Patidar, RavindraKumar Gupta,Gajendra Singh Chandel , International Journal of Advanced Research in Computer Science and Software Engineering 3(3), March - 2013, pp. 517-521
[9] K.K.Aggarwal, Yogesh Singh, Arvinder Kaur, Ruchika Malhotra,: “Empirical Study of Object-Oriented Metrics”, in Journal of Object Technology, vol. 5. no. 8, Novmeber-December 2006, pp. 149-173
[10] Gopal Goyal, Sachin Patel: Importance of Inheritance and Interface in OOP Paradigm Measure through Coupling Metrics, International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868, Foundation of Computer Science FCS, New York, USA, Volume 4– No.9, December 2012
[11] Dr. K.P. Yadav, Ashwini Kumar, Sanjeev Kumar: bject Oriented Metrics Measurement Paradigm, IJMIE Volume 2, Issue 6 ISSN: 2249-0558
[12] Chen, J-Y., Lum, J-F.: "A New Metrics for Object-Oriented Design." Information of Software Technology 35,4(April 1993):232-240.
[13] M. Lorenz, J. Kidd, “Object Oriented Software Metrics”, Prentice Hall, NJ, (1994)