RESEARCHERID
THOMSON REUTERS

[Gupta* *et al.*, 6(5): May, 2017]
IC™ Value: 3.00

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

OBJECT ORIENTED DESIGN METRICS FOR DESIGN AND COMPLEXITY ANALYSIS

Sonam Gupta*, Prof. Anish Lazrus

* Shri Shankaracharya group of Institution Dept. of Information and Technology Bhilai, Chhattisgarh, India

Shri Shankaracharya group of Institution Dept. of Information and Technology Bhilai, Chhattisgarh, India

DOI: 10.5281/zenodo.802769

ABSTRACT

Quality assurance is one of the imperative non-functional programming requirements which numerous software items neglect to fulfill. Current software market is driven for the most part by earnestness and rivalry. This represents a major issue to software quality affirmation, consumer loyalty and reliability of the product items. One of the techniques to guarantee programming quality is a measurements based approach. Programming measurements have been utilized to quantitatively assess programming items. Software measurements proposed and utilized for procedural worldview have been discovered deficient for protest situated programming items mainly in light of the recognizing components of the question arranged worldview, for example, legacy and polymorphism. In this paper, we have analyzed various software metrics based on difefrent criteria. The result of analysis suggests user or developer that how the software should be change or improved to comply with standards.

KEYWORDS: Software Engineering Metrics, Object Oriented Models, Inheritance Tree.

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.

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.

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.

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

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.

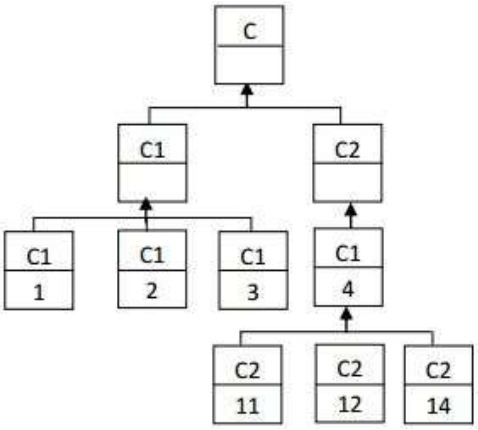

Fig.1. value of DIT in class hierarchy

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.

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.

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 [14].

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

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

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.

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).

PROBLEM IDENTIFICATION

In past years, many new technologies are introduced offering many challenges in designing of software development. Innovation is constantly changed [15]. So for a decent plan, it is common to adjust with new advancements. Presently it is the period of object oriented design, on the grounds that different properties of object oriented design (Inheritance, modularity and so on) support the adjustment without changing the past or existing modules. In any case, one ought to dependably be cautious about a few properties of object oriented design, which can make the plan more mind boggling, for instance "inheritance" property.

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

Architects can't have the capacity to utilize object oriented design in a manner that it will help him if there should arise an occurrence of later with the change of advances yet won't make the program more intricate. An excess of strategy makes a framework complex. Martin Proposes four essential manifestations tell whether plans are decaying. They are not orthogonal, but rather are identified with each other in ways that will get to be distinctly self-evident. They are: unbending nature, delicacy, stability, and consistency.

Here this paper proposed various metrics for software development process. And also how these metrics helps in analyzing the software design patterns. The design patterns helps in organizing the software and identifying loose coupling and strong coupling of objects between classes.

ANALYSIS OF METRICS

We have analyzed 5 different metrics using Java classes in which 4 are Chidamber and Kemerer metrics and one is non C & K metrics [16].

WMC: Weighted methods per class
DIT: Depth of Inheritance Tree
CBO: Coupling between object classes
RFC: Response for a Class
Ce: Efferent coupling

Java source code is used for performing experiments. We have performed experiments to evaluate the performance of java source code named jUnit 4.10 on a java environment. This chapter presents results performed using 6 different metrics and its various values.

TABEL I: jUnit Package Details

SNO	Attributes	Values
1	Packages	31
2	Classes	100+
3	Methods	701+
4	Constructor	94
5	Fields	162+
6	Types	145

WMC

It generally tells that how much effort from developer side are required to main particular class. Class 2,6,13, 22 has higher WMC means it is more complex than any other class. The classes 4, 5, 8 and 20 has lower value of WMC means it has greater polymorphism than others. They are strongly connected. The lower limit for WMC in RefactorIT is default 1 because a class should consist of at least one function and the upper default limit is 50.

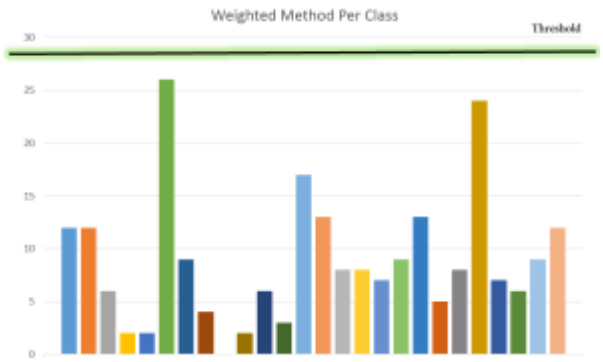

Fig. 2. Shows the WMC of all 26 classes compared to threshold

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

DIT

Classes 3, 4, 5, 8, 10, 11, 15, 16, 17, 18 and 23 has DIT as 0. It simply means that they are the super class. The other classes has various DIT value, its tells how far that class is from the super class. The more DIT will be, the more variable and method it will contains or inherit. The higher the DIT, the more will be the usability of the code. The min and max threshold value should be 0 and 1 respectively for code to be reusable.

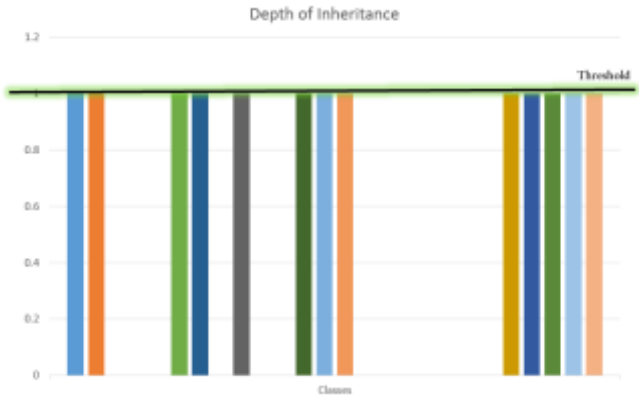

Fig.3. shows the threshold value and original value of DIT metrics

CBO

It generally tell, average number of classes used per class in particular package. Nominal range is between 1 to 4. Class 1 has 17 and class 6 has 10 CBO values, means it has crosses the nominal range. So, we have to remove and place on other packages. It also means that the classes with higher CBO are tightly coupled. The developer must decouple the classes to improve performance.

RFC

If RFC classes has highest value then it means that there exists some complexity in that particular class. Class 6 has highest value of RFC, means it is more complex than any other class. RefactorIT recommends max 50 RFC value. It should not exceed this value. The highly complex in sense that, if a method calls happen, it recursively calls to another method and this continues. So the RFC value should be between 0 to 50.

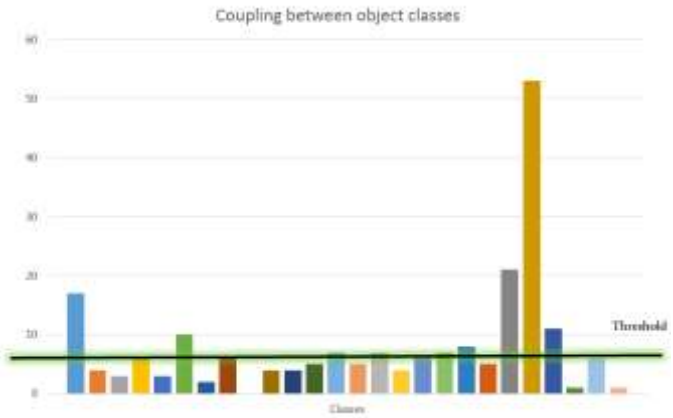

Fig. 4. Shows majority of classes falls in the normal range of CBO metrics

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

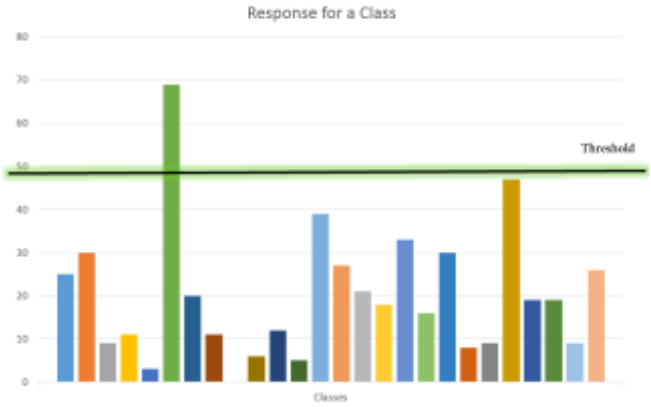

Fig. 5. Shows comparison between desired threshold with actual threshold value of RFC class

Ce

They are also known as outgoing dependencies. A large efferent coupling can indicate that a package is unfocussed and may also indicate that it is unstable since it depends on the stability of all the types to which it is coupled. RefactorIT recommends an upper limit of 20. We have all of our metrics are under 20. So we don't have complexity exists at this metrics.

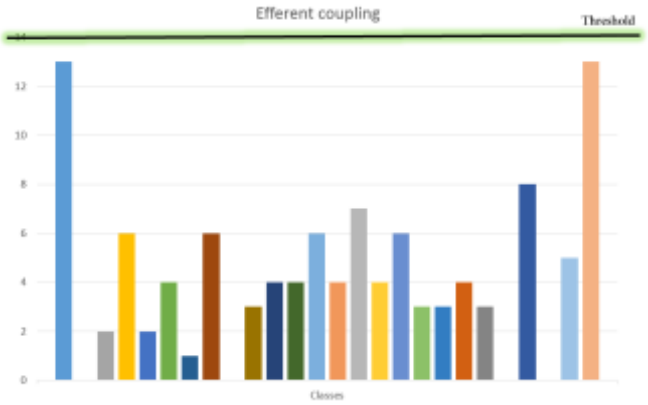

Fig. 6. Presents the Ce metrics wrt its threshold for different classes

CONCLUSION

This paper analyzed over 26 different classes of java source code. Analysis carried out using OO design metrics. The calculation are based on these metrics. After experiment this thesis aimed to analyses the different metrics based on its calculated metrics. This analysis are presented at analysis section. The analysis helps any software developer or programmer to effectively find mistakes in the software and correct it wisely so that in future the problem don't re-appear. Hence, different metrics shows that how the software metrics should be normalized w.r.t. to desired calculation for better use of code and error free code.

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.

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

[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, (199
[14] 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
[15] 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
[16] 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.

CITE AN ARTICLE

Gupta, S., & Lazrus, A., Prof. (2017). OBJECT ORIENTED DESIGN METRICS FOR DESIGN AND COMPLEXITY ANALYSIS. *INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY, 6*(5), 803-809. doi:10.5281/zenodo.802769