Volume 5, Issue 4, April 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Study of the Efficacy of Various Kinds of Object Oriented Metrics to Ensure Better Understandability and Maintainability of Object Oriented Design

P. Ashok Reddy[1], Dr. K. Rajasekhara Rao[2], Dr. M. Babu Reddy[3]
[1] Sr. Asst. Professor of Comp. Applications LBRCE-Mylavaram, Krishna Dt., AP, India
[2]Director, Sri Prakash Educational Institutions, TUNI, AP, India
[3]Asst. Professor of Computer Science Krishna University, Machilipatnam, AP, India

Abstract: In the Glossary of Software engineering, Object-Oriented design is becoming other essential task in software development environment and software Metrics are necessary in software engineering for measuring the complexity of software, estimating size, quality and project efforts. Various tools are used for measuring the estimations about lines of codes, function points, and object points. This paper highlights mostly the classification of metrics like procedural metrics and the object-oriented metrics for improving understandability and modifiability.

The central role of software development plays in the delivery and application of information technology, managers are increasingly focusing on process improvement in the software development area. The focus on process improvement has increased the demand for software measures, or metrics with which to supervise the process. The need of such metrics is particularly acute when an organization is adopting a new technology for which established practices have yet to be developed. This research addresses needs through the development and implementation of a suite of metrics for OO design.

Keywords: Software Metrics,Procedure Oriented Metrics, Object-Oriented Metrics, Class Metrics, System Metrics.

I. INTRODUCTION

Object-Oriented Analysis and Design of software provide many benefits such as reusability, decomposition of problem into simply understandable object and the aiding of future modifications[3]. Object-Oriented design is more valuable in software development environment and object oriented design metrics is an essential feature to measure software quality over the environment. Object-oriented is a classifying approach that is capable to classify the problem in terms of objects and it may provide many paybacks on reliability, adaptability, reusability and decomposition of problem into easily understandable objects and providing some future modifications.

Metrics provide insight necessary to create and design model through the test. It also provide a quantative way to access the quality of internal attributes of the product, so it enables the software engineer to access quality before the product is fabricated. Metrics are the crucial source of information through which a software developer takes a decision for design good software[9]. Some metrics may be transformed to serve their purpose for a new environment. Software metrics are the tools of measurement.

Software metrics are mostly or generally characterized by the software engineering product (example: design, source code, and test case), software engineering process (example: analysis, design and coding) and software engineering people (example: the efficiency of an individual tester or the productivity of an individual designer)

II. REVIEW OF SOFTWARE QUALITY METRICS

In this review of Software Quality Metrics the following types are to be addressed.

1. Size related Metrics
2. Complexity Metrics
3. Quality Metrics

2.1.Size Related Metrics

These are the metrics which can help to quantify the software size. There are two types of software metrics which are used to measure the software size:

a) **Lines of code (LOC):** It is the oldest metrics which is used to measure the module size in terms of lines of codes

b) **Function point Metrics:** It was measured lines of code when the code is available and hence cannot be used in early stage. This was totally depends on the user input, user output, inqueries and deliberate the values to compute the value to measure program size and thus effort required for the development.

2.2. Complexity Metrics

These metrics are essential to describe the complexity which has direct impact on the quality of design.

© 2015, IJARCSSE All Rights Reserved

a) **Module Complexity** : complexity measurement may be known as module logical complexity. The basic goal of the metrics is to evaluate the testability and maintainability of the software module. This metrics can also be used as a indicator of reliability in a software intensive system.
b) **Information flow : M**etrics are also necessary to measure the number of local information about input flows (fan-in) and output flows (fan-out)

2.3.Quality Metrics : The quality metrics defines about quality of the approach that was followed.

a) **Defect Metrics:** Here there is no effective procedure for counting the defects in the program, number of design changes, number of intended errors and the error detected by the code inspections and the number of program tests may be treated as an alternative measures to the defects.
b) **Reliability Metrics:** The internal product quality is generally measured by the number of bugs in the software or by how long the software can run before encountering in a crash.
c) **Maintainability Index:** It **is** defined by a number of functions that predicts software maintainability. At this point a relation is maintained in between average volume per module , average cyclomatric complexity per module and average lines of code per module .

III. PROCEDURAL METRICS

Software metrics plays a significant role in project coordination and project management. With the help of software metrics different set of attributes of a project can be measured. Software metrics are also helpful in the area that is prone to an error. Different types of metrics like size metrics, quality metrics, object oriented metrics etc. Procedure oriented metrics measures different attributes of a project or smaller pieces of code. Here, function points are intended to be a measure of program size and program environment; thus, effort required for development.

IV. OBJECT-ORIENTED DESIGN METRICS

4.1. List of Object-Oriented Design Metrics

Object-oriented measurements are being used to evaluate and predict the quality of software. The validation of these metrics requires convincingly demonstrating that the metric measures what it purports to measure (for example, a coupling metric really measures coupling) and the metric is associated with an important external metric, such as reliability, maintainability and fault-proneness Often these metrics have been used as an early indicator of these externally visible attributes, because the externally visible attributes could not be measures until too late in the software development process.

The list of Object Oriented Design Metrics are:

1. Methods per Each Class(MPEC)
2. Attributes per Each Method (APEM)
3. Depth of Inheritance per Each Class (DIPEC)
4. Level of Inheritance per Each Class (LIPEC)
5. Cohesion and Coupling among the Classes (CCAC)
6. Relationships among the classes in Class Diagram (RCCD)
7. Number of Instances of the Class (NIC)
8. Eliminating the redundancy among the classes (RAC)
9. Inheritance among the Attributes (IAA)
10. Inheritance among the Methods (IAM)

4.2. Limitations of Object-Oriented Metrics

It is to be noted that the validity of these metrics can sometimes be criticized .Many things, including fatigue and mental and physical stress, can impact the performance of programmers with resultant impact on external metrics[10]. "The only thing that can be reasonably stated is that the empirical relationship between software product metrics are not very likely to be strong because there are other effects that are not accounted for, but as has been demonstrated in a number of studies, they can still be useful in practice.

4.3. Object Oriented Design and Development

To develop the object oriented development among the classes it needs to focus on following things

a) Identification of Classes
b) Identification of the Semantics
c) Identification of the Relationship among the classes
d) Implementation of Classes

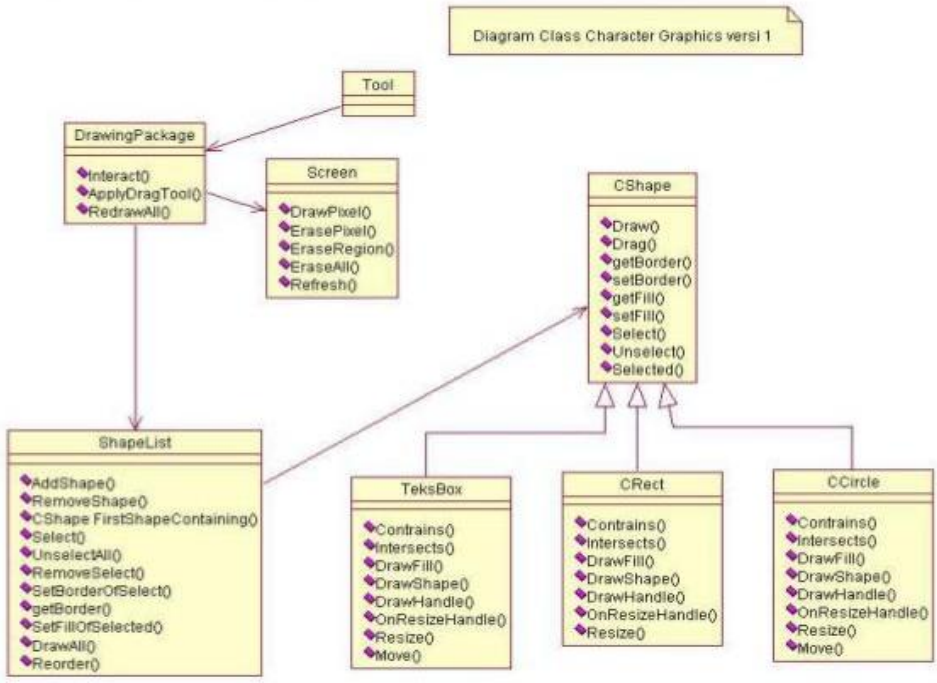
V. METRICS FOR OBJECT-ORIENTED SOFTWARE ENGINEERING (MOOSE)

It is proposed some metrics that have generated a significant amount of interest and are currently the most well known object-oriented suite of measurements for Object-Oriented software.The object oriented software engineering consists of six metrics that assess different approaches of characteristics for object oriented approach. Those are

© 2015, IJARCSSE All Rights Reserved

a) Weighted Methods per Class (WMC)
b) Depth of Inheritance (DI)
c) Number of Children(NOC)
d) Coupling among Objects (CAO)
e) Response set of Each Class (REC)
f) Lack of Cohesion in the Methods (LOCM)

Example Class Diagram



5.1: Metric 1: Weighted Methods per Class (WMC)

Definition: Consider a Class C1, with methods $M_1, M_2 \ldots M_n$ *that are defined in the class*. Let $cm_1, cm_2 \ldots cm_n$ be the complexities of corresponding methods, then:

$$WMC = \sum_{i=1}^{n} Cm_i$$

Viewpoints

• The number of methods and the complexity of methods involved is a predictor of how much time and effort is necessary to develop and maintain the class.
• The larger the number of methods in a class the greater the potential impact on children, since children will inherit all the methods defined in the class.
• Classes with large numbers of methods are likely to be more application specific, limiting the possibility of reuse.

For the above Class Diagram Weighted Method per Class(WMC) is as follows

Classname	**WMC**
Tool.java	1
ShapeList.java	30
Screen.java	13
DrawingPackage.java	19
Cshape.java	10
Crect.java	15
Ccircle.java	20
TextBOx.java	13
Total WMC	121

5.2 Metric 2: Depth of Inheritance Tree (DIT)

Definition*:* Depth of inheritance of the class is the DIT metric for the class. In cases involving multiple inheritance, the DIT will be the maximum length from the node to the root of the tree.

© 2015, IJARCSSE All Rights Reserved

Viewpoints:

• The deeper a class is in the hierarchy, the greater the number of methods it is likely to inherit, making it more complex to predict its behavior.

• Deeper trees constitute greater design complexity, since more methods and classes are involved.

• The deeper a particular class is in the hierarchy, the greater the potential reuse of inherited methods.

For the above Class Diagram Depth of Inheritance Tree (DIT) is as follows

Classname	**DIT**
Tool.java	0
ShapeList.java	0
Screen.java	0
DrawingPackage.java	0
Cshape.java	0.75
Crect.java	0
Ccircle.java	0
TextBOx.java	0
Total DIT	0.75

5.3 Metric 3: Number of children (NOC)

Definition: NOC = number of immediate sub-classes subordinated to a class in the class hierarchy.

Viewpoints:

- As NOC grows, reuse increases – but the abstraction may be diluted
- Depth is generally better than Breadth in class hierarchy , since it promotes reuse of methods through Inheritance
- NOC gives an idea of the potential influence a class may has on the design
- Classes with large number of children may require more testing

For the above Class Diagram Number of Children (NOC) is as follows

Classname	**NOC**
Tool.java	0
ShapeList.java	0
Screen.java	0
DrawingPackage.java	0
Cshape.java	3
Crect.java	0
Ccircle.java	0
TextBOx.java	0
Total NOC	0.75
Median	0
Standard Deviation	1.06
Min; Max	0;3

5.4 Metric 4: Coupling between object classes (CBO)

Definition: CBO for a class is a count of the number of other classes to which it is coupled.

Two classes are coupled when methods declared in one class use methods or instance variables defined by another class.

Viewpoints:

• CBO is the number of collaborations in between two classes

- As collaboration increases, reuse decreases
- High fan-outs represent class coupling to other classes/objects and thus are undesirable
- High fan-ins represent good object designs and high level of reuse
- Not possible to maintain high fan-in and low fan-out s across the entire system

For the above Class Diagram, Coupling between object classes (CBO) is as follows.

Classname	**CBO**
Tool.java	1
ShapeList.java	2
Screen.java	1

© 2015, IJARCSSE All Rights Reserved

DrawingPackage.java	3
Cshape.java	4
Crect.java	1
Ccircle.java	1
TextBOx.java	1
Total CBO	14
Median	1
Standard Deviation	1.16
Min; Max	1;4

5.5 Metric 5: Response for a Class (RFC)

Definition: RFC = | RS | where RS is the response set for the class.

Viewpoints:

• If a large number of methods can be invoked in response to a message, the testing and debugging of the class becomes more complicated since it requires a greater level of understanding on the part of the tester.

• The larger the number of methods that can be invoked from a class, the greater the complexity of the class.

• A worst case value for possible responses will assist in appropriate allocation of testing time.

For the above Class Diagram, Response for a Class (RFC) is as follows

Classname	**RFC**
Tool.java	3
ShapeList.java	20
Screen.java	8
DrawingPackage.java	19
Cshape.java	45
Crect.java	16
Ccircle.java	19
TextBOx.java	17
Total RFC	147

5.6 Metric 6: Lack of Cohesion in Methods (LCOM)

The LCOM is a count of the number of method pairs whose similarity is 0 minus the count of method pairs whose similarity is not zero. The LCOM value provides a measure of the relative disparate nature of methods in the class.

Viewpoints:

• Cohesiveness of methods within a class is desirable, since it promotes encapsulation.

• Lack of cohesion implies classes should probably be split into two or more sub-classes.

• Any measure of disparateness of methods helps to identify flaws in the design of classes.

• Low cohesion increases complexity, thereby increasing the likelihood of errors during the development process.

For the above Class Diagram, Lack of Cohesion in Methods (LCOM) is as follows

Classname	**LCOM**
Tool.java	0
ShapeList.java	0
Screen.java	0
DrawingPackage.java	1
Cshape.java	24
Crect.java	0
Ccircle.java	0
TextBOx.java	0
Total LCOM	25

VI. METHODOLOGY FOR OBJECT-ORIENTED DESIGNS

Step 1: Select a metrics or metric set to measure the attributes of object-oriented design.

Step 2: Form a metric set for a procedure to measure the effectiveness of an object-oriented design of a particular domain.

© 2015, IJARCSSE All Rights Reserved

Step 3: Calculate and obtain the attribute values of the metric set.
Step 4: Obtained values of metrics are tabulated for all classes of the system for easy usage and manipulation of metric data.

VII. GUIDING FACTORS OF OBJECT ORIENTED METRICS

The following are the characteristics of Object Oriented Metrics

- Localization operations used in many classes
- Encapsulation metrics for classes, not modules
- Information Hiding should be measured and improved
- Inheritance adds complexity and hence it need to be measured
- Object Abstraction metrics represent level of abstraction

Inside the Object Oriented Metrics the software should be measured through the parameters like:

1.Size
- ✓ Population (Number of classes, operations)
- ✓ Volume (dynamic object count)
- ✓ Length (e.g., depth of inheritance)
- ✓ Functionality (Number of user functions)

2.Complexity
- ✓ How classes are interrelated

3. Coupling
- ✓ Number of collaborations between classes, number of method calls, etc.

4.Sufficiency
- ✓ Does a class reflect the necessary properties of the problem domain

5.Completeness
- ✓ Does a class reflect all the properties of the problem domain (for reuse)

6. Cohesion
- ✓ Do the attributes and operations in a class achieve a single, well-defined purpose in the problem domain

7.Primitiveness (Simplicity)
- ✓ Degree to which class operations can't be composed from other operations

8.Similarity
- ✓ Comparison of structure, function, behavior of two or more classes

9.Volatility
- ✓ The likelihood that a change will occur in the design or implementation of a class

VIII. CONCLUSION AND FUTURE WORK :

An effort has been initiated to have close look into the various kinds of parameters which are very much essential to fine tune the design process especially in the Object Oriented environments by keeping understandability and maintainability in view. For bringing the better insight into the usefulness of various Object Oriented metrics, empirical study has been conducted on sample set of Java programs. In this paper, a comprehensive study has been carried out to observe the impact of object oriented metrics on the quality of software applications. They provided a basis for measuring the necessary influencing parameters like size, complexity, performance and quality, etc.., Similar type of studies need to be carried out with different data sets to give generalized results across different organizations with large scale object-oriented software systems.

REFERENCES

[1] C. Shyam, Kemerer, F. Chris, "A Metrics Suite for Object- Oriented Design" M.I.T. Sloan School of Management, pp. 53-315, 1993.
[2] R.S.Pressman,"Siftware Engineering-A practioners Approach" Fourth Edition, Mc. Graww Hill International Edition 1997.
[3] C. Neelamegam, M. Punithavali, "A survey on object oriented quality metrics", Global journal of computer science and technologies, pp 183-186, 2011.
[4] A. Deepak, K. Pooja, T. Alpika, S. Sharma, "Software quality estimation through object oriented design metrics", IJCSNS International journal of computer science and network security, april 2011, pp 100-104.
[5] E.V.Berrad, "Metrics fot Object-Oriented Software engineering", The Object Agency.
[6] Amith Sarma , S.K.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 ISSN (Online):
[7] E.V.Berrad, "Metrics for Object-Oriented Software engineering", The Object Agency.
[8] V.R. Basili, L.C. Briand, and W.L. Melo, "A Validation of Object- Oriented Design Metrics as Quality Indicators," IEEE Trans. Software Eng., vol. 22, pp. 751-761, 1996.
[9] S.R. Chidamber, "Metrics for Object-Oriented Software Design," PhD thesis, MIT, 1994.
[10] S.R. Chidamber and C.F. Kemerer, "A Metrics Suite for Object- Oriented Design," IEEE Trans. Software Eng., vol. 20, pp. 476-493, 1994.

© 2015, IJARCSSE All Rights Reserved

[11] [Oct 1996], Victor R. Basili, Fellow, IEEE, "A Validation of Object- Oriented Class Metrics As Quality Indicators". IEEE Transactions On Software Engineering, Vol. 22.
[12] [June 1994] ,Shyam R. Chidambaram and Chris F. Kemmerer, "A Metrics Suite for Object Oriented Design" , IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 20, NO. 6
[13] Karin Erni , "Applying Design-Metrics to Object- Oriented Frameworks"
[14] [Jan 2012], Mrinal Singh Rawat, "Survey on Impact of Software Metrics on Software Quality", Department of Computer Science MGM's COET, Noida, India International Journal of Advanced Computer Science and Applications, Vol. 3, No.
[15] [May 2003] , Hilda B. Klasky, "A Study Of Software Metrics " A thesis submitted to the Graduate School-New Brunswick Rutgers, The State University of New Jersey
[16] [2009], "Software Measurements And Metrics: Role In Effective Software Testing
[17] [2005], Linda Westfall, "12 Steps to Useful Software Metrics"
[18] "Design and Development of a Procedure for new Object-Oriented Design Metrics " by K.P.Srinivasan and Dr.T.Devi, by IJCA (0975-8887) Volume 24-8 June 2011.
[19] "Comparative Study Of Static Metrics Of Procedural And Object Oriented Programming Languages" by Manik Sharma , Lakbhir Singh and Chandini Sarma by IJC & T with Volume 2 No.1 Feb 2012
[20] " Empirical Study of Object-Oriented Metrics" by K.K.Aggarwal, Yogesh Singh, Arvinder Kaur, Ruchika Malhotra School of Information Technology, GGS Indraprastha University

ABOUT AUTHORS

Mr. P. Ashok Reddy, has received his Master's Degree from JNTU Kakinda with Computer Science & Engineering specialization and at present he pursuing Doctor of Philosophy from Acharya Nagarjuna University, AP, India in the area of Software Engineering. He has been actively involved in teaching and research for the past 9 years and now he is working as Sr. Asst. Professor in LBRCE-Mylavaram, AP, India.

Dr. K Rajasekhara Rao, has received his Doctor of Philosophy in Computer Science & Engineering from Acharya Nagarjuna University, AP, India. He has been actively involved in teaching and research for the past 30 years and now he is working as Director, Sri Prakash Educational Institutions, Tuni, AP, India. His research interests include: Embedded Systems, Software Engineering, Algorithm Complexity analysis and Parallel Computing. Three of his scholars have successfully earned their PhD degrees in Computer Science & Engineering domain and 8 more scholars are pursuing their PhD work under his guidance from various Universities.

Dr. M. Babu Reddy, has received his Master's Degree and Doctor of Philosophy from Acharya Nagarjuna University, AP, India with Computer Science & Engineering specialization. He has been actively involved in teaching and research for the past 15 years and now he is working as Asst. Professor of Computer Science, Krishna University, Machilipatnam, AP, India. His research interests include: Machine Learning, Software Engineering, Algorithm Complexity analysis and Data Mining.

© 2015, IJARCSSE All Rights Reserved