ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 3, May 2013

Study and Evaluation for Quality Improvement of Object Oriented System at Various Layers of Object Oriented Matrices

N. A. Nemade[1], D. D. Patil[2], N. V. Ingale[3]

Assist. Prof. SSGBCOET Bhusawal[1], H.O.D. (CSE) SSGBCOET Bhusawal[2], Research Scholar SSGBCOET Bhusawal[3].

Abstract*— The fundamental of any engineering discipline is measurement. For making quality management decisions object-oriented design metrics can be used which is considerable evidence. This leads to saving substantious costs of allocated resources in testing or estimate a maintenance effort of project. Many object-oriented metrics have been designed for C++ which is most preferred language for many object-oriented systems. This paper focuses on an evaluation of object oriented metrics in C++. For this, two projects considered as inputs for the study – The first project is a library management system for college and the second is a graphical editor which can be used to describe and create a scene. The metric values have been calculated by semi automated tool. The resulting values have been analyzed to provide significant insight about object oriented characteristics of projects. The metric values have been calculated using a semi automated tool. The resulting values have been analyzed to provide significant insight about the object oriented characteristics of the projects. This is very long process. This project will be applicable in well compiled java program and it should have valid comments to measure the cohesion .The objective of this project is to create a common platform for all design quality metrics to make the application software more scalable and maintainable. In this paper, we include the system design of my work, solution steps according to the problem definition statement. Finally conclude by testing C++ and Java project.

***Index Terms*— LCOM, MOOD, Object-Oriented, System metrics.**

I. INTRODUCTION

Software modularization, Object-Oriented (OO) decomposition in particular, is an approach for improving the organization and comprehension of source code. In order to understand OO software, software engineers need to create a well-connected representation of the classes that make up the system. Each class must be understood individually and, then, relationships among classes as well. One of the goals of the OO analysis and design is to create a system where classes have high cohesion and there is low coupling among them. These class properties facilitate comprehension, testing, reusability, maintainability, etc. Software cohesion can be defined as a measure of the degree to which elements of a module belong together. Cohesion is also regarded from a conceptual point of view. In this view, a cohesive module is a crisp abstraction of a concept or feature from the problem domain, usually described in the requirements or specifications. Such definitions, although very intuitive, are quite vague and make cohesion measurement a difficult task, leaving too much room for interpretation. In OO software systems, cohesion is usually measured at the class level and many different OO cohesion metrics have been proposed which try capturing different aspects of cohesion or reflect a particular interpretation of cohesion. We propose a new measure for class cohesion, named the Conceptual Cohesion of Classes (C3), which captures the conceptual aspects of class cohesion, as it measures how strongly the methods of a class relate to each other conceptually. The conceptual relation between methods is based on the principle of textual coherence [Quality Metrics Tool for Object Oriented Programming IJCTE 2010].

II. TYPES OF LEVEL

A. System metrics level

There are system metrics that can be derived from class metrics with statistics, as relative measures, identifying systems that deviate from the norm. Unusual trends or characteristics of the system under construction can be spotted and corrected. The MOOD (Metrics for Object oriented Design) set of metrics of Abreu which operate at System level. They refers to a basic structural mechanism of the OO paradigm as encapsulation, inheritance, polymorphism and message-passing [A Study of Software Metrics IJCEM January 2011]. The set of metrics are:

B. Encapsulation Level

1 Method Hiding Factor (MHF)

Definition MHF is defined as the ratio of the sum of the invisibilities of all methods defined in all classes to the total number of methods defined in the system under consideration. The invisibility of a method is the

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 3, May 2013
percentage of the total classes from which this method is not visible. In other words, MHF is the ratio of hidden methods –protected or private methods- to total methods. Intent MHF is proposed to measure the encapsulation (the relative amount of information hidden).It have found that as MHF increases, the defect density and the effort spent to fix is expected to decrease. Inherited methods are not considered.

2 Attribute Hiding Factor (AHF).

Definition AHF is defined as the ratio of the sum of the invisibilities of all attributes defined in all classes to the total number of attributes defined in the system under consideration. In other words, AHF is the ratio of hidden attributes –protected or private- to total attributes. Intent AHF is also proposed to measure the encapsulation (amount of information hidden) [Metrics Identification for Measuring Object Oriented Software Quality IJSCE November 2012].

C. Coupling Level

Coupling is the use of methods or attributes defined in a class that are used by another class. Classes interact with other classes to form a subsystem/system and this interaction can indicate the complexity of the design. Representative metrics of this set are:

1 Coupling Between Objects (CBO)

Definition CBO for a class is a count of the number of other classes to which it is coupled. Coupling between two classes is said to occur when one class uses methods or variables of another class. COB is measured by counting the number of distinct non-inheritance related class hierarchies on which a class depends. C&K view the more coupling in class, the more difficult to reuse. Classes should be as independent from other classes as possible in order to promoting reuse. Classes always have interdependencies, but if they are large, to understand the design can be a nightmare and therefore, the reuse can be more expensive than rewrite. Classes with excessive coupling makes more difficult the maintenance and the more rigorous the testing needs to be done. Strong coupling complicates a system since a class is harder to understand, change or correct by itself if it is interrelated with other classes. A system with low-coupling reduces complexity, improves modularity and promotes encapsulation.

D. Class Level

These metrics identify characteristics within the class, highlighting different aspects of the class abstractions and help identify where remedial action may be taken. Representative metrics of this set are:

1 Response For a Class (RFC).

Definition RFC is the count of the set of all methods that can be invoked in response to a message to an object of the class or by some method in the class. This includes all methods accessible within the class hierarchy. RFC counts the occurrences of calls to other classes from a particular class. The response set of a class is a set of methods that can potentially be executed in response to a message received by an object of that class. The cardinality of this set is a measure of the attributes of objects in the class. Since it specifically includes methods called from outside the class, it is also a measure of the potential communication between the class and other classes. Intent According to C&K, RFC is a measure of the complexity of a class through the number of methods and the amount of communication with other classes. RFC is an indicator of the effort of testing and debugging. The larger RFC, the greater the complexity because a large number of methods can be invoked in response to a message. Therefore, more allocation of testing time is required since it requires a greater level of understanding. There is ambiguity in definition of this metric forcing the user to interpret [Empirical Analysis of CK Metrics for Object-Oriented Design Complexity: Implications for Software Defects IEEE APRIL 2003].

2 Lack of Cohesion in Methods (LCOM)

Definition LCOM measures the extent to which methods reference the classes instance data. Example: Intent LCOM is a quality measure for the cohesiveness of a class by measuring the number of common attributes used by different methods. LCOM measure indicates the fitness of each class abstraction. A high value of LCOM implies lack of cohesion, namely, low similarity and the class may be a composition of unrelated objects. Methods within a class is highly desirable cohesive, since classes cannot be divide and it increase the degree of encapsulation whereas low cohesiveness increase complexity, thereby likelihood of errors occurring during development process is increased. It arises two problems with this metric. First, two classes can have a LCOM=0 while one has more common variables than other. And second, there are not guidelines about interpreting the values. Therefore, Henderson-Sellers have suggested a new LCOM measure. A value near zero for this metric indicates very high cohesion, where most methods refer to most instance variables [An Effective Analysis of Object Oriented Metrics in Software Quality IJCTIS December 2011].

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 3, May 2013

III.ADVANTAGE OF SOFTWARE MATRICES

- In Comparative study of various design methodology of software systems.
- For analysis, comparison and critical study of various programming language with respect to their characteristics.
- In comparing and evaluating capabilities and productivity of people involved in software development.
- In the preparation of software quality specifications.
- In the verification of compliance of software systems requirements and specifications.
- In making inference about the effort to be put in the design and development of the software systems.
- In getting an idea about the complexity of the code.
- In taking decisions regarding further division of complex module is to be done or not.
- In providing guidance to resource manager for their proper utilization.
- In comparison and making design tradeoffs between software development and maintenance cost.

IV.LIMITATIONOF SOFTWARE MATRICES

- The application of software metrics is not always easy and in some cases it is difficult and costly.
- The verification and justification of software metrics is based on historical/empirical data whose validity is difficult to verify.
- These are useful for managing the software products but not for evaluating performance of the technical staff.
- The definition and derivation of Software metrics is generally based on assuming which are not standardized and may depend upon tools available and working environment.
- Most of the predictive models rely on estimates of certain variables which are often not known exactly.
- Most of the software development models are probabilistic and empirical.

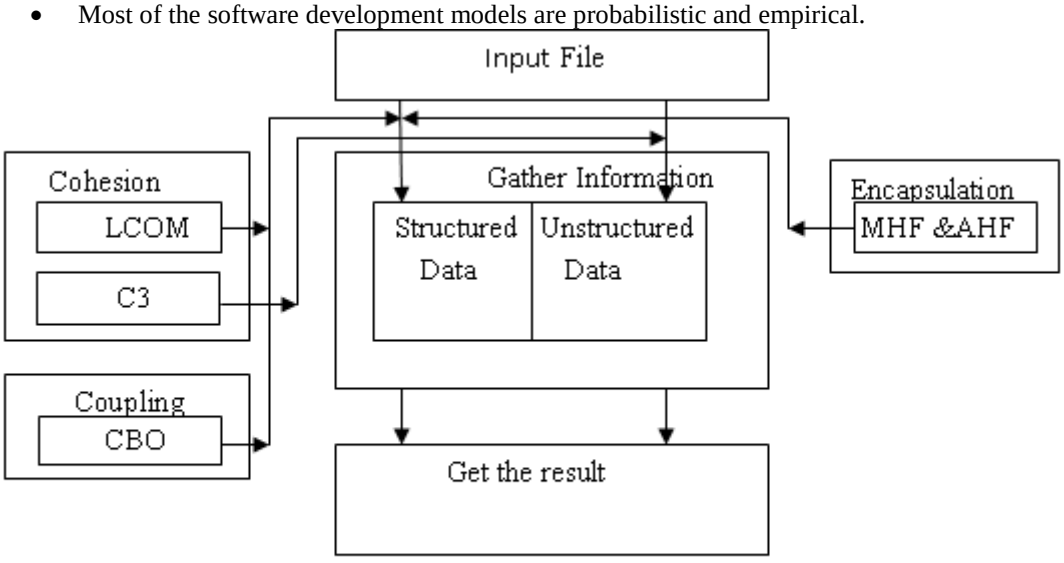

Fig.1 : Block diagram of proposed system.

V.SYSTEM STEPS

Figure 1 show the block diagram of the proposed system which consists of the input file, cohesion, coupling, encapsulation, display result and gathering of information (structured and unstructured form). The working of the overall system, cohesion, coupling, and encapsulation is as given below.

A. Steps for solving the problem

1. Input any Java or CPP project file.
2. Gather the information in the form of structured and unstructured data.
3. Apply the Cohesion or coupling or encapsulation methods (Algorithm).
4. Get the appropriate result of the system.

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 3, May 2013

B. Steps for Cohesion

1. Input any java or CPP project file.
2. Gather the information in the form of structured and unstructured data.
3. Apply LCOM5 procedure on structured data and C3 procedure on unstructured data.
4. Get the appropriate result of cohesion.

C. Steps for Coupling

1. Input any java or Cpp project file.
2. Gather the information in the form of structured and unstructured data.
3. Apply the CBO procedure on structured data.
4. Get the appropriate result of coupling.

D.Steps for Encapsulation

1. Input source file of Java or CPP project.
2. Compile the file by using MHF and AHF methods.
3. Display the fields and methods used in the given project.
4. Find the metrics.
5. Get the appropriate result of encapsulation.

VI CONCLUSION

A metrics program is based on the goals of an organization will help to communicate, measuring progress, and eventually attain those goals. People will work to accomplish their important belief. Well-designed metrics with documented objectives can help an organization in obtaining the information which needs to continuously improve its software products, processes, and services while maintain a focus on important. The valuable aids includes practical, systematic, start-to-finish of selection, design and implement software metrics. The Assessment tool is used to assess the quality of the java application. The tool can be enhanced to assess other Object Oriented languages such as C++, C sharp, etc., The enhanced assessment tool will be helpful in assessing the quality in advance to develop awareness for quality issues such as reliability, testability and maintainability. Metrics are important to judge the complexities and reliability issues of any object oriented system. These will certainly helpful in reducing the cost and effort incurred in the design of any object oriented system and one can determine the level of its reliability and robustness. Object-oriented systems classes in different programming languages contain identifiers and comments which reflect concepts from the domain of the software system. This information can be used to measure the cohesion of software to extract this information for cohesion measurement; this paper defines the conceptual cohesion of classes, which captures new and complementary dimensions of cohesion compared to a host of existing structural metrics. Principal component analysis of measurement results on three open source software systems statistically supports this fact. In addition, the combination of structural and conceptual cohesion metrics defines better models for the prediction of faults in classes than combinations of structural metrics alone. and a coherent internal description. Latent Semantic "Indexing can be used in similar manner to measuring the coherence of natural languages.

ACKNOWLEDGMENT

"No duty is more urgent than that of returning thanks……"

My Thanks to…….

- To my guide Mrs. N .A .NEMADE for inspiration, advice and encouragement.
- To all my teachers for valuable ideas and supervising my theses at S.S.G.B.C.O.E.T., Bhusawal.
- To Mum and Dad and all the family for giving me the best start in my life and for giving me the opportunities that have got me to where I am.
- To my H.O.D.

REFERENCES

[1] Aman Kumar Sharma, Arvind Kalia, Hardeep Singh, "Metrics Identification for Measuring Object Oriented Software Quality", International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-5, November 2012, pp 255-258.

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 3, May 2013

[2] Dr. P. PonMuthuramalingam and M. Yamunadevi, "An Effective Analysis of Object Oriented Metrics in Software Quality", International Journal of Computing Technology and Information Security, Vol.1, No.2, pp.43-47, December, 2011. ISSN: 2231-1998 © 2011, pp 44-47.

[3] Gurudev Singh, Dilbag Singh, Vikram Singh, "A Study of Software Metrics", IJCEM International Journal of Computational Engineering & Management, Vol. 11, January 2011 ISSN (Online): 2230-7893, pp 22-27.

[4] Mythili Thirugnanam* and Swathi.J.N. ," Quality Metrics Tool for Object Oriented Programming", International Journal of Computer Theory and Engineering, Vol. 2, No. 5, October, 2010, ISBN No: 1793-8201, pp 712-717.

[5] Ramanath Subramanyam and M.S. Krishnan, "Empirical Analysis of CK Metrics for Object-Oriented Design Complexity: Implications for Software Defects", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 29, NO. 4, APRIL 2003, ISSN: 0098-5589/03,pp 297-310.