

# Analysis of Cohesion and Coupling Metrics for Object Oriented System

Annushri Sethi<sup>1</sup>, Prof. Narendra Pal Singh Rathore<sup>2</sup>

<sup>1</sup>Student, Department of Computer Science, TCET, Indore, India

<sup>2</sup>Professor, TRUBA College of Engineering and Technology, Indore-Rao Bypass Road, India

## ABSTRACT

In software engineering modularization is way to divide software project in to multiple independent and discrete modules. After complete module conquers make as software. Means modules design use divide and conquer rule. Cohesion and Coupling measure the quality of design of modules and interaction between modules. In this paper studies have been done to identify complexity between inheritance and interface by applying the cohesion and coupling metrics. Two program of c# implementing one with inheritance and other with interface are taken and measurement is done. The metrics value obtained is compared to prove which concept is good and beneficial for c# developer to use.

**Keywords:** Cohesion, Coupling, Modularization, Inheritance, Interface.

## I. INTRODUCTION

When a software system program is modularized, its tasks area unit divided into many modules supported some characteristics. As we know, modules area unit set of directions place along so as to attain some tasks. They are although, thought-about as single entity however could confer with one another to figure along. There are a unit measures by that the standard of a style of modules and their interaction among them is often measured. These measures area unit known as coupling and cohesion.

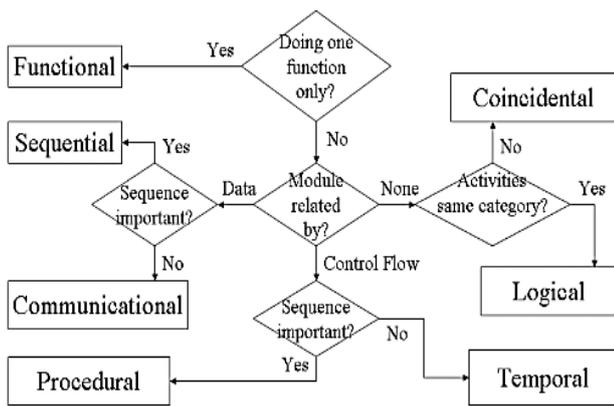
Cohesion is a very important attribute similar to the standard of the abstraction captured by the category into account. Sensible abstractions generally exhibit high cohesion. Cohesion refers to the degree of the relationships among the members during a category. A category is cohesive once its members area unit extremely related .A extremely cohesive module is one whose components have a detailed relationship among them so as to supply the only real practicality of the module. On the contrary, a coffee cohesive module has some components that have very little relation with others that indicates that the module appears to supply many unrelated functionalities.

It is extensively accepted that the upper the cohesion of a module is, the simpler the module is to develop, maintain, and reuse, and therefore the less fault prone

it's. it's a very important object-oriented software system quality attribute. The degree of sophistication cohesion offers a sign for the standard of sophistication style. In object-oriented paradigm, the category cohesion are often thought because the mensuration of connection among the members of sophistication. Metrics area unit suggests that for attaining a lot of correct estimations of project milestones, and developing a computer code that contains stripped-down faults. it's wide received that object bound development needs a unique manner of thinking than ancient structured development and software system comes area unit shifting to object bound code. Object bound metrics to measure properties of object bound Code.

### 1.1 Cohesion:

Cohesion may be a live that defines the degree of intra-dependability inside components of a module. The bigger the cohesion, the higher is that the program style.in below figure 1 describe how to determine cohesion module.

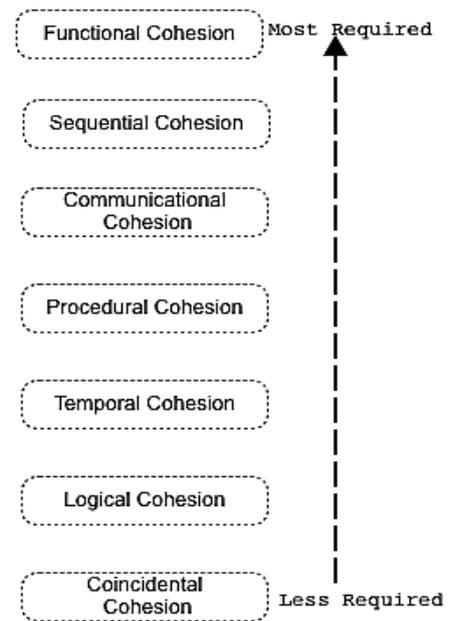


**Figure 1 : Determine Cohesion Modules**

Their area unit seven kinds of cohesion, namely –

1. **Coincidental cohesion-** its unplanned and random cohesion, which could be the results of breaking the program into smaller modules for the sake of modularization. as a result of it's unplanned, it's going to serve confusion to the programmers and is usually not accepted.
2. **Logical cohesion**-once logically classified components area unit place along into a module, it's known as logical cohesion.
3. **Temporal Cohesion** - Once components of module area unit organized such they're processed at the same purpose in time, it's known as temporal cohesion.
4. **Procedural cohesion** - Once components of module area unit classified along, that area unit dead consecutive so as to perform a task, it's known as procedural cohesion.
5. **Communicative cohesion** - Once components of module area unit classified along, that area unit dead consecutive and work on same information (information), it's known as communicative cohesion.
6. **Successive cohesion** - once components of module area unit classified as a result of the output of one part is input to a different and then on, it's known as successive cohesion.
7. **Practical cohesion** - it's thought-about to be the best degree of cohesion, and it's extremely expected. Components of module in practical cohesion area unit classified as a result of all of them contributes to one well defined perform. It can even be reused.

### Types Of Cohesion

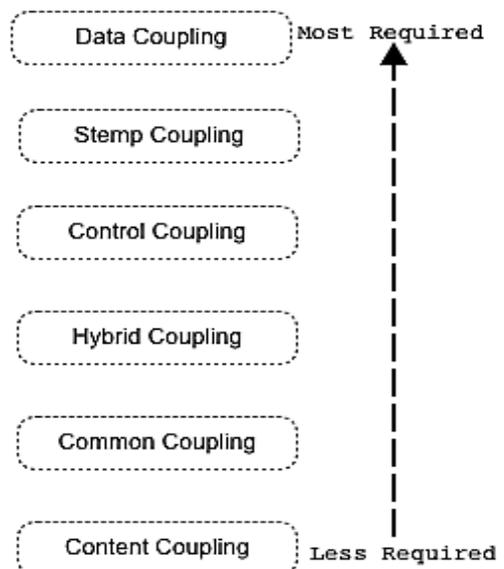


**Figure 2 : Type of Cohesion and its importance**

### 1.2 Coupling:

Coupling may be a live that defines the amount of inter-dependability among modules of a program. It tells at what level the modules interfere and act with one another. The lower the coupling, the higher the program.

### Types Of Coupling



**Figure 3 : Type of Coupling and its importance**

Their area unit five levels of coupling, namely –

1. **Content coupling** - once a module will directly access or modify or confer with the content of another module, it's known as content level coupling.
2. **Common coupling**- once multiple modules have scan and write access to some international information, it's known as common or international coupling.
3. **Control coupling**- Two modules area unit known as control-coupled if one among them decides the perform of the opposite module or changes its flow of execution.
4. **Stamp coupling**- once multiple modules share common system and work on completely different a part of it, it's known as stamp coupling.
5. **Information coupling**- Data coupling is once two modules act with one another by suggests that of passing information (as parameter). If a module passes system as parameter, then the receiving module ought to use all its elements.

## II. METHODS AND MATERIAL

### A. Literature Review

The concept of an interface in object-oriented programming is quite old. Software engineering has been using interfaces for more than 25 years. Many metrics are available to measure class, method, inheritance, polymorphism and system level. There is no significant work on the code of human computer interfaces. In literature, relatively little information has been published on interface metrics. Those metrics provide only little information about the quality and usability of the interfaces. Sue of interface leads to the high cohesion and make the code more reusable.

In Year -2010, V. Krishnapriya and Dr. K. Ramar have measured interface concept by using coupling metrics on design based and have proved interface is more effective in use then inheritance to increase reusability of a code in object oriented programming [1]. In this paper, measurement of inheritance and interface is calculated using cohesion metrics using a example and prove the usage of interface increased the reusability James m. Bieman and byungkyookang published a Paper on

Cohesion and reuse in object oriented system explaining the TCC and LCC on C++ Program [4].

There are many metrics to find class cohesion but no standard metric or definition has been generally accepted, out of available [Fenton &Pfleeger 1998], [Counsell et al. 2002] and [Etzkorn et al. 2004]. A reasonable metric to measure class cohesion should give an insight to the relatedness among the methods of a class while considering the impacts of inheritance paradigm on local class cohesion.

Like class cohesion, there is no standard metric or definition for class coupling [Fenton &Pfleeger 1998]. However, In OO design class coupling is a measurement of class dependence on other classes. We will attempt to measure a class coupling on the basis of UML relationships.

Metrics for measuring class cohesion and class coupling are supposed to share same input data for their respective measurements. By the same set of input data we mean class member attributes, member methods, and usage of attributes by the methods. We will attempt to find mutual relationships between class cohesion and class coupling metrics by analyzing the results of experiment statistically.

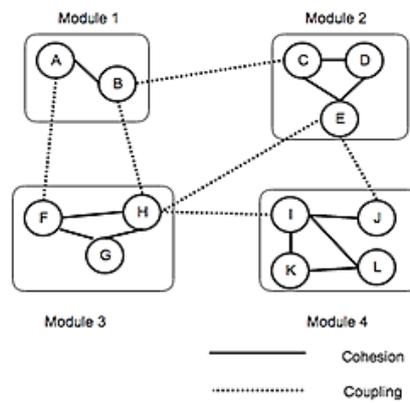
Literature on the subject of the software evolution clearly introduces the erosive trends in the software architecture while meeting the changes imposed by the

software evolution. In this thesis, we will attempt to identify such erosive trends with the help of class cohesion and coupling metrics. Based on the literature review, we suppose that both class cohesion and coupling should follow deteriorating trends while evolution in the software architecture.

### B. Cohesion and Coupling Matrices In Object Oriented System

Cohesion is the degree to which methods within a class are related to one another and work together to provide well-bounded behavior. Effective object oriented designs maximize cohesion because cohesion promotes encapsulation. Coupling is a measure of the strength of association established by a connection from one entity to another. Classes are coupled when a message is

passed between objects; when methods declared in one class use methods or attributes of another class. Inheritance is the hierarchical relationship among classes that enables programmers to reuse previously defined objects including variables and operators.



**Figure 3:** Relationship between Cohesion and Coupling

### III. RESULTS AND DISCUSSION

Author Name / Title	Journal	Strength	Weakness
N. Rajkumar1 "Measuring Cohesion And Coupling In Object Oriented System Using Java Reflection"	ARN Journal of Engineering and Applied Sciences	This paper proposes a set of new measures to find coupling and cohesion in a developmental system using Java reflection components to assess the usability. It will predict the fault in an object-oriented system.	Next version will calculate coupling and cohesion metrics for UML representations
Martin Hitz "Measuring Coupling and Cohesion In Object-Oriented Systems "	<a href="http://www.isys.uni-klu.ac.at/PDF/1995-0043-MHBM.pdf">http://www.isys.uni-klu.ac.at/PDF/1995-0043-MHBM.pdf</a>	This distinction refers to dynamic dependencies between objects on one hand and static dependencies between implementations.	important aspects of software quality at run-time and during the maintenance phase, respectively.
Aaron B. Binkley "A classical view of object-oriented cohesion and coupling"	<a href="http://citeseerx.ist.ps u.edu/viewdoc/download?doi=10.1.1.99.4519&amp;rep=rep1&amp;type=pdf">http://citeseerx.ist.ps u.edu/viewdoc/download?doi=10.1.1.99.4519&amp;rep=rep1&amp;type=pdf</a>	Evidence is starting to accumulate that this paradigm is indeed as effective as has been suggested	Most of the metrics used in conjunction with the object-oriented paradigm are, in fact, classical metrics.
Mr. KailashPatidar "Coupling and Cohesion Measures in Object Oriented Programming"	International Journal of Advanced Research in Computer Science and Software Engineering	A large numbers of metrics have been built and proposed for measuring properties of object-oriented software such as size, inheritance, cohesion and coupling. The coupling is an important aspect in the evaluation of reusability and maintainability of components or services.	To achieve consistent and satisfying results, empirical data obtained from reallife software engineering projects
Shweta Sharma "A review of Coupling and Cohesion metricsin Object Oriented Environment"	International Journal of Computer Science & Engineering Technology (IJCSSET)	This paper focuses on two very significant factors of complexity measurement of software which are coupling and cohesion. An extensive study of approximately all types of coupling and cohesion metrics has been reported in this paper	Very little work has been done in areas of dynamic coupling and cohesion metrics and need further more investigations

### Measure Cohesion for class using Inheritances and Interfaces

Inheritance is one of the fundamental concepts of Object Orientated programming, in which a class "gains" all of the attributes and operations of the class it inherits from, and can override some of them, as well as add more attributes and operations of its own. In Object Oriented Programming, inheritance is a way to compartmentalize and reuse code by creating collections of attributes, thing and behaviors called objects that can be based on previously created objects.

Lack of Cohesion in Methods (LCOM) as the number of pairs of methods operating on disjoint sets of instance variables, reduced by the number of method pairs acting on at least one shared instance variable.

### IV. CONCLUSION

The purpose of this thesis is to finding the approach and way to identify complexity between inheritance and interface programming through cohesion metrics in object oriented programs. Metrics measure certain properties of software system by mapping them to numbers (or to other symbols) according to well-defined, objective measurement rules. Code Metrics are measurements of the static state of the project's Code and also used for assessing the size and in some cases the quality and complexity of software. Analysis and maintenance of Object-Oriented (OO) software is expensive and difficult. Thus, measuring the relationships has become a prerequisite to develop efficient techniques for analysis and maintenance. Various cohesion metrics have been proposed and used in past empirical investigations; however none of these have taken the run-time properties of a program into account. As program behavior is a function of its

operational environment as well as the complexity of the source code, static metrics may fail to quantify all the underlying dimensions of coupling and cohesion. In our future work, we can apply various other cohesion metrics to identify better complexity between inheritance and interface programming.

[10] Pradeep Kumar Bhatia, Rajbeer Mann, “ An Approach to Measure Software Reusability of OO Design”, Proceedings of 2nd International Conference on Challenges & Opportunities in InformationTechnology(COIT-2008),RIMT-IET,MandiGobissndgarh, March 29, 2008.

## V. REFERENCES

- [1] V. Krishnapriya, K. Ramar, "Exploring the Difference Between Object Oriented Class Inheritance and Interfaces Using Coupling Measures," ace, pp.207-211, 2010 International Conference on Advances in Computer Engineering, 2010
- [2] K.K.Aggarwal, Yogesh Singh, Arvinder Kaur, RuchikaMalhotra. "Empirical Study of Object-Oriented Metrics",2006
- [3] Martin Hitz, BehzadMontazeri."Measuring Coupling and Cohesion.In Object-Oriented Systems" in Angewandte Informatik (1995)
- [4] James M. Bieman and Byung-kyookang."Cohesion and Reuse in Object Oriented System" Department of Computer Science, Colorado State University Fort Collins,Colorado,1995
- [5] Shyam R. Chidamberand Chris F. Kemerer" A Metrics Suite For object Oriented Design" IEEE Transactions on software Engineering, Vol. 20, No. 6, June 1994
- [6] KrishnaprasadThirunarayan." Inheritance in Programming Languages" Department of Computer Science and Engineering ,Wright State University ,Dayton, OH-45435
- [7] ArtiChhikara Maharaja Agrasen College, Delhi, India. R.S.Chhillar "Applying Object Oriented Metrics to C#(C Sharp) Programs"Deptt. Of Computer Sc.And Applications, Rohtak, India.SujataKhatriDeenDyalUpadhyaya College, Delhi, India(2011)
- [8] Christopher L. Brooks, Chrislopher G. Buell, "A Tool for Automatically Gathering Object-Oriented Metrics", IEEE, 1994
- [9] Friedrich Stiemann, Philip Mayer and Andreas Meibner, "DecouplingClasses with Inferred Interfaces", Proceedings of the 2006 ACM Symposium on Applied Computing, P.No:1404 – 1408.