Volume 3, Issue 3, March 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Coupling and Cohesion Measures in Object Oriented Programming

Mr. Kailash Patidar[1] Prof.RavindraKumar Gupta[2] Prof.Gajendra Singh Chandel[3]
1. *M.Tech Scholar Dept. of Computer Science, SSSIST SEHORE,*
2. *PG Coordinator Computer Science, Dept., SSSIST SEHORE,*
3. *Head of Dept. Computer Science, Dept. SSSIST SEHORE,*

Abstract— Many coupling and cohesion measures have been introduced in various surveys to identify and measure the design complexity of object oriented systems.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. The coupling metrics find complexity between inheritance and interface programming. In this paper presents measurements of object after that find the coupling and cohesion between objects, measure the association between numbers of classes, check thedirect dependencies, indirect dependencies, IO dependencies,number of out and in metrics in object oriented programming. A measurement is performing between inheritance and interface programs. This paper also proposes a model to measure the degree of coupling and cohesion due to these dependencies.

Keywords—Design Metrics,Class Cohesion Metrics, Cohesion among Methods of a Class, Class Coupling Metrics, Direct dependencies, indirect dependencies.

I. Introduction

Software engineering aims at development of high-quality software and tools to promote qualitysoftware that is stable and easy to maintain and use[3]. In order to assess and improve softwarequality during the development process, developers and managers use, among othermeans, 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. In the OO paradigm of software development cohesion means extent to which the public methods of class perform the same task [Bieman& Kang 1995], whereas coupling means the degree of dependence of a class on other classes in the system. If we look into the existing OO metrics for cohesion, coupling, size and reuse, we will notice that all of these metrics use the shared input data [1].

According to object oriented programming, the class provides encapsulation and abstraction and the interface provides abstraction and cannot inherit from one class but can implement multiple interfaces. The above said differences are minor and they are very similar in structure, complexity, readability and maintainability of source code [2]. Here, the difference in usage of class inheritance and interface concepts are measured for java programs by coupling metrics. Density of source code directly relates to cost and quality. For measuring complexities, we have cohesion and coupling models. The coupling models presented in the literature show many possible interactions that can occur between objects in the software systems and offer metrics to measure complexity. Software engineering best practices promote low coupling between components in order to decrease direct or indirect dependencies and facilitate evolution. This paper presents a comparison between object oriented interfaces and inheritance class. This paper also proposes a model to measure the degree of coupling and cohesion due to these dependencies.

II. Literature Survey

The literature survey, that we have mainly covered can be categorized into three areas namely, cohesion, coupling and software evolution. Object-oriented measurement has become an increasingly popular research area. Metrics are powerful support tools in software development and maintenance. They are used to assess software quality, to estimate complexity, cost and effort, to control and improve processes. The metrics that are important to calculate reusability are related to inheritance, cohesion and coupling.

Traditional Metrics

McCabe Cyclomatic Complexity (CC):Cyclomatic complexity is a measure of a module control flow complexity based on graph theory [4]. Cyclomatic complexity of a module uses control structures to create a control flow matrix, which in turn is used to generate a connected graph. The graph represents the control paths through the module. The complexity of the graph is the complexity of the module [4], [5]. Fundamentally, the CC of a module is roughly equivalent to the number of decision points and is a measure of the minimum number of test cases that would be required to cover all

execution paths. A high Cyclomatic complexity indicates that the code may be of low quality and difficult to test and maintain.

Source Lines of Code (SLOC):The SLOC metric measuresthe number of physical lines of active code, that is, noblank or commented lines code [6]. Counting the SLOC isone of the earliest and easiest approaches to measuringcomplexity. Ingeneral the higher the SLOC in a module the lessunderstandable and maintainable the module is.

Comment Percentage (CP):The CP metric is defined asthe number of commented lines of code divided by thenumber of non-blank lines of code. Usually 20% indicatesadequate commenting for C++ [7]. A high CP valuefacilitates in maintaining a system.

III . Object Oriented Programming And Metrics

Object oriented programming and design are veryimportant in today's environment. It provides generalizedsolutions for many problems in addition to many benefitslike reusability, decomposition of problems into small easilyunderstandable objects and also helps to performmodifications in future and to do functional extensions in already built systems [8]. 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. Object oriented metrics evaluate the object oriented concept: methods, classes, cohesion, coupling and inheritance. Object oriented metrics focus on the internal object structure. Object oriented metrics measure externally the interaction among the entities. Object oriented metrics measures the efficiency of an algorithm. Object oriented metrics are primarily applied to the concepts of classes, coupling, and inheritance. Preceding each metric, a brief description of the object oriented structure is given. A class is a template from which objects can be created. This set of objects shares a common structure and a common behavior manifested by the set of methods. A method is an operation upon an object and is defined in the class declaration. A message is a request that an object makes of another object to perform an operation. The operation executed as a result of receiving a message is called a method. 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.

IV. Class 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. In classical inheritance where objects are defined by classes, classes can inherit property other classes. The new classes, known as subclasses or derived classesinherit attributes and behavior of the pre-existing classes, which are referred to as super classes, ancestor classes or base classes. The inheritance, relationships of classes gives rise to a hierarchy. The inheritance concept was invented in 1967 for Simula [9]. Interfaces allow only method definitions and constant attributes. Methods defined in the interfaces cannot have implementations in the interface. Classes can implement. The interface by providing bodies for the methodsdefined in the interface. An interface is a contract between a client class and a server class [9]. It helps to decouple the client from the server. Any intended change on the methods defined in the interface will impact both the client and server classes. Possible changes are asfollows:

1) Changing the name of a method,
2) Changing the signature of a method
3) Changing the return type of a method

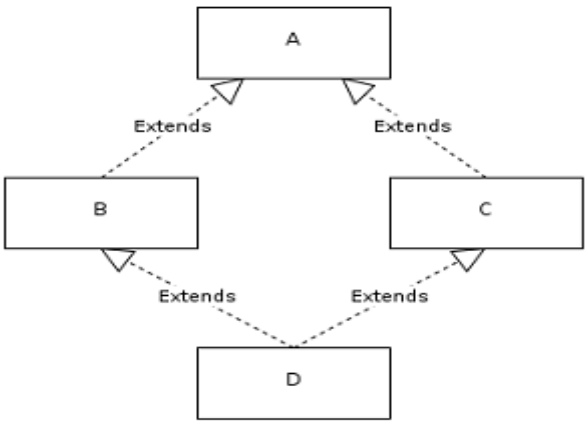

Fig.1 **Diamond problem of multiple inheritances**

The java programming language does not permit multiple inheritance but interfaces provide an alternative. In java a class can inherit from only one class but it can implement more than one interface. Therefore, objects can have multiple types: the type of their own class and the types of all the interfaces that they implement. This means that if a variable is declared

to be the type of an interface, its value can reference any object that is instantiated from any class that implements the interface.

V. Coupling Measures

Coupling refers to the degree of direct and indirect dependencies between parts of the design. To measure coupling in class diagrams there are three types of metrics [10]. In this paper two CK metric is added to measure coupling performance. A measure of coupling is more useful to determine the complexity. The higher the inter object coupling, the more rigorous the testing needs to be. In this paper, three Genero metrics are used to validate the proposed approach check the performance.

Number of children metrics: -Number of children metric defines number of sub-classes subordinated to a class in the class hierarchy. This metric measures how many sub-classes are going to inherit the methods of the parent class. Number of children metric relates to the notion of scope of properties. If Number of children metric grows it means reuse increases. On the other hand, as Number of children metric increases, the amount of testing will also increase because more children in a class indicate more responsibility. So, Number of children metric represents the effort required to test the class, reuse and maintain.

Coupling Between Number of Objects:- A class is coupled with another if the methods of one class use the methods or attributes of the other class. An increase of Coupling between Number of Objects indicates the reusability of a class will decrease. Multiple accesses to the same class are counted as one access. Only method calls and variable references are counted.

Number of Dependencies In: -The Number of Dependencies In metric is defined as the number of classes that depend on a given class [11]. This metric is proposed to measure the class complexity due to dependency relationships. The greater the number of classes that depend on a given class, the greater the inter-class dependency and therefore the greater the design complexity of such a class. When the dependencies are reduced the class can function more independently.

Fig.2 Number of Dependencies In

Number of Dependencies Out:- The Number of Dependencies Out metric is defined as the number of classes on which a given class depends. When the metric value is minimum the class can function independently.

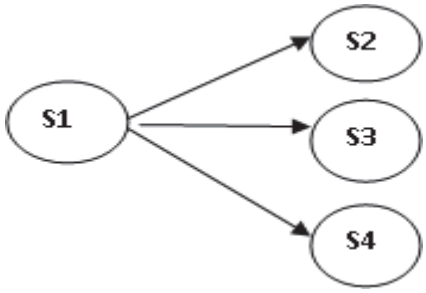

Fig.3 Number of Dependencies Out

Number of Association: - The Number of Association per Class metric is defined as the total number of associations a class has with other classes or with itself. When the number of associations are less the coupling between objects are reduced. This metric was introduced by Brian.

Direct Dependency: - The direct dependency is direct connection between services. This kind of dependency may exist between services directly when a service calls other services or a service is calledby other services [12]. Direct Dependency share the local data without any mediator.

Indirect Dependency: - The indirect dependency between services may occur at:

(a)When there is indirect connection or transitive connection between services and

(b) When there is a sharing of global data between services.

VI. Metric For Class Cohesion

To describe the cohesion in term of OO paradigm of software development, we first need to understand how the construct of a class stands in OO paradigm. A class may be inherited from zero or more classes in case of Multiple Inheritances in C++ and a class may be derived by zero or more classes. An inherited class is also referred as base class or super class orparent class, whereas derived class is also referred as subclass or child class. Class is composed of its

member. Member stands for member attributes and methods of a class. Members of a class may have different access rules, based on the access rules; members can be public, protected and private. Privatemembers cannot be accessed directly from the outside of class; however pointer to private methods or attributes may cause violation to this rule. Protected and public members of inherited class become the part of derived classes and hence such members can be accessed by derived class. In C++, protected members of the class can also be accessed by the friend classes. However, public members of the class can be accessed by all other classes in the system. Good software design dictates that types and methods should have highcohesion and low coupling. High coupling indicates a designthat is difficult to reuse and maintain because of its manyinterdependencies on other types.

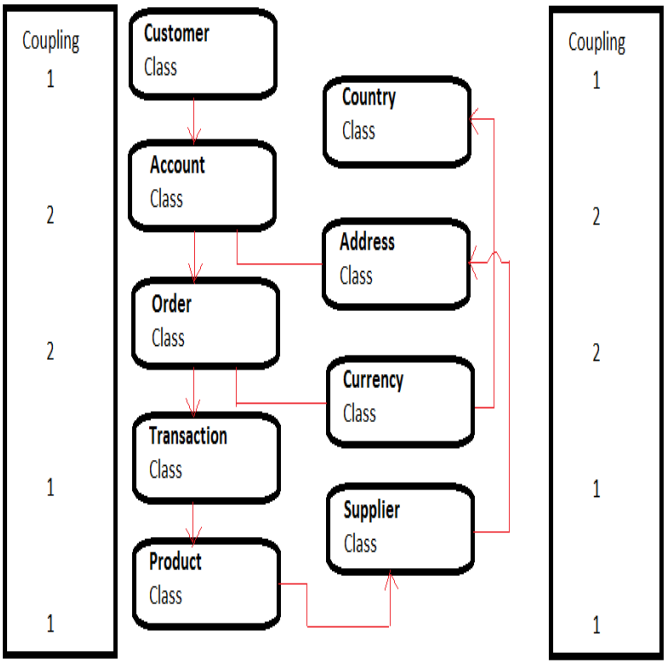

Fig4. Metric for class cohesion

VII. Proposed Approach

Software metrics can be gathered at different levels. By measuring metrics the software process and its products can quantitatively assess the merit of process improvements. Interfaces in java will allow only method definitions and constant attributes. Methods defined in the interfaces do not have implementations in the interface. Classes in java implement the interfaces. Finding commonality among classes makes it more effective for object oriented programming. So, this paper expresses commonality and differences between the usages of object oriented class inheritance and interfaces.

Goal: - Exploring the difference between class inheritance and interface in java programming through coupling metrics and cohesion metrics.

Hypothesis: -Five object oriented metrics are used for coupling and cohesion measures in object oriented class, inheritance and interface programs.

1. Two java programs are used with inheritance concept in this paper.
2. These programs are introduced with maximum possible interface.
3. All five metrics are applied to both inheritance and interface programs.
4. The results are compared between inheritance and interface coupling measures then measure the degree of coupling and cohesion due to these dependencies.

In this paper we proposed a novel Class, Object and Inheritance based Coupling Measure to find better OOP Paradigm using JAVA programming language. By this approach we find the better OOP paradigm. Our Algorithm consist of four phases

1) Authentication
2) Select two Object Oriented Programming Files
3) Count no of Classes, Object and Inheritance
4)Based on the analysis provided in the database we deduce that which programming approach is better in the current situation.

Our Algorithm implemented in future and shows the expected outcomes in near future.

VIII. Conclusion

This paper presents an idea on how to reduce coupling in object oriented programming. It is helpful for the developers to check which concept is best between inheritance and interface. Despite of the very interesting research work and studies

on coupling measures, there is still a little understanding of the motivation and empirical hypotheses behind many of the measures. It is reported that relating the measures is a difficult task in most of the cases and especially to conclude for which applications they can be used. Our Algorithm consist of four phases 1) Authentication 2) Select two Object Oriented Programming Files 3) Count no of Classes, Object and Inheritance 4)Based on the analysis provided in the database we deduce that which programming approach is better in the current situation. Our Algorithm implemented in future and shows the expected outcomes in near future.

References

[1] *James M. Bieman and Byung-Kyooh Kang "Cohesion and reuse in an object-oriented system", ACM Press New York, NY, USA, Pages: 259 – 262, 1995.*

[2] *Mathew Cochran,"Coding Better: Using Classes VsInterfaces",January 18th, 2009.*

[3] *V. Krishnapriya, Dr. K. Ramar, "Exploring the Difference between Object Oriented Class Inheritance and Interfaces Using Coupling Measures", 2010 International Conference on Advances in Computer Engineering, 978-0-7695-4058-0/10 $26.00 © 2010 IEEE.*

[4] *McCabe and Associates, Using McCabe QA 7.0, 1999, 9861Broken Land Parkway 4th Floor Columbia, MD 21046.*

[5] *McCabe, T. J., "A Complexity Measure", IEEE Transactions on Software Engineering, SE 2(4), pages 308- 320, December 1976.*

[6] *Lorenz, Mark & Kidd Jeff, Object-Oriented Software Metrics, Prentic, Hall, 1994.*

[7] *Rosenberg, L., and Hyatt, L., "Software Quality Metrics for Object- Oriented System Environments", Software assurance Technology Center, Technical Report SATC-TR-95-1001,NASA Goddard Space Flight Center, Greenbelt, Maryland 20771.*

[8] *Rene Santaolaya Salgado, Olivia G. Fragosco Diaz, Manuel A. Valdes Marrero, Issac M. Vaseuqz Mendez and Shiela L. Delfin Lara, "Object Oriented Metric to Measure the Degree of Dependency Due to Unused Interfaces", ICCSA 2004, LNCS 3046, P.No: 808- 817,2004 @ Springer, Verlag Berlin Heidelberg.*

[9] *http://en.wikipedia.org/wiki/Object-oriented programming.*

[10] *Marcela Genero, Mario Piattini and Coral Calero," A Survey of Metrics for UML Class Diagrams", in Journal of Object Technology,Vol. 4, No. 9, Nov-Dec 2005.*

[11] *Marcela Genero, Mario Piattini and Coral Calero," A Survey of Metrics for UML Class Diagrams", in Journal of Object Technology, Vol. 4, No. 9, Nov-Dec 2005.*

[12] *T. Karhikeyan, J. Geetha "A Metrics Suite and Fuzzy Model for Measuring Coupling in Service Oriented Architecture"IEEE2012 International Conference on Recent Advances in Computing and Software Systems page no.254-259.*

[13] *SelimKebir, Abdelhak-DjamelSeriai, Sylvain Chardigny and AllaouaChaoui "Quality-Centric Approach for Software Component Identification from Object-Oriented Code"IEEE 2012 Joint Working Conference on Software Architecture & 6th European Conference on Software Architecture page no. 181-190.*

