VOL. 10, NO. 7, APRIL 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

MEASURING COHESION AND COUPLING IN OBJECT ORIENTED SYSTEM USING JAVA REFLECTION

N. Rajkumar[1], C. Viji[1] and S. Duraisamy[2]
[1] Department of Computer Science and Engineering, SVS College of Engineering, Coimbatore, TN, India
[2] Department of Computer Applications, Sri Krishna College of Engineering and Technology, Coimbatore, TN, India

ABSTRACT

Creating an efficient and effective system is a motto of a software engineer. The companies are spending 60% of cost for producing good quality software, software metrics require to measure qualitative in terms of software performance and reliability related characteristics like dependencies, coupling and cohesion etc. 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. Coupling and cohesion metrics are calculated by considering a number of relationships of a class. In this paper these metrics are calculated by using structural parameters like classes, methods and attributes. The structured information embedded in the source code. Classes, methods and attributes are retrieved in a package by using Java reflection. The retrieved information helps to measure the coupling and cohesion.

Keywords: coupling, cohesion, latent semantic indexing, Java reflection, fault prediction, LCOM.

1. INTRODUCTION

Object oriented metrics are useless if they are not mapped to a quality parameter. Many numbers of quality models are recommended to map parameters of object oriented software like Extensibility, Reusability, efforts, manageability and cost. To find the [1] quality of the product we should know more about the interdependencies of parameters of metrics and software quality parameters [3]. Figure: 1 shows the interdependencies of the metrics parameters and software quality parameters [4, 9] by measuring object oriented metrics.

We proposed to measure a traditional metrics parameters coupling and cohesion and it is a structure based, prescribed for object oriented systems. In structured design as well as programming two main attribute coupling and cohesion acts a major role. In every good quality object oriented software the coupling level should be low and cohesion level should be high. The coupling and cohesion measures capture the degree of interaction and relationship among the structural elements in source code, such as classes, methods and attributes of object oriented system. These attributes and methods are embedded in the source code. This information is retrieved by using the Java reflection. That will produce information about the name of the package, classes, methods and attributes that will help to measure coupling and cohesion.

Most approaches to measure quality metrics have automation as one of their goals as it is not practical to manually measure these metrics in large systems. OO analysis and design methods decompose the problem addressed by the software system development into classes in an attempt to control complexity. High cohesion in classes and low coupling between classes are design principles for reducing the system complexity. The most desirable quality metrics are cohesion and coupling. We are trying to measure these in our proposed approach. The source code of a software system contains structured data. Our approach is based on the structured information embedded in the source code. This information captures the domain related information of the software and adds a new layer of programming language semantics and the information extracted from the source code is very useful.

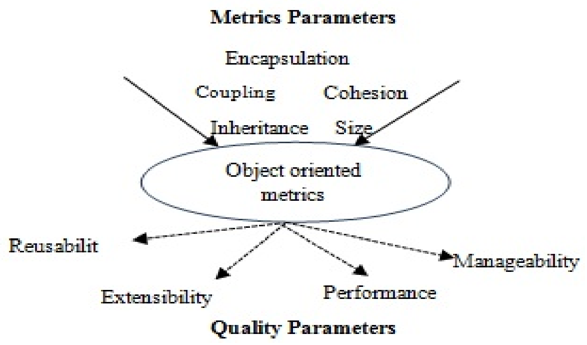

Figure-1. Relation between metrics and quality parameters.

The classes of structural metrics are used to measure coupling and cohesion. There are several methods to measure coupling and cohesion. Here we use LCOM formula foran inverse measure of cohesion. The higher value refers low cohesion .We propose two coupling measurement are WICoup (Weighted Intransitive Coupling) and WTCoup (Weighted Transitive Coupling). This measure will help to predict the fault if a class.

2. RELATED WORK: LATENT SEMANTIC INDEXING

LSI is a corpus-based [2, 16] statistical method for indexing and retrieval method that uses a mathematical technique called singular value decomposition (SVD) to identify patterns in the relationships between the terms and concepts contained in an unstructured collection of text. A key feature of LSI is its ability to extract the conceptual content of a body of text by establishing associations between those terms that occur in similar contexts.

LSI used to retrieve the unstructured information of a class like identifiers and comment lines. This information is created by the developer for future references. It will generate a text document (natural language) for the entire class using the comment. The entire document is indexed, and that give a conceptual relation between the methods and in attribute. That will help to measure conceptual cohesion and coupling.

The unstructured information is created by the developer. If a developer is not aware of this information, there is a defect in the measurement. This work uses Java reflection for structured information retrieval. This information is used to measure coupling and cohesion .That will help to predict the fault of an object oriented system.

3. OVERVIEW OF JAVA REFLECTION

Java Reflection is quite powerful and can be very useful. Reflection is commonly used by programs which require the ability to examine or modify the runtime behaviour of applications running in the Java virtual machine. This is an advanced feature and should be used only by developers who have a strong knowledge of the fundamentals of the language.

Java Reflection makes it possible to investigate classes, interfaces, fields and methods at runtime without knowing class name or package name. At compile time it is also possible to create a new object, invoke methods and get/set field values using reflection. We can access Java classes, methods, attributes, annotations at runtime. Reflection API is a powerful technique to find out program environment and investigate the class itself. Reflection API was included in Java version 1.1. The class of Reflection API is the part in the package of java. lang. reflect and the methods of Reflection API is the part in the package of java. lang. class. It allows the user to get the complete information about classes, constructors, fields and various methods being used, and it also investigates the name of subclass and super class.

Its main functionalities are as follows:

- Get class
- Get all fields and methods in the class includes private ones
- Invoke methods

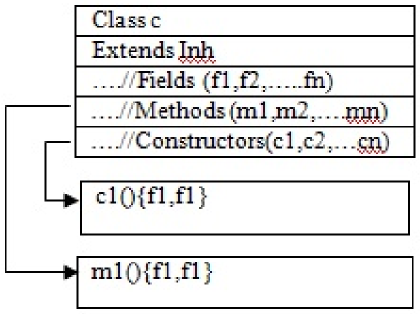

Figure-2. Example class.

For example, Figure-2 shows that a class containing no of fields, methods and constructors. The class can be inherited by anotherclass. Each method and constructor can access fields.

```
//Creating an object
Class obj = Class.forName("class url");
// Find all the fields in the class
Field[] flds = c.getDeclaredFields();
// Find all the methods in the class
Method[] mms = obj.getDeclaredMethods();
// Find all constructor names in the class
Constructor[] cns = obj.getDeclaredConstructors();
// Find superclass
 Class parent = obj.getSuperclass();
```

By using the Java reflection we can retrieve all declared fields, methods and constructors in a class. It is also possible to identify the parent class and grandparent class. Number of methods that access the field can be found by investigating a signature of a field inside the methods and constructors. This will help to measure the metrics.

4. COHESION

The cohesion describes the relationship within a class. **The cohesion** of a single module/component is the degree to which its responsibilities form a meaningful unit, higher cohesion is better for better quality and reliability.

Types of cohesion

1. Coincidental Cohesion: Module elements are not related to the other.
2. Logical Cohesion: Components perform similar activities as selected from outside module.
3. Temporal Cohesion: operations connected only by general time performed.
4. Procedural Cohesion: Components involved in different but consecutive activities, each with different data.Communicational Cohesion : isolated operations except need same data or input
5. Sequential Cohesion: operations on the same data in significantother; output from one function is input to the next.
6. Informational Cohesion: a module executes a number of activities, each with its own entry point, with independent code for each activity, all executed on the same data structure.
7. Functional Cohesion: all componentscontribute to a single.

Reason for high cohesion

1. Easy to understanding modules.
2. Easy to maintain a system, changes in one module do not require changes in related modules.
3. Easy to reusing a module because most applications will need the random set of operations provided by a module.

VOL. 10, NO. 7, APRIL 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

4.1. Cohesion Metrics

Cohesion is the measure to which the methods in a class are related to another and work together to provide a set of actions. LCOM measures the degree ofsimilarity of methods by fieldsin the class.

The Lack of Cohesion in Methods Calculation

LCOM 1: The original object oriented cohesion metric as given by Chidamber and Kemmerer represents an inverse measure for cohesion [6, 8]. Take each pair of methods in the class and determine the set of fields they each access. If they do not share the same field, the count R increases by one. If they share at least one field access, S increases by one. Each pair of methods taken into account:
RESULT = (R > S)? (R - S): 0
A low value indicates the class method having a good relationship between other.

LCOM 2: This is an improved version of LCOM1 as is given by Chidamber and Kemmerer.
Definitions used
mno of methods in class
ano of fields in class
mAno of methods that access a field Ai
Sum (mA): sum of mA over attributes of a class
LCOM2 = 1 - sum (mA)/ (m*a)
If the number of methods or fields in a class is zero, LCOM2 is undefined.

LCOM 3: This is another improvement on LCOM1 and LCOM2 as given by Li & Henry, 1993.
It is defined as follows:
LCOM3 = (m - sum (mA)/a) / (m-1)
The LCOM3 value between 0 and 1. LCOM3>1 indicates low cohesion in the class.

LCOM 4: This is another improvement on LCOM, LCOM2 and LCOM3 as given by Hitz&Montazeri. It is defined as follows:
LCOM4 = (m – [sum (meA) –m]/ac) / (m-1)
LCOM4=1 indicates a cohesive class (good).
LCOM4>=2 bad & split method.
LCOM4=0 there are no methods in a class (bad).

LCOM5: This is another improvement on LCOM4 as is given by Henderson-Sellers, LCOM2, LCOM3 and LCOM4 It is defined as follows:
LCOM5 = {[(1/a) (sum (mA))] – m} / (1 – m)
If the result is 0, it is considered as a perfect cohesion.

5. COUPLING

The coupling describes the relationship between the modules or classes, it indicates the strength of program units. [10, 11] Highly coupled program units are depending on each other, loosely coupled units are independent or almost independent. [5] Lower coupling is better for high quality software.

Types of coupling

1. Content/Pathological Coupling :A module uses/changes data in another
2. Control Coupling : Two modules communicating with a control flag (first tells second what to do via flag)
3. Common/Global-data Coupling : Two modules are interacting via global data
4. Stamp/Data-structureCoupling: Interacting via a data structure passed as a parameter. The data structure holds more info than the receiver needs.
5. Data Coupling: Interacting via parameter passing. The passed parameters are requested by the receiver.
6. No data coupling: independent modules.
7. Subclass Coupling: The relationship between a child and its parent. The child is associatedwith its parent, but the parent isn't associatedwith the child.
8. Temporal coupling: When two actions are communicating with each other and act as a module.

Reasons for low coupling

1. A change in one module usually services a ripple effect of changes in other modules.
2. Assembly of the modules might need more effort and/or time due to the increased inter-module dependency.
3. A specific module might be harder to reuse and/or test because dependent modules must be included

5.1 Coupling metrics

Coupling is defined as Interactions between classes by invoking methods and accessing variables [15]. This coupling metric takes account of the degree of coupling, functional complexity and transitive.
In this proposed metric there are two kinds of measures.

- ➢ WICoup (Weighted Intransitive Coupling)
- ➢ WTCoup (Weighted Transitive Coupling)

Weighted direct Coupling 2

This measure only considers the direct coupling between the classes. The system consists of classes
C= {C1, C2, C3…….Cm}.
The class consist of a set of fields and methods.
Let Mj be the set of methods of class Cj, Vj be the set of fields of class Cj and MVj,i be the set of fields and methods of Ci invoked by Cj.
MVj is the set of all methods and fields in other classes that is invoked by class Cj can be defined:

$$MVj = \bigcup_{\text{[illegible]}} MVj,i$$

The result of MVj is more, if the class invoke more methods and fields outside the class.CoupD(i.J) is the measure of direct coupling between class Ci ,Cj.is defined as

$$CoupD(i,J) = \frac{|MVi,j|}{|MVi| + |Vi| + |Mi|}$$

Coupling measure for an entire system consistsof nno of classes.

$$WICcupD = \sum_{i,j=1}^{n} \frac{CoupD\,(i,j)}{n^2 - n}$$

Weighted transitive coupling

Suppose that coupD(i,j) and coupD(j,k) is finite values but that coupD(i,k) is zero.there is no relation between Ci and Ck.Ci coupled with Cj and Cj coupled with Ck. there is indirect coupling between Ci and Ck due to the path p.

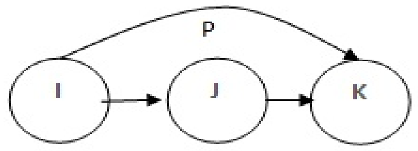

Figure-3. Example of indirect coupling.

For example in fig 3 there is an indirect coupling between class I and K through the path P.

$$CoupT(i,k,p) = \prod_{e_{i,j} \in p} coupD(i,k)$$

$$CoupT(i,k,p) = \prod_{e_{i,k} \in p} \frac{|MVi,k|}{|MVi| + |Vi| + |Mi|}$$

Here e i,k denotes the number of classes betweeni and k . The largest path is taken into account and hence define Coup (i,k) ,the strength of coupling between the two classes,Ci,Ck to be

Coup (i,k)=CoupT(i,k,P_{max}(i,j))

Where P_{max}(i,j) =arg max p$\in \pi$ CoupT(i,k,p) and π is the set of all paths from Ci to Ck.
The weighted transitive coupling is defined as

$$WTCoup = \sum_{i,j=1}^{n} \frac{Coup\,(i,k)}{n^2 - n}$$

For n is the number of classes in the system.

Case study: Information retrieval and metric measurement

All the structural cohesion and coupling measures were computed using Java reflection.java reflection is used as a reverse engineering tool that contains the components for analyzing Java code and extracting the information. The structural information needed to compute cohesion and coupling has been also extracted using Java reflection. It extracted all types of fields, methods and constructors from classes in source.

In order to validate and demonstrate the metric calculation, a case study has been completed. From Figure-4 consider an example a system consist of classes, including fields, methods and constructors. Number of field accessing methods can be found by using the signature of the field inside the method. classA and classD having indirect coupling between them.

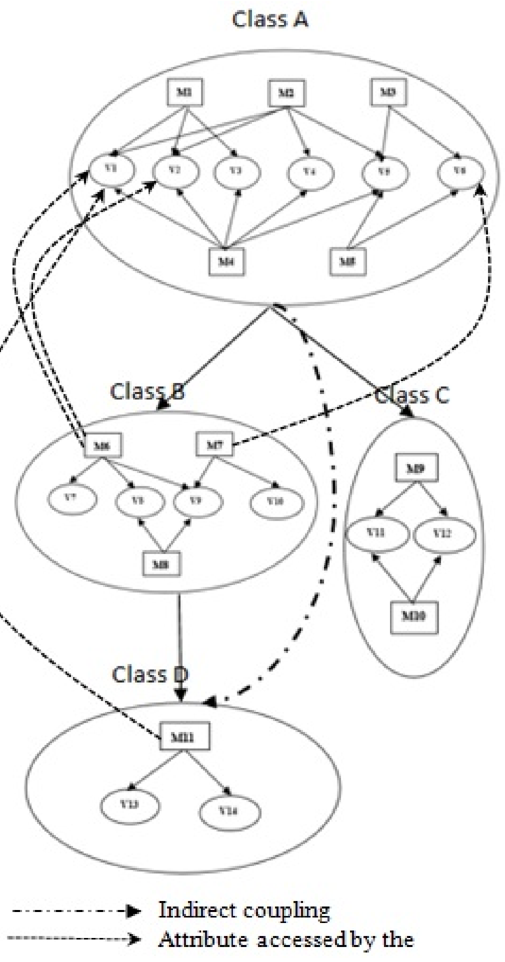

Figure-4. Example for case study.

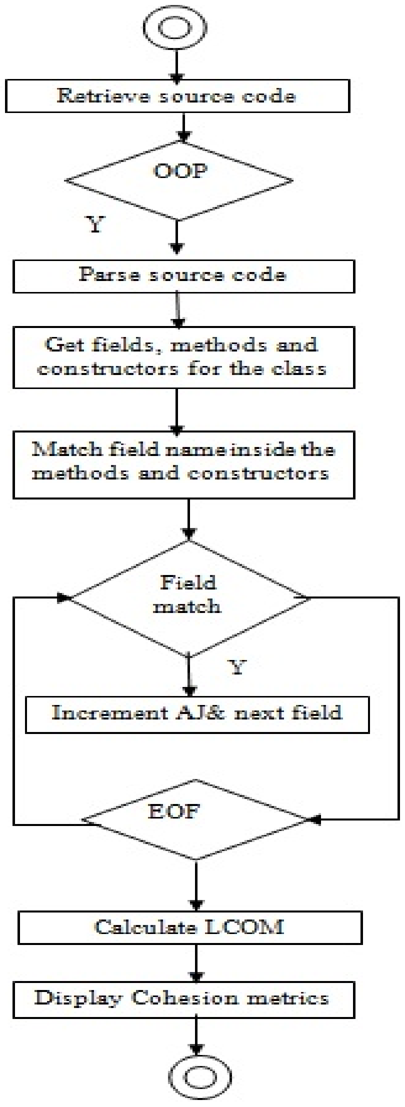

Figure-5. Cohesion measurement program flow.

The system consist of classes (A, B, C), methods (m1, m2,…m11) and fields (v1, v2,…. v14). In this example class A inherited by class B & C. class B is inherited by class. Cohesion measurement only concentrates on a single class, it does not consider inherited classes.

Cohesion measurement

Java reflection retrieves all the fields, methods and constructors. Each field's signature inspected in the method definition. If a field is used inside a method, the value of a variable Aj incremented by one. These steps repeated for all fields and methods.Fig: 5 cohesion measurement program flowgives the clear idea about it.
In our example case study consider classA
No of methods=5.
No of fields =6.
No of Methods that access attribute =16.

LCOM1=(R > S)? (R - S): 0
R=2, S=8.
=0.
LCOM2 = 1 - sum (mA) / (m*a)
= 0.466
LCOM3 = (m - sum (mA)/a) / (m-1)
=0.583
LCOM4 = (m – [sum (meA) –m]/ac) / (m-1)
=0.583
LCOM5 = {[(1/a) (∑u (Aj))] – m} / (1 – m)
= 0.1716

Table-1. Result of various cohesion measurement.

Class	LCOM1	LCOM 2	LCOM 3	LCOM 4	LCOM 5
Class A	0	0.466	0.583	0.583	0.171
Class B	0	0.416	0.625	0.625	0.5
Class C	0	0	0	0	1
Class D	0	0	∞	∞	∞

From the Table-1 contain the result of various cohesion measurement for an example case study.

Coupling measurement

In our example case study The system consist of classes (A, B, C), methods (m1, m2,..,m11) and fields (v1, v2,…. v14). In this example class A inherited by class B & C. class B is inherited by class. Coupling is a relationship between the classes. So we have to find parent class and grandparent for a class.java reflection help to find a parent class for a class.

Before measuring coupling we have to find fields, methods and constructors for each class.
If the class is inherited, find parent and grant parent classes. Find the signature of base class field in derived class. If a variable found inside the method increment the value of MVji. That is the set of all methods and fields in other classes that is invoked by class Cj

Consider the case study example, Class A inherited by Class B. Both classes having direct coupling. Method m6 access the field v1& v2. Method m7 access the field v6 and method m6 access m4. Coupling between classes B & A is
Mvji =3.
The value should be larger, if the class invoke more methods and fields

$$CoupD(i,j) = \frac{|MVi,j|}{|MVi| + |Vi| + |Mi|}$$

$$CoupD(i,j) = \frac{|3|}{|11| + |4| + |3|}$$

CoupD(i,j) =0.167.

$$WICoupD = \sum_{i,j=1}^{n} \frac{CoupD(i,j)}{n^2 - n}$$

WICoupD = 0.014

There is an indirect coupling between class A & D.

CoupT(i,k)=0.0714

WTCoup =0.006

The results in this study using structural information to measure cohesion and coupling. They may help to find the quality of software and predict the fault.

6. CONCLUSIONS

This paper focus on object oriented quality metric parameters. To extract this information for cohesion and coupling measurement, Java reflection can be used in a manner similar to measuring these metrics. Java reflection defines the novel structural information retrieval method for coupling and cohesion measurement. The low coupling and high cohesion improve the product quality and reliability. The outcome result helps to predict the fault of an object oriented system after the coding phase is completed.

The reflection retrieval method is also applicable C# and PHP. In future, we hope to predict the fault in earlier phases. The next version will calculate coupling and cohesion metrics for UML representations of software. The investigation for a new tool has started to calculate metrics for both UML and source code representation.

REFERENCES

[1] Amos. Dange, Prof. Dr. S. D. Joshi. 2011. ”Fault Prediction in Object Oriented System Using the Coupling and Cohesion of Classes”,IJCSMS International Journal of Computer Science and Management Studies. Vol.11, Issue 02, pp. 48-51, Aug.

[2] Andrian Marcus and Denys Poshyvanyk. 2008.” Using the Conceptual Cohesion of Classes for Fault Prediction inan Object-Oriented Systems,” IEEETRANS ACTIONSON SOFTWARE ENGINEERING, Vol.34, NO.2, pp 287-300, march /april.

[3] Arisholm E., Briand L. C. and Foyen A. 2004. "Dynamiccoupling measurement for OO software". IEEETSE. Vol. 30, no.8, pp. 491-506.

[4] Bansiya J. and Davis C. G. 2002. "A hierarchical model for object-oriented design quality assessment", IEEETSE. Vol. 28, no.1, pp.4-17.

[5] Briand L., Devanbu P. and Mello W. 1997. ”An investigation into coupling measures for C++,” proceedings of ICSE.

[6] Briand, S. Morasca and V. R. Basili. 1996 “Property-Based Software Engineering Measurements”. IEEE Trans. Software Eng. Vol. 22, no.1, pp.68-85, Jan.

[7] Briand L.C., Wüst J., Daly J. W. and Porter V. D. 2000. "Exploringthe relationship between design measures and software quality in object-oriented systems", Journal of System and Software. Vol. 51, no.3, pp.245-273.

[8] Chidamber S. R. and Kemerer C. F. 1994. "A Metrics Suite for Object Oriented Design”, in IEEE transaction in software engineering,. Vol. 20, No. 6, June.

[9] Deepak Arora, PoojaKhanna and Alpika Tripathi, Shrpra Sharma and Sanchika Shukla. 2011. Software quality estimation through object oriented design metrics,” In international journal of computer science and network security. Vol. 11, No 4, April.

[10] Guigui, Paul D. Scott. 2009.”Measuring software component reusability by coupling and cohesion,” In journal of computers, Vol.4, No. 9, September.

[11] Kavitha, Dr. A. Shanmugam. 2008.”Dynamic coupling measurement of object oriented software using trace events,” In IEEE Transaction, 2008.

[12] Lawrie D., Field H. and Binkley D. 2006. "Leveraged Quality Assessment Using Information Retrieval Techniques", in ICPC' 06, pp.149-158.

[13] Mishra B., ShuklaK. K. 2011. “Impact of attribute selection on defect proneness prediction in OO software,”2nd International Conference on Computer and Communication Technology (ICCCT), IEEE Conference Publications, pp. 367-372.

[14] Subramanyam and Krishnan. 2003. ”Empirical analysis of CK metrics for OOD complexity: Implication for software defect,” IEEE transaction on software engineering.

[15] Sukainah Husein and Alan Oxley. 2009. ”A coupling and cohesion metrics suite for object oriented software,” international conference on computer technology and development, IEEE conference publication.

[16] Újházi B., Ferenc R., Poshyvanyk D. and Gyimothy T. 2010.”New Conceptual Coupling and Cohesion Metrics for Object-Oriented Systems,” 10th IEEE Working Conference, pp. 33 – 42.