I.J. Information Technology and Computer Science, **2014, 08, 72-78**
Published Online July 2014 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2014.08.10

Coupling Metric for Understandability and Modifiability of a Package in Object-Oriented Design

Sandip Mal
Department of Computer Science and Engineering, Central University of Jharkhand, India
Email: sandip.mal@cuj.ac.in

Kumar Rajnish
Department of Information Technology, BIT, Mesra, India
Email: krajnish@bitmesra.ac.in

***Abstract*— This paper presents a new coupling metric (*Coup*), which is based on the formal definition of methods and variables of classes, and packages. The proposed metric has been validated theoretically against Briand properties as well as empirically using packages taken from two open source software systems and four experienced teams. We measure Coup value by our own CC tool. An attempt has also been made to present a strong correlation between *Coup* values and *understandability of the packages* and between *Coup* values and *modified classes of the packages*. The results indicate that Coup is used to predict understandability and modifiability of a package in Object-Oriented design. Finally this paper proves that Coup is a better predictor of *understandability* and *modifiability* of a package than other existing coupling metrics in the literature.

***Index Terms*— Package, Coupling, Metric, Understandability, Modifiability, Quality Factor, Object-Oriented

I. Introduction

Software engineering is an engineering discipline that is concerned with all aspects of software production. Software products consist of developed programs and associated documentation. Essential product attributes are modifiability, dependability, understandability and usability. Coupling measures the degree of interaction and relationships among source code elements, such as classes, methods, attributes, and packages in Object-Oriented (OO) software systems. It has important applications in software development and maintenance. They are used to help developers, testers and maintainer's reason about software complexity and software quality attributes. The current research on modeling and measuring the relationships among software components through coupling analysis is insufficient. Coupling measures are incomplete in their precision of definition and quantitative computation. One of the main goals behind OO analysis and design is to implement a software system where classes of a package have low coupling among them. This paper presents a coupling metric between classes and show how classes of a package are coupled with each other. This coupling measure is well correlate with quality factors like understandability and modifiability.

For OO systems, most of the coupling metrics have been defined up to class level [1-8] and only a few metrics exist for measurement of coupling at the higher levels of abstraction in OO systems [9-11]. Other work related to packages or other higher abstraction levels has been carried out in [12-19] [21]. Many researchers have been proposed coupling metrics and their reviews are available in the literature but the most important metrics have been selected for our comparative study. Table 1 describes the details of metric chosen for comparative study.

Table 1. Existing Coupling Metrics

Name	Definition
CBO [2] [23]	Classes are coupled if methods or instance variables in one class are used by the other. CBO for a class is number of other classes coupled with it.
RFC [2] [23]	Count of all methods in the class plus all methods called in other classes.
Fan out[24]	The fan out of a class is the number of its immediately subordinate classes.

The rest of the paper is organized as follows. Section 2 describes some basic definitions and new coupling metric with a working example. Section 3 provides theoretical validation of proposed metrics against Briand properties. Section 4 presents a case study on open source software system. Section 5 presents conclusion and future work.

II. Proposed Coupling Metric and Working Example

Before going to define proposed coupling metric, some basic definition about class, method, variable, and package has been discussed first. This basic definition is useful for theoretical and empirical validation of proposed coupling metric.

1. ***Package***: Package is a set of classes, sub packages and interfaces (Java/C# application) as their elements. Further, these sub-packages also may contain classes, sub packages and interfaces as their elements. This leads to a hierarchical structure of packages in a software system. But this study ignores the sub packages and considers only the first level of hierarchical structure.

2. ***Empty Package***: A package that have no elements in it and hence, there is no relations with other packages. It is denoted by (Ɵ, Ɵ).

3. ***Class***: Class is a set of global variable V = {v_1, v_2, ..., v_n} and set of methods M = {m_1, m_2, ..., m_n}.

4. ***Method***: In OO programming, a method is a subroutine (or *procedure*) associated with a class. Methods have the special property that at runtime, they have access to data stored in an instance of the class.

5. ***Global Variable:*** Suppose Packages of a system contains a set of classes $C = \{C_1, C_2, \ldots C_m\}$. CoupM(i,j) is the coupling value by sharing methods between Class i and j. Similarly CoupV(i,j) is the coupling value by sharing variable between Class i and j.

CoupM(i,j)= $|Ci(m) \cap Cj(m)|$
CoupV(i,j)= $|Ci(v) \cap Cj(v)|$

CoupA(i,j) value is the net coupling value between two classes i and j and calculated by taking summation value of CoupM(i,j) and CoupV(i,j).

$$\text{CoupA}(j,j) = \begin{cases} 1 & \text{if CoupM}(i,j) + \text{CoupV}(i,j) > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\text{Coup}= \frac{\sum_{i,j=1}^{p} CoupA(i,j)}{2}$$

We divide the summation of CoupA(i,j) by 2 because CoupA(i,j) and CoupA(j,i) is same things. Coup is the coupling value of a package of p classes.

```
package p1;
public class C1{
            public int a, b, c;
public void m1(int i){
     }
public void m2(int i){                    }
}
public class C2 extends C1{
                    a++;
   m1(int i){
     }
}
public class C3 extends C1 {
   public void m1(int i){
     }
public class C4 extends C1 {
   a++; c++;
}
```

CoupM(C1,C2) = 1, CoupM(C2,C1) = 1
CoupM(C1,C3) =1, CoupM(C3,C1) = 1

The value of CoupM(C1,C4) , CoupM(C2,C3), CoupM(C2,C4), and CoupM(C3,C4) is zero because there is no common method.

CoupV(C1,C2) = 1, CoupV(C2,C1) = 1
CoupV(C1,C4) =1, CoupV(C4,C1) = 1

The value of CoupM(C1,C3) , CoupM(C2,C3), CoupM(C2,C4), and CoupM(C3,C4) is zero because there is no common variable.

CoupA(C1,C2) =1, CoupA(C2,C1) = 1
CoupA(C1,C3) =1, CoupA(C3,C1) = 1
CoupA(C1,C4) =1, CoupA(C4,C1) = 1

So, Coup=6/2=3

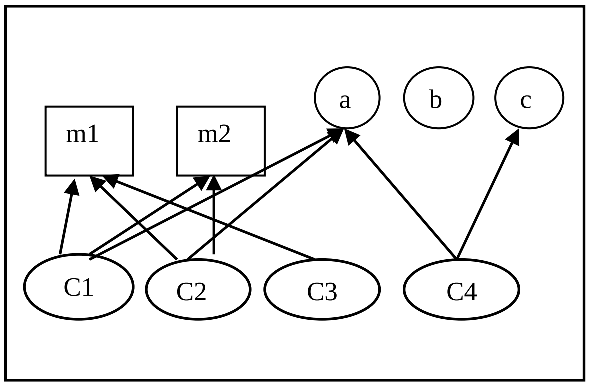


Fig. 1. Example of a Package p1

III. Theoretical Validation

The proposed coupling measures are validated theoretically by analyzing their mathematical properties. For this purpose, five properties given by Briand *et al.* in [20] are used and these properties provide a useful guideline in construction and validation of coupling measures in a precise manner and these properties are necessary to prove the usefulness of a coupling measure.

Property 1: Non-Negativity

The value of coupling between a pair of classes and Coup value of package will always be non-negative.

Coup ≥ 0

Thus, Coup satisfies Property 1.

Property 2: Null Value

If the number of class in the package is zero or one or there is no common variable or no common method between two classes in a package, then Coup will be null. So Coup satisfies property 2.

Property 3: Monotonicity

If we add one common method or common variable then CoupA value will never decrease. So Coup value will never decrease if we add common method or variable. So Coup also satisfies property 3.

Property 4: Merging of Classes

This property states that merging of two classes must not increase coupling of resulting system because some of the relationships may disappear on merger. Let *P* be a package an object-oriented system, and C1, C2 be two classes in *P*. Let C be the class which is obtained by merging of C1 and C2. Let *P'* be the new package of object-oriented system. Then, in all case Coup of p is greater than or equal to Coup of p'. Thus, Coup also satisfies Property 4.

Property 5: Merging of Unconnected Classes

This property states that merging of two unconnected classes must not increase coupling of resulting system. When two or more classes having no common method or variable between them are merged, coupling cannot increase because apparently unconnected classes are being encapsulated together in a single class. Let *A* and *B* be two classes in a package *P*. Let *A+B* be the class, which is the union of A and B. Let *P'* be the new package of OO system. If no relationships exist between class A and B, then Coup of package p is greater than Coup of *P'*.

Thus, Coup satisfies this property.

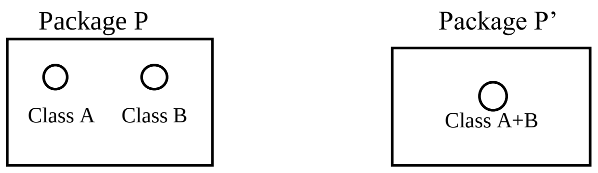


Fig. 2. Example of Merging of unconnected class

IV. Case Study of Coup on Open Source Software System

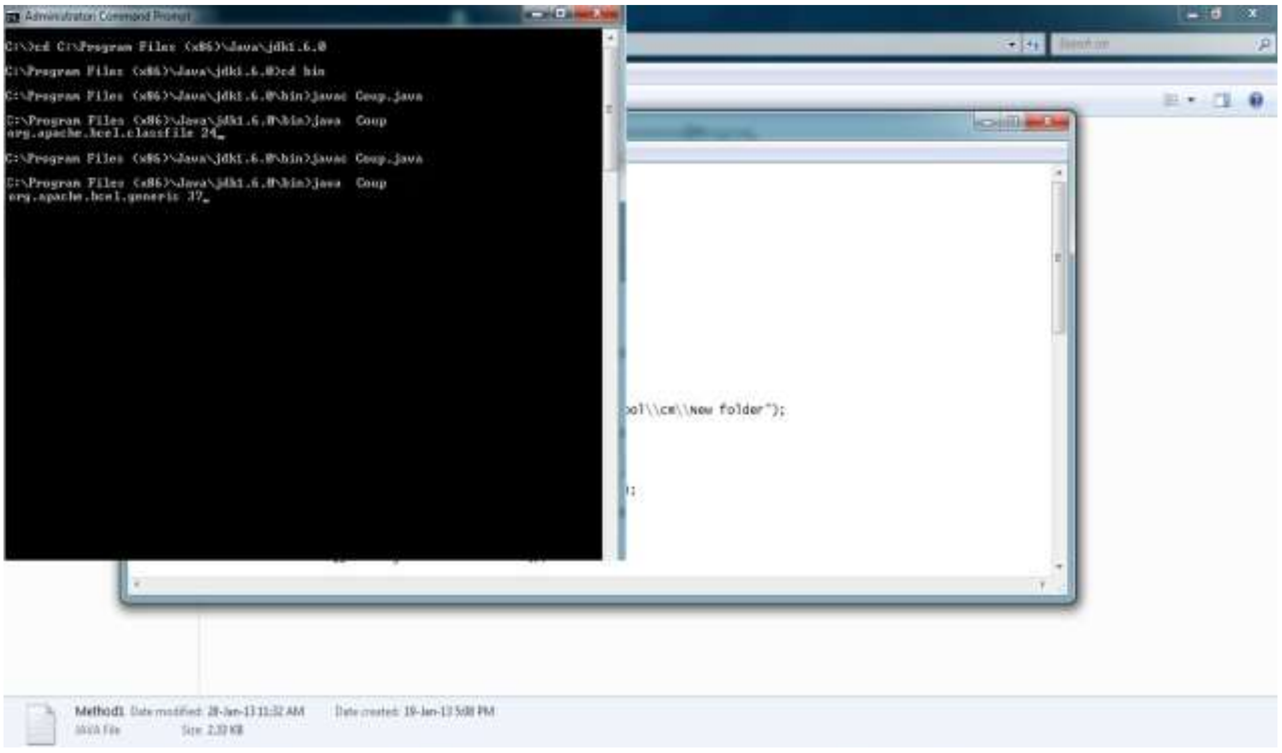
Two open source software projects have been chosen for case-study. XGen [25] Source Code Generator, that creates Java source code from a simple XML document and its main function is to generate JDBC compliant beans that allow object level persistence to relational databases and The Byte Code Engineering Library (Apache BCEL) [26] is intended to give users a convenient way to analyze, create, and manipulate (binary) Java class files (those ending with .class).The basic data about these two projects are given in Table 2. For Coup analysis 4 package of BCEL and 7 package of XGen have been taken. BCEL have 367 classes in 4 packages and XGen have 73 classes in 7 packages. Table 3 and Table 4 lists the names of packages of BCEL, XGen and the number of classes contained in each package.

Table 2. Information about Project Taken for Case Study

Software Project	Apache BCEL	XGen 0.5.0
No of Package	4	7
No of Classes	367	73

A. Results

The proposed Coupling Metrics (Coup) has been applied to seven packages taken from XGen and four packages taken from BCEL software systems. The Coup value with their number of classes is given in Table 3 and Table 4. Our CC tool calculates the Coup value of each package. Snapshot of result has given in figure 3.


Administrator: Command Prompt
Method1 Date modified: 28-Jan-13 11:32 AM Date created: 19-Jan-13 5:08 PM
JAVA File Size: 2.32 KB

```
C:\>cd C:\Program Files (x86)\Java\jdk1.6.0

C:\Program Files (x86)\Java\jdk1.6.0>cd bin

C:\Program Files (x86)\Java\jdk1.6.0\bin>javac Coup.java

C:\Program Files (x86)\Java\jdk1.6.0\bin>java  Coup
org.apache.bcel.classfile 24_

C:\Program Files (x86)\Java\jdk1.6.0\bin>javac Coup.java

C:\Program Files (x86)\Java\jdk1.6.0\bin>java  Coup
org.apache.bcel.generic 37_
```

```
ool\\cm\\new folder");
```

Fig. 3. Snapshot of Result calculating by CC tool

The results given in Table 3 and Table 4 may not be always true that a package with the large number of classes have more Coup, as an example, package org.apache.bcel.classfile and workzen.xgen.engine.

Table 3. Number of classes and Coup values of four packages Taken from Apache BCEL

Sl. No.	Name of Package	No. of Classes	Coup
1	org.apache.bcel.classfile	51	24
2	org.apache.bcel.generic	225	37
3	org.apache.bcel.util	28	29
4	org.apache.bcel.verifier	63	15

Table 4. Number of classes and COP values of seven packages Taken from XGen

Sl. No.	Name of Package	No. of Classes	Coup
1	workzen.xgen.ant	5	4
2	workzen.xgen.engine	2	7
3	workzen.xgen.loader	7	22
4	workzen.xgen.model	17	19
5	workzen.xgen.test	23	16
6	workzen.xgen.type	15	12
7	workzen.xgen.util	4	9

B. *Empirical Validation*

Four teams of three members each have been set up and assigned these packages to three teams. These members are well experienced in Java programming. The teams set rank to the each package based on a heuristic score. The heuristic score is the time taken to understand the package and effort require to find out Coup value manually by the teams. The packages were ranked 1 to 11 based on their increasing heuristic score and Coup rank is the rank according to increasing Coup value. Table 5 shows the Coup ranking and the ranking given by four teams.

The Correlation Coefficient, R_s is used here to test the significance of the correlation between Coup rank and individual team's rank. 0.65 cutoff has been considered for validation of 11 packages. Spearman Correlation Coefficient for two set is calculated in the following formula.

Table 5. Coup ranking and Teams ranking of eleven packages taken from two open source system

Sl. No.	**Name of Package**	**Team Rank**				**Coup Rank**
		1	**2**	**3**	**4**	
1	workzen.xgen.ant	1	3	1	3	1
2	workzen.xgen.engine	3	4	2	2	2
3	workzen.xgen.loader	10	9	10	11	8
4	workzen.xgen.model	5	11	7	8	7
5	workzen.xgen.test	9	8	9	7	6
6	workzen.xgen.type	4	7	4	5	4
7	workzen.xgen.util	2	1	3	1	3
8	org.apache.bcel.classfile	6	6	5	4	9
9	org.apache.bcel.generic	11	10	11	10	11
10	org.apache.bcel.util	7	5	8	9	10
11	org.apache.bcel.verifier	8	2	6	6	5

$$R_s = 1 - \frac{6\sum d2}{n(n2-1)} \qquad -1.00 \le R_s \le +1.00$$

Following null hypothesis has been set to test the result.

H_0: *There is no correlation between the Coup rank of a package and the rank given by teams to the packages.*

H_1: *The alternative hypothesis, there is significant positive correlation between the Coup rank of a package and the given by teams to the packages.*

Table 6 shows that team 2 cannot evaluate Coup value properly. Packages with high Coup value is better design then the low Coup value. This was felt for team 2.

Table 6. Correlation Coefficient between Coup ranking and Teams ranking of eleven packages

	Team 1	Team 2	Team 3	Team 4
$6\sum d^2$	276	516	204	288
R_s	0.79	0.60	0.84	0.78
$R_s > 0.65$	√	×	√	√

This study also shows a well correlation between Coup and the quality factors such as understandability, modifiability. This property of Coup indicates the usefulness of the proposed metrics. First, we eliminate team 2 and calculate the effort required to fully understand the functionality of these packages by these three valid teams and rank the effort from 1 to 10. A higher rank indicates that more effort spent on understanding the package. Table 7 shows the effort required by each team and average effort. Then all the packages have been given to most experienced team to modify the package. We select the team 3, whose Correlation Coefficient is larger (0.84) than other team as a most experienced team. The team adds, delete or modify some classes to add, delete and modify some feature of the packages and recalculate the Coup values. Table 8 shows the number of modified classes and new Coup values.

Table 7. Effort and Coup values of seven packages Taken from two open source system

Sl. No.	Name of Package	Team			Average Effort
		1	3	4	
1	workzen.xgen.ant	4	2	3	3.0
2	workzen.xgen.engine	3	3	2	2.7
3	workzen.xgen.loader	6	5	6	5.7
4	workzen.xgen.model	6	7	8	7.0
5	workzen.xgen.test	4	6	5	5.0
6	workzen.xgen.type	3	2	3	2.7
7	workzen.xgen.util	2	3	2	2.3
8	org.apache.bcel.classfile	4	5	5	4.7
9	org.apache.bcel.generic	8	8	9	8.3
10	org.apache.bcel.util	5	6	6	5.7
11	org.apache.bcel.verifier	4	5	3	4.0

Table 8. Number of modified classes and new Coup values of packages Taken from two open source system

Sl. No.	Name of package	No. of Modified Classes		Coup
		Add Class	Delete Class	
1	org.apache.bcel.classfile	0	2	26
2	org.apache.bcel.generic	0	5	41
3	org.apache.bcel.util	1	0	27
4	org.apache.bcel.verifier	2	0	19
5	workzen.xgen.ant	1	0	4
6	workzen.xgen.engine	1	0	5
7	workzen.xgen.loader	0	1	26
8	workzen.xgen.model	0	2	15
9	workzen.xgen.test	3	0	15
10	workzen.xgen.type	0	2	13
11	workzen.xgen.util	2	0	8

Following two null hypotheses has been set to test the result.

H_{U0}: ρ=0. *There is no correlation between the Coup and the effort required to understand the package.*

H_{U1}: ρ≠0. *There is strong positive correlation between the Coup and the effort required to understand the package.*

H_{M0}: ρ=0. *There is no correlation between Coup and the number of modified classes to understand the modifiability of a package.*

H_{M1}: ρ≠0. *There is strong positive correlation between Coup and the number of modified classes to understand the modifiability of a package.*

To test theses hypothesis, first calculate the amount of correlation between two variables Coup and Average Effort required for understanding the package. Secondly calculate the amount of correlation between the numbers of modified classes and Coup. For this purpose, Spearman's rank correlation method has been used. Table 9 gives the correlation results. The correlation coefficient between two variables Coup and Average Effort comes out to be 0.87 at 0.01 significance levels and the correlation coefficient between two variables Coup and modifiability comes out to be 0.58 at 0.01 significance levels. The results indicate a strong correlation in both cases. Thus, we reject the null hypothesis H_{U0}, and $H_{M0.}$ This strong correlation indicates that Coup is a good predictor of quality factor understandability and modifiability.

Table 9. Spearman’s Correlation between Coup and Average effort & Coup and Modifiability

Variable	Coup &Average Effort	Coup & Modifiability
Correlation coefficient (r)	0.87	0.58

A comparison of Coup with the other existing coupling metrics, given in Table 1, has been described. Table 10 gives the CBO, RFC and Fan out value of each package of two open source software systems. All the considered metrics are class level coupling metrics. For measuring coupling of a package, we take average of coupling values of classes in a package. Table 10 provides the average of Coup metric values of classes in a package and table 11 provides average of Coup metric values after modifying the packages. JHAWK tool has been used to find out the metrics values.

Table 10. CBO, RFC, MPC and Fan In values of 11 packages

Sl. No.	Name of Package	CBO	RFC	Fan out
1	workzen.xgen.ant	8.8	30.8	6
2	workzen.xgen.engine	1	25	7
3	workzen.xgen.loader	2.33	32	5
4	workzen.xgen.model	1.12	22.82	3
5	workzen.xgen.test	0.96	32.54	5
6	workzen.xgen.type	2.53	1.6	3
7	workzen.xgen.util	0	22	5
8	org.apache.bcel.classfile	10.21	24.9	4
9	org.apache.bcel.generic	9.46	13.26	4
10	org.apache.bcel.util	1.25	13.75	4
11	org.apache.bcel.verifier	4.56	26.69	5

Table 11. CBO, RFC, MPC and Fan In values of 11 packages after modification of packages

Sl. No.	Name of Package	CBO	RFC	Fan out
1	workzen.xgen.ant	9.2	31	6
2	workzen.xgen.engine	2.3	25.6	8
3	workzen.xgen.loader	2	31.42	5
4	workzen.xgen.model	1.42	23	5
5	workzen.xgen.test	1	32.94	4
6	workzen.xgen.type	1.98	1.2	3
7	workzen.xgen.util	1	27	6
8	org.apache.bcel.class	9.54	22.8	5
9	org.apache.bcel.generic	7.62	10	3
10	org.apache.bcel.util	2	14.35	5
11	org.apache.bcel.verifier	5	26.59	4

Table 12. Spearman's Correlation between Coup and Average effort & Coup and Modifiability

	CBO	RFC	Fan out
Average Effort	0.238	-0.043	0.476
Modifiability	0.216	-0.049	0.526

Table 12 describes the Spearman's Correlation between Coup and Average effort & Coup and Modifiability of CBO, RFC, and Fan out. Positive values indicate that metric is useful to predict the factors (Average Effort, Modifiability). CBO and Fan out is the accepted metrics for measuring Average Effort and Modifiability but RFC is not a good measure for Average Effort and Modifiability as it has negative Correlation. Table 9 shows Correlation Coefficient of Coup (0.87 & 0.58), which is larger than the Correlation Coefficient of CBO, RFC and Fan out. So we can say that Coup is the better measure of Average Effort and Modifiability than CBO, RFC, and Fan out.

V. Conclusion and Future Work

In this paper, an attempt has been made to propose a new coupling metric which is based on formal definitions, properties and relations of classes. The structure of packages has also been taken into consideration during the measurement of coupling of a package. The proposed metrics has been validated theoretically as well as empirically. The theoretical validation of *Coup* satisfies all the properties presented by *Briand*. In addition to the proposal and theoretical validation, this paper has also presented empirical data on *Coup* from two open source software system *(Apache BCEL, XGen 0.5.0).* Both systems developed in Java. From Table 9, it is found that there is a strong correlation between *Coup* and *understandability (0.87, significant label 0.01)* and between *Coup* and *modifiability (0.58, significant label 0.01).* So, this study clearly provided that *Coup* is the valid indicator of external quality attributes of the software such as understandability, modifiability and also better than the other existing coupling metrics. This firmly believes us that this work will encourage other researchers and developers to use the results obtained from this study to predict and measure several other software quality attributes.

The future scope includes some fundamental issues

- To analyze the nature of proposed metric with performance indicators such as design, maintenance, effort and system performance.
- Another interesting study would be together different coupling metric at various intermediate stages of the project. This would provide insight into how application reusability, maintainability, testability evolves and how it can be managed and controlled through the use of metrics.

References

[1] S R Chidamber, C F. Kemerer, Towards a metrics suite for object oriented design. In *Proc. the 6th ACM Conf. Object-Oriented Programming: Systems, Languages and Applications*(*OOPSLA*), Phoenix, AZ, Oct. 6-11, (1991), pp.197-211.

[2] S R Chidamber, C F. Kemerer, A metrics suite for object oriented design. IEEE Transactions on Software Engineering, 20(6) (1994): 476-493.

[3] N. I. Churcher, M J. Shepperd, Comments on `A metrics suite for object-oriented design'. IEEE Transactions of Software Engineering, 21(3) (1995), 263-265.

[4] M. Hitz, B. Montazeri, Measuring coupling in object-oriented systems. Object Currents, 1(4) (1996) ,124-136.

[5] L. Briand, P Devanbu, W. Melo, An investigation into coupling measures for C++. In Proc. 19th Int. Conf. Software Eng., Boston, May 17-23, 1997, pp.: 412-421.

[6] K.Rajnish and V. Bhattacherjee, Class Cohesion: An Empirical and Analytical Approach, International Journal of Science and Research (IJSR), Victoria, Australia, Vol.2, No. 1, 2007, pp.53-62, http://www.international.au.in.

[7] S. Mal and K.Rajnish, Applicability of Weyuker's Property 9 to Inheritance Metric.International Journal of Computer Applications, Foundation of Computer Science, USA, Volume 66– No.12, March 2013.

[8] B.S. Henderson, L. L. Constantine, I. M. Graham, Coupling and Cohesion (towards a valid metrics suite for object oriented analysis and design, Object oriented systems, vol. 3, 143-158, 1996.

[9] Y S Lee, B S Liang, S F Wu, F J Wang. Measuring the coupling and cohesion of an object-oriented program based on information flow. In Proc. International Conference on Software Quality, Maribor, Slovenia, Nov. 6-9, 1995, pp.81-90.

[10] S. Mal and K.Rajnish, "Theoretical validation of New Class Cohesion Metric against Briand Properties". Accepted for the publication in International Conference on Advanced Computing, Networking, and Informatics (ICACNI-2013), published by Advances in Intelligent and soft Computing, springer, Raipur, India.

[11] G Gui, P D Scott, Coupling and cohesion measures for evaluation of component reusability. In Proc. International Workshop on Mining Software Repositories, Shanghai, China, May 22-23, 2006, pp.18-21.

[12] S. A. E. bad, and M. Ahmed, An Evaluation Framework for Package-Level Cohesion Metrics, International Conference on Future Information Technology, Singapore vol.13 (2011), pp-239-243.

[13] L C Briand, J W Daly, J K WÄust, A unified framework for coupling measurement in object-oriented systems. IEEE Transactions on Software Engineering, 1999, 25(1): 91-121.

[14] E Allen, T Khoshgoftaar. Measuring coupling and cohesion of software modules: An information theory approach. In Proc. the Seventh International Software Metrics Symposium, London, UK, April 4-6, 2001, pp.124-134.

[15] T Xu, K Qian, X He, Service oriented dynamic decoupling metrics. In Proc. the 2006 International Conference on Semantic Web and Web Services (SWWS'06), Las Vegas, USA, June 26-29, 2006, pp.170-176.

[16] V Gupta, J. K Chhabra, Package coupling measurement in object-oriented software. Journal of computer science and technology 24(2): 273-283 Mar. 2009.

[17] X Franch, J P. Carvallo, A quality-model-based approach for describing and evaluating software packages. In Proc. IEEE Joint International Conference on Requirements Engineering (RE'02), Essan, Germany, Sept. 9-13, 2002, pp.1-8.

[18] H. Washizaki, H. Yamamoto, Y. Fukazawa, A metrics suite for measuring reusability of software components. In Proc. the Ninth International Software Metrics Symposium (METRICS'03), 2003.

[19] W. Li, S. Henry, Object-oriented metrics that predict maintainability. Journal of Systems and Software, 1993, 23(2): 111-122.

[20] L. Briand, S. Morasca, V. Basili, Property-based software engineering measurement. IEEE Transactions of Software Engineering, (1996), 22(1): 68-86.

[21] http://sourceforge.net/projects/xgen/

[22] http://jakarta.apache.org

[23] E. Brito, F Abreu, and W. Melo, Evaluating the impact of OO Design on Software Quality. Proc. Third International Software Metrics Symposium. (Berin 1996).

[24] http://www.virtualmachinery.com/jhawkmetricsclass.html

Authors' Profiles

Mr. Sandip Mal is working as an assistant professor in Central University of Jharkhand, India. He is doing his Research from Birla Institute of Technology, Department of Computer Science and Engineering, Mesra, Ranchi, Jharkhand, India.He received his M.E (Software Engineering) degree from Birla Institute of Technology, Mesra, Ranchi, Jharkhand, India in year 2012 and B.Tech degree from West Bengal University of Technology, West Bengal, India in the year of 2008. His research area is Object-Oriented Metrics, Software Engineering and Programming Languages.

Dr. Kumar Rajnish is an Assistant Professor in the Department of Information Technology at Birla Institute of Technology, Mesra, Ranchi, Jharkahnd, India. He received his PhD in Engineering from BIT Mesra, Ranchi, Jharkhand, India in the year of 2009. He received his MCA Degree from MMM Engineering College, Gorakhpur, State of Uttar Pradesh, India. He received his B.Sc Mathematics (Honours) from Ranchi College Ranchi, India in the year 1998. He has 24 International and National Research Publications. His Research area is Object-Oriented Metrics, Object-Oriented Software Engineering, Software Quality Metrics, Programming Languages, and Database System.

How to cite this paper: Sandip Mal, Kumar Rajnish,"Coupling Metric for Understandability and Modifiability of a Package in Object-Oriented Design", International Journal of Information Technology and Computer Science(IJITCS), vol.6, no.8, pp.72-78, 2014. DOI: 10.5815/ijitcs.2014.08.10