Analysis of the Software Code based upon Coupling in the Software

Harjot Singh Virdi, Balraj Singh

Department of Computer Science
Lovely Professional University
Punjab, India, 144806

***Abstract*- Coupling metrics play an important role in determining the quality of the software. In this paper we studied the different types of coupling i.e. Static Coupling and the Dynamic Coupling. We also studied how these metrics perform under different environments and we calculated the mean and the standard deviation for the results produced and graphically represented them. We also described the basic difference between both types of coupling in this paper.**

I. INTRODUCTION

Software metrics have become an essential in measuring the quality of the software projects. Coupling plays an important role in quantitatively evaluation and improves internal quality attributes of the software products. In the object oriented approach it indicates the dependency of the one class on the other class. The changes in one class might affect the functioning of the dependent classes. Coupling can be reduced by promoting cohesiveness of the class [1]. The cohesion in software measures how closely related the responsibilities, methods and data of the class to each other. So the main goal behind the achieving the good quality software is achieving high cohesion and low coupling in software systems.

In the modern programming systems the extensive use of dynamic language features such as polymorphism, dynamic class loading, dynamic class generation, etc. is done to make software as a part of integrated development environment. Loosely coupled components of the system here facilitate comprehensive activities, testing efforts, reuse, and maintenance tasks.

Automatically measuring of the coupling and the cohesion can reduce the measurement effort, subjectivity, and possible errors. For this many coupling and cohesion metrics are introduced for the syntactic and the semantic analysis, examples for this are Lint, Analyst4j, VizzAnalyser, and Sonar. In this paper we will study coupling and cohesion relationships from different frameworks and selecting the metrics that links to respective relationships and how can be these metrics be implemented.

This paper is divided into four parts. The first part is the introduction section which gives an overview of the term coupling and the software metrics. In the second part we discussed the related work done by various authors in the past. The third part describes the approach followed for the analysis of the software. Finally the fourth part gives the conclusion and discusses the future work.

II. RELATED WORK

Stevens et al. [1] was first to introduced the concept of coupling and defined coupling as the measure of the strength of the association established by a connection form one module to another. This means that stronger the coupling between modules, more difficult are they to understand, change and correct and thus the more complex the resulting software system.

Eder et al. [2] defined the relationships that contribute to coupling. These are three in types: interaction coupling, component coupling and inheritance coupling. The interaction coupling describes the relationship caused by message passing and method invocation. The component coupling refers to the relationship caused by the abstract data types. The inheritance coupling refers to the relationship caused by inheritance or if one class is an ancestor of another class.

Page and Jones [3] defined coupling into eight different levels according to their effects on the understandability, maintainability, modifiability and reusability of the coupled modules.

Offutt et al. [4] extended the eight levels of coupling to twelve thus providing a finer grained measure of coupling. They also described algorithms to automatically measure the coupling level between each pair of units in a program. The coupling levels are defined between pairs of units A and B. For each coupling level the parameters are classified by the way they are used. Uses are classified into computation uses (*C-uses*) [5], predicate uses (*P-uses*) and indirect uses (*I-uses*) [4].

A *C-use* occurs when a variable is used on the right side of an assignment statement, in an output statement, or a procedure call. A *P-use* occurs when a variable is used in a predicate statement.

An *I-use* occurs when a variable is used in an assignment to another variable and the defined variable is later used in a predicate. The *I-use* is considered to be in the predicate rather than in the assignment.

Briand et al. [6] also provided a framework which is applicable for a high level design measurement. He defines two types of cohesion, Data-Data Interaction (DD-*interaction*) and Data-Method Interaction (DM- *interaction*) [7]. DD-*interaction* gives relationships caused by interactions between data declaration. Data declarations consisted of abstract local and global type data, class and global attributes. DM-*interaction* gives relationship caused by interactions between data which are at method level. Method level

attributes are consisted of local attributes, method return types and method parameters.

We studied that measurement of the coupling can be classified into two different categories:

- *Static Coupling metrics* measure what all may happen when a set of code will be executed and compute different aspects of the source code.
- *Run-time / Dynamic Coupling metrics* measure what exactly happens when a set of code us executed. It evaluates the source code behaviour and run- time characteristics as well as complexity.

For the static measures Chidamber and Kemerer [8] proposed the six metrics for the object - oriented systems which include two metrics for the measuring coupling, these are, *Coupling between Object* (CBO) and *Response for Class* (RFC).

Coupling Between Objects (CBO) is defined for a class as a count of the number of other classes to which it is coupled. This also includes the coupling due to the inheritance and relates to the notion method of one use method or instance variables of other [8].

Response for Class (RFC) is defined the number of methods in the Response Set (RS). The Response Set (RS) for the class is the set of methods that can potentially be executed in the response to a message received by the object of class [8].

$$RS=M\cup_{all\ i}\{R_i\} \quad (1)$$

Here R_i is the set of methods called by the method i and M is the set of all methods in the class [8].

There is very little research in the field of the run-time coupling measures till date. Here we will describe the two most widely accepted and used approaches for run-time measures:

Yacoub et al. [9] proposed a set of dynamic coupling metrics based design to evaluate the change proneness of the design. These are used to determine quality of the system in the early development phase. They are based on execution scenarios, "the measurements are calculated for parts of the design model that are activated during the execution of a specific scenario triggered by an input stimulus". He defined two metrics as followed:

- *Export Object Coupling* ($EOC_x(o_i,o_j)$) for an object o_i with respect to an object o_j, is defined as the percentage of number of messages sent from o_i to o_j with respect to the total number of messages exchanged during the execution of a scenario x [9].
- *Import Object Coupling* ($IOCx(o_i, o_j)$) for an object o_i with respect to an object o_j ,is the percentage of the number of messages received by object o_i that were sent by object o_j with respect to the total number of messages exchanged during the execution of a scenario x [9].

Arisholm et al. [10] also defined the metrics for run-time coupling; these are given the table I below:

TABLE 1
Abbreviations for the Dynamic Coupling Metrics for the Arisholm et al. [10].

Variable	Description
IC_CC	Import, Class Level, Number of Distinct Classes
IC_CM	Import, Class Level, Number of Distinct Methods
IC_CD	Import, Class Level, Number of Dynamic Messages
EC_CC	Export, Class Level, Number of Distinct Classes
EC_CM	Export, Class Level, Number of Distinct Methods
EC _CD	Export, Class Level, Number of Dynamic Messages
IC_OC	Import, Object Level, Number of Distinct Classes
IC_OM	Import, Object Level, Number of Distinct Methods
IC_OD	Import, Object Level, Number of Dynamic Messages
EC_OC	Export, Object Level, Number of Distinct Classes
EC_OM	Export, Object Level, Number of Distinct Methods
EC _OD	Export, Object Level, Number of Dynamic Messages

Each dynamic coupling metric name starts with either I or E, this is to distinguish between import coupling and export coupling, based on the direction of the method calls. The third letter C or O distinguishes whether the entity of measurement is object or class. The remaining letter distinguishes three types of coupling. The first metric, C, counts the number of distinct classes that a method in a given class/object uses or is used by. The second metric, M, counts the number of distinct methods invoked by each method in each class/object while the third metric, D, counts the total number of dynamic messages sent or received from one class/object to or from other classes/objects [10].

III. APPROACH

The Open Source real world problems such as Velocity (Apache Jakarta Project) [11], Ant (Apache Ant Project) [12] and Xalan-Java (Apache XML Project) [13] are used to study coupling metrics. The source code of all the problems is publicly available and we compiled all the codes with javac compiler from Oracle Java SDK 1.7. Similar study was carried out by the Arisholm et al. [10] with single program Velocity only. In our experiment we calculated the mean and the standard variation of the values for the different coupling

metrics which is given in the figure 1 below, the formula used for the calculation of mean is below:

$$\overline{X} = \frac{\sum_{i=1}^{n} x_i}{n} \quad (2)$$

Where, $\overline{X}$ is the calculated mean for value x_i, value of i ranges from 1,2,3,.....n and n is the number of times the values measured for the given metrics.

For the calculation the standard variation of the values the formula we used is as follows:

$$\sigma = \left[\frac{\sum_{i=1}^{n}(x_i - \overline{X})^2}{n}\right]^{1/2} \quad (3)$$

Where, σ is the standard deviation and the $\overline{X}$ is the calculated mean for value x_i, value of i ranges from 1,2,3,.....n and n is the number of times the values measured for the given metrics.

The results obtained demonstrated that the results obtained by the static coupling measures are not same as with the run-time coupling measures and also additional information over and above the static CBO can be extracted. This holds good with the finding of Arisholm et al. [10] for the single Velocity program.

TABLE 2
Mean and Standard Deviation Table for Velocity

	MEAN	SD
CBO	7.52	8.02
IC_CC	4.27	6.91
IC_CM	8.52	10.74
IC_CD	20.35	33.14
EC_CC	3.79	4.45
EC_CM	7.46	9.25
EC_CD	26.23	28.23

TABLE 3
Mean and Standard Deviation Table for Ant

	MEAN	SD
CBO	8.39	7.84
IC_CC	4.01	7.98
IC_CM	7.64	8.56
IC_CD	16.69	17.03
EC_CC	2.35	3.45
EC_CM	7.11	7.67
EC_CD	21.07	20.49

TABLE 4
Mean and Standard Deviation Table for Xalan

	MEAN	SD
CBO	9.07	9.82
IC_CC	4.17	4.45
IC_CM	8.60	9.01
IC_CD	35.62	38.09
EC_CC	2.91	3.78
EC_CM	6.48	7.68
EC_CD	42.07	45.23

As standard deviation measures the absolute dispersion, so we can say that where there is small standard deviation there exists the high degree of uniformity as well as homogeneity in observations and for large values of standard deviation vice versa exists. So these results obtained directly affect the quality, it means, lower the deviation higher the software quality.

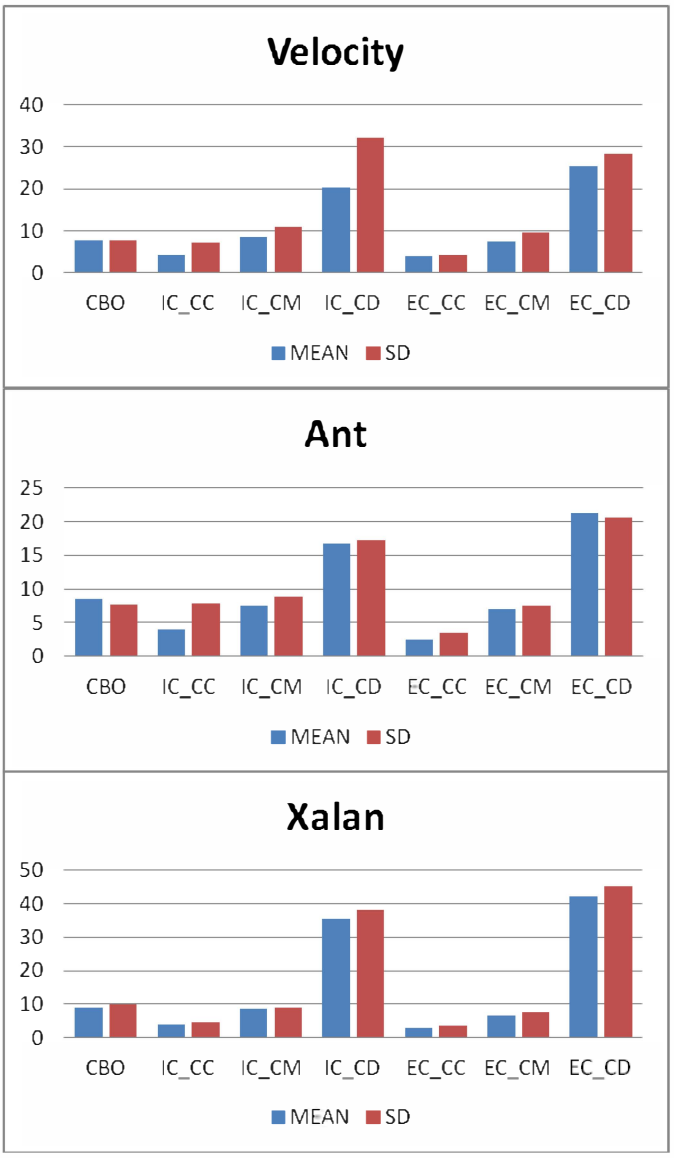

Fig. 1 Graphical Representation of Results for Different Real World Programs.

The calculation of the coefficient of variation is calculated for the value obtained individually for the metrics for different problems, according to the formula given below:

$$C_v = \overline{X} / \sigma \quad (4)$$

Where, C_v is the coefficient of Variation. $\overline{X}$ and σ are the corresponding values of mean and the standard deviation calculated earlier. The graph was obtained for the coefficient

of variation for Velocity, Ant and Xalan, as shown in the figure 2 below.

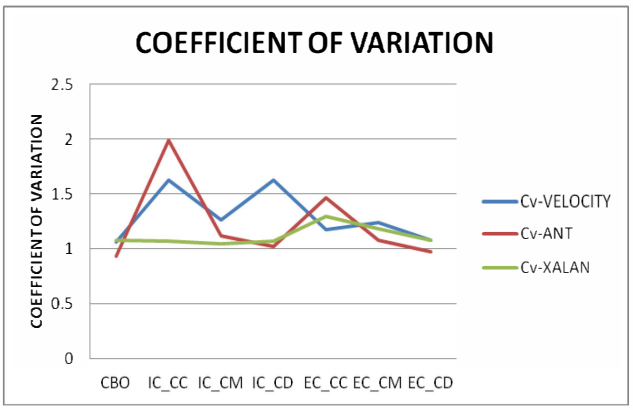

Fig. 2. Graphical Representation of the Coeffiecient of Variation.

We interpreted from the above that when the same set of coupling measures is applied in the different environment the results so obtained are different. Further it can be interpreted from the above graph that the results for Xalan deviates less as compared with Velocity and Ant problems, as the graph line for Xalan follows lesser ups and downs (i.e. its almost straight as the deviation value is less). Also, we can say that Arisholm et al. [10] findings can be applicable across variety of programs.

Here are some common differences observed between the static and the run-time coupling measures:

- Run-time measures are effort intensive i.e. more complex to perform than static measures.
- The static measures are less expensive as compared to the run-time measures.
- Static measures don't reflect the actual situation when the dynamic binding and polymorphism is present in the code.
- Static analysis can be performed during development phases side by side whereas the run-time analysis can be done only after the project completion.

IV. CONCLUSION AND FUTURE WORK

This paper presented the results of static and dynamic coupling measures on the three different real world problems. We calculated the mean and standard deviation for obtained values of result. We also discussed how the value of standard deviation is usefully in judging the representativeness of the mean and quality of software system.

In the future, we will replicate our experiment on large number of software systems and improve the quality of the system by making the deviation factor low.

ACKNOWLEDGMENT

The author would like to thank the faculty of Department of Computer Science, Lovely Professional University, Punjab (India) for their able guidance.

REFERENCES

[1] W.P. Stevens, G.J. Myers, and L. L. Constantine. Structured design. IBM Systems Journal, 13(2):115-139, 1974.
[2] J. Eder, G. Kappel, and M. Schrefl, "Coupling and cohesion in object-oriented systems," University of Klagenfurt, Technical report 1994.
[3] M. Page-Jones. The Practical Guide to Structured Systems Design. Yourdon Press, New York, NY, 1980.
[4] A.J. Offutt, M.J. Harrold, and P. Kolte. A software metrics system for module coupling. The Journal of Systems and Software, 20(3):295-308, 1993.
[5] D. Gregg, J. Power, and J. Waldron. Platform independent dynamic Java virtual machine analysis: the Java Grande Forum benchmark suite. Concurrency and Computation: Practice and Experience, 15(3-5):459-484, March 2003.
[6] L. C. Briand, S. Morasca, and V. R. Basili, "Defining and Validating Measures for Object-Based High-Level Design," IEEE Transactions on Software Engineering, vol. 25, pp. 722-743, 1999.
[7] L. C. Briand, J. W. Daly, and J. Wu¨st, "A unified framework for cohesion measurement in object oriented systems," Empirical Software Eng.: An Int'l Journal, vol. 3, pp. 65-117, 1998.
[8] S.R. Chidamber and C.F. Kemerer. A metrics suite for object-oriented design. IEEE Transactions on Software Engineering, 20(6):467-493, June 1994.
[9] S.M. Yacoub, H.H. Ammar, and T. Robinson. Dynamic metrics for object oriented designs. In Software Metrics Symposium, pages 50-61, Boca Raton, Florida, USA, Nov 4-6 1999.
[10] E. Arisholm, L.C. Briand, and A. Foyen. Dynamic coupling measures for object oriented software. IEEE Transactions on Software Engineering, 30(8):491-506, 2004.
[11] Jakarta. The Apache Jakarta Project. http://jakarta.apache.org/.
[12] The Apache Ant Project. Ant. http://ant.apache.org/.
[13] The Apache XML Project. Xalan. http://xml.apache.org/xalan-j/.
[14] Anthony Hock-koon, Mourad Oussalah, Defining Metrics for Loose Coupling Evaluation in Service Composition, 2010 IEEE International Conference on Services Computing.
[15] D.P. Darcy, C.F. Kemerer, S.A. Slaughter, and T.A. Tomayko. The structural complexity of software: An experimental test. TOSE, 31(11):982-995, 2005.
[16] J.D. McGregor and D.A. Sykes. A Practical Guide to Testing Object-oriented Software. Addison Wesley, March 2001.
[17] L.C. Briand, W.L. Melo, and J. Wust. Assessing the applicability of fault proneness models across object-oriented software projects. IEEE Transactions on Software Engineering, 28(7):706-720, 2002.
[18] Mark D. Hansen, SOA using Java Web Services, Prentice Hall, Pearson Education, Inc, RR Donnelley in Crawfordsville, Indiana, April 2007.
[19] Mikhail Perepletchikov and Caspar Ryan, A Controlled Experiment for Evaluating the Impact of Coupling on the Maintainability of Service-Oriented Software, IEEE transactions on software engineering, vol. 37, no. 4, July/August 2011.
[20] R.T. Alexander and J. Offutt. Coupling-based testing of O-O programs. *The Journal of Universal Computer Science*, 10(4):391-427, 2004.
[21] S. Husein and Alan Oxley, A Coupling and Cohesion Metrics Suite for Object-Oriented Software, 2009 International Conference on Computer Technology and Development, IEEE.
[22] Simon Allier, Stephane Vaucher, Bruno Dufour, and Houari Sahraoui, Deriving Coupling Metrics from Call Graphs, 2010 Working Conference on Source Code Analysis and Manipulation, 978-0-7695-4178-5/10 , 2010 IEEE ,DOI 10.1109/SCAM.2010.25.
[23] Y.K. Malaiya, M.N. Li, J.M. Bieman, and R. Karcich. Software reliability growth with test coverage. *IEEE Transactions on Reliability*, 51(4):420426, December 2002.