ISSN:0975-9646

V.S.Bidve et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 3 (2) , 2012,3839-3842

Simplified Coupling Metrics for Object-Oriented Software

V.S.Bidve , Akhil Khare

Information Technology Department, BVCOE, Pune, India

Abstract— **Coupling in software has been linked with maintainability and existing metrics are used as predictors of external software quality attributes such as fault-proneness, impact analysis, ripple effects of changes, changeability, etc. Many coupling measures for object-oriented (OO) software have been proposed, each of them capturing specific dimensions of coupling.**

In this paper, we describe and evaluate some recently innovated coupling metrics for object-oriented (OO) design. We present an investigation into the run-time behavior of objects in Java programs, using specially adapted coupling metrics. These new metrics seek to quantify coupling at different layers of granularity that is at class-class and object-class level. For each measure, we indicate the type of coupling it uses what factors determine the strength of coupling, if it is an import or export coupling measure how indirect coupling is accounted for and how inheritance is dealt.

Keywords— **metrics, coupling, object-oriented, measurement, class**

I. INTRODUCTION

Increasingly, object-oriented measurements are being used to evaluate and predict the quality of software. A growing body of empirical results supports the theoretical validity of these metrics [1]. The validation of these metrics requires convincingly demonstrating that (1) the metric measures what it purports to measure (for example, a coupling metric really measures coupling) and (2) the metric is associated with an important external metric, such as reliability, maintainability and fault-proneness [2]. 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. Several of Chidamber and Kemerer's OO metrics appear to be useful to predict class fault-proneness during the early phases of the life-cycle [3]. There is a need of comprehensive framework of coupling measurement which includes all aspect of coupling. The components like inheritance and polymorphism are essential for dynamic coupling measurement [4, 5].

In this paper we consider the unified framework proposed by Lionel C. Briand, John W. Daly, and Jurgen Wust [6]. Most of measures considered for implementation are from the unified framework.

The following section outlines the related work for object-oriented coupling metrics. Section 3 describes our approach and the proposed measures. In section 4 we describe the proposed metrics with all is features. In section 5 we describes implementation details of the tool that we developed to compute our metrics as well as mathematical properties of the measures. Section 6 concludes the paper and discusses the future work.

II. RELATED WORK

Coupling measurement is a very rich and interesting body of research work, resulting in many different approaches using structural coupling metrics [7, 2, 8, 9], dynamic coupling measures [12], evolutionary and logical coupling [10, 11], coupling measures based on information entropy approach [2], coupling metrics for specific types of software applications like knowledge based systems [13], and more recently systems developed using aspect-oriented approach [14].The structural coupling metrics have received significant attention in the literature.

These metrics are comprehensively described and classified within the unified framework for coupling measurement [6]. The best known among these metrics are CBO (coupling between objects) and CBO1 [2, 8], RFC (response for class) [2] and RFC∞ [8], MPC (message passing coupling) [15], DAC (data abstraction coupling) and DAC1 [15], ICP (information-flow-based coupling) [9], the suite of coupling measures by Briand et al. (IFCAIC, ACAIC, OCAIC, FCAEC, etc) [7]. Other structural metrics like Ce (efferent coupling), Ca (afferent coupling), COF (coupling factor), etc. are also overviewed in [6]. Many of the coupling measures listed above are based on method invocations and attribute references. For example, the RFC, MPC, and ICP measures are based on method invocations only. CBO and COF measures count method invocations and references to both methods and attributes. The suite of measures defined by Briand et al. [7] captures several types of interactions between classes like class-attribute, class-method, as well as method-method interactions. The measures from the suite also differentiate between import and export coupling as well as other types of relationships like friends, ancestors, descendants etc.

Dynamic coupling measures were introduced as the refinement to existing coupling measures due to gaps in addressing polymorphism, dynamic binding, and the presence of unused code by static structural coupling measures [4].

III. PROPOSED MEASURES

In this section, we are discussing a framework proposed in unified framework for coupling measurement for coupling in object-oriented systems from implementation point of view. The objective of the unified framework is to support the comparison and selection of existing coupling measures with respect to a particular measurement goal [6].

The framework consists of six criteria, each criterion determining one basic aspect of the resulting measure. Out of these six criteria we are considering five criteria for implementation [6].

The six criteria of the framework are:

1. The type of connection, i.e., what constitutes coupling.
2. The locus of impact, i.e., import or export coupling.
3. Granularity of the measure: the domain of the measure and how to count coupling connections.
4. Stability of server.
5. Direct or indirect coupling.
6. Inheritance: inheritance-based vs. non-inheritance-based coupling, and how to account for polymorphism, and how to assign attributes and methods to classes.

These criteria are necessary to consider when specifying a coupling measure. Here we discuss the above criteria with its meaning; also measures under each criterion are listed out in the following discussion. Here we are trying to simplify each criterion with the help of its meaning. Measure under each criteria is selected which has minimum or no overlapping with other measures. Here we are trying to avoid the redundancy in the dynamic coupling measurement. Each measure selected in this section will be measured in the implementation section of this paper using real time object oriented application.

1. *The type of connection: It is mechanism by which two classes are coupled. The coupling can be due various mechanisms which are given in the Table 1.*

TABLE I
MEASURES SELECTED FROM TYPES OF CONNECTION

Mechanism	**Measures considered under unified framework**	**Measures considered in this paper**
Attribute in one class is of another class type	DAC, DAC´, IFCAIC, ACAIC, OCAIC,FCAEC, DCAEC, OCAEC	DAC
Method in one class has a type of parameter of other class type	IFCMIC, ACMIC, OCMIC, FCMEC,DCMEC, OCMEC	-
Local variable of a method of one class is of another class type	-	
Parameter of method of one class is of another class type	-	
Method of one class references attribute of another class type	CBO, CBO´, COF	CBO, COF
Method of one class invokes method of another class	CBO, CBO´, RFC□, RFC, RFC´, MPC,COF, ICP, NIH-ICP, IH-ICP, OMMIC,IFMMIC, AMMIC, OMMEC, FMMEC,DMMEC	RFC, MPC, COF, ICP
One class uses another class	-	-

In the Table 1 there are seven mechanisms of coupling are given and each mechanism comprises many types of measures. The first mechanism has DAC, DAC´ and other component coupling type of measures. All these measures have class-attribute interaction also DAC´ count classes used as a type of attributes. Definition of DAC tells the same thing (count number of attribute and parameter having a class type). So we are considering DAC only to avoid overlapping of measures for first mechanism. Similarly we are considering CBO, COF, RFC, ICP measures only from the table 1 in order avoid redundancy and overlapping of measures between similar mechanisms.

2. Locus of impact: it is nothing but direction of request for coupling.

Import: classes, methods, attributes in a role of client (users).

Export: classes, methods, attributes in a role of server.

TABLE 2
IMPORT AND EXPORT COUPLING MEASURES

Direction	**Measures considered under unified framework**	**Measures considered in this paper**
Import	CBO, CBO´, RFC□, RFC, RFC´, MPC, DAC, DAC´, COF, ICP, IH-ICP, NIH-ICP, IFCAIC, ACAIC, OCAIC, IFCMIC, ACMIC, OCMIC, IFMMIC, AMMIC, OMMIC	Import
Export	CBO, CBO´, COF, FCAEC, DCAEC, OCAEC, FCMEC, DCMEC, OCMEC, OMMEC, FMMEC, DMMEC	Export

There are many measures under import and export category but there should be some separate count of import and export coupling for a class. The separate count is useful in order to predict quality using export and import coupling. So we are taking import and export as separate types of measures.

3. Granularity: level of detail at which information is gathered.

All measures considered in this category under unified framework are already considered in other criteria measures of this paper. So no measure is new under this criterion.

4. Stability of server class: how stable the class is.

Two different category of class stability

- Unstable Classes: these are classes which are subject to development or modification in the project.
- Stable Classes: classes that are not subject to change in the project.

Most of classes are unstable classes and there is no point to consider library classes in the stability measurement. So we are not considering any measure under this category.

5. Direct or indirect connections:

All measures considered in this paper are direct measures. The indirect measure considered in unified framework is RFC´. The same measure we are considering here to count indirect coupling.

6. Inheritance:

There are four options given in unified framework to deal with inheritance. The four options are listed below.

- count inheritance-based coupling only
- count non-inheritance-based coupling only
- count inheritance-based and non-inheritance based coupling separately
- count inheritance-based and non-inheritance based coupling, making no distinction

Here we count the coupling due to inheritance only (option 1) by using option 4 and option 2. Also we are taking one more measure i.e. depth of inheritance (DIT) which will be useful to count maximum inheritance path from the class to the root class [7, 3].

Polymorphism:

The second point which is more important is polymorphism. There are many measures which accounts for polymorphism are considered in unified framework like CBO, CBO′, RFC∞, RFC, RFC′, COF. So we are not considering any additional measure for polymorphism.

Apart from these measures we are considering some more measures which are useful measures of software quality metrics.

1. Weighted methods per class (WMC): It measures the total number of methods defined in class. A high WMC has been found to lead to more faults [7, 3]**.**
2. Number of Children (NOC): Number of immediate sub-classes of a class. High NOC has been found to indicate fewer faults. This may be due to high reuse, which is desirable [7, 3].

The use of each measure is already explained by previous authors so we are not going in those details. We are directly implementing these measurements and checking the values using java package.

IV. CONSTRUCTION OF THE COUPLING MEASURES

In the above section we selected required measures for the implementation in our system. The selected measures are formalized in this section in order to implement those measures and in the next section of this paper. The measures are given in the table 3.

V. SYSTEM ARCHITECTURE AND RESULTS

To implement measures described in table 4. We have developed a java code which analyses the java packages and find out the values of above measures. Occurrence of event increments the value of measure by one, also an event can have one or more aspects related with it as given in table 4. Every measure is collected class-wise.

We have developed five chief classes in our system to find out measurement values. The classes are given with their functions below,

1. MetricsFilter.java: This class collects the classes from given package.
2. ClassVisitor.java: This class works as metrics container for all classes.
3. MethodVisitor.java: This class works as visitor of the class the method.
4. ClassesMetrics.java: Collects details needed for calculating a class's metrics.
5. ClassMetricsContainer.java: Store metrics of all visited classes.

For our java project any java package can be used as an input. Here, we are taking our project itself as input for measurement. This project contains many packages we are showing the result of measurement of only one package i.e. cm.metrics package in the table 4.

Interpretation of the results

As shown in table 4 we can collect all the measures from the definitions provided by unified framework and Chidamber & Kemerer metrics suite. The measures which are considered here are sufficient to predict all quality attributes. Most of the redundant measures are avoided in this work.

TABLE 3
PROPOSED COUPLING MEASURES

Measure	**Events**	**Aspects considered under measure**
CBO	Methods invocation, attribute reference.	Inheritance, import, export, polymorphism.
COF	Methods invocation, attribute reference.	Import, export, polymorphism.
RFC	Methods invocation.	Inheritance, import, polymorphism.
MPC	Methods invocation.	Inheritance, import.
ICP	Methods invocation.	Parameter passed, inheritance, import.
DAC	Attribute reference.	Inheritance, import.
RFC'	Methods invocation.	Inheritance, import, polymorphism, indirect coupling.
IMPORT	Methods invocation, attribute reference, class used.	Every imported event
EXPORT	Methods invocation, attribute reference, class used.	Every exported event
Coupling due to inheritance only	Methods invocation, attribute reference.	Coupling due to all aspects including inheritance (count each aspect once) – coupling due all aspects except inheritance.
DIT	----------	maximum inheritance path from the class to the root class
WMC	----------	number of methods defined in class
NOC	----------	number of immediate sub-classes of a class

TABLE 4
CLASS WISE COUNT RESULTS OF EACH MEASURE USING OUR JAVA PROJECT

↓Measure \ class →	ClassVisit-or	Test	ClassMetrics-Container	PrintPlain-Result	Output-Handler	Metrics-Filter	Method-Visitor	Class-Metrics
CBO	178	0	18	3	1	40	129	0
COF	176	0	18	3	1	40	127	0
RFC	95	5	25	8	1	39	40	59
MPC	10	0	2	1	1	2	10	0
ICP	84	0	9	1	0	27	41	0
DAC	12	0	3	2	1	19	16	0
RFC’	104	7	30	12	1	46	43	64
IMPORT	14	0	3	2	1	7	21	0
EXPORT	2	0	5	2	5	4	1	7
Coupling due to inheritance only	2	0	0	0	0	0	2	0
DIT	3	2	2	2	2	2	3	2
WMC	18	2	5	2	1	9	11	48
NOC	0	0	0	0	0	1	0	1

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we have analyzed two coupling metrics proposed by unified framework [paper] and Chidamber & Kemerer. We have selected and the measures which are sufficient to predict complexity of object-oriented software. We have collected the class-wise values of each measure from our code implemented in java. The values of each measure are useful to analyze the complexity of any class.

The paper simplified the work of coupling measurement. The proposed metrics could be further refined by taking more detailed formalism for each measure.

REFERENCES

[1] Abreu, F. B. e., "The MOOD Metrics Set," presented at ECOOP '95 Workshop on Metrics, 1995.

[2] Chidamber, S. R. and Kemerer, C. F., "Towards a Metrics Suite for Object Oriented Design", in Proceedings of OOPSLA'91, 1991, pp. 197-211.

[3] S.R. Chidamber, C.F. Kemerer, “Towards a Metrics Suite for Object Oriented design”, in A. Paepcke,(ed.) *Proc. Conference on Object-Oriented Programming: Systems, Languages and Applications* (OOPSLA’91), October 1991. Published in SIGPLAN Notices, 26 (11), 197-211, 1991.

[4] Harrison, R., Counsell, S. J., and Nithi, R. V., "An Evaluation of the MOOD Set of Object-Oriented Software Metrics," *IEEE Transactions on Software Engineering*, vol. 24, pp. 491-496, June 1998.

[5] Briand, L. C., Daly, J., and Wüst, J., "A Unified Framework for Coupling Measurement in Object Oriented Systems", *IEEE Transactions on Software Engineering*, vol. 25, no. 1, January 1999, pp. 91-121.

[6] Briand, L. C., Devanbu, P., and Melo, W. L., "An investigation into coupling measures for C++", in Proc. Of International Conference on Software engineering (ICSE'97), Boston, MA, May 17-23 1997, pp. 412 - 421.

[7] Chidamber, S. R. and Kemerer, C. F., "A Metrics Suite for Object Oriented Design", IEEE Transactions on Software *Engineering*, vol. 20, no. 6, 1994, pp. 476-493.

[8]] Lee, Y. S., Liang, B. S., Wu, S. F., and Wang, F. J., "Measuring the Coupling and Cohesion of an Object-Oriented Program Based on Information Flow", in Proceedings of International Conference on Software Quality, Maribor, Slovenia, 1995.

[9] Gall, H., Jazayeri, M., Krajewski, J., "CVS Release History Data for Detecting Logical Couplings", *6th* International Workshop on Principles of Software Evolution *(IWPSE'03)* Sept. 1 - 2, 2003, pp. 13 - 23.

[10] Zimmermann, T., Zeller, A., Weissgerber, P., and Diehl, S., "Mining Version Histories to Guide Software Changes", IEEE Transactions on Software Engineering, vol. 31, no. 6, June 2005, pp. 429-445.

[11] Arisholm, E., Briand, L. C., and Foyen, A., "Dynamic coupling measurement for object-oriented software", *IEEE* Transactions on Software Engineering, vol. 30, no. 8, August 2004, pp. 491-506.

[12] Kramer, S. and Kaindl, H., "Coupling and cohesion metrics for knowledge-based systems using frames and rules", ACM Trans. on Soft. Engineering and Methodology *(TOSEM)*, vol. 13, no. 3, July 2004, pp. 332-358.

[13] Zhao, J., "Measuring Coupling in Aspect-Oriented Systems", in Proc. of 10th IEEE International Soft. Metrics Symposium (METRICS'04), Chicago, USA, 2004.

[14] Li, W. and Henry, S., "Object-oriented metrics that predict maintainability", *Journal of Systems and Software*, vol. 23, no. 2, 1993, pp. 111-122

[15] Basili, V. R., Briand, L. C., and Melo, W. L., "A Validation of Object Orient Design Metrics as Quality Indicators," *IEEE Transactions on Software Engineering*, vol. 21, pp. 751-761, 1996.

[16] Briand, L., Emam, K. E., and Morasca, S., "Theoretical and Empirical Validation of Software Metrics," 1995.

[17] Briand, L., Ikonomovski, S., Lounis, H., and Wust, J., "Measuring the Quality of Structured Designs," *Journal of Systems and Software*, vol. 2, pp. 113-120, 1981.

[18] Schneidewind, N. F., "Methodology for Validating Software Metrics," *IEEE Transactions on Software Engineering*, vol. 18, pp. 410-422, 1992.

[19] Jacobson, I., Christerson, M., Jonsson, P., and Overgaard, G., *Object-Oriented Software Engineering: A Use Case Driven Approach*. Wokingham, England: Addison-Wesley, 1992.

[20] Korson, T. D. and Vaishnavi, V. K., "An Empirical Study of Modularity on Program Modifiability," *Empirical Studies of Programmers*, pp. 168-86, 1986.

AUTHORS

V. S. Bidve: Computer engineering from University of Aurangabad and the M. Tech pursuing from BVUCOE, Pune. He has nine years of teaching experience in Pune and Mumbai. He is now working as a lecturer in the Department of information technology SKNCOE, Pune.

A. **R. Khare**: Has completed bachelor degree in computer engineering from Bhopal University, India and M. Tech. from same University. Pursuing Ph. D. from JNU, Jodhpur In the field of computer engineering. Working as Assistance Professor in Information Technology Department of BVCOE, Pune. Having 10+ years of teaching experience. Working as a PG coordinator for IT department and guiding number of students for their project work and various academic activities.