ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 11, May 2014

Coupling Appraisal in Object-Oriented Systems

Priya Nigam, Rachna Mishra
Department of Computer Science & Engg.
Oriental College of Technology, Raisen Road, Bhopal

Abstract*— The metrics "Coupling is a quantification of interdependence of two objects. Coupling in system has been interacted with maintainability and subsisting metrics suite are utilized as prognosticator of extramural system quality attributes such as fault-propensity, impact scrutiny, undulate effects of changes, variability, etc. Many coupling quantifications for object-oriented (OO) systems have been propound, each of them apprehend peculiar dimensions of coupling. Current system engineering gives attention towards only mundane deportment with postulation that all faults can be abstracted during development. In this paper, we describe and evaluate some recently innovated coupling metrics for object-oriented (OO) design. We present an exploration into the run-time comportment of objects in Java programs, utilizing specially habituated coupling metrics. These incipient metrics seek to quantify coupling at different layers of gravelly that is at class-class and object -class level. For each quantification, we designate the type of coupling it utilizes what factors determine the vigor of coupling, if it is an import or export coupling quantification how indirect coupling is accounted for and how inheritance is dealt.

***Keywords*— system metrics, coupling, object-oriented, appraisal, class**

I. INTRODUCTION

Increasingly, object-oriented appraisals are being utilized to evaluate and predict the quality of system. A growing body of empirical results supports the theoretical validity of these system metrics [1]. The validation of these metrics requires convincingly demonstrating that (1) the metric quantifications what it purports to quantification (for example, a coupling metric really quantifications coupling) and (2) the metric is associated with an important extramural metric, such as reliability, maintainability and fault-propensity [2]. Often these metrics have been utilized as an early indicator of these extramurally visible attributes, because the extramurally visible attributes could not be quantifications until too late in the system development process. Several of Chidamber and Kemerer's OO metrics appear to be useful to predict class fault-propensity during the early phases of the life-cycle [3]. There is a need of comprehensive framework of coupling appraisal which includes all aspect of coupling. The components like inheritance and polymorphism are essential for dynamic coupling appraisal [4, 5].

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

This paper uses metrics and complexity terms as defined by IEEE standards. The IEEE defines a *quality factor* to be an attribute of a program that aspects its quality [37, 38]. These were called *extramural attributes* by Briand et al. [15]. Examples include maintainability, reliability, and complexity. Factors are often qualities on which we wish to base a decision. Some can be quantification directly and some cannot. For example, Schneidewind defines a factor as a type of metric that provides a direct quantification of a software quality" [60], implying that he assumes factors can be quantification. A *quality metric* is an attribute that can be quantification by one or more functions whose inputs are data obtained from software artefacts (such as design documents, program source, or requirements documents). These were called *internal attributes* by Briand et al. [15]. An *appraisal* is a specific function for computing values for a metric. Metrics often represent information that is not directly relevant to the developers (such as how many executable lines are in a program), but are utilized to *estimate* factors that are directly relevant to developers (such as how hard it is to change the program). A metric is said to be *validated* for a factor if it has been statistically shown to accurately estimate the factor [60]. Only measurable factors can be validated {we cannot show statistically what we cannot quantification.

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

II. RELATED WORK

Coupling appraisal 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 quantifications [12], evolutionary and logical coupling [10, 11], coupling quantifications based on information entropy approach [2], coupling metrics for peculiar types of system 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

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 11, May 2014

appraisal [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 quantifications 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 quantifications listed above are based on method invocations and attribute references. For example, the RFC, MPC, and ICP quantifications are based on method invocations only. CBO and COF quantifications count method invocations and references to both methods and attributes. The suite of quantifications 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 quantifications from the suite also differentiate between import and export coupling as well as other types of relationships like friends, ancestors, descendants etc.

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

III. PROPOUND QUANTIFICATIONS

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

In this section, we are discussing a framework propound in unified framework for coupling appraisal 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 quantifications with respect to a particular appraisal goal [6]. The six criteria of the framework are:

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

Now we have developed a tool for the conceptual coupling appraisal, named IRC^{2}M (**I**nformation **R**etrieval based **C**onceptual **C**oupling **Measurement**), to automate the computation of the conceptual coupling quantifications for C++ programs. IRC^{2}M's indexing component is based on IRiSS, an IR-based tool utilized for source code browsing and exploration. IRC^{2}M employs the following steps to compute CoCC (see Figure 1):

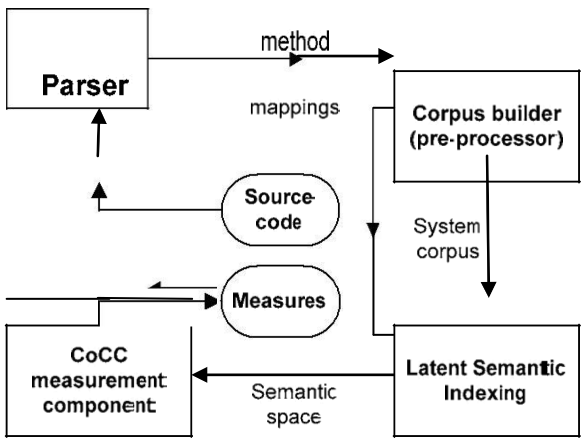

Fig 1. Architecture of the IRC^{2}M tool

The source code is parsed and the system corpus is constructed by including methods from the source code as documents. For each method in the software system, there will be one document in the corpus. Mappings between methods, classes, and their indexes respectively in the system corpus are generated in this step. Pre-processing of the system corpus is performed to eliminate common keywords, stop words, and to split identifiers;

- LSI constructs a term-by-document matrix from the generated system corpus. Then it applies SVD to this matrix to construct the LSI subspace. New document vectors are obtained by orthogonally projecting the corresponding vectors from the original vector space onto the new LSI semantic space.
- Once the methods from the software system are represented in the LSI space, the conceptual similarities between methods are computed. The CoCC and CoCC$_m$ quantifications use different measuring mechanisms to determine how the classes are related conceptually in the context of the software system. CSBC and CSBC$_m$ are also computed in this last step.

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

- 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.QUANTIFICATIONS SELECTED FROM TYPES OF CONNECTION

Mechanism	Quantifications considered under unified framework	Quantifications 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 quantifications. The first mechanism has DAC, DAC´ and other component coupling type of quantifications. All these quantifications have class-attribute interaction also DAC´ count classes utilized 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 quantifications for first mechanism. Similarly we are considering CBO, COF, RFC, ICP quantifications only from the table 1 in order avoid redundancy and overlapping of quantifications between similar mechanisms.

- ***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 which is given in table 2.s

TABLE 2. IMPORT AND EXPORT COUPLING QUANTIFICATIONS

Direction	Quantifications considered under unified framework	Quantifications 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 is much quantification 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 quantifications.

- ***Granularity: level of detail at which information is gathered***

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

- ***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 appraisal. So we are not considering any quantification under this category.

Direct or indirect connections

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

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 quantification 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 quantifications which accounts for polymorphism are considered in unified framework like CBO, CBO′, RFC∞, RFC, RFC′, COF. So we are not considering any additional quantification for polymorphism.

Apart from these quantifications we are considering some more quantifications which are useful quantifications of system quality metrics.

- Weighted methods per class (WMC): It quantifications the total number of methods defined in class. A high WMC has been found to lead to more faults [7, 3]**.**
- 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 quantification is already explained by previous authors so we are not going in those details. We are directly implementing these appraisals and checking the values using java package.

IV. CONSTRUCTION OF THE COUPLING QUANTIFICATIONS

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

Quantification	**Events**	**Aspects considered under quantification**
CBO	Methods invocation, attribute References	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 utilized.	Every imported event
EXPORT	Methods invocation, attribute reference, class utilized.	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

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 11, May 2014

TABLE 3. PROPOUND COUPLING QUANTIFICATIONS

V. SYSTEM ARCHITECTURE AND RESULTS

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

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

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

For our java project any java package can be utilized as an input. Here, we are taking our project itself as input for appraisal. This project contains many packages we are showing the result of appraisal 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 quantifications from the definitions provided by unified framework and Chidamber & Kemerer metrics suite. The quantifications which are considered here are sufficient to predict all quality attributes. Most of the redundant quantifications are avoided in this work.

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

↓Quantification \ 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

REFERENCES

[1] N.Kayarvizhy and S. Kanmani "Analysis of Quality of Object Oriented Systems using objects oriented Matrices" IEEE 2011 978-1-4244-8679-3/11.

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

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

[4] 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.

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

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

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

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

[9]] 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 System Quality, Maribor, Slovenia, 1995.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 11, May 2014

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

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

[12] Arisholm, E., Briand, L. C., and Foyen, A., "Dynamic coupling appraisal for object-oriented system", IEEE Transactions on System Engineering, vol. 30, no. 8, August 2004, pp. 491-506.

[13] 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.

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

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

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

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

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

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

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

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