Global Journal of Computer Science and Technology
Software & Data Engineering
Volume 12 Issue 13 Version 1.0 Year 2012
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Impact of Mediated relations as Confounding Factor on Cohesion and Coupling Metrics: For Measuring Fault Proneness in Oo Software Quality Assessment

By Amjan.Shaik, Dr.C.R.K.Reddy & Dr.A.Damodaram
JNTUH, Hyderabad, Andhra Pradesh, India

Abstract - Mediated class relations and method calls as a confounding factor on coupling and cohesion metrics to assess the fault proneness of object oriented software is evaluated and proposed new cohesion and coupling metrics labeled as mediated cohesion (MCH) and mediated coupling (MCO) proposed. These measures differ from the majority of established metrics in two respects: they reflect the degree to which entities are coupled or resemble each other, and they take account of mediated relations in couplings or similarities. An empirical comparison of the new measures with eight established metrics is described. The new measures are shown to be consistently superior at measure the fault proneness.

GJCST-C Classification: *D.2.8*

Strictly as per the compliance and regulations of:

Impact of Mediated relations as Confounding Factor on Cohesion and Coupling Metrics: For Measuring Fault Proneness in Oo Software Quality Assessment

Amjan.Shaik[α], Dr.C.R.K.Reddy[σ] & Dr.A.Damodaram[ρ]

Abstract - Mediated class relations and method calls as a confounding factor on coupling and cohesion metrics to assess the fault proneness of object oriented software is evaluated and proposed new cohesion and coupling metrics labeled as mediated cohesion (MCH) and mediated coupling (MCO) proposed. These measures differ from the majority of established metrics in two respects: they reflect the degree to which entities are coupled or resemble each other, and they take account of mediated relations in couplings or similarities. An empirical comparison of the new measures with eight established metrics is described. The new measures are shown to be consistently superior at measure the fault proneness.

I. Introduction

Object Oriented (OO) design and code, for instance, [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]. These metrics offer ways to evaluate the excellence of software and their use in former phases of software development can help organizations in evaluating large software development quickly, at a low cost [3]. But how do we know which metrics are functional in capturing important quality attributes such as Degree of Fault prone, effort, efficiency or amount of maintenance adaptations. Experiential studies of real systems can provide relevant answers. There have been few empirical studies evaluating the effect of object-oriented metrics on software quality and constructing models that utilize them in predicting quality attributes in the system, such as [16, 17, 18, 19, 5, 20, 21, 22, 23, 8, 12, 24]. More data based by empirical studies, which are capable of being verified by observation or experiment are needed. The evidence gathered through these empirical studies is today considered to be the most powerful support possible for testing a given hypothesis.

A well designed component, in which the functionality has been appropriately distributed to its various subcomponents, is more likely to be fault free and will be easier to adapt. Appropriate distribution of function underlies two key concepts of object-oriented design: coupling and cohesion. Coupling is the extent to which the various subcomponents interact. If they are highly interdependent then changes to one are likely to have significant effects on the behavior of others. Hence loose coupling between its subcomponents is a desirable characteristic of a component. Cohesion is the extent to which the functions performed by a subsystem are related. If a subcomponent is responsible for a number of unrelated functions then the functionality has been poorly distributed to subcomponents. Hence high cohesion is a characteristic of a well designed subcomponent.

Many metrics have been proposed to measure the coupling and cohesion to predict the fault-prone and maintainability of software. However, few studies had been done using coupling and cohesion to assess the quality of components.

In this context we therefore analyzed the mediated relations of the classes and method calls as a confounding factor for coupling and cohesion metrics and proposing two new metrics called Mediated coupling and Mediated cohesion to measure the fault proneness to assess the quality of the software.

The rest of the paper organized as, in section II the traditional cohesion and coupling metrics revealed, which followed by section III that explores transitivity as a confounding factor.

II. The Coupling and Cohesion in OO Programming

a) Measuring Coupling

The term coupling is usually used in a derogatory manner in design review meetings. Even so, it's not possible to design aefficient OO application without coupling. At any time if one object interacts with another object, then it is coupling. In reality, what you need to try to minimize is coupling factors. Strong coupling means that one object is strongly coupled with the implementation details of another object. Strong coupling is discouraged because it results in less

Author α : Research Scholar, Department of CSE, JNTUH, Hyderabad, Andhra Pradesh, India. E-mail : amjan_shahi@yahoo.com
Author σ : Professor and HOD of CSE CBIT, Gandipet, Hyderabad, Andhra Pradesh, India. E-mail : crkreddy@gmail.com
Author ρ : Professor of CSE and Director, Academic Audit Cell, JNTUH, Kukatpally, Hyderabad, Andhra Pradesh, India.
E-mail : damodarama@rediffmail.com

flexible, less scalable application software. However, coupling can be used so that it enables objects to talk to each other while also preserving the scalability and flexibility.

Though this seems like a difficult task, OO metrics can help you to measure the right level of coupling.

Coupling between Objects (CBO): CBO is defined as the number of non-inherited classes associated with the target class. It is counted as the number of types that are used in attributes, parameters, return types, throws clauses, etc. Primitive types and system types (e.g. Java.lang.*) is not counted.

Data Abstraction Coupling (DAC): DAC is defined as the total number of referring types in attribute declarations. Primitive types, system types, and types inherited from the super classes are not counted.

Method Invocation Coupling (MIC): MIC is defined as the relative number of classes that receive messages from a particular class.

$$MIC = \frac{nMIC}{N-1}$$

Where N isthe total number of classes defined within the project.

nMIC is the total number of classes that receive a message from the target class.

Demeter's Law: Ian Holland first proposed the Law of Demeter. The class form of Demeter's Law has two versions: a strict version and a minimized version. The strict form of the law states that every supplier class of a method must be a preferred supplier. The minimization form is more permissive than the first version and requires only minimizing the number of acquaintance classes of each method.

Definition 1 (Client): Method M is a client of method f attached to class C, if in M message f is sent to an object of class C, or to C. If f is specialized in one or more subclasses, then M is only a client of f attached to the highest class in the hierarchy. Method M is a client of some method attached to C.

Definition 2 (Supplier): If M is a client of class C then C is a supplier to M. In other words, a supplier class to a method is a class whose methods is called in the method. In Listing 1, the Product class is a supplier class to the client class Order.

Definition 3 (associate Class): A class $C1$ is an acquaintance class of method M attached to class $C2$, if $C1$ is a supplier to M and $C1$ is not one of the following:

The same as $C2$;

A class used in the declaration of an argument of M

A class used in the declaration of an instance variable of $C2$

Definition 4 (Preferred-supplier class): Class B is called a preferred-supplier to method M (attached to the class C) if B is a supplier to M and one of the following conditions holds:

B is used in the declaration of an instance variable of C

B is used in the declaration of an argument of M, including C and its super classes.

B is a preferred acquaintance class of M.

b) Measuring Cohesion

In OO methodology, classes contain certain data and exhibit certain behaviors. This concept may seem fairly obvious, but in practice, creating well-defined and cohesive classes can be tricky. Cohesive means that a certain class performs a set of closely related actions. A lack of cohesion, on the other hand, means that a class is performing several unrelated tasks. Though lack of cohesion may never have an impact on the overall functionality of a particular class—or of the application itself—the application software will eventually become unmanageable as more and more behaviors become scattered and end up in the wrong places.

Thus, one of the main goals of OO design is to come up with classes that are highly cohesive. Luckily, there's a metric to help to verify that the designed class is cohesive.

The LCOM Metric: Lack of Cohesion in Methods

The Lack of Cohesion in Methods metric is available in the following three formats:

LCOM1: Take each pair of methods in the class and determine the set of fields they each access. If they have disjointed sets of field accesses, the count P increases by one. If they share at least one field access, Q increases by one. After considering each pair of methods:

$$\mathrm{RESULT} = (\mathrm{P} > \mathrm{Q})?\ (\mathrm{P} - \mathrm{Q}):\ 0$$

A low value indicates high coupling between methods. This also indicates the potentially high reliability and good class design. Chidamber and Kemerer provided the definition of this metric in 1993.

LCOM2: This is an improved version of LCOM1. Say you define the following items in a class:

m : number of methods in a class

a : number of attributes in a class.

mA : number of methods that access the attribute a.

$\text{sum}(mA)$: sum of all mA over all the attributes in the class.

$$LCOM2 = 1 - \text{sum}(mA)/(m*a)$$

If the number of methods or variables in a class is zero (0), LCOM2 is undefined as displayed as zero (0).

LCOM3: This is another improvement on LCOM1 and LCOM2 and is proposed by Henderson-Sellers. It is defined as follows:

$$LCOM3 = (m - \text{sum}(mA)/a) / (m-1)$$

where m, a, mA, $\text{sum}(mA)$ are as defined in LCOM2.

The following points should be noted about LCOM3:

The LCOM3 value varies between 0 and 2. LCOM3>1 indicates the shortage of cohesion and is considered a kind of alarm.

If there is only one method in a class, LCOM 3 is undefined and also if there are no attributes in a class LCOM3 is also undefined and displayed as zero (0).

Each of these different measures of LCOM has a unique way to calculate the value of LCOM.

An extreme lack of cohesion such as LCOM3>1 indicates that the particular class should be split into two or more classes.

If all the member attributes of a class are only accessed outside of the class and never accessed within the class, LCOM3 will show a high-value.

A slightly higher value of LCOM means that you can improve the design by either splitting the classes or re-arranging certain methods within a set of classes.

III. Mediated Relations of Classes and Method Calls as Confounding Factor

a) Confounding Factor

The term confounding refers to a situation in which an association between an independent variable and a dependent variable is thought to be the result of the influence of a third variable[17]. The suggestion is that an apparent association between the independent and dependent variables may be partly or completely accounted for by a third variable. By the same token, the absence of an apparent association between independent and dependent variables may be the result of a failure to account for the effects of a third variable. The third variable that distorts the true association between the independent and dependent variables is usually called a confounding variable. The distortion that results from perplexing may lead to overestimation or underestimation of an association, depending on the direction and magnitude of the relations that the confounding variable has with the independent and dependent variables [18].

To quantitatively analyze the confounding factor, a number of confounding factor analysis models using various modeling techniques, such as linear, logistic, and probity regression, have been developed [16], [17], [19], [20], [21], [22]. Among these models, the confounding factor analysis model based on linear regression techniques has been widely used in health sciences and epidemiological research [16], [19], [20]. Compared to models based on other modeling techniques, the linear-regression-based model has two main advantages: 1) A number of statistical methods have been developed for this model to test for a confounding variable [16], [19] and 2) it is easy to determine whether a confounding variable leads to overestimation or underestimation of the true association between the independent and dependent variables [16], [20].

b) Mediated relation as dependent variable

The objective of this study is to empirically investigate to identify the cohesion and coupling metrics under consideration of mediated class relations and method calls as confounding factors and assessing the association between these cohesion and coupling metrics and degree of fault-proneDegree of Fault prone is an important external quality attribute and identifying faults-prone classes is very useful because: 1) It enables software developers to take focused preventive actions that can reduce maintenance costs and improve quality and 2) it helps software managers to allocate resources more effectively. In this study, Degree of Fault prone denotes the extent of class responsibility in component failure. We need to select the depth of the transitivity in class relations and method calls as the dependent variable for our study.

IV. Mediated Coupling Between Objects[MCBO]

We begin by regarding any object-oriented software system as a directed graph, in which the vertices are the classes comprising the system. Suppose such a system comprises a set of classes $C \equiv (C_i \in C \mid \{i = 1..m\})$. Let $m(C_j) \equiv \{m(C_j)_i \in m(C_j) \mid (i = 1..n)\}$ be the methods of the class C_j, and $mI_{(C_j \to C_i)}$ the set of methods and instance variables in class C_i invoked by class C_j for $j \neq i$. An edge from C_j to C_i exists if and only if the $mI_{(C_j \to C_i)} > 0$, which can be used to generate the weight of that directed edge. The graph is directed since $mI_{(C_j \to C_i)}$ is not necessarily equal to $mI_{(C_i \to C_j)}$. Let consider that $mI_{(C_j \to C)}$ is the set of all methods and instance variables in other classes of C that are invoked by class C_j. '$mI_{(C_j \to C)}$' can be represented as follows:

$$mI_{(C_j \to C)} = \bigcup_{i=1}^{m} mI_{(C_j \to C_i)}$$

a) Finding a Degree of Directed Coupling (DDC)

The directed edge weight $cw_{(C_j \to C_i)}$ between classes C_j and C_i can be represented as

$cw_{(C_j \to C_i)} = \frac{mI_{(C_j \to C_i)}}{mI_{(C_j \to C)}}$, the directed edge weight also can refer as degree of direct coupling (DDC) between two classes cw is always between 0 and 1.

b) Finding a degree of mediated coupling (DMC)

Based on this degree of direct coupling between two classes, we can generalize the process of detecting the degree of mediated coupling mcw between any two classes C_j and C_k exists such that $mI_{(C_j \to C_k)} \cong 0$, which follows:

$$mcw_{(C_j \to C_k, p)} = 1 - \frac{1}{\left(\sum_{i=1}^{|e_{j \to k}|} cw_i \right)} \; (iff \; \mathrm{mI}_{(C_j \to C_k)} \cong 0)$$

In above equation

$e_{j \to k}$ is the set of DDCs of edges, which are building path p between class C_j and C_k

cw_i is DDC of an edge i that belongs to $e_{j \to k}$.

p is one of the path out of set of paths P between C_j and C_k

c) Applying Confounding factor

The confounding factor of path p is $cf_{(p)}$, that assessed as follows:

$$cf_{(C_j \to C_k, p)} = \frac{|e(p)| - 1}{|e(p)|}$$

Here in the above equation

$e_{(p)}$ is set edges that belongs to the path p.

Then the generalized degree of mediated coupling between class C_j and C_k $mcw_{(C_j \to C_k)}$ can be found as follows

$$mcw_{(C_j \to C_k)} = 1 - \frac{1}{\left(\sum_{i=1}^{|P|} (mcw_{(C_j \to C_k, i)} + cf_{(C_j \to C_k, i)}) \right)}$$

The following hypothesis is a convention from the empirical study conducted on applications that are confirmed as fault prone:

If 'mcw'is the degree of mediated coupling between two objects O_1 and O_2 then $(mcw \times 100)\%$ is the percentage of O_1 and O_2 objects in application's fault proneness.

V. Mediated Cohesion (Mch)

The proposed cohesion metric is based on transitive function calls. The Hypothesis of the proposed cohesion metric can be defined as:

If a method A invoking a method B and method B is invoking method C, then the connection between A and C can be considerable and their cohesiveness is transitive if and only if A,B and C belongs to a same class or classes in an inheritance hierarchy .

We build a graph based on the function calls between the functions of the same class.

The edge between any two functions represents thetotal number of similar properties used similar functions invoked in both functions.

Finding Degree of Direct Cohesion(DDCH)

The Degree of Direct Cohesion Between two functions that represents the edge weight can be generalized as follows:

Let $pM_{(i)}$ is set of properties used in a method M_i of the class C such that $pM_i \in pC$ and $M_i \in mC$, here pC is set of properties declared in the class C and mC is a set of methods belongs to the class C. Let $mM_{(i)}$ is set of methods invoked in method M_i of the class C such that $mM_i \in mC$.

Let $pM_{(j)}$ is set of properties used in the method M_j of the class C such that $pM_j \in pC$ and $M_j \in mC$, here pC is set of properties declared in the class C and mC is a set of methods belongs to the class C. Let $mM_{(j)}$ is set of methods invoked in method M_j of the class C such that $mM_j \in mC$.

If $(M_j \in mM_i \parallel M_i \in mM_j)$ then there an edge exists between these two methods. The graph is not a directed graph, since edge weight is not changing under any direction of direct connection between the two functions. The DDCH that referred as edge weight can be measured as follows:

$$chw_{(M_i \oplus M_j)} = \frac{\left(\left(\frac{|pM_i \cap pM_j|}{|pM_i \cup pM_j|}\right)+\left(\frac{|mM_i \cap mM_j|}{|mM_i \cup mM_j|}\right)\right)-1}{\left(\left(\frac{|pM_i \cap pM_j|}{|pM_i \cup pM_j|}\right)+\left(\frac{|mM_i \cap mM_j|}{|mM_i \cup mM_j|}\right)\right)}$$

,here $1 \geq chw_{(M_i \oplus M_j)} \geq 0$

a) Finding Degree of Mediated Cohesion (DMCH)

Based on this degree of direct Cohesion between two methods, we can generalize the process of detecting the degree of mediated cohesion $mchw$ between any two methods M_j and M_k of same class exists such that $chw_{(M_j \oplus M_k)} \cong 0$, which follows:

$$mchw_{(M_j \oplus M_k, p)} = 1 - \frac{1}{\left(\sum_{i=1}^{|e_{j \to k}|} chw_i\right)}$$

$$(iff \ \text{chw}_{(M_j \oplus M_k)} \cong 0)$$

In above equation

$e_{j \to k}$ is the set of DDCHs of edges, which are building a path p between methods M_j and M_k

chw_i is DDCH of an edge i that belongs to $e_{j \to k}$.

p is one of the path out of set of paths P between M_j and M_k

b) Applying Confounding factor

The confounding factor of the path p is $cf_{(p)}$, that assessed as follows:

$$cf_{(M_j \oplus M_k, p)} = \frac{|e(p)|-1}{|e(p)|}$$

Here in the above equation

$e_{(p)}$ is set edges that belongs to the path p .

Then the generalized degree of mediated cohesion between methods M_j and M_k $mchw_{(M_j \oplus M_k)}$ can be found as follows

$$mchw_{(M_j \oplus M_k)} = 1 - \frac{1}{\left(\sum_{i=1}^{|P|} (mchw_{(M_j \oplus M_k, i)} + cf_{(M_j \oplus M_k, i)})\right)}$$

Since the class level cohesiveness is significant to predict the fault proneness than the method level cohesiveness.

The class level confounding factor of a class C measures as follows:

$$ccf_{(C)} = 1 - \frac{1}{\left(\frac{|P'|}{|mC|}\right)}$$

Since the the ration between number paths build in the graph and number of methods exists indicates the cohesiveness, if the majority of paths between same classes can be considered as a confounding factor. Hence the above equation justifies the measurement of the class level confounding factor.

Here in this equation $|P'|$ represents the total number of paths build between the methods of a class C, $|mC|$ total number of methods in class C.

Then the class level mediated cohesiveness can be measured as follows:

$$mchw_{(C)}=1-\frac{1}{\left(\sum_{i=1}^{|P'|} mchw_{(p_i)}\right)+ccf(C)}$$

Here in the above equation

$mchw_{(p_i)}$ is the degree of mediated cohesion between methods that build path p_i

The following hypothesis is a convention from the empirical study conducted on applications that are confirmed as fault prone:

If '$mchw$'is the degree of mediated cohesionof class C then $(mchw\times 100)\%$ is class C fault proneness in application's fault proneness.

VI. Results Analysis

We conducted experiments on applications build under SDLC standards. We make sure the heterogeneity in number of classes of the applications considered for experiments. We measured the Fault proneness prediction accuracy of the MCBO and MCH as fallow:

$$\text{S(MCBO)} = \frac{\text{Classes correctly predicted as fault prone}}{\text{Classes actually fault prone}}$$

$$\text{S(MCH)} = \frac{\text{Classes correctly predicted as fault prone}}{\text{Classes actually fault prone}}$$

$$\text{S(MCBO} \oplus \text{MCH)} = \frac{\text{(Correctly predicted as fault prone by MCBO and MCH}}{\text{Actually fault prone}}$$

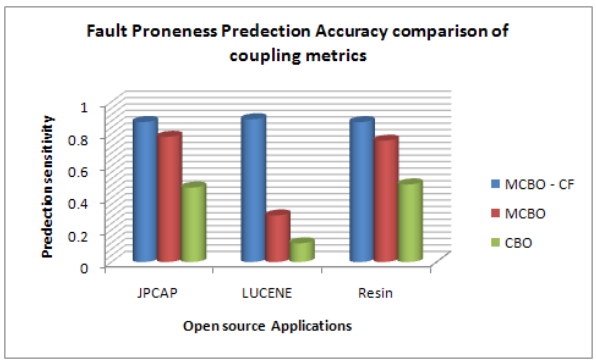

Fig. 3 : Fault proneness prediction sensitivity of Mediated Coupling Between Objects(MCBO), MCBO with confounding factor(MCBO - CF) and CBO

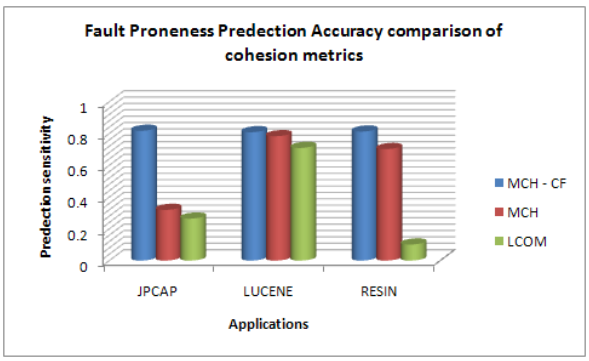

Fig. 4 : Fault proneness prediction sensitivity of Mediated cohesion (MCH), MCH with confounding factor(MCH - CF) and LCOM

Here in fig 3 we can observe the performance of the MCBO with the confounding factor in predicting the sensitivity of fault proneness, which stands with approximately 90% and miles ahead when compared to CBO. If path lengths are not considered as confounding factors then the sensitivity of MCBO is as low as CBO. This we can observe in the case of Lucene. Since the Lucene is having considerable variations in path lengths between any two classes that are connected in a transitive manner. In other two applications JPCAP and RASIN are having a minimal number of paths between two classes and also the variation between any two paths is negligible. The similar kind of performance can be observed for MCH with confounding factor. Fig 4 indicating the advantage of MCH with the number of paths as confounding factors over LCOM. The significance of the number of paths as confounding factor can be observed in the case of JPCAP. In majority classes the number of paths builds between same methods of the class. Hence the performance of the MCH without confounding factor is as low as LCOM(see fig 4).

VII. Conclusion

These results clearly demonstrate that the proposed metrics MCBO and MCH for coupling and coherence are very good predictors for fault proneness. It is clearly identified that

1. Mediated coupling between two objects is having an impact of the number of connections and path length variation as confounding factors.
2. Mediated Cohesion between the methods of a class is having an impact of the number of paths build between any two methods of a class as confounding factors.

These two metrics MCBO and MCH are measuring as numeric values rather in binary quantity.

The performance of mediated coupling between objects and mediated cohesion of class is miles ahead over CBO and LCOM, The number of paths and length of the paths concluded as confounding factors that influence the performance of the MCBO and MCH.

References Références Referencias

1. K.K.Aggarwal, Yogesh Singh, ArvinderKaur, RuchikaMalhotra, "Analysis of Object-Oriented Metrics", International Workshop on Software Measurement (IWSM), Montréal, Canada , 2005.
2. L.Briand, J.Daly and J. Wust, "A Unified Framework for Cohesion Measurement in Object-Oriented Systems", Empirical Software Engineering, 3, 65-117, 1998.
3. L.Briand ,J.Daly and J. Wust, "A Unified Framework for Coupling Measurement in Object-Oriented Systems", IEEE Transactions on software Engineering, vol. 25, 91-121, 1999.
4. J.Bieman, B.Kang, "Cohesion and Reuse in an Object-Oriented System", Proc. CM Symp. Software Reusability (SSR'94), 259-262, 1995.
5. M.Cartwright, M.Shepperd, "An Empirical Investigation of an Object- Oriented Software System", IEEE Transactions of Software Engineering. vol.26, Issue 8, 786 – 796, Aug. 2000.
6. S.Chidamber and C.Kemerer, "A metrics Suite for Object-Oriented Design", IEEE Trans. Software Engineering, vol. SE-20, no.6, 476-493, 1994.
7. S.Chidamber, C. Kemerer, "Towards a Metrics Suite for Object Oriented design", Proc. Conference on Object-Oriented Programming: Systems, Languages and Applications (OOPSLA'91), Published in SIGPLAN Notices, vol 26 no. 11, 197-211, 1991.
8. R.Harrison, S.J.Counsell, R.V.Nithi, "An Evaluation of MOOD set of Object- Oriented Software Metrics", IEEE Trans. Software Engineering, vol. SE-24, no.6, 491-496, 1998.
9. B.Henderson-sellers, "Object-Oriented Metrics, Measures of Complexity", Prentice Hall, 1996.
10. M.Hitz, B. Montazeri, "Measuring Coupling and Cohesion in Object-Oriented Systems", Proc. Int. Symposium on Applied Corporate Computing, Monterrey, Mexico, 1995.
11. A.Lake, C.Cook, "Use of factor analysis to develop OOP software complexity metrics", Proc. 6th Annual Oregon Workshop on Software Metrics, Silver Falls, Oregon, 1994.
12. W.Li, S.Henry, "Object-Oriented Metrics that Predict Maintainability', Journal of Systems and Software, vol. 23, no.2, 111-122, 1993.
13. Y.Lee, B.Liang, S.Wu, F.Wang, "Measuring the Coupling and Cohesion of an Object-Oriented program based on Information flow", International Conference on Software Quality, Maribor, Slovenia 1995.
14. M.Lorenz, J.Kidd, "Object-Oriented Software Metrics", Prentice-Hall, 1994.
15. D.Tegarden, S. Sheetz, D.Monarchi, "A Software Complexity Model of Object- Oriented Systems", Decision Support Systems, vol. 13 no.3-4, 241-262, 1995.
16. V.Basili, L.Briand, W.Melo, "A Validation of Object-Oriented Design Metrics as Quality Indicators", IEEE Transactions on Software Engineering, vol. 22 no.10, 751-761, 1996.
17. A.Binkley and S.Schach, "Validation of the Coupling Dependency Metric as a risk Predictor", International Conference on Software Engineering (ICSE), 452- 455, 1998.
18. L.Briand ,J.Daly, V.Porter, J. Wust, "Exploring the relationships between design measures and software quality", Journal of Systems and Software, vol. 5, 245- 273, 2000.
19. L. Briand, J. Wüst, H. Lounis, "Replicated Case Studies for Investigating Quality Factors in Object-Oriented Designs, Empirical Software Engineering: An International Journal, vol 6, no 1, 11-58, 2001.
20. S.Chidamber, D. Darcy, C. Kemerer, "Managerial use of Metrics for Object- Oriented Software: An Exploratory Analysis", IEEE Transactions on Software Engineering, vol.24, no.8, 629-639, 1998.
21. K.ElEmam, S. Benlarbi, N.Goel , S. Rai, "A Validation of Object-Oriented Metrics", Technical Report ERB-1063, National Research Council of Canada (NRC), 1999.
22. K.ElEmam, W. Melo, J. Machado, "The Prediction of Faulty Classes Using Object-Oriented Design Metrics", Journal of Systems and Software, vol. 56, 63- 75, 2001.
23. T.Gyimothy, R. Ferenc I. Siket, "Empirical validation of object-oriented metrics on open source software for fault prediction", IEEE Trans. Software Engineering, vol. 31, Issue 10, 897 – 910, Oct. 2005.
24. Yu Ping, Ma Xiaoxing, LuJian "Predicting Degree of Fault prone using OO Metrics: An Industrial Case Study, CSMR 2002, Budapest, Hungary, 99-107.

This page is intentionally left blank