International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 14 (2015) pp 34687-34698

Comprehensive survey of coupling metrics in object-oriented software

Mr. V. S. Bidve[#1], Dr. P. Sarasu[#2]
[1]*Ph.D. Scholar,* [2]*Director R & D*
[#]*Veltech Dr. RR & Dr. SR Technical University,*
Avadi, Chennai, India.
[1]vijay.bidve@gmail.com
[2]sarasujivat@gmail.com

Abstract- Coupling is the dependency relationship between the classes of an Object-Oriented (OO) system. Coupling is an important metric of the OO system as it has a strong impact on software quality. In the last two and half decades, lots of work has been done in the field of coupling. In the existing literature of coupling, every author has described a diverse set of coupling metrics. As a result, multiple metrics have emerged due to the different ways of measuring coupling of each author. In this paper, we take a comprehensive review of the existing coupling metrics. Our aim is to review maximum metrics available in the field of Coupling. The inadequacies in the form of overlapping and limitations of the existing metrics are discussed. In the end, the paper presents a summary of the existing metrics using a table of classification.

Keywords- coupling, measures, metrics, static, dynamic, and object-oriented.

1. INTRODUCTION

Coupling in the OO software is the only focus of the study of this paper. The metrics such as cohesion or complexity have not been considered in the paper. Coupling is the dependency relationship between the classes of an object-oriented system; this dependency relationship has been measured differently by several authors. As per the related work, coupling can be of various types including static, dynamic, import, export, inheritance, direct, indirect, class level, and object level, etc.

This paper makes an attempt to review the literature in depth in order to analyze the existing coupling measures. We try to uncover the different facets of coupling measures described by the various authors with different names and scope. As per our observation, none of the work done so far presents a comprehensive and normalized metrics of coupling measurement. In some metrics, there is overlapping among the measures and in the remaining all the aspects of coupling are not considered. In this paper, we are trying to collect all the aspects of coupling from the literature illustrated so far. The overlapping and differences amongst the measures have focused in order to get a normalized coupling metrics. The contribution of this paper is briefly described as follows:

- To review related work in order to find the various aspects of coupling.
- Evaluation of the related work to bring into notice the overlapping and differences among the existing coupling measures.
- To present a comprehensive and normalized metrics using the earlier metrics.

The remainder of the paper is organized as follows. The section 2 describes the current state of coupling and gives motivation to this study. In section 3, a detailed review of the related work is articulated. The section 4 evaluates the existing work. The normalized summary of the existing work has been given in section 5. The paper is concluded in section 6 with future work.

2. MOTIVATION

Coupling measurement in the object-oriented software has become an important research area. Lots of work is available in the field of coupling as stated by various researchers, but every researcher has presented their/his work differently. As a result, many different coupling metrics are described so far. Every metric has a different set of measures, and there is a lot of overlapping among the measures. Also, different names are used for the same type of coupling mechanisms by various authors. Many metrics are measuring the limited aspects of coupling and the others are having either overlapping amongst them or with the other metrics. Hence, there is a need to take a review of the existing metrics in order to extract all the measures that are unambiguous, distinct, and normalized.

3. A DETAILED REVIEW OF RELATED WORK

As already discussed in the introduction, we have considered only coupling measures for the study, and not the cohesion or complexity measures. Starting from the very first object-oriented metrics [1], we are trying to cover the maximum literature in the area of coupling. We are considering the literature serially, with respect to its evolution year for our study. In this review, at the end of the discussion of every author's metrics we have provided our comments or a summary in the form of a table concerning each metrics.

A. Chidamber and Kemerer Metrics

The Chidamber and Kemerer (CK) [1], have presented metrics for an object oriented-design. The Chidamber and Kemerer [2], came with the improved version of metrics by considering more aspects of each measure (including inheritance) and with theoretical and empirical validations. The CK metrics is not a language specific metrics. The measures introduced by Chidamber and Kemerer are as below.

1. *Weighted Method per Class (WMC):* In this measure the complexity of each method is considered as a unity and WMC=n i.e. total number of methods defined in a class.

WMC is an indicator of the time and effort required to develop an object. If the numbers of methods (WMC) in a class are more, there is a greater impact on the descendent classes.

2. *Depth of Inheritance Tree (DIT):* DIT is a measure of how many ancestor classes are affecting a class. Deeper the class in the hierarchy, the greater the number of methods it inherits and the class become more complex.
3. *Number of Children (NOC):* Number of subclasses of a class in the hierarchy. NOC gives an idea of the influence of a class on design [1]. Greater the number of children more the reuse of class; also, there is a possibility of improper abstraction of the parent class. NOC gives an idea of the influence of a class on design [2].
4. *Coupling between the object (CBO):* This is the most important measure in the field of coupling. It is defined at first as a count of the number of non-inheritance couples with the other classes. In CBO, coupling is counted if the method of one class uses a method or instance variable of another class. Larger the number of couples, the design is highly sensitive to changes and maintenance is more difficult [1]. In later work inheritance is considered in CBO [2].
5. *Response For a Class (RFC):* It is a count of the methods available to the object. Greater the number of methods invoked by an object, more the complexity of the object.

Table 1: SUMMARY OF CK METRICS [1][2]

Sr. No.	Measure	Means of Coupling as per Definition of Measures	Comments
1	WMC	No Coupling	No coupling is considered except **method invocation, attribute reference,** and **inheritance**.
2	DIT	Inheritance	
3	NOC	Inheritance	
4	CBO	Method-Method, Method-Attribute	
5	RFC	Inheritance	

B. Dr. Linda H. Rosenberg, Lawrence E Hyatt

The Linda et al. [3], have proposed an object-oriented metrics with respect to class and inheritance.

1) Class metrics: Coupling metrics defined by the authors is as given below.

- Method: As per the author, a method is an operation upon the object and is defined in the class declaration. The measure related to the method is Weighted Methods per Class (WMC). WMC is a total number of implemented methods in a class or the sum of the complexities of methods.
- Message: As per the author, a message is a request that an object makes to another object to perform an operation [3]. The operation that is executed after receiving a message is called a method. The metric with methods and messages is a Response for Class (RFC). RFC is a count of all the methods that are invoked in response to an object of a class. All the methods accessible to the object of a class are included in RFC. Inheritance is considered and mentioned explicitly for RFC by the authors.
- Coupling: As per the author coupling is a measure of strength of association due to a connection from one entity to another. Classes (Objects) are coupled by three ways:

1. When a message is passed between any two objects, then the objects are said to be coupled.
2. Classes are said to be coupled when the methods declared in one-class use the methods or attributes of the other class.
3. Due to inheritance there can be tight coupling between the super-classes and subclasses.

As per the author this coupling leads to a new metric called as Coupling between Object (CBO). CBO is a count of the number of other classes to, which a class is coupled [3]. CBO is measured by counting the number of non-inheritance related classes to, which a class is coupled.

2) Inheritance Metrics: Another metric described by the author is inheritance metrics. Inheritance enables the reuse of previously defined classes. The authors have defined the two metrics of inheritance as below.

- Depth of Inheritance Tree (DIT): It is a count of the maximum length from the class to the root of the tree.
- Number of Children (NOC): It is a measure of a number of immediate subclasses subordinate to a class.

TABLE 2: SUMMARY OF LINDA et al. METRICS [3]

Sr. No.	Measure	Means of Coupling	Comments
1	WMC	No Coupling	No coupling is considered except **method invocation, attribute reference,** and **inheritance**.
2	RFC	Inheritance	
3	CBO	Method-Method, Method-Attribute	
4	DIT	Inheritance	
5	NOC	Inheritance	

C. J. Eder, G. Kappel, M. Schrefl

Eder *et al.* [4], described three different dimensions of coupling properties such as Interaction Coupling, Component Coupling, and Inheritance Coupling.

1) Interaction Coupling: In Interaction Coupling Methods are coupled in terms of invocation of each other and/or sharing of data. The interaction coupling again is categorized into five types:

- Content Coupling: One method directly accesses parts of the internal structure of the method of another class, e.g. friend function in C++. Content coupling is the worst form of Coupling.
- Common Coupling: If methods communicate via an unstructured, global, or shared data space. This type of coupling is pathological from the author's point of view.
- External Coupling: It is common coupling by structuring global, or shared data space. This coupling occurs due to the access of public instance variables by multiple objects.
- Control Coupling: Methods communicate with each other via parameter passing.

- Stamp Coupling: A method depends on some externally defined data structure and changes if this data structure changes.
- Data Coupling: Two methods are data coupled if they communicate only by parameters.
- No Direct Coupling: Two methods do not directly depend on each other.

2) *Component Coupling:* In Component coupling one class is used as a domain of some instance variable of another class. The class can be a domain of an instance variable, parameter, local variable of a method or local variable of a method of the method of another class in component coupling. The Component Coupling again is categorized into four types:

- Hidden coupling: The object of one class is used in the implementation of the method of another class.
- Scattered Coupling: One class is used as a domain in the definition of a local variable or an instance in the implementation of another class.
- Specified Coupling: One class is used in the specification of another class i.e. one class is the component of another class.
- Nil: No direct component coupling between the classes.

3) *Inheritance Coupling:* Two classes are inheritance coupled if one class is a direct or indirect subclass of another class. The Inheritance Coupling is categorized into four types:

- Modification: In modification inherited information is changed or deleted.
- Refinement: Inherited information is only changed due to predefined rules.
- Extension: Only new methods are added in a subclass with no modification or refinement in the existing inherited methods.
- Nil: No inheritance relationship between two classes.

Eder et al. [4], have covered maximum aspects of coupling measurement, but there is overlapping among some measures. In interaction coupling, method invocation (Method-Method) and parameters passing are logically equivalent [20]. Coupling due to shared global space (data) cannot be considered as a direct coupling between two classes from the programming point of view as they are not using the components of one another. Hence, coupling due to parameter passing and global data sharing is not required to be considered. In component coupling the main concept is one class is a data type of variables of another class. Hence, all coupling means of this type can be combined under the main heading of component coupling. The summary of Eder et al. [4] metric is given in Table 3.

D. M. Hitz, B. Montazeri

Hitz et al. [5], has emphasized on the distinction between object level coupling (OLC) and class level coupling (CLC).

Table 3: SUMMARY OF EDER et al. [4] METRICS

Sr. No.	Measure	Means of Coupling	Similar coupling means in earlier metrics	Comments
1	Interaction Coupling	Method-Method, Method-Attribute, Parameter Passing,	CBO [1, 2]	**Friend, and Component coupling** are the new Coupling measures.
		Friend Method, Shared Global Space	---	
2	Component Coupling	One class is domain (type) of another class's instance variable, method parameter, method's local variable, class's local variable	---	
3	Inheritance Coupling	Inheritance	NOC, DIT [1, 2]	

Table 4: SUMMARY OF HITZ ET AL. [5] METRICS

Sr. No.	Measure	Means of Coupling	Similar coupling means in earlier metrics	Comments
1	Class Level Coupling (CLC)	SC is type in CC	Component Coupling [4]	Except **Class-Attribute** Coupling all means are used previously
		Local variable of SC is used in Method of CC,	Method-Attribute [1, 2], Friend [4]	
		SC is a superclass of CC	Inheritance [1, 2]	
		SC is type of parameter of method of CC	Component Coupling [4]	
		CC accesses global variables of SC	---	
2	Object Level Coupling (OLC)	O is accessing interface or instance variable of X	Method-Method, Method-Attribute [1, 2]	
		X is parameter to O's method	Component Coupling [4]	
		In message passing in O are of type X	Component Coupling [4]	

1) *Class Level Coupling (CLC):* CLC is coupling resulting from state dependencies between classes at a designed time. This coupling involves the following types: Scope of access to the Server Class (SC) objects within the Client Class (CC) with different possible variants is as below.

- SC is the type of an instance variable of CC.
- A local variable of type SC used within a method of CC.
- SC is a superclass of CC.
- SC is the type of parameter of the method of CC.
- CC accesses the global variable of class SC.

2) *Object Level Coupling (OLC):* Method of one object using methods or instance variables of a non-native object constitutes OLC. Three dimensions are given to depict the OLC between objects say O and X.

- O is accessing to the interface of X or may refer to at least one instance variable of X.
- Object X is accessed by O such that, X is a parameter to one of O's methods, a non-native part of O or a global object.
- In the message passing, arguments to the message of O are of type X.

E. Abreu, F. B.

The Abreu [6], described MOOD Metrics (Metrics for Object Oriented Design), which includes six measures.

1) *Method Hiding Factor (MHF):* MHF measures the invisibilities of methods in classes. MHF of a method is a percentage of the total classes to, which the method is not visible. MHF is a fraction where the numerator is the sum of invisibilities of all the methods defined in all classes, and the denominator is the total number of methods defined in the project.
2) *The Attribute Hiding Factor (AHF):* AHF measures the invisibilities of attributes in classes. An attribute can be called visible if it can be accessed by another class or object. The numerator of AHF is the sum of the invisibilities of attributes defined in all classes and the denominator is the total of the attributes defined in the project.
3) *Method Inheritance Factor (MIF):* MIF is a ratio of the inherited methods to all the methods in the system. MIF=0 means the inheritance mechanism is not used. MIF is a measure of reuse via inheritance.
4) *Attribute Inheritance Factor (AIF):* AIF is a ratio of the inherited attributes to all the attributes in the system.

TABLE 5: MEASURES PROPOSED BY BRIAND et al. [8]

Type of Relationship	Type of Interaction	Locus	Measure
IF	CA	Import Coupling (IC)	IFCAIC
A			ACAIC
O			OCAIC
IF	CM		IFCMIC
A			ACMIC
O			OCMIC
IF	MM		IFMMIC
A			AMMIC
O			OMMIC
F	CA	Export Coupling (EC)	FCAEC
D			DCAEC
O			OCAEC
F	CM		FCMEC
D			DCMEC
O			OCMEC
F	MM		FMMEC
D			DMMEC
O			OMMEC

Table 6: SUMMARY OF BRIAND et al. [8] METRICS

Sr. No.	Measure	Means of Coupling	Similar coupling means in earlier metrics	Comments
1	(Class-Attribute) CA	One class has attributes of another class type	Component Coupling [4]	There is an addition of coupling means in CM, and MM measures.
2	(Class-Method) CM	Method signature of one class uses another class	Component Coupling [4]	
		Return type of method of one class is another class	---	
3	(Method-Method) MM	Method of one class calls another class method	CBO [1, 2]	
		Method of one class as a parameter of another class's method	---	

5) *Polymorphism Factor (PF):* PF is the ratio of a total overridden method to the total possible methods overridden.
6) *Coupling Factor (CF*): CF is a ratio of the total number of non-inheritance coupling to the total number of coupling in a class.

Comments: The Abreu [6], has not described any coupling means explicitly. Inheritance is the only coupling that can be extracted from MOOD metrics. But Inheritance is already covered in the earlier metrics [1, 2] hence, no novel means of coupling is found in this metrics.

F. R. Harrison, S. Counsell, R. Nithi

The Harrison et al. [7], defined the Number of Associations (NAS) metric, which measures the number of associations of each class considering association lines originating from a class on an OMT (Object Modelling Technique) diagram. The author assumed that NAS and CBO are identical; the only difference is that CBO counts the coupling frequency due to repetition that NAS does not count. As per the author's opinion this metrics considers inheritance and from this design document we can collect the metric data.

Comments: As per Harrison et al. [7], the NAS metrics is identical to the CBO metrics and the CBO [1, 2] metrics is already considered in our discussion. Harrison et al. [7] have not provided any novel means of coupling.

G. L. Briand, P. Devanbu, and W. Melo

Briand et al. [8], expressed three aspects of coupling between the classes in OO systems developed with C++. The three aspects are locus, type, and relationship as given below.

1) *Relationship:* It refers to the type of relationship such as friendship, inheritance, or any other.
2) *Locus:* It refers to locus of impact; i.e. the impact of change flow towards a class (import) or away from the class (export). Changes in the ancestor flow towards a class is called as import and changes to a class flow to the descendants is called export coupling. A class exports impact to its friends and descendants and imports impact from the ancestor and classes having the class as a friend.
3) *Type:* Type refers to the type of interaction between classes. It can be of Class-Attribute, Class-Method, or Method-Method interaction.

- Class-Attribute (CA) Interaction: One class has attributes of another class type this is a Class Attribute relationship between the two classes.
- Class-Method (CM) Interaction: The method signature of one class uses another class or return type of method of one class is another class.
- Method-Method (MM) Interaction: Method of one class calls another class method or the method of one class is used as a parameter of another class method; then there is MM interaction in the two classes.

In addition to the above metrics, the following functions are defined by the authors.

- Friend^{-1}/Inverse Friend (IF): This function returns the set of classes that have class c as a friend.
- Ancestor (A): Ancestors classes of a class are returned by this function.
- Friend (F): This function returns all the classes, which are friends of c.
- Descendants (D): This function returns all the descendant classes of class c.
- Other (O): System(S) – Friend^{-1} – Ancestor – Friend – Descendants.

H. Briand, Lionel C., John W. Daly, and Jurgen K. Wust

Briand et al. [9], have proposed a new framework for coupling in object-oriented systems. In the framework, they have given six criteria of coupling,

1. Type of connection. It constitutes class-attribute, class-method, or method-method type of connection.
2. Locus of impact: i.e. import or export coupling.
3. Granularity of measure: It consists of a domain of the measure and how exactly the connection is counted. The domain can be class, method, object, etc.
4. Stability of server: Server class can be stable or unstable.
5. Direct or indirect coupling: This is also an issue whether to count direct or indirect measure. RFC is the indirect measure, where the remaining all are the direct measures.
6. Inheritance: Coupling can be inheritance-based or non-inheritance-based.

Comments: These criteria are already described for e.g. type of connection and locus is already explained by Briand et al. [8]; granularity is covered by Hitz et al. [5]. Inheritance is an important coupling measure covered by most of the above authors [3, 4, 6, 8]. Stability of server is not a point of our study because stable classes are the only library classes, which need not to be considered for coupling. We cannot make a distinction between direct and indirect because of polymorphism and inheritance. All these coupling means are already covered in the above discussed metrics.

I. Yacoub, Sherif M., Hany H. Ammar

The Yacoub et al. [10], formulated dynamic coupling metrics using some terminologies and a scenario is as given below.

- o_i: Instance of a class (an object)
- x: A scenario
- PS_X: Probability of a scenario
- $M_x(o_i, o_j)$: Set of messages sent form object o_i to object o_j during the execution of scenario x.
- MT_x: Total number of messages exchanged between objects during the execution of scenario x.

Using the above terminologies the authors have defined two types of metrics-

1) *Export Object Coupling* $EOC_x(o_i, o_j)$*:* This is a percentage of the number of messages sent from o_i to o_j with respect to the total number of messages exchanged during the execution of scenario x [10]. The authors have further extended as Object Request for Service i.e. $OQFS_x(o_i)$ to measure the percentage of the total number of messages sent by the object *oi* to all the other objects in the design [10].
2) *Import Object Coupling* $IOC_x(o_i, o_j)$*:* This is a percentage of the number of messages received by o_i from o_j with respect to the total number of messages exchanged during the execution of scenario x [10]. This is also extended to Object Response for Service i.e. $OPFS_x(o_i)$ to measure the percentage of the total number of messages received by the object *oi* from all the other objects in the design.

Comments: The authors have considered message passing by the objects as a means of coupling. Object oriented programming doesn't differentiate between message passing and method invocation [20]. As the CBO measure is already defined by the earlier authors for method invocation. Hence, EOC and IOC measures don't contribute to a new means of coupling.

J. Erik Arisholm, Lionel C. Briand, and Audun Føyen

Erik et al. [11], used three decision criterions to classify dynamic coupling measures.

- Entity: The entity of measurement may be an object or a class.
- Granularity: The granularity is an aggregation level it can be again class, object, scenario, or use case.
- Scope: The scope is what to include/exclude in the measurement. The library and framework classes are excluded from coupling measurement.

The authors used the following elements to define coupling metrics:

1) Sets:

- C: Sets of classes in the system. Classes can be of type Application Classes (AC), library Classes (LC), and Framework Classes (FC).
- O: Set of objects instantiated by the system while executing all the scenarios.
- M: Set of methods in the system.
- N: Total number of lines of code.

2) Relations:

- D and A are descendant and ancestor classes. D and A are having relation on C.
- ME: Set of possible messages in the system. ME $\subseteq$ O $\times$ M $\times$ N $\times$ O $\times$ M is a message at source object and method sending a message, a line of code (N). The target object and method.
- IV: set of possible method invocations. IV $\subseteq$ M $\times$ C $\times$ M $\times$ C is of the form invoking method and its class with a method invoked and its class.

3) The consistency rule

The rule is ME $\subseteq$ O $\times$ M $\times$ N $\times$ O $\times$ M and IV $\subseteq$ M $\times$ C $\times$ M $\times$ C.

Assumptions: The measurement entity is either object or class, and the direction of coupling is import or export.

The author has given three ways of measuring coupling.

- Dynamic Messages: Distinct messages sent from (or received) by one object to another.
- Distinct method: Distinct methods invoked by each method in each object.
- Distinct classes: Number of distinct classes that a method in a given object uses.

Comments: Message passing, method invocation, and class references are used as a coupling means by Erik et al. [11]. The same coupling means are used by the earlier authors. Any additional coupling means is not defined by the Erik et al. [11].

K. Huan Li

The Huan Li [12], has given three types of coupling relationships of object oriented systems, which is as below.

1) *Invocation Coupling:* Method implemented in one class invokes the method implemented in another class is called invocation coupling.
2) *Reference Coupling:* If one class is an attribute of the other class, a class is a parameter of the method, or local variable of a method of another class, then, an interaction between the two classes is called Reference Coupling.
3) *Inheritance Coupling*: If one class is a parent or children of another class, then the coupling is of inheritance type.

The author considered that these three types of coupling are of equal strength. The new concept defined by the author of dependency frequency *f(i, j)= count(i, j)* is the total count of coupling that class i accesses class j. The author also defined global coupling; that measures indirect coupling between classes.

TABLE 7: MEASURES PROPOSED BY ERIK et al. [11]

Direction	Entity	Measure Strength	Measure
Import	Object	Dynamic messages	IC_OD
		Distinct Methods	IC_OM
		Distinct Classes	IC_OC
	Class	Dynamic messages	IC_CD
		Distinct Methods	IC_CM
		Distinct Classes	IC_CC
Export	Object	Dynamic messages	EC_OD
		Distinct Methods	EC_OM
		Distinct Classes	EC_OC
	Class	Dynamic messages	EC_CD
		Distinct Methods	EC_CM
		Distinct Classes	EC_CC

Comments: The three metrics described by the author are already defined by J. Eder et al. [4], and Briand et al. [8]. The author defined dependency frequency, which counts the coupling frequency, which is also defined by Harrison et al. [7]. The dependency frequency is useful to measure the strength of coupling, but our focus is to find the different ways by, which coupling occurs. The global data coupling is not a direct coupling between two classes hence, it is not considered for our study.

L. Husein, Sukainah, Oxley Alan

Husein et al. [13], have presented a consolidated framework using Eder [4] and Braind [8] metrics. They have considered *E(c)* as a set of elements in the class c.

E(c) = m(c) ∪ v(c)

Where, *m(c)* is a set of methods and *v(c)* is a set of attributes of class c. Also, *mA* is a method class of A and *vA* is an attribute of class A. The consolidated framework proposed by the authors is as follows.

TABLE 8: MEASURES PROPOSED BY HUSEIN et al. [13]

Classification	Relationship	Description	Name of Measure
Method Invocation Coupling	Method Invocation	mA is invoked by mB	MI
Global Data Coupling	Sharing of data	vA is invoked by vB	AR
Data Abstraction Coupling	Attribute of class or method as ADT	B is the type of vA	DA
	Method parameter as ADT	B is the type of a formal parameter of mA	MP
	Method return type as ADT	B is a return type of mA	MR
Inheritance Coupling	Inheritance	Class A is an ancestor of class B	IH

TABLE 9: SUMMARY OF HUSEIN et al. [13] METRICS

Sr. No.	Measure	Means of Coupling	Similar coupling means in earlier metrics	Comments
1	MI	Method of one class invokes method of another class	CBO [1, 2]	Measure names AR, MR are new, but coupling means are described earlier.
2	AR	Attribute of one class uses global attribute of another class	--- Used by Hitz [5]	
3	DA	A class is a type of attribute of another class	Component Coupling	
4	MP	A class is a type of parameter of method of another class	Component Coupling	
5	MR	A class is return type of method of another class	--- Used by Briand [8]	
6	IH	Inheritance	Inheritance	

M. Michael English, Tony Cahill, Jim Buckley

The Michael [14], divided the coupling metrics in two groups.

1) *Metrics based on interaction between classes:* This is the first group of metrics that depends on the interaction between classes. The coupling of type method-member (MM) interactions; which includes method-method (MP) or method attributes (MA) interactions.
2) *Friend based metrics:* This metrics is used to count the friendship relationship between classes. The author suggests that instead of declaring the whole class as a friend; the required members should be declared as a friend. The metric suggested for this is Actual Friend Methods (AFM). AFM is a count of the number of methods in a class that accesses the hidden member of classes that declared the current class (method) as a friend. Another friend based metric is Members Accessed by Friends (MAF); which counts the number of hidden members in a class accessed by the other classes.

The authors combined import and export coupling with each measure to achieve coupling due to import and export of the respective type.

Comments: Method-Member (MM) uses the same means of coupling as CBO [1, 2]. Also, coupling due to friend relationship is also described earlier by Eder et al. [4].

N. Suresh Yeresime, Jayadeep Pati and Santanu Ku Rath

Suresh et al. [15], have taken a review of the earlier software quality metrics. Important object-oriented coupling metrics including WMC, DIT, NOC, CBO, and RFC are considered by Suresh et al. [15]. Also, Suresh et al. [15] referred some object-oriented design metrics, which includes MHF, AHF, MIF, AIF, PF, and CF. All these measures are already discussed in the earlier part of this paper.

Some additional object-oriented measures described by the authors are as follows:

- Number of Operations Overridden by subclass (NOO): The subclass replaces operations inherited from the parent. More vale of NOO implies a design problem.
- Specialization Index (SI): This is a degree of specialization for each of the subclass in an object-oriented system. SI can be achieved by adding or deleting or overriding.
- Percent public and protected (PAP): This metric is a count of the percentage of class attributes that are visible to the other classes.
- Public access to data members (PAD): This metric is a count of the number of methods that can access another class's attributes.

Comments: These additional measures describe properties of a single class, or some elements of a class. Any coupling means is not described in these measures. Hence, these measures are not considered for our study.

O. A Mitchell, JF Power

The runtime coupling measures are proposed by Mitchell et al. [16, 17, 18, 19]. The runtime coupling measures are as below.

- The authors extended the definition of CBO and proposed a new measure called Dynamic Coupling between Object (DCBO), which counts the number of couples with the other classes at runtime [16].
- The author explained Degree of dynamic coupling between two classes as a percentage of the number of times one class accesses methods or instances variables from the other class to its total number of accesses to methods or instances of all classes [17].

The runtime Coupling Dimensions proposed for Java Programs [18] by the authors are as below.

- The former directions (import, export) of coupling metrics is extended to define two runtime metrics as runtime import coupling (R_I) and runtime export coupling (R_E) between objects.
- Strength of Coupling: Is a count of the total accesses of a class to methods or instance variables of another class. The authors also described degree of class as, accesses from an outside class called as runtime import degree of coupling (RD_I) and metrics captured from the server class is called as runtime export degree of coupling (RD_E).

Mitchell et al. [19] defined Runtime Coupling between Objects (RCBO) as a metrics to measure runtime class-class coupling at object level.

Comments: The Mitchell et al. [16, 17, 18, 19] described runtime coupling measures RCBO, and DCBO. These measures do not give any new means of coupling rather these measures use the same coupling means as that of CBO [1, 2].

4. EVALUATION OF RELATED WORK

In the above section III, we have reviewed the work of various authors; all this work is evaluated in this section. The evaluation is done on the basis of means (mechanisms) of coupling used by the authors. Measures described by each author with respect to coupling means are considered for evaluation. Table 10 gives the detailed overview of all the related work. The rows of the table give the mechanism that constitutes coupling (i.e. means of coupling) and columns show the respective authors. The measures, which are not described by the authors for a particular coupling means are marked using three dashes in Table 10. The measures, which do not contribute to any coupling means has called as additional measures. The additional measures are given in the last row of Table 10. The additional measures are mostly used to describe properties of an individual class rather than coupling between two classes.

The comparison of the earlier work given in Table 10 shows that the method-method, method-attribute, and the inheritance coupling mechanisms are commonly used by most of the authors. Hence, these three coupling mechanisms are most important, and are given by row number 1, 3, and 6 respectively in Table 10. The mechanism given in row 2 is used to show the method as the parameter of a method of another class, but this mechanism is considered by the Briand [8, 9] only. The mechanism 2 also belongs to the method-method type of coupling as here the method of one class uses the method of another class. The mechanism in, which one class is used in the implementation of a method of another class (class-method) is given by the rows 4 and 5 of the Table 10. The class-method mechanism used in row 4 of Table 10 has been described by five authors [4, 5, 8, 12, 13] and the mechanism used in row 5 is by only two authors [8, 13] only. The mechanism of row 4 is an important one as it is considered by many authors. The mechanism 5 is similar to mechanism 4 because both have the class-methods type of coupling interaction. The row 7 gives a mechanism for message passing between two classes (with and without inheritance). The message passing mechanism is also not described by most of the authors; this may be due to the logical similarity between message passing and method invocation (used by the mechanisms in row 1 and 2 of Table 10). Message passing and method invocation are logically equivalent [20], hence mechanisms stated by rows 1, 2, and 7 are having similar behavioral properties. The friendship relation is another important aspect of an object-oriented programming/program. There can be a strong coupling between the classes due to friend relationship. The coupling due to friendship given by row 8 of Table 10 is described by four authors [4, 8, 9, 14] only. The friendship is a more strong type of coupling as it shares private data members with another class; but only four authors have described this mechanism. The mechanisms given by rows 9 to 12 in Table 10 are using one class as either a domain or in the specification of another class. This is a class-attribute type of coupling described by the mechanisms given in the rows 9 to 12 of Table 10. The Class-attribute mechanism has not been used by most of the authors.

It has been observed from Table 10 that, no mechanism is described by all the authors, and a few measures are described by at most four to five authors only. This means, that most of the authors have not considered all the means of coupling and the mechanisms used by every author are different.

There are some more measures described by the authors called as additional measures in our work. The additional measures are mostly describing the properties of a single class or some element of a class for e.g. The WMC measure gives the count of the total weighted methods of a single class. WMC doesn't lead to coupling between two classes. Similarly, MHF, AHF, MIF, AIF, PF, and CF [6] measures give the value of the property of a single class. NOO, SI, PAP, PAD [15] measures also describe the property of an individual class. Dependency Frequency (DF) is a measure defined by Huan [12] for coupling frequency, but DF is not a coupling mechanism it is only a frequency count. IC, EC, $R_{I,}$ and R_E measures are giving directions of coupling either import or export, but not the means of coupling. Hence, the additional measures IC, EC, $R_{I,}$ and R_E cannot be considered as coupling measures because any measure can be either import or export.

5. Issues and Challenges

There are many issues and challenges related to coupling metrics reviewed in this paper. These issues and challenges are discussed in this part of the paper.

From Table 10, it is clear that, every reviewed author has defined a different coupling metrics set. Mechanisms used to measure coupling are different of each author. All means of coupling discussed in Table 10 are not considered by every author. Hence, no coupling metrics described so far is comprehensive.

There are many differences among the metric (measure) names used for the similar coupling means of various authors for e.g. the Method-Method coupling mechanism used in row 1 of Table 10 is named as CBO by CK [1, 2] and Linda [3], but the name interaction coupling is used by Eder [4], OLC by Hitz [5], MM by Briand [8, 9] and so on. The same is the case with all the other remaining measures. Many authors who defined the measures for a similar coupling mechanism have used different metric names. Most of the authors have not defined the measures for all the mechanisms; it means that they have not considered all the facets of coupling. As a result every author has come up with limited and a different set of coupling measures.

Overlapping amongst the coupling mechanisms is another issue with the reviewed metrics. The mechanisms used by the various authors are correlated and perhaps overlapped with respect to their behavioral properties. The mechanisms used in row 1 and row 2 of Table 10 have similar properties i.e. method of one class is invoking (calling) method of another class. In the first method is called within a method and in the second method is called as a parameter. Form the behavioral properties point of view both the mechanisms 1 and 2 are used for the same purpose (i.e.

TABLE 10: COMPARISON OF RELATED WORK

Sr. No.	Author → Mechanism for coupling ↓	C & K Metrics [1][2]	Dr. Linda et al. [3]	J. Eder et al.[4]	Hitz et al. [5]	Abreu, F. B. [6]	Harr-ison, et al. [7]	Briand, et al. [8]	Briand, et al. [9]	Yacoub et al. [10]	Erik et al. [11]	Huan Li [12]	Husein, et al. [13]	Michael et al.[14]	Suresh et al. [15]	Aine et al. [16][17] [18][19]
1	Method of one class uses/invokes method of another class	CBO	CBO	IRC	OLC	---	NAS	MM	MM	---	IC_OM, IC_CM, EC_OM, EC_CM	IVC	MI	MM	---	DCBO RCBO
2	Method of one class is used as a parameter of method of another class	---	---	---	---	---	--	MM	MM	---	---	---	---	---	---	---
3	Method of one class uses Instance Variable of another class	CBO	CBO	IRC	OLC	---	NAS	---	CM	---	IC_OC, IC_CC, EC_OC, EC_CC	---	---	MM	---	DCBO RCBO
4	One class is used in the implementation of a method of another class	---	---	CMC	CLC	---	---	CM	---	---	IC_OC, IC_CC, EC_OC, EC_CC	REF	MP	---	---	---
5	Return type of method of one class is another class	---	---	---	---	---	---	CM	---	---	---	---	MP	---	---	---
6	One class is a superclass of another class	CBO, DIT, NOC, RFC	RFC, DIT, NOC	IHC	CLC	---	NAS	A, D	A, D	---	A, D	IHC	IH	---	IHC	---
7	Message passing between objects of two different classes with & without inheritance	---	RFC	IRC	OLC	---	---	---	---	EOC IOC	IC_OD, IC_CD, EC_OD, EC_CD	---	---	---	---	---
8	Method of one class can directly access parts of the internal structure of another class method (friend)	---	---	IRC	---	---	---	Friend	Friend	---	---	---	---	AFM, MAF	---	---
9	One class is the domain of the instance	---	---	CMC	CLC	---	---	CA	CA	---	---	REF	DA	---	---	---

	variable, local variable of another class															
10	One class is used in specification of another class as a component	---	---	CMC	---	---	---	---	---	---	---	---	---	---	---	---
11	One class accesses global variables of another class	---	---	---	CLC	---	---	---	CA	---	---	---	---	---	---	---
12	Attribute of one class uses the attribute of another class	---	---	---	---	---	---	---	---	---	---	---	AR	---	---	---
13	Additional Measures	WMC	WMC	---	---	MHF, AHF, MIF, AIF, PF, CF	---	IC, EC	---	---	---	DF	IC, EC	---	NOO, SI, PAP, PAD	R_I, R_E

IRC- Interaction Coupling, CMC- Component Coupling, IHC/IH- Inheritance Coupling, IVC- Invocation Coupling, REF- Reference Coupling, DF- Dependency Frequency count, IC- Import Coupling, EC-Export Coupling, OD-Dynamic Messages, OM- Distinct methods, OC- Distinct Classes.

TABLE 11: PROPOSED CLASSIFICATION OF COUPLING MEANS

Sr. No.	Type of Interaction	Coupling mechanism/ means	Related measures defined so far
1	Method-Method	Method of one class uses/invokes method of another class	CBO, IRC, OLC, NAS, MM, IC_OM, IC_CM, EC_OM, EC_CM, IVC, MI, MM, DCBO RCBO.
		Method of one class is used as a parameter of method of another class	
		Message passing between objects of two different classes with & without inheritance	
2	Method-Attribute	Method of one class uses Instance Variable of another class	CBO, IRC, OLC, NAS, CM, IC_OM, IC_CM, EC_OM, EC_CM, MM, DCBO, RCBO.
3	Class-Method	One class is used in the implementation of method of another class	CMC, CLC, CM, IC_OC, IC_CC, EC_OC, EC_CC, REF, MP.
		Return type of method of one class is another class	
4	Class-Class	One class is a superclass of another class (Inheritance)	CBO, DIT, NOC, RFC, IHC, CLC, NAS, A, D, IH.
5	Class-Class/ Class-Method/ Class-Attribute	Method of one class can directly access parts of the internal structure of another class method (friend)	IRC, Friend, AFM, MAF.
6	Class-Attribute	One class is the domain of the instance variable, local variable of another class	CMC, CLC, CA, REF, DA, AR.
		One class is used in specification of another class as a component	
		One class accesses the global variables of another class	
		Attribute of one class uses the attribute of another class	

method-method interaction). Hence, these two mechanisms can be merged as they are based on a similar property of method invocation. Similarly, the mechanism used in row 4 and row 5 has a similar property of using the other class for the method implementation of a class. The mechanism of row 5 is used by only Briand [8] and Husein [13]. Hence, as the mechanisms 4 and 5 have similar properties and mechanism 5 is used by very few authors (not worth to define separately) we can merge them. The mechanism used in row 7 of Table 10 is of message passing. From the object- oriented programming point of view there is no difference between message passing and method invocation [20]. Hence, the mechanism used in row 7 of Table 10 can be merged with the mechanism used in row 1 and row 2. The mechanisms used in row 9 an row 10 have a similar property of using one class as a domain or in the specification of another class hence, these two mechanisms can be merged.

At last the mechanisms used in row 11 and row 12 have similar properties of one class that is accessing the attributes of another class. Further, the mechanisms used in row 9 to row 12 have similar types of interactions i.e. a class is using attributes of another class or a class as a domain in another class, hence, these four mechanisms can be merged together.

There is a need to remove the overlapping amongst the coupling mechanisms on the basis of the similarities between the behavioral properties. We have classified the coupling mechanisms of Table 10 on the basis of their similarities between the behavioral properties. The proposed classification is given in Table 11. Coupling mechanisms are grouped as per the similarities between their types of interactions in Table 11 for e.g. mechanisms 1, 2, 7 of Table 10 are merged as they belong to the method-method interaction. Similarly the mechanisms 4 and 5, 9 to 12 are merged on the similarity basis as they belong to class-method and class-attribute mechanisms respectively. Also, it has been observed that a single mechanism has been referred with multiple names (measures) as given in the last column of Table 11. Hence, there is difficulty in uniquely identifying a mechanism with its measure name. Here is a need to assign a unique identification (name) to each mechanism in the form of a measure/ metric. Also, sometimes a single measure (name) is used to refer more than one mechanism for e.g. as shown in Table 11 a CBO measure (name) is used to refer Method-Method, Method-Attribute, and the Inheritance type of interactions. Hence, there is a need to assign unique name to each coupling mechanism/Interaction.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have reviewed the related work of coupling in a comprehensive manner and collected the different mechanisms and measures of coupling. In the review we found multiple measures for the same type of coupling mechanisms and the coupling mechanisms have lots of overlapping amongst themselves. It has been observed that overlapping between the mechanisms can be removed by combining the mechanisms with similar behavioral properties. Also, there is a need to define measures uniquely with respect to their coupling mechanisms.

In the future, all the coupling mechanisms will be assigned a unique name based on their use/behavior. Also, each mechanism will be used to find the strength of coupling between the classes of an object-oriented software system. The strength measurement will be used to find the impact of coupling on different quality attributes of object-oriented softwares.

7. ACKNOWLEDGMENT

We are thankful of Dr. P. N. Mahalle and Prof. S. K. Pathan for receiving useful comments about this study. Also, we are thankful of Veltech Dr. RR & Dr. SR Technical University staff for their huge support.

REFERENCES

[1] 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.

[2] S.R. Chidamber, C.F. Kemerer, "A Metrics Suite for Object Oriented Design", *IEEE Transactions on Software Engineering*, 20 (6), 476-493, 1994.

[3] Dr. Linda H. Rosenberg, Lawrence E Hyatt, "Software Quality Metrics for Object Oriented System Environments", *A report of SATC's research on OO metrics*, 1994.

[4] J. Eder, G. Kappel, M. Schrefl, "Coupling and Cohesion in Object-Oriented Systems", Technical Report, University of Klagenfurt, 1994.

[5] M. Hitz, B. Montazeri, "Measuring Coupling and Cohesion in Object-Oriented systems", in Proc. Int.Symposium on Applied Corporate Computing, Monterrey, Mexico, October 1995.

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

[7] Harrison, Rachel, Steve Counsell, and Reuben Nithi. "Coupling metrics for object-oriented design." *Software Metrics Symposium, 1998. Metrics 1998. Proceedings. Fifth International*. IEEE, 1998.

[8] L. Briand, P. Devanbu and W. Melo, "An Investigation into Coupling Measures for C++," *Proc. 19th Int', l Conf. Software Eng., ICSE ',* 97, Boston, pp. 412-421, May 1997.

[9] Briand, Lionel C., John W. Daly, and Jurgen K. Wust, "A unified framework for coupling measurement in object-oriented systems." *Software Engineering, IEEE Transactions on* 25.1 (1999): 91-121.

[10] Yacoub, Sherif M., Hany H. Ammar, and Tom Robinson. "Dynamic metrics for object oriented designs." *Software Metrics Symposium, 1999. Proceedings. Sixth International*. IEEE, 1999.

[11] Erik Arisholm, Lionel C. Briand, and Audun Føyen, "Dynamic Coupling Measurement for Object-Oriented Software", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 30, NO. 8, AUGUST 2004.

[12] Huan Li, "A Novel Coupling Metric for Object – Oriented Software Systems", *IEEE International*

Symposium on International Journal of Computer Applications (0975 – 8887) Volume 27– No.10, August 2011Knowledge Acquisition and Modeling Workshop, pp. 609-612, 2008.

[13] Husein, Sukainah, Oxley Alan, "A Coupling and Cohesion Metrics Suite for Object-Oriented Software", *International Conference on Computer Technology and Development,* vol.1, no., pp.421-425, 13-15 Nov. 2009.

[14] Michael English, TonyCahill, JimBuckley, "Construct specific coupling measurement for C++ software", Computer Languages, Systems & Structures 38 (2012), 300–319.

[15] Suresh Yeresime, Jayadeep Pati and Santanu Ku Rath, "Review of Software Quality Metrics for Object-Oriented Methodology", *Proceedings of International Conference on Internet Computing and Information Communications, Advances in Intelligent Systems and Computing* 216, Springer India 2014, 267-278.

[16] A Mitchell, JF Power, "Toward a definition of run-time object-oriented metrics", In *Seventh ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering*. Darmstadt, Germany 2003.

[17] A Mitchell, JF Power, "Runtime Coupling Metrics for the Analysis of Java Programs", preliminary results from SPEC and Grand suites 2003.

[18] A Mitchell, JF Power, "An empirical investigation into the dimensions of run-time coupling in java programs", In *Third Conference on the Principles and Practice of Programming in Java (*pp. 9-14) 2004. Las Vegas, Nevada, USA.

[19] Mitchell Aine, James F. Power. "Using object-level run-time metrics to study coupling between objects." *Proceedings of the 2005 ACM symposium on applied computing*. ACM, 2005.

[20] http://programmers.stackexchange.com/questions/46642/differences-between-messages-and-methods.