ANALYSIS AND THEORETICAL VALIDATION OF OBJECT-ORIENTED COUPLING METRICS

Jarallah Alghamdi and Moshood Omolade Saliu
Department of Information and Computer Science
King Fahd University of Petroleum & Minerals, Dhahran 31261, Saudi Arabia
Email: {jaralla, saliu}@ccse.kfupm.edu.sa

ABSTRACT

Various object-oriented (OO) coupling metrics have been proposed for capturing the level of class coupling in object-oriented systems. Since product complexity plays a major role in determining the quality of software, metrics developed to characterize internal attributes such as coupling need to be validated to determine the usefulness of the measures. In this paper, we present an analysis of some OO coupling metrics. An interaction coupling metric, the modified coupling metric (MCC), is proposed as an extension of coupling between object (CBO) [1] and message passing coupling (MPC) [2]. We also perform a theoretical validation of the suite of OO inheritance coupling metrics proposed in [3] using well-known coupling properties.

KEY WORDS
Software Metrics, Coupling, Theoretical Validation

1. INTRODUCTION

Coupling is the degree of interdependence between modules. Coupling is an attribute of pairs of modules, rather than of the design as a whole [4]. It has been shown that coupling is significantly related to the probability of detecting a fault in a class during testing.

In OO design, three types of coupling may exist between classes: inheritance coupling, interaction coupling, and component coupling [1].

Two types of validation are recognized: theoretical (internal) and empirical (external) validation. Demonstrating that a metric measures what it purports to measure using some properties is a form of theoretical validation. Empirical validation requires many studies to be performed to accumulate convincing evidence that a metric is valid [5].

A major criticism of past OO metrics is that little attention has been paid to their theoretical or empirical validation [6].

Our first contribution in this paper is the extension of the interaction coupling metrics (CBO and MPC) proposed in [1] and [2], respectively, following an analysis that revealed their deficiencies. We propose a modified interaction coupling metric (MCC). Another main contribution of this paper is the theoretical validation of the suite of inheritance coupling metric proposed in [3].

The paper is organized as follows. Section 2 gives background on coupling in OO systems. Section 3 presents our interaction coupling metric (MCC). Section 4 contains an analysis of a suite of coupling metrics in [3]. In Section 5, we present the set of properties for complexity measures, and a theoretical validation of the metrics in [3] follows. Section 6 gives our observations on the coupling frameworks used for validation and the suite of metrics discussed. The conclusions and recommendations come in Section 6.

2. OBJECT-ORIENTED COUPLING

The software development industry today places a lot of emphasis on software quality, which has led to a large body of work in the area of software measurement, to enable evaluation, and prediction of software quality. Coupling is one of the most famous internal product attributes studied since the advent of structured programming, and more recently in the context of object-orientation.

Coupling was first introduced in the context of structured development techniques in 1974. Today, there are over thirty different OO coupling measures [7].

The principle of low coupling was brought to OO design by Coad and Yourdon in 1991 [7]. Chidamber and Kemerer [8] gave the first formal definition of coupling between classes, and concluded that “…any evidence of a method of one object using methods or instance variables of another object constitutes coupling.”

There are three existing and quite different frameworks for OO coupling. These frameworks act as building
blocks for definition of coupling metrics, because they introduce taxonomy of coupling characteristics. The three frameworks [1][9][10] were later integrated into one unique theoretical framework by Briand et al. in [7]. This new framework is suggested to serve as a mechanism for categorizing and comparing existing measures and their potential use.

Property-based measurement frameworks that set up criteria for validation of coupling were proposed in [7][11][12][13]. Our validation of the OO coupling metrics to be presented in this paper is based on a subset of these properties. Similar validation has been done in [10][11] on coupling metrics proposed in [4][8][11].

3. INTERACTION COUPLING METRIC

The proposed interaction coupling metric is an extension of CBO [1] and the message-passing MPC [2]. CBO considers the number of collaborated classes, while MPC considers in addition the frequency of service request.

3.2 The Proposed Modified Interaction Coupling Metric (MCC)

The major factors that may affect the degree of interaction coupling between classes in an OO system are: Number of collaborated classes, Frequency of service request, and Number of parameters per message. Both CBO and MPC did not consider the third factor.

The approach employed in our metric requires a listing of all the collaborations in the system, i.e. messages passed by all the classes along with their arguments. Our modified interaction coupling metric (MCC) is given as:

$$\text{MCC} = \sum_{i=1}^{n} \text{NOP}_i$$

NOP$_i$ Number of parameters in message i, and n no. of messages passed by the class to the others.

Thus MCC captures the fan-out as defined in CBO, the number of messages passed to different methods within the same class as in MPC and the number of parameters. An example shown in Figure 1 below to justify this claim:

A tabular representation of the interaction between all the classes in the system, including the number of parameters passed and frequency of service is given in Table 1.

If we take interaction coupling of class C5 that receives three service requests from different methods of class C3, each containing 1, 4, and 1, parameters respectively, MPC calculates the coupling of class C5 as 3, the same value it reports for class C6 with three service requests and 3 parameters each. Our metric differentiates between the two complexities with $\text{MCC}_5 = 5$ and $\text{MCC}_6 = 9$.

Also, for classes C8 and C10, both CBO and MPC give the same values. But MCC again reports different values, $\text{MCC}_8 = 2$ and $\text{MCC}_{10} = 4$, taken into consideration the number of parameters passed.

<table>
<thead>
<tr>
<th>Class</th>
<th>CBO</th>
<th>MPC</th>
<th>MCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C5</td>
<td>1,4,1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>C6</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>C7</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C8</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C9</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C10</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 1: The hypothetical system of Figure 1.

4. ALGHAMDI COUPLING METRICS

In this section, we present the coupling metrics in [3].

4.1 Inheritance Coupling Metric

AlGhamdi et al. proposed three different inheritance coupling measures: Individual class coupling (ICC), Class-to-class coupling (CTC), and Overall system coupling (OSC).

Two basic constituents were used as building block in establishing a definition matrix and coupling matrix for the framework. The constituents are: Class inheritance hierarchy which can be represented as an entity-relationship diagram, and Class signatures (i.e., methods interfaces) and attributes (i.e., instance variables).

The definition matrix defines the OO system as a set of classes, and each class defined by a set of attributes and methods. Each element in a given class in the definition matrix has a weight attached, and the weight is calculated considering the two constituents mentioned above. A coupling matrix is generated from the definition matrix, and this leads to an automatic coupling measurement.

For a detailed description of the steps in generating the definition and coupling matrices, we refer the reader to [3]. Below is a diagrammatic view of the process to giving a succinct and clearer picture.
Overall System Coupling (OSC)

The coupling of the whole OO system is the average of the coupling of all classes of the system:

$$C = \frac{\sum_{i=1}^{m} C_i}{m}$$

An example inheritance coupling hierarchy is shown in figure 4 below:

![Inheritance Hierarchy for a hypothetical system](image)

4.1.2 Enhancing the inheritance coupling metrics

The concept of usability of attributes and methods was introduced to the definition matrix of the original framework in [14]. The authors conjectured that some classes can actually inherit attributes and methods without using them. Thus, the formula for calculating each entry in the definition matrix was multiplied by zero if that entry corresponds to an inherited attribute or method that was never used and 1 otherwise. It allows for the coupling between classes that are not related through inheritance to be determined as zero.

5. EVALUATION CRITERIA AND THEORETICAL VALIDATION

This section presents two property-based measurement evaluation frameworks and theoretical validation of the inheritance coupling metrics in [3].

5.1 Property-based Frameworks for Coupling Metrics Evaluation

We briefly present the Weyuker’s set of properties as modified by Chidamber and Kemerer [8] to use classes rather than programs, and the coupling properties defined by Briand et al.[11].

5.1.1 Modified Weyuker’s Properties [8]

Chidamber and Kemerer employed Weyuker’s properties in validating their coupling metrics [8]. The set of properties were modified for relative appropriateness for OO development.

Three of the nine original properties by Weyuker were dropped as they do not apply to OO metrics. The exclusion of these properties makes the property numbers used here no longer consistent with the original property numbers. The remaining six properties are repeated below.
Property 1: Noncoarseness. Given a class P and a metric μ another class Q can always be found such that: μ(P) ≠ μ(Q).

Property 2: Nonuniqueness (Notion of Equivalence). There can exist distinct classes P and Q such that μ(P) = μ(Q).

Property 3: Design Details are Important. Given two class designs, P and Q, which provide the same functionality, does not imply that μ(P) = μ(Q). The specifics of the class must influence the metric value.

Property 4: Monotonicity. For all classes P and Q, the following must hold: μ(P) ≤ μ(P+Q) and μ(Q) ≤ μ(P+Q) where P+Q implies combination of P and Q. (It should be noted that P+Q is the combination of two classes, whereas μ(P) + μ(Q) is the addition of the metric value of P and the metric value of Q).

Property 5: Nonequivalence of Interaction. ∃P, ∃Q, ∃R, such that μ(P) = μ(Q) does not imply that μ(P+R) = μ(Q+R).

Property 6: Interaction Increases Complexity. ∃P and ∃Q, such that μ(P) + μ(Q) < μ(P+Q).

5.1.2 Briand et al. Properties [7],[11]

Briand et al. presented five properties to characterize coupling in an intuitive manner. While the properties were said not to be sufficient to say that a measure which fulfills them will be useful, the reverse was observed to be true.

There are two kinds of coupling measure can be measured, import or export coupling (or both), OuterR(c) will denote the relevant set of relationships from or to class c (or both). In the same vein, InterR(C) = \cup_{c \in C} OuterR(c) is defined to be the set of inter-class relationships in a whole system C.

The five proposed coupling properties, Coupling 1 to Coupling 5, are presented thus:

Coupling 1: Nonnegativity. The coupling [of a class c | of an OO system C] is nonnegative:

[ Coupling(c) ≥ 0 | Coupling(C) ≥ 0 ]

Coupling 2: Null value. The coupling [of a class c | of an OO system C] is null if [ OuterR(c) | InterR(C) ] is empty:

[ OuterR(c) = ∅ ⇒ Coupling(c) = 0 | InterR(C) = ∅ ⇒ Coupling (C) = 0 ]

Coupling 3: Monotonicity. Let C be an OO system and c ∈ C be a class in C. We modify c to form a new class c′ such that OuterR(c) ⊆ OuterR(c′). Let C′ be the OO system which is identical to C except that class c is replaced by class c′. Then

[ Coupling(c) ≤ Coupling(c′) | Coupling (C) ≤ Coupling(C′) ]

Coupling 4: Merging of classes. Let C be an OO system, and c₁, c₂ ∈ C two classes in c. Let c′ be the class which is the union of c₁ and c₂. Let C′ be the OO system which is identical to C except that classes c₁ and c₂ are replaced by c′. Then

[ Coupling(c₁) + Coupling (c₂) ≥ Coupling(c′) | Coupling (C) ≥ Coupling(C′) ]

Coupling 5: Merging of unconnected classes. Let C be an OO system, and c₁, c₂ ∈ C two classes in c. Let c′ be the class which is the union of c₁ and c₂. Let C′ be the OO system which is identical to C except that classes c₁ and c₂ are replaced by c′. If no relationships exist between classes c₁ and c₂ in C, then

[ Coupling(c₁) + Coupling (c₂) = Coupling(c′) | Coupling (C) = Coupling(C′) ]

5.2 Theoretical Validation

In this section, we theoretically validate the inheritance coupling metrics in [3] based on these generic properties. In order to validate the CTC coupling metric, we require both the procedures involved in formulating the definition matrix as well as the coupling matrix.

5.2.1 Validation using Briand et al. Properties

We consider each of the properties in turn and validate CTC based on the formulation in the definition and coupling matrices, as follows:

Coupling 1: Nonnegativity

Each entry in the definition matrix can only be zero for inaccessibility of attributes/methods or greater than zero for any inherited attributes/methods. This implies that α and β, can only be positive. Meanwhile, each entry in the coupling matrix is a sum which consists of multiplicative factors of entries dα, dβ and constants α and β. Since entries and constants are positive, the product will be positive. Thus by commutative property of addition, the resulting entry in the coupling matrix is always positive. This implies that CTC coupling cannot be less than zero. Therefore, property coupling I is satisfied.

Coupling 2: Null value.

Suppose we have a system in which the import and export coupling (OuterR(c)) is null, it implies there is no
inheritance relationship between the classes of the system. The weight of each of the attributes and methods of the classes would make some entries in the definition matrix to be nonzero. And we therefore have entries for $\alpha$ and $\beta$.

In the coupling matrix, calculating $C_{ij}$ requires corresponding columns for the two classes in the definition matrix to be multiplied, i.e., $\forall k, d_{ik} * d_{jk}$, it is the case that $C_{ij}$ would always be zero in the coupling matrix for any two classes that do not share attributes or methods in common (i.e, $i \neq j$). Thus, we have zero entry in the coupling matrix for CTC coupling of different classes. The separation of each class ($C_{ii}$) will be positive because of the entries in the definition matrix, and specifically the values will always be 1. Given $C_{ii}=1$, it implies that ICC = 1 - $C_{ii}$, CTC, and OSC are all zeroes, so property coupling 2 is satisfied.

An example is two classes $c_1$ and $c_2$ of a system that are not inheritance coupled, with $c_1$ having attributes a, b and method M11 and $c_2$ with attributes c, d and method M21. The definition matrix given below;

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>M11</th>
<th>M21</th>
<th>$\alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.14</td>
</tr>
<tr>
<td>C2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0.25</td>
</tr>
</tbody>
</table>

And we have the following coupling matrix;

<table>
<thead>
<tr>
<th></th>
<th>C1</th>
<th>C2</th>
<th>Class</th>
<th>ICC</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>1.0000</td>
<td>0.0000</td>
<td>C1</td>
<td>0.00</td>
</tr>
<tr>
<td>C2</td>
<td>0.0000</td>
<td>1.0000</td>
<td>C2</td>
<td>0.00</td>
</tr>
</tbody>
</table>

**OSC 0.00**

**Coupling 3: Monotonicity.**

Suppose we add an extra relationship to an existing class in the inheritance hierarchy, the root class and all super classes of the class with extra relationship, including the class itself will have their entries in the definition matrix increased. The increase for super classes is either due to increased in the depth, d, or NOC$_k$.

The increase in the values along the rows of the definition matrix for the classes stated above causes the value of constant $\alpha$, to reduce for each corresponding row, since $\alpha$ is the inverse of a sum that has increased. Also, the value of $\beta$, reduces for all the columns affected.

The classes for which the corresponding $\alpha$ values are reduced in the definition matrix have their entries in the coupling matrix reduced as a result of lower values of $\alpha$ and $\beta$. While those other classes whose $\alpha$ values did not change have their corresponding entries in the coupling matrix reduced as a result of reduction in some of the $\beta$ values, ($\forall i, j, k$ we need $\beta_k$ for $C_{ij}$).

The reduction in CTC causes the ICC of the class with extra relationship to increase since it is $(1 - C_{ii})$. ICC of all the other classes increases because the normalization in the coupling matrix formula. Thus, OSC is increased and property coupling 3 is satisfied.

As an example, suppose we add an extra relationship from class $C_4$ of Figure 4 to class $C_8$ with attribute 1 of weight 2 and method M81 of weight 1 as shown in Figure 5 below, the value of the entries for class $C_1$ and $C_4$ in the definition matrix increases thereby reducing $\alpha$ along row containing $C_1$ and $C_4$ in the definition matrix.

![Figure 5: Class C8 added to class C4 of Figure 4](image)

These enable us to compare the coupling values of classes $C_4$ and $C_7$ with the original system of Figure 4.

**Coupling 4: Merging of classes.**

If two classes in an inheritance hierarchy are merged to form a single class, the entries in the definition matrix for all the super classes of the resulting merged class are decreased. Because the value of d for the super class is lowered if the merged classes include a leaf class, and the NOC$_k$ at level k where the merging took place will reduce for all the super classes. The reduction in the entries for the super classes in the definition matrix results in an increased value of $\alpha$. The $\beta$ values will be increased since all entries for the super classes are reduced, and the merger has eliminated duplicate entries for methods/attributes in the columns of the two merged classes. All sub classes of the merged classes would have their corresponding $\alpha$ values in their rows unchanged, because the number of inherited attributes/methods remains intact.

In the coupling matrix, however, the increased value of $\beta$ causes all entries in the coupling matrix that is CTC coupling to increase proportionately, since $\beta$ affects each entry in the coupling matrix. All the super classes would have their values increased much more as a result of the increased value of $\alpha$.

The new class that constitutes the two merged classes would have an increased separation $C_{ii}$, than any of its component classes because of increased $\beta$ value, and more entries in its row in the definition matrix. Thus the ICC is less than the sum of the ICC of its component classes, and also the OSC of the new system is lower than the OSC of the original system. Therefore, property coupling 4 is satisfied.
C1 in S with attributes a, b and method M11(a), C2 with attribute c and method M21(a, c), and C3 with attribute d and method M31(d, b) as shown in the figure 6, have C2 and C3 are merged to form class C′ in system S′.

![Figure 6: System S](image)

![Figure 7: System S′](image)

The weights of attributes a, b, c and d are 1, 5, 10 and 1 respectively. While the weights of methods M11, M21 and M31 are 1, 2 and 3 respectively. The sum of ICC of C2 and C3 is greater than C′, and the OSC for system S is greater than system S′.

**Coupling 5: Merging of unconnected classes.**

The only way two classes can be unconnected in an inheritance hierarchy of a single system is either through multiple inheritance or one of the two classes is isolated and does not belong to the inheritance hierarchy of the system. Merging two unconnected classes c1 and c2 to form c′, increases entries in the row of c′ in the definition matrix as discussed for property coupling 4 above.

Entries for the descendants of the new class c′ in the definition matrix are increased, since they inherit more attributes and methods. All other entries will not be affected. These increments cause reduction in values of α of the new class and its descendants. The value of β reduces for columns having additional attributes/methods from the merged class, while others remain unchanged.

For the multiple inheritance case, the entries in the coupling matrix depend on the weights of the additional attributes/methods, since the reduction in β has same effects on all entries and α reduction having minimal effect. The coupling formula contains the product v_i, d_{ik} * d_{jk}, resulting in increased CTC. The c_{ii} for each of the classes is reduced because the calculation depends only on the product of each entry in the row of that class (i.e., d_{ik} * d_{jk}, i = j) which has reduced α value. The reduction in c_{ii} causes increase in the ICC of class c′ and all its descendants.

The coupling of class c′ cannot be equal to the sum of its component classes because the effect is propagated down the inheritance hierarchy through normalization. The OSC is reduced as it is averaged over less number of classes. This implies that property coupling 5 is not satisfied because coupling is already affected.

An example is given in Figures 8 and 9 for system S consisting of class c1 with attributes/methods a, b and M11, class c2 with attributes/methods c, M21, class c3 with attributes/methods d, M31 and class c4 with attributes/methods e, M41. In the second system S′, class c4 is merged with class c1 since they are unconnected.

In the second case where one of the component classes in the merger is isolated, the CTC coupling of the new class C′ with all other classes of the systems reduces as opposed to the increment observed for the multiple inheritance case. Also, the separation of the resulting merged class C′, C_{ii}, and its CTC coupling to its descendants are increased as against the reduction for the first case considered. The coupling of all other classes of the system remains unchanged. The increment in C_{ii} is because most of the values of β remain unchanged and the reduction in α value for its row is proportionate to the weights of the methods which also constitute to C_{ii}. The reduction in CTC (C_{ii} i ≠ j) results from multiplication of corresponding columns of class i and class j (with no extra entry through the merging) in the definition matrix, and the reduced values of α and β cannot increase CTC.

In this case as well, property coupling 5 is still not satisfied, since ICC and OSC changed after merging.

![Figure 8: System S](image)

![Figure 9: System S′](image)

Figure 10 gives an example with an isolated class C8.

![Figure 10: An inheritance hierarchy and one Isolated class C8](image)

From the two cases discussed it is unlikely that property coupling 5 can be satisfied. The weights of attributes/methods from the unconnected class have a lot to contribute to the resulting coupling measure. Again, the inheritance hierarchy demands that all the sub-classes would have share from the effect of the merger.

To conclude our validation of the coupling metrics using Briand et al. set of properties, the ICC and OSC violates property coupling 5. This might not be unconnected with the weight assignments procedure as well as the fact that attributes and methods are major contributors in the definition and coupling matrices.

### 5.2.2 Theoretical Validation using modified Weyuker’s Properties

Here, we perform a theoretical validation using the modified Weyuker’s properties.
Property 1: Noncoarseness.
In the definition matrix of an inheritance relationship, the sum of the entries in all the rows cannot be equal because of $d$, $w_k$, and NOC$_k$ considered. The difference in the values of $\alpha$ results in different CTC of all the classes. At least two classes can be found with different $c_n$, meaning ICC(c1) $\neq$ ICC(c2). Therefore property 1 is satisfied.

Property 2: Nonuniqueness (Notion of Equivalence).
If we have two sibling classes' c1 and c2, having the same number of attributes/methods of the same weights implemented differently in each, then we will have the same entries in the definition matrix. The CTC for each of the two classes to every other class in the system will be same, as they are affected by the same values of $\alpha$ and $\beta$ in the coupling matrix. We also have ICC(c1) = ICC(c2) and property 2 is satisfied.

Property 3: Design Details are Important.
If we have two classes c1 and c2 providing the same functionality but different number and type of attributes/methods, their corresponding entries in the definition matrix would be different. The different values of $\alpha$, $\beta$, $d_a$ and $d_b$ will result in different values of CTC and ICC, meaning their metric values are not necessarily equal. Thus property 3 is satisfied.

Property 4: Monotonicity.
The Weyuker monotonicity property is related to merging of classes properties of Briand et al. From our validation using Briand et al. properties, and the examples given there, the satisfaction of this monotonicity property depends on whether the two combined classes are connected classes or unconnected classes.

If the two combined classes are unconnected, then the mononicity property will always hold in the context of Weyuker’s definition. Thus property 4 is satisfied.

For the second case where the classes combined are connected, Weyuker property 4 is not satisfied. The reason is simply as stated earlier that, the separation of the new class from the other classes in the system is higher than any of its component classes since the number of classes it is connected to is reduced and duplicate entries would have been eliminated, so the ICC of the combined class is less than either of its component classes.

Property 5: Nonequivalence of Interaction.
As discussed for nonuniqueness (Property 2) it is possible for any two classes to have same coupling values. Suppose we have two classes c1 and c2 of equal coupling values, combining the same class c3 with c1 and c2 may yield different results for c1c3 and c2c3 according to Weyuker property 5. In an inheritance relationship, this type of situation can only be captured if c3 is unconnected with the original system (else it becomes interaction coupling) or both c1 and c2 have multiple inheritance relationship with class c3 as shown below:

![Figure 11: Multiple inheritance](image)

From merging of classes, if C3 share more methods with one of the classes than the other, then the relationship holds that the coupling of the combined classes would be different for the two original classes. If they are unconnected, that is, C3 has no attributes/methods in common with either C1 or C2 then the entries in the definition matrix would proportionately change to reflect the new attributes/methods for classes c1 and c2. This changes that reduce the values of $\alpha$ and $\beta$ affect both classes to the same degree in the coupling matrix, and their coupling measures will be same again. In either case, Weyuker property 5 is satisfied.

Property 6: Interaction Increases Complexity.
Again, as discussed under coupling 4 and coupling 5 of Briand et al. properties in our validation, this property can not be satisfied by the metric. Weyuker property 6 conjectures that interaction between classes increases complexity. This is not necessarily true, because if two connected classes are merged, then in the worst case there will be no relationship eliminated. The common relationship between them would have been eliminated and coupling of the combined class will definitely be less than that of the component classes. Thus it is not likely that property 6 of Weyuker would hold for any inheritance coupling metric.

We refer the reader to Briand et al.[11], where it was stated that this particular Weyuker property is the most controversial of all properties. This is evident from our validation in Section 5.2.1.

6. OBSERVATIONS ON VALIDATION FRAMEWORKS AND THE METRICS

Some of the properties put forward by Weyuker, while they have been tuned to apply to OO paradigm have some specifics that are still applicable to procedural paradigm. For example, if we consider their nonequivalence of interaction (property 5) to always hold, then coupling measure are compelled to be sensitive to at least one kind of relationship which is desirable, anyway. But if the property is considered not to hold (the existential quantifier does not cover all cases), then coupling measures are forced to be related to flow of control since it will eliminate the fact that merging of two classes can yield extra relationship.
Most of the Weyuker properties are implicitly embedded within the Briand properties. We therefore agree with Briand et al. [7] that their properties aggregate most of the previous properties in the literature and we also believe that they are better defined and targeted at OO systems in particular.

Now, to the inheritance coupling metrics, we observe that, different primitive data types have different weights assigned. Although one would have expected that all primitive data types have similar weights.

In general, the weight assignments need to be investigated empirically, because it constitutes so much to the coupling values. For example C6 in Figure 5 shows C2 to be more separated from the system than any of the other two, including the root because it has higher entries in the definition matrix, which mostly comes from weight assignment. This makes one wonder if the number of methods/attributes in a class should be a major factor in determining coupling, whereas coupling should be between classes of a system and not methods/attributes of a class.

7. CONCLUSION

We have presented our proposed interaction coupling metric resulting from an analysis of the metrics in [1][2]. We have also discussed an analysis and theoretical validation of the coupling metrics proposed [1]. While acknowledging the contributions made in the field of OO design metrics, we have shown through our analysis and theoretical validation that the set of metrics in [1] can still be improved by following some clearly stated measurement principles. We suggest that an empirical validation be performed on the set of metrics to be able to establish their usefulness. Most importantly, the effects of the weights assigned to attributes and methods influence to a greater extent the coupling values. Therefore, further clarification through empirical validation would help ascertain the reasonability of the weighting scheme.

8. ACKNOWLEDGEMENT

The Authors wish to acknowledge King Fahd University of Petroleum and Minerals (KFUPM) for utilizing the various facilities in carrying out this research.

REFERENCES