UML Class Diagrams: Similarity Aspects and Matching

Mojeeb Al-Rhman Al-Khiaty and Moataz Ahmed

Abstract—Model matching is at the core of different model management operations such as model evolution, consolidation, and retrieval. An accurate identification of the similarity and differences between the elements of the matched models leads to an accurate model matching, which, in turn, leads to better model management. Software metrics are the software engineer means to quantify the similarity between the elements of the matched models. In this paper, we empirically validate the use of different metrics for capturing the similarity and the differences between the elements of two matched UML class diagrams. The paper empirically investigates the improvement of the similarity assessment of the class diagrams through the weight calibration of compound metrics. The results, reported based on two case studies, show the superiority of the compound metrics over the individual metrics.

Index Terms—Model matching, similarity metrics, reuse, weight calibration.

I. INTRODUCTION

Models in software development allow engineers to downscale the complexity of the software systems. They are the developer means for reasoning about the requirements, communicating with stakeholders, documenting the system to ease the maintenance task, generating test cases, etc [1]. As early stage artifacts, models also provide great reuse potential [2]. Typically, for each software system, there is a set of models that describe its structural, behavioral, and functional perspectives. Our focus in this paper is the structural perspective, modeled by the UML (Unified Modeling Language) class diagram (see Fig. 1 as an example of UML class diagrams).

Overtime, engineers unavoidably find themselves dealing with large collections of models. These models represent different development concerns [3]. Additionally, these models are considered as a main source of knowledge which is captured from the minds of involved people. This knowledge is re-practiced each time new software is created, yet, when comparing software systems, we usually find 60% to 70% of a software product’s functionality is common [4]. Thus, without effective management, it is possible to build a new system from scratch, yet a similar situation has been built before. This results in duplicated artifacts, and thus redundant maintenance cost and time. Therefore, it is of paramount importance to have a systematic way to access and reuse existing software models in an efficient way. One way with a great potential is to merge these models into a reference model that unifies there overlaps and explicates their differences. Another way is to have an efficient repository along with efficient retrieval mechanism. In both cases, model matching is a fundamental operation.

Model matching is the task of identifying semantic correspondences between elements of two models [3], [5]. The task is error-prone and time consuming. It is error-prone due to the fact that these models, while representing similar functionalities, are modeled independently by different developers, and thus inconsistency, design differences, and conflicts among them are expected. Therefore, their similarity and differences must be accurately quantified to have an accurate match.

Different metrics as well as different matching algorithms have been proposed in the literature to identify the similarity and the differences of the models to be matched, especially for UML diagrams [6]–[11]. In prior works, we presented the use of Simulated Annealing algorithm (SA) [12] and the greedy algorithm [13] for model matching using different similarity metrics. Two types of metrics were used, individual metrics (e.g., a metric which measures the similarity between two classes based on the lexical names of the two classes) and compound metrics (e.g. a metric which measures the similarity between two classes based on a combination of their names and their neighborhood information). Compound metrics, with even weight assignments for their constituents (individual metrics), showed more consistent performance, across the two case studies used, than did the individual metrics. However, their accuracy was limited by the confounding effects of some of their constituents. Calibrating the weights of the constituents of the compound metric can control the confounding effect resulting from some constituents and thus make the compound metric more sound measure [13], [14]. In this paper, we empirically validate the effect of the weight assignments for the different compound metrics, introduced in our prior work, on the matching accuracy. Using two case studies, we empirically show how the appropriate weight settings for the compound metric can improve the matching accuracy with consistent performance across the two case studies. Since our focus here is the UML class diagram, the word ‘model’ henceforth will refer to a UML class diagram, which represents the structural view of a software system. Additionally, the words “element” and “class” are used exchangeably.

The paper is organized as follows. In Section II, we introduce the comparison framework. Section III presents the empirical investigation setup and the data collection. Empirical results and analysis are presented in Section IV. The conclusion of this paper and some future directions are summarized in Section V.

Manuscript received August 20, 2014; revised October 31, 2014. This work was supported by the Deanship of Scientific Research at King Fahd University of Petroleum and Minerals (KFUPM), Saudi Arabia, under Research Grant 11-INF1633-04.

The authors are with King Fahd University of Petroleum and Minerals, Dhahran, 31261, Saudi Arabia (e-mail: alkhiaty@kfupm.edu.sa, moataz@kfupm.edu.sa).

II. COMPARISON FRAMEWORK

Model comparison is the task of assessing the degree of the similarity and differences between the elements of the compared models [15]–[17]. Crucial to an efficient similarity assessment is to have a set of similarity metrics that considers the different aspects of the compared models, thus their similarity and differences are best quantified. Accurate quantification of the similarity between the elements of the compared models is a prerequisite for an accurate matching [14]. The matching algorithm uses the similarity scores between the elements of the matched models, as quantified by the similarity metrics, to match each element in the smaller model (model with less number of classes, e.g., Fig. 1 (a)) to its most similar element in the other model. When each element in the smaller model is matched to a distinct element in the other model the match is called injective.

A. Similarity Aspects

The similarity degree (ranging from 0.0 to 1.0) between the elements of the compared models (e.g., $M_0$ and $M_1$ in Fig. 1) can be represented in a two dimensional matrix, which we call here elements similarity matrix ($ES$), see Fig. 2. We use three types of similarity information: lexical naming information (also called shallow semantic [7]), internal information (also called deep semantic [7]), and neighborhood information. The shallow lexical information is used to measure the naming similarity between the compared elements (classes). The internal information is used to measure the similarity of the properties (i.e., attributes) and the behaviors (i.e., operations) of the compared elements. The neighborhood information is used to measure the similarity of structural relationships of the compared elements with their neighbors. Using either of this information individually to capture the similarity between the elements of the matched models may not usually lead to an accurate match. For example, two classes may have similar names, but they may totally have different properties and behaviors, and vice-versa. Therefore, relying on the naming similarity may not be enough to say whether two classes are similar or not. Similar argument can be said when relying only on the internal information. The confounding effect of generic attributes (e.g. name and ID) and methods (e.g. setters and getters) can misleadingly affect the accuracy of the metrics in capturing the actual similarity between the elements of the compared models. This ultimately will lead to a wrong match. This can also happen when relying only on the neighborhood information, as two dissimilar classes from two different models may have similar, or even identical, neighbors, and vice-versa.

Compound metrics (as a combination of different individual similarity metrics) allow us to consider different similarity aspects at the same time, and thus are expected to outperform the individual metrics. However, one of the main issues of the compound metrics is the weights assigned to each constituent of the metric [13], [14].

In all of the metrics, concepts (classes’ names, operations’ names, attributes’ names, and names of the relations between classes) are compared based on their semantic similarity (e.g. synonyms, hyponyms) according to the WordNet [18] is-a hierarchy of concepts. More precisely, our similarity assessment is based on a composed semantic path-based measure that considers both the local homogeneity as well as the granularity of the concepts in the WordNet hierarchy of concepts. This composed measure we used is the mean of two of the path-based measures supported by the WordNet: Path Length and Wu & Palmer [19], where:

Path length ($PL$) between two concepts $c_1$ and $c_2$ is defined as the inverse of the shortest path between synsets of two concepts.

Wu & Palmer ($WuP$) between two concepts $c_1$ and $c_2$ is defined as:

$$WuP(c_1, c_2) = \frac{2 \times \text{len}(\text{root}, c_1)}{\text{len}(c_1, c_2) + \text{len}(c_2, c_1) + 2 \times \text{len}(\text{root}, c_1)},$$

where $c_3$ is the Least Common Subsumer ($LCS$) of $c_1$ and $c_2$ in the hierarchy of concepts. Thus, our composed semantic path-based similarity measure of two concepts, $c_1$ and $c_2$, can be defined as follows:

$$SSC(c_1, c_2) = (PL(c_1, c_2) + WuP(c_1, c_2)) / 2.$$  \hspace{1cm} (1)

The following section presents technical definitions of the similarity metrics used in this work.
B. Similarity Metrics

Lexical Name Similarity metric (NS) measures the similarity between the names of two classes, $C_1$ and $C_2$, based on their semantic similarity as quantified by “(1)”.

$$NS(C_1, C_2) = SSC(\text{Name}(C_1), \text{Name}(C_2)).$$

(2)

Attributes’ Similarity metric (ASim) measures the similarity between two sets of attributes, $A_1$ and $A_2$, of two classes, $C_1$ and $C_2$, respectively, as follows:

$$ASim(C_1, C_2) = \max\{\sum_{i=1}^{k} aSim(a_i, a_i)\} |A_2|.$$  

(3)

where $a_i \in A_1$ and $a_j \in A_2$, $|A_1| \leq |A_2|$. The similarity $aSim(a_i, a_j)$ between two attributes, $a_i$ and $a_j$, is computed based on their semantic similarity as quantified by “(1)”.

Operations’ Similarity metric (OSim) measures the similarity between two sets of operations, $O_1$ and $O_2$, of two classes, $C_1$ and $C_2$, respectively, as follows:

$$OSim(C_1, C_2) = \max\{\sum_{k=1}^{l} oSim(o_k, o_k)\} |O_2|.$$  

(4)

where $o_k \in O_1$ and $o_l \in O_2$, $|O_1| \leq |O_2|$. The similarity $oSim(o_k, o_l)$ between two operations, $o_k$ and $o_l$, is computed based on their semantic similarity as quantified by “(1)”.

Internal Similarity metric (IS) measures the internal similarity of two classes, $C_1$ and $C_2$, as a weighted similarity of their attributes’ and operations’ similarity.

$$IS(C_1, C_2) = w_o \times ASim(C_1, C_2) + w_w \times OSim(C_1, C_2),$$

(5)

where $w_o$ and $w_w$ represent arbitrary weights assigned to the attributes’ and operations’ similarity, respectively.

Neighborhood Similarity metric (NHS) measures the neighborhood similarity of two classes, $C_1$ and $C_2$, having the two sets of neighbors $N_1$ and $N_2$, respectively, as follows:

$$NHS(C_1, C_2) = \max\{\sum_{k=1}^{N} NSim(n_k, n_k)\} |N_2|.$$  

(6)

where $n_k \in N_1$ and $n_l \in N_2$, $|N_1| \leq |N_2|$. The neighbor similarity $NSim(n_k, n_l)$ between two neighbors $n_k$ and $n_l$ is measured as a weighted similarity of the relation type similarity ($rtSim$), the relation name similarity ($rnSim$), and the neighbor name similarity ($nnSim$):

$$NSim(n_k, n_l) = w_r \times rnSim(n_k, n_l) + w_n \times nnSim(n_k, n_l)$$

(7)

where $w_r$ represents the weight assigned to neighbor name similarity, $w_r$ represents the weight assigned to relationship type similarity, $w_r$ represents the weight assigned to relationship type similarity. Relation types are compared using the similarity information presented in Table I, which is inspired from [10]. When evaluating the relation type similarity we consider the similarity of the two ends of the relation.

C. Class Similarity Metrics

Based on the metrics presented in Section II.B, we compute the class similarity using seven similarity metrics: NS, IS, NHS, NNHS, INHS, and NINHS. The first three are, respectively, defined in equations “(2)”, “(5)”, and “(6)”. The last four are defined as follows.

$$NIS(C_1, C_2) = w_o \times NS(C_1, C_2) + w_o \times IS(C_1, C_2).$$

(8)

$$NNHS(C_1, C_2) = w_n \times NS(C_1, C_2) + w_n \times NNHS(C_1, C_2).$$

(9)

where $w_o$ and $w_n$ are weights assigned to Name Similarity (NS), Internal Similarity (IS), and Neighborhood Similarity (NHS), respectively.

III. EMPIRICAL INVESTIGATION DESIGN

Having a compound similarity metric as a combination of different other individual metrics, each constituent in the combination should be assigned a weight that makes it accurately capture the similarity/dissimilarity between the compared elements. The weights assignments of the constituents of the compound metric is crucial for the accuracy of the metric, as will be witnessed by our empirical investigation. Toward this aim, we conduct a set of experiments for weight collaboration. Our objective is to select the most appropriate weights for the different constituents of the compound similarity metric so that each class in a certain model will be matched to its most similar class in the other model based on similarity value of the calibrated metric.
TABLE I: LOOKUP TABLE OF SIMILARITIES BETWEEN RELATIONSHIPS’ ENDS IN THE CLASS DIAGRAM

<table>
<thead>
<tr>
<th>Relationship’s End</th>
<th>OAS</th>
<th>MAS</th>
<th>OAG</th>
<th>OCO</th>
<th>GES</th>
<th>GEC</th>
<th>IRS</th>
<th>IRC</th>
<th>DES</th>
<th>DEC</th>
<th>RES</th>
<th>REC</th>
<th>NRE</th>
</tr>
</thead>
<tbody>
<tr>
<td>OAS</td>
<td>1.0</td>
<td>0.0</td>
<td>0.89</td>
<td>0.89</td>
<td>0.0</td>
<td>0.55</td>
<td>0.0</td>
<td>0.33</td>
<td>0.0</td>
<td>0.55</td>
<td>0.0</td>
<td>0.23</td>
<td>0.0</td>
</tr>
<tr>
<td>MAS</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.55</td>
<td>0.0</td>
<td>0.33</td>
<td>0.0</td>
<td>0.55</td>
<td>0.0</td>
<td>0.23</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>OAG</td>
<td>0.89</td>
<td>0.0</td>
<td>1.0</td>
<td>0.89</td>
<td>0.0</td>
<td>0.55</td>
<td>0.0</td>
<td>0.33</td>
<td>0.0</td>
<td>0.55</td>
<td>0.0</td>
<td>0.23</td>
<td>0.0</td>
</tr>
<tr>
<td>OCO</td>
<td>0.89</td>
<td>0.0</td>
<td>0.89</td>
<td>1.0</td>
<td>0.0</td>
<td>0.55</td>
<td>0.0</td>
<td>0.33</td>
<td>0.0</td>
<td>0.55</td>
<td>0.0</td>
<td>0.23</td>
<td>0.0</td>
</tr>
<tr>
<td>GES</td>
<td>0.0</td>
<td>0.51</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.4</td>
<td>0.0</td>
<td>0.72</td>
<td>0.0</td>
<td>0.4</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>GEC</td>
<td>0.51</td>
<td>0.0</td>
<td>0.51</td>
<td>0.0</td>
<td>0.51</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.4</td>
<td>0.0</td>
<td>0.72</td>
<td>0.0</td>
<td>0.4</td>
</tr>
<tr>
<td>IRS</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.21</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.49</td>
<td>0.0</td>
<td>0.83</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>IRC</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.21</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.49</td>
<td>0.0</td>
<td>0.83</td>
<td>0.0</td>
</tr>
<tr>
<td>DES</td>
<td>0.0</td>
<td>0.51</td>
<td>0.0</td>
<td>0.0</td>
<td>0.72</td>
<td>0.0</td>
<td>0.68</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.79</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>DEC</td>
<td>0.51</td>
<td>0.0</td>
<td>0.51</td>
<td>0.0</td>
<td>0.72</td>
<td>0.0</td>
<td>0.68</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.79</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>RES</td>
<td>0.0</td>
<td>0.17</td>
<td>0.0</td>
<td>0.0</td>
<td>0.38</td>
<td>0.0</td>
<td>0.89</td>
<td>0.0</td>
<td>0.66</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>REC</td>
<td>0.17</td>
<td>0.0</td>
<td>0.17</td>
<td>0.0</td>
<td>0.38</td>
<td>0.0</td>
<td>0.89</td>
<td>0.0</td>
<td>0.66</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>NRE</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

OAS = Owned Association; MAS = Member Association; OAG = Owned Aggregation; OCO = Owned Composition; GES = Generalization Supplier; GEC = Generalization Client; IRS = Interface Realization Supplier; IRC = Interface Realization Client; DES = Dependency Supplier; DEC = Dependency Client; RES = Realization Supplier; REC = Realization Client; NRE = No Relation End

TABLE II: BASIC STATISTICS ABOUT CASE STUDY 1 AND CASE STUDY 2

<table>
<thead>
<tr>
<th>Case Study</th>
<th>Number of class diagrams</th>
<th>Number of pairs to be matched</th>
<th>Number of classes in the largest model</th>
<th>Number of classes in the smallest model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case Study 1</td>
<td>5</td>
<td>10</td>
<td>71</td>
<td>49</td>
</tr>
<tr>
<td>Case Study 2</td>
<td>4</td>
<td>6</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

The matching accuracy is commonly measured in terms of precision "(12)"; recall "(13)"; and accuracy "(14)". However, it is a general problem that evaluating the accuracy of the matching algorithms depends heavily on the particular matching goal [20]. Within the context of the goal of this paper, we will consider all pairs more similar than certain threshold to be matched, and all pairs less similar to be not matched [21]. Therefore we can define the three measures as follows. Let $TP$ be the number of true positives (i.e. number of pairs of classes, correctly matched, with similarity score above a threshold of 80%), $TN$ be the number of true negatives (i.e. the number of classes in each model that are correctly unmatched, or are matched incorrectly, due to injectivity, but with low similarity score), $FP$ be the number of false positives (i.e. the number of pairs of classes incorrectly matched with similarity score above a threshold), $FN$ be the number of false negative (i.e. the number of pairs of classes incorrectly unmatched, or matched correctly with low similarity score), then:

$$
\text{precision} (%) = 100 \times \frac{TP}{TP + FP}.
$$

(12)

$$
\text{recall} (%) = 100 \times \frac{TP}{TP + FN}.
$$

(13)

$$
\text{accuracy} (%) = 100 \times \frac{TP + TN}{TP + TN + FP + FN}.
$$

(14)

The weights of the compound metrics of $n$ constituents are assigned values from 0.0 to 1.0, updated by ±0.05, such that, $w_{1} + w_{2} + ... + w_{n}=1$. For example, the weight of the constituents in “(7)” is calibrated according to the following pseudo code:

```plaintext
for $w_{mn} \leftarrow 0.0$ to 1.0 step 0.05
for $w_{mn} \leftarrow 0.0$ to 1.0 - $w_{mn}$ step 0.05
```

Similar procedure is applied for other compound metrics. The weight adjustment experiment was conducted as follows. A pair of models was randomly selected. The weights then are assigned in the same way exemplified by the above pseudo code. For each weight assignment, the similarity score for each pair of classes from the two models is computed, producing the $ES$ matrix. Then the match from each class in the smaller model to its most similar, unmatched, class in the other model is found. The resulting match is evaluated in terms of precision, recall, and accuracy. The weight setting that gives the best matching accuracy is used for matching the other pairs of models, and the matching accuracy is reported as a validation.

The metrics were extracted from two case studies. The first case study (referred to here as Case Study 1) represents within a domain class diagrams reversed engineered from an open source system, ezmorph\(^1\), consisting of 12 releases. To allow for differences between the reversed engineered class diagrams of the different releases, we picked 5 non-consecutive releases (0.8, 0.9, 1.0, 1.0.4, and 1.0.6) of this system. The second case study (referred to here as Case Study 2), adopted from [22], represents across domains class diagrams consisting of four class diagrams with similar structures (as they represent admission systems) but in different ontologies (Computer Repair Shop, Hospital

\(^1\) http://sourceforge.net/projects/ezmorph/?source=directory
registration, Student Admission, and Admission in a General Institution). Each diagram of the four class diagrams is given to a software engineering graduate student to enrich it with some attributes and operations. Table II provides basic statistics about the two case studies.

We developed a Java-based tool that takes, as input, a set of class diagrams in XMI format. The tool then parses the XMI files to extract the required similarity information for the similarity metrics, and then the tool assesses the pair-wise similarity between the classes of each pair of input models based on the corresponding similarity metrics. The pair-wise similarity scores between the classes of each pair of models are presented to the matching algorithm (adopted from [13]) as a two dimensional similarity matrix, $ES$. The matching algorithm takes, as input, the $ES$ matrix and produces as output the optimal match between the classes of the matched models, along with their similarity scores. Fig. 2 depicts exemplified steps of the matching algorithm used. Given the $ES$ matrix representing the similarity between the classes of two models, $M_i$ and $M_j$, the algorithm, in each of its steps, looks for the highest similarity score, in the $ES$ matrix, for which the corresponding classes are not matched so far. Then the algorithm matches these classes and marks them as matched. The algorithm repeats its steps until all the classes of the smaller model are matched.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

Fig. 3 shows the matching accuracy obtained based on “(6)” (NHS metric) at different weight settings for “(7)” For this experiment of weight calibration, class diagrams were selected randomly from Case Study 2. Having these diagrams from different ontologies, the confounding effects of the lexical naming of the relation name and the neighbor name is minimized.

As we can see from Fig. 3, the best accuracy was obtained at the following ranges of the weights: $w_{nn} \in \{0.05, 0.10, \ldots, 0.30\}; w_{rn} \in \{0.0, 0.05, \ldots, 0.35\}; w_{rt} \in \{0.65, 0.70, \ldots, 0.95\}$. Taking the median within each set (conditioning that $w_{nn} + w_{rn} + w_{rt} = 1$) we can suggest the following weight settings: $w_{nn} = 0.15; w_{rn} = 0.15; w_{rt} = 0.70$.

Fig. 4 shows the matching accuracy obtained at different weight settings for “(11)”, the NINHS metric, based on Case

![Fig. 3. Models’ matching accuracy at different weight settings for the neighborhoods similarity metric NHS’ constituents, “(6)” and “(7)”.]

![Fig. 4. Models’ matching accuracy at different weight settings for the similarity metrics NINHS’ constituents, “(11)”, across domains.]

![Fig. 5. Models’ matching accuracy at different weight settings for the similarity metrics NINHS’ constituents, “(11)”, within a domain.]


Study 2 (across domains class diagrams). As it is clear from Fig. 4 that the low accuracy was obtained when the weight assigned to the class name (i.e. the value of \( w_n \)) is high. However, when \( w_n \) is assigned low weight values (0 ≤ \( w_n \) ≤ 0.25, \( w_i \) and \( w_h \) can be assigned any values > 0 such that \( w_i + w_h = 1 - w_n \)), we usually get high matching accuracy between the classes of the matched class diagrams. The highest matching accuracy of 100% was achieved at the weight values: \( w_n \in \{0, 0.5, \ldots, 0.15\}; w_{nh} \in \{0.35, 0.40, \ldots, 1.0\}; \) and \( w_i \in \{x: x=1-(w_n + w_{nh})\} \). The low accuracy obtained due to high values assigned to the class name similarity (NS) can be attributed to the fact that models across different domains (which is the case in Case Study 2), though their structural similarity, have different ontologies, and thus they are lexically different. This will lead to high false negative, which in turn, leads to low accuracy.

The results in Fig. 6 show the matching accuracy as measured by the thee different accuracy measures, precision (at the top of the figure), recall (the second row in the figure), and accuracy (at the bottom of the figure). The figure shows the matching accuracy under Equal versus Calibrated weight assignment of the compound metrics constituents, for both Case Study 1 (the left side of the figure) and Case Study 2 (the right side of the figure). In the Equal weight assignment, the weights are assigned evenly to the constituents of the compound metric while in the Calibrated weight assignment, the weights are assigned based on an empirical investigation as shown in Fig. 3, Fig. 4, and Fig. 5. The sum of all the weights must be 1. The results showed the superiority of the compound metrics, with appropriate weight assignment, over both the individual metrics and the compound metrics with evenly assigned weights.

Across the two case studies, the two compound metrics NNHS and NINHS showed high and consistent accuracy under the calibrated weight assignment. However, the performance of the two metrics NIS and INHS is not consistent across the two case studies. As per the INHS metric, the reason can be attributed to the fact that in Case Study 1 the confounding effect of the generic methods or attributes, or of the empty methods’ list or attributes’ list, if comes together with the similarity of the neighborhood for some classes, can lead to increasing the false positives (hence decreasing the precision and accuracy). Regarding the NIS metric, the metric is based on two lexical metrics (NS and IS). The lexical similarity across domains (which the situation in Case Study 2) is expected to be low, leading to a decrease in the accuracy. However, both metrics, NIS and INHS, are performing better than their constituents across the two case studies and under the calibrated weight assignment.

V. CONCLUSION AND FUTURE WORK

This paper focused on the similarity aspects of the UML class diagrams, namely, the similarity metrics. The paper empirically investigated the improvement of the similarity
assessment of the class diagrams through the weight calibration of the compound metrics. The result showed the superiority of the calibrated compound metrics over both the individual metrics and the compound metrics with equally assigned weights. The results also showed that the importance of the structural (neighborhood) information for assessing similarity across domains, and the importance of the lexical (ontology-based information) for assessing the similarity within a domain. As a future work, additional similarity information, such as data types, is to be considered. Additionally, the results reported in this paper needs to be investigated using more case studies. Moreover, the results in this study needs to be reproduced using an AI-based heuristics such as genetic algorithms.

REFERENCES


Mojeeb Al-Rhman Al-Khiaty is a PhD candidate in computer science and engineering at King Fahd University of Petroleum and Minerals (KFUPM), Saudi Arabia. He received his master of science (MS) degree in computer science from KFUPM in June 2009. He obtained his bachelor of science (BS) degree with honors in mathematics and computer from Sana’a University, Yemen, in June 1999. His research interests include software metrics, software reuse, and soft computing.

Mootaz A. Ahmed received his Ph.D. in computer science from George Mason University in 1997. He is currently an associate professor with the Information and Computer Science Department of King Fahd University of Petroleum and Minerals, Saudi Arabia. He also serves as the chief technology officer, LEROS Technologies Corporation, Fairfax, Virginia. He is associated as an adjunct/guest professor with a number of universities in the USA, Italy, and Spain too. During his career, he worked as a software architect in several software houses. His research interests include automated software engineering, especially software testing, software reuse, and cost estimation, and software metrics and quality models. He has supervised a number of theses and published a number of technical papers in refereed journals and conferences in these areas.