Automated Class Diagram Assessment using Semantic and Structural Similarities

By Evi Triandini

Automated Class Diagram Assessment using Semantic and Structural Similarities

Reza Fauzan[1,2]   **Daniel Siahaan[1]***   **Siti Rochimah[1]**   **Evi Triandini[3]**

[1]*Department of Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia*
[2]*Department of Electrical Engineering, Politeknik Negeri Banjarmasin, Banjarmasin, Indonesia*
[3]*Department of Information System, ITB STIKOM Bali, Denpasar, Indonesia*
* Corresponding author's Email: daniel@if.its.ac.id

Abstract: Class diagrams show classes in software and the relationships between those classes. A class diagram is a unified modeling language diagram commonly used in education. Thus, an assessment of class diagrams is essential for teachers who usually have students produce class diagrams based on predetermined projects. Teachers assess student-produced class diagrams based on an answer key. However, teachers have a problem with a lack of consistency in assessment as teachers can use different standards between answers. This research attempts to approach class diagram assessment automatically. The proposed approach consists of two assessments: semantic and structural similarities. Semantic similarity is calculated using lexical information in the class diagram, and structural similarity is calculated using the diagram's structure, ignoring its lexical information. Our results show that experts see semantic and structural similarities equally during assessment. The proposed approach shows substantial agreement with experts in class diagram similarity assessment. Therefore, the proposed approach can automatically assess class diagram similarity as reliably as experts can.

Keywords: Automated assessment, Class diagram, Semantic similarity, Structural similarity, UML diagram.

1. Introduction

Class diagrams are a part of a unified modeling language (UML) that displays a software's static structure by showing the classes within a piece of software and the logical relationships between those classes. Measurement of similarities between UML diagrams has been widely studied [1,2]; however, there are several issues with measuring class diagram similarity, such as reuse [3,4], clone detection [5-7], and assessment [8]. Class diagram reuse is useful because software engineers do not need to make diagrams from scratch, clone detection is useful when checking for plagiarism within diagrams, and assessment is useful for assisting teachers in assessing student diagram design assignments.

Several previous studies have measured similarities between two class diagrams. Park and Bae [9] began by comparing the lexical information within class diagrams, dividing lexical information into class names, relationship names, and related class information. However, lexical similarity is only calculated syntactically. The studies [10,11] continued by dividing similarities into between-structure similarities in class names and structures. Class name similarity compares the names within each class using WordNet to find differences in semantic meaning. The structural similarity between classes compares each type of relationship, such as associations, generalizations, and dependencies. Robles [12] built an ontology domain to look for similar class diagrams. He searched for each word in the class diagram in that ontology domain. However, the comparison of words is only syntactic, and reuse is less expected in more complex class diagrams. Qiu [13] divided class diagram similarities into two categories: structural similarities and property similarities. Structural similarities are seen in lexical relationships between classes. Property similarities are seen in the lexical attributes and operations of

each class. This study showed that combining class structures and class properties is a good approach for calculating similarity. Al-Khiaty and Ahmed [14-17] measured similarities of class properties and class diagram structures. He combined class names and information within the class (e.g., attribute and operation names) as class properties. The lexical evaluation of relations in the class diagram is used to calculate class diagram structure. Adamu [18,19] built on previous studies by combining several UML diagrams. He proposed a framework for measuring similarity using the diagram's meta model [4]. He then combined the similarity of several UML diagrams using different similarity parameters [18]. Lexical information is used as a measurement parameter in class diagrams [18,20]. Yuan [21] measured structural similarity by ignoring lexical information to convert a class diagram into a graph. He noted that graphs could effectively measure structural similarities between class diagrams. Huang [22] also converted a class diagram into a graph. Therefore, a graph can be used to represent a class diagram.

UML diagrams can be modeled in certain forms. Based on previous studies, we can conclude that a UML diagram has two primary types of information: lexical information and structural information. Lexical information consists of all the words in the diagram, such as class names, property names, attribute names, and relation types. In contrast, structural information is the UML diagram's structure or form without any lexical information.

In this study, we focused on assessing the similarity of UML class diagrams, which are commonly used as teaching materials. Teachers give assignments to students to build class diagrams based on a project determined by the teacher. Similarity assessment is used to assist teachers in assessing the class diagrams that students build. However, teachers can have difficulty assessing class diagrams from students because class diagrams consist of many classes that have many relationships between them [23], [24]. Many components in a class diagram can affect the consistency of teacher assessments. In addition, teacher fatigue can also affect assessment consistency. Automatic assessment is a solution to maintain teacher consistency [25]. Jebli [26] attempted to assess class diagrams by obtaining lexical information from class diagrams using machine learning and ignoring diagram structure. However, structure is an important parameter for assessing UML diagram similarity [27].

In this study, we developed an approach for automatically assessing the class diagram similarities as reliably as an expert can (i.e., a teacher who conducts assessments). This study is a continuation of previous studies [28] that only assessed semantic similarity. The proposed approach divides similarity into semantic and structural similarities. We combined these two similarities to assess like an expert. We divided similarity into two because the information obtained from class diagrams consists of lexical and structural information. Our approach can show the inclination of a teacher's perspective in assessing whether to look at lexical information or class diagram structure. The tendency of perspective in assessment is obtained from the similarity weight, which has the best agreement value with the expert. The proposed semantic similarity method compares each component's lexical information using simple natural language processing (NLP) and compares within-class and between-class structures using graph edit distance (GED). The assessment of the proposed method agreement with the expert is a substantial agreement. Therefore, our approach assesses students' answers based on the answer key as reliable as an expert.

The rest of this article consists of several sections: Section 2 presents the semantic similarity assessment. Section 3 presents the structural similarity assessment. Section 4 presents the combination of previous assessment into class diagram similarity assessment. Section 5 presents the result and evaluation. Section 6 presents our discussion about the evaluation and findings. Section 7 presents the conclusion of our study.

2. Semantic similarity assessment

The concept of semantic similarity assessment involves seeing the similarity of two class diagrams based on lexical information obtained from those class diagrams. Information in class diagrams can be divided into property information and relationship information. This study discusses the lexical aspects of each item of information. For example, for class diagrams d1 and d2, all property information from d1 will be searched to find similarities to property information from d2. Operating information has many items, and each item from d1 will be compared with each item from d2. After obtaining the semantic similarity from each item, a greedy algorithm is needed to locate the ideal value of the previously calculated set of similarity values.

2.1 Semantic similarity

We determined semantic similarity using NLP [29]. We used a common NLP approach because words will be compared in one phrase consisting of two words on average. As previously explained,

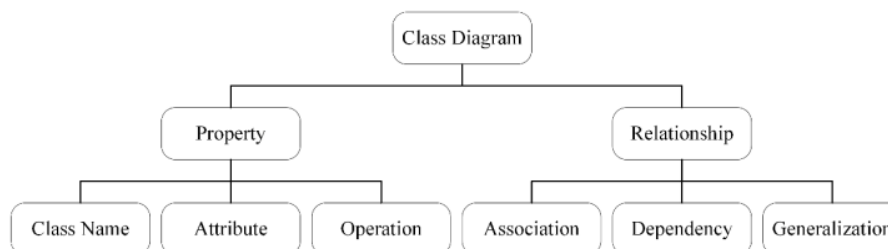Figure. 1 Natural language processing in UML diagram semantic similarity assessment

Figure. 2 Class diagram elements

semantic calculations are performed in the smallest part of each class diagram component. For example, the class name is one of the lowest parts of a class diagram, and class names have different characters from ordinary lexical information. Classes can have more than one word for a name, but the next word must be in uppercase. Therefore, NLP for semantic similarity has specific process details.

Fig. 1 shows the calculation procedure. The first part of this procedure is the tokenization process. Next, each word is given a part of speech (POS) sign. The following procedure is to eliminate stop words and lemmatization. The final part of the procedure is to calculate the similarity of words using cosine similarity.

The tokenization process involves breaking a group of words into individual units and is performed to adjust to the input words to be calculated. From the previous explanation, the lexical form of the class name, for example, *getData*, cannot be directly processed. Therefore, a unique method is needed for this tokenization process. First, if the next letter consists of an uppercase letter, the word will be broken up. Second, if uppercase letters are found in a sequence, the word will not be broken up until uppercase letters accompanied by lowercase letters are found.

POS tagging is intended to make the token's type known. Token types include nouns, verbs, adjectives, and so on. This process is assisted by Stanford NLP [30] using Stanford POS tagger with an English language model. Tagging uses POS from the Penn Treebank model to obtain each word's label. Before conducting the POS tagging process, all tokens are converted into lowercase letters.

The main objective in implementing the stop word elimination process is reducing the number of words in a report, which affects NLP speed and performance. In NLP, stop words are words that are ignored in processing, which are usually stored in

stop lists. Eliminating stop words in this study is aimed at increasing the similarity between two series of words based on their true meanings.

The following process is known as lemmatization, which is the process of turning tokens into essential words. Lemmatization aims to increase accuracy in the next process and was assisted by Stanford NLP. Calculation using cosine similarity [29,31] is the last process of calculating between-word similarity. Cosine similarity, as implemented in this study, consists of three stages: vector making, calculation of word similarity, and calculation of the cosine values. Each word will be counted as a similarity to other words to obtain words with different meanings. If several similar words are found, only the first word will be included in the vector. Determining words with different meanings requires that the similarity value of the word does not exceed a certain threshold. The similarity of words is calculated using the Wu and Palmer Semantic Similarity Measure [32], which uses WordNet [33] to calculate the depth.

2.2 Class diagram semantic similarity

Before measuring the similarity between two class diagrams, we first divided class diagram elements into property information and relationship information [28]. Fig. 2 shows class diagram elements. Property information consists of class names, attributes, and operations, and relationship information includes association, dependency, and generalization.

Furthermore, Fig. 2 shows that the class diagram semantic similarity (*classSem*) consists of the similarity of class diagram properties (*propSim*) and relationships (*relSim*). Fig. 3 shows the flow of semantic similarity calculations. Both class diagrams will be processed in XML metadata interchange format using a tool. Each diagram is extracted based on class diagram components, and each extraction
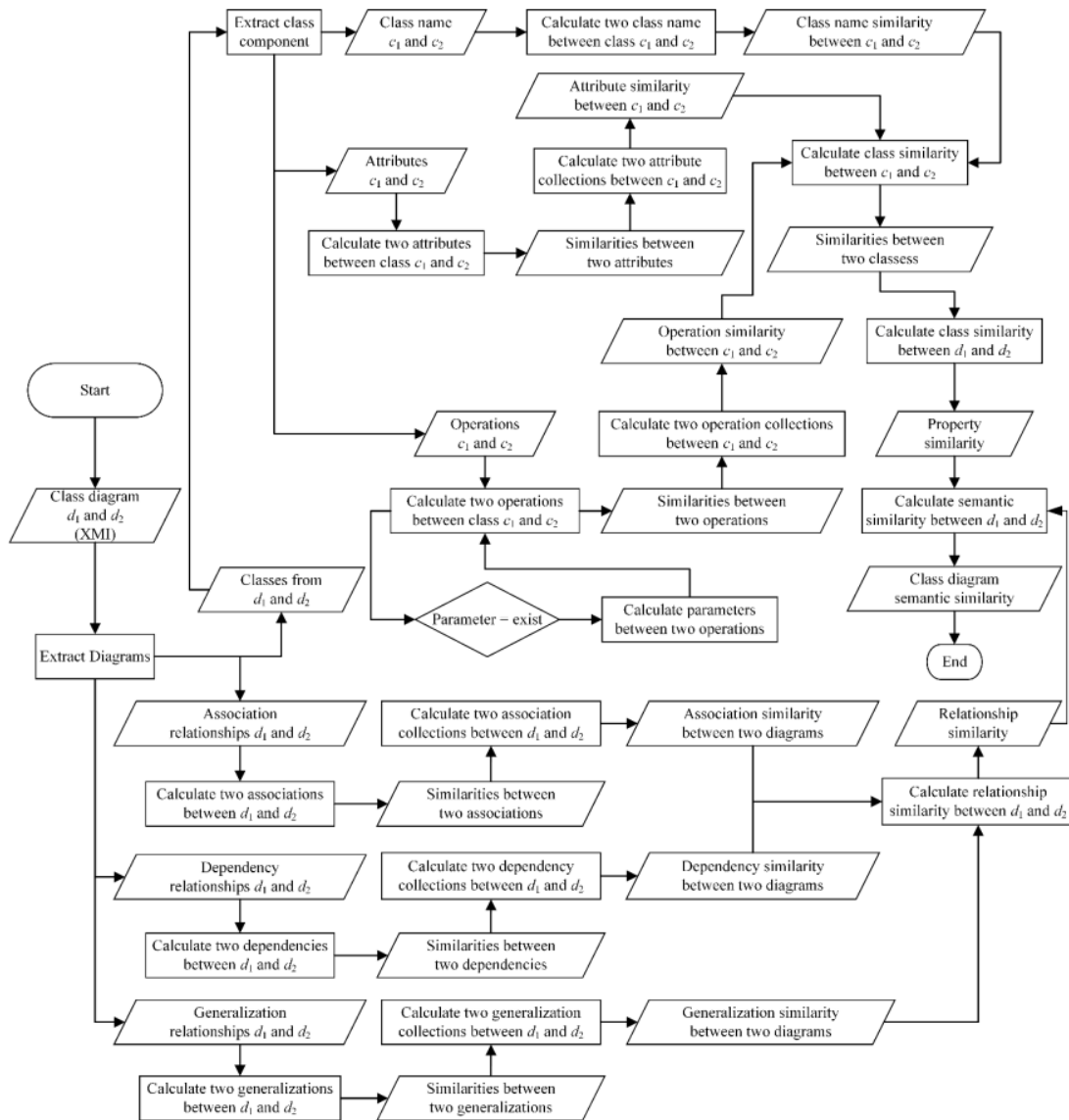
Figure. 3 Flowchart of semantic similarity assessment

result is counted as a semantic similarity. Components are calculated from the smallest to largest. The *classSem* between class diagrams d_1 and d_2 can be expressed as Equation 1.

$$classSem(d_1, d_2) = (1 - \rho_{sem}) \times propSim(d_1, d_2) + \rho_{sem} \times relSim(d_1, d_2) \quad (1)$$

Eq. (1) shows how to assess similarities between class diagrams d_1 and d_2. Properties and relationships have a different level of interest using ρ_{sem}, the value of which ranges from 0 to 1. Then, properties in class

diagram d_1 consist of several classes. Each class in class diagram d_1 will be compared with the classes in class diagram d_2. The *propSim* between the two class diagrams is expressed as Eq. (2).

Class collection C1 in class diagram d1 will be compared with class collection C2 in class diagram d2. The ith-class (ci) in class diagram d1 is then compared with the jth-class (cj) in class diagram d2. The similarity between the two classes is called cSim. We used changePivot to change each value in the row and column with a maximum value to 0. Algorithm 1 explains changePivot.

56

$$propSim(d_1, d_2) = \frac{2 \times \left(\sum_{k=1}^{Min(|C_1|,|C_2|)} changePivot\left(Max(\sum_{i=1}^{|C_1|}\sum_{j=1}^{|C_2|} cSim(c_i,c_j))\right)\right)}{|C_1|+|C_2|}. \quad (2)$$

$$aSim(c_1, c_2) = \frac{2 \times \left(\sum_{k=1}^{Min(|A_1|,|A_2|)} changePivot\left(Max(\sum_{i=1}^{|A_1|}\sum_{j=1}^{|A_2|} daSim(a_i,a_j))\right)\right)}{|A_1|+|A_2|}. \quad (4)$$

$$opSim(c_1, c_2) = \frac{2 \times \left(\sum_{k=1}^{Min(|OP_1|,|OP_2|)} changePivot\left(Max(\sum_{i=1}^{|OP_1|}\sum_{j=1}^{|OP_2|} dopSim(op_i,op_j))\right)\right)}{|OP_1|+|OP_2|}. \quad (6)$$

Algorithm 1: *changePivot*

Input: two-dimension   matrix   and
pivot/coordinate maximum value (x,y)
   1. Select pivot
   2. M(x, :) = 0
   3. M(:, y) = 0
Output: changed matrix

Line 1 in Algorithm 1 takes the pivot or coordinates with the maximum value in the similarity matrix. Line 2 changes all the values in the row pivot (x) to 0. Line 3 changes all the values in the column pivot (y) to 0. The cSim between classes c1 and c2 can be expressed as Eq. (3)

$$cSim(c_1, c_2) = w_{cn} \times cnSim(c_1,c_2) + w_a \times aSim(c_1,c_2) + w_{op} \times opSim(c_1,c_2). \quad (3)$$

The similarity between the two classes is influenced by class name similarity (*cnSim*), attribute similarity (*aSim*), and operation similarity (*opSim*). Weight w_{cn} is the weight of *cnSim* between classes c_1 and c_2. Weight w_{ap} is the weight of *aSim* between classes c_1 and c_2. Weight w_o is the weight of *opSim* between classes c_1 and c_2. The total of these weights is 1. The weight composition was adopted from Al-Khiaty [14], which showed that w_a and w_{op} were only slightly greater than w_{cn}. Therefore, we used weight compositions $w_{cn} = 0.4$, $w_a = 0.3$, and $w_{op} = 0.3$. The *cnSim* value measures the lexical similarity of class names of two classes with semantic similarity described in Section 2.1. Eq. (4) shows *aSim* between classes c_1 and c_2.

We then compared attribute collection A_1 from class c$_1$ with attribute collection A_2 from class c$_1$. We then used *changePivot* to change the values of rows and columns in the similarity matrix following Algorithm 1. Attributes then consist of a modifier, an attribute name, and a data type. We must perform a detailed attribute similarity calculation (*daSim*), which contains these three items. The *daSim* between attributes a_1 and a_2 can be expressed as Eq. (5).

$$daSim(a_1, a_2) = w_{md} \times sim(md_1,md_2) + w_{na} \times CosineSim(na_1,na_2) + w_{ty} \times sim(ty_1,ty_2). \quad (5)$$

Weight w_{md} is the weight of similarity between modifiers md_1 and md_2. Weight w_{na} is the weight of attribute name similarity between attribute names na_1 and na_2. Weight w_{ty} is the weight of data type similarity between data types ty_1 and ty_2. The total of w_{md}, w_{na}, and w_{ty} is 1, and the weights used are $w_{md} = 0.1$, $w_{na} = 0.7$, and $w_{ty} = 0.2$. We emphasized name attributes because experts see them more clearly. Calculations of similarities between two modifiers and between two data types are simple. If the modifiers of the data types are the same, the similarity value is 1. If the modifiers or data types are different, the similarity value is 0. The similarity between attribute names can be calculated from the lexical information of each attribute using semantic similarity, as shown in Section 2.1.

The *opSim* between all operations in c_1 and all operations in c_2 is expressed as Eq. (6). Operation collection OP_1 in class c_1 is compared with operation collection OP_2 in class c_2. We then calculated the similarity of each *i*-th operation (op_i) of class c_1 with that of the *j*-operation (op_j) of class c_2. We then used *changePivot* to change the values of rows and columns in the similarity matrix following Algorithm 1. Operations consist of a modifier, an operation name, a data type, and a parameter. Therefore, we needed a detailed operation similarity (*dopSim*) between operations op_1 and op_2, which contains these four things. *dopSim* between operations op_1 and op_2 can be expressed as Eq. (7).

$$dopSim(op_1, op_2) = w_{mdO} \times sim(mdO_1,mdO_2) + w_{naO} \times CosineSim(naO_1,naO_2) + w_{tyO} \times sim(tyO_1,tyO_2) + w_{parO} \times prSim(op_1,op_2). \quad (7)$$

Weight w_{mdO} is the weight of operation modifier similarity between operation modifiers mdO_1 and mdO_2. Weight w_{naO} is the weight of operation name similarity between operation names naO_1 and naO_2.

$$prSim(op_1, op_2) = \frac{2 \times \left(\sum_{k=1}^{Min(|PR_1|,|PR_2|)} changePivot \left(Max (\sum_{i=1}^{|PR_1|} \sum_{j=1}^{|PR_2|} CosSim(pr_i, pr_j)) \right) \right)}{|PR_1| + |PR_2|}. \tag{8}$$

$$raSim(d_1, d_2) = \frac{2 \times \left(\sum_{k=1}^{Min(|RA_1|,|RA_2|)} changePivot \left(Max (\sum_{i=1}^{|RA_1|} \sum_{j=1}^{|RA_2|} draSim(ra_i, ra_j)) \right) \right)}{|RA_1| + |RA_2|}. \tag{10}$$

Weight w_{tyO} is the weight of operation data type similarity between operation data types tyo_1 and tyO_2. Weight w_{parO} is the weight of parameter similarity between operations op_1 and op_2. We used arbitrary weights of $w_{mdO} = 0.1$, $w_{naO} = 0.7$, $w_{tyO} = 0.1$, and $w_{parO} = 0.1$, and emphasized the name of the operation. Calculation of similarity between modifiers and data types in operations is the same as that between modifiers and data types in attributes. The similarity between operation names can be calculated from the lexical information of each operation using semantic similarity, as shown in Section 2.1. We cannot directly calculate the similarity of parameters ($prSim$) because an operation can have more than one parameter. $prSim$ is expressed as Eq. (8).

Parameter collection PR_1 from operation op_1 is compared with parameter collection PR_2 from operation op_2. The ith-parameter (pr_i) of the first operation is calculated by the jth-parameter (pr_j) of the second operation. We then used $changePivot$ to change the values of rows and columns in the similarity matrix following Algorithm 1. Similarity between pr_i and pr_j is calculated using semantic similarity, as shown in Section 2.1. The $relSim$ between class diagrams d_1 and d_2 can be expressed as Eq. (9).

$$relSim(d_1, d_2) = w_{ra} \times raSim(d_1, d_2) + w_{rd} \times rdSim(d_1, d_2) + w_{rg} \times rgSim(d_1, d_2). \tag{9}$$

Here, w_{ra} is the weight of association relationship similarity ($raSim$) between class diagrams d_1 and d_2, w_{rd} is the weight of dependency relationship similarity ($rdSim$) between class diagrams d_1 and d_2, and w_{rg} is the weight of generalization relationship similarity ($rgSim$) between class diagrams d_1 and d_2. Relationships are separated because it is unfair to equally compare relationships when the relationships being compared are different. The weight determination is adopted from previous studies [34] by looking at the availability of elements to be calculated. If neither diagram has generalizations, the weight for generalizations is 0. The weight value to be calculated is the result of sharing the group of relationships sought by the sum relationship of the two diagrams. For example, d_1 has three associations and one dependency, and d_2 has two associations.

Thus, the weight of the association is 5/6, the weight of dependency is 1/6, and the weight of generalization is 0. The $raSim$ value between class diagrams d_1 and d_2 can be expressed as Eq. (10).

The association relationship collection RA_1 from class diagram d_1 is compared with the association relationship collection RA_2 from class diagram d_2. The ith-association (ra_i) from class diagram d_1 is calculated along with the jth-association (ra_j) from class diagram d_2, and $changePivot$ is used to change the row and column values in the similarity matrix following Algorithm 1. Furthermore, an association relationship consists of the source class, relation name, aggregation/composition, and target class. Therefore, we need a detailed association relationship similarity ($draSim$) between association relationships ra_1 and ra_2, which can be expressed as Eq. (11).

$$draSim(ra_1, ra_2) = w_{ra1} \times CosSim(src_1, src_2) \\ + w_{ra2} \times CosSim(nm_1, nm_2) \\ + w_{ra3} \times sim(ow_1, ow_2) \\ + w_{ra4} \times CosSim(tgt_1, tgt_2). \tag{11}$$

$draSim$ consists of source class src, relationship name nm, relation ownership ow, and target class tgt. Here, w_{ra1} is the weight of source class similarity between source classes src_1 and src_2, w_{ra2} is the weight of relationship name similarity between relationship names nm_1 and nm_2, w_{ra3} is the weight of ownership similarity between ownerships ow_1 and ow_2, and w_{ra4} is the weight of target class similarity between target classes tgt_1 and tgt_2. The total weight is 1, and the weights used are $w_{ra1} = 0.3$, $w_{ra2} = 0.2$, $w_{ra3} = 0.2$, and $w_{ra4} = 0.3$. The weight of the class name (source class and target class) is determined to be greater than that of the ownership and relationship name because they are always available in class diagrams. In contrast, relationship names are rarely written on class diagrams and ownership consists only of composition or aggregation. Class name and relationship name similarities use semantic similarity, as shown in Section 2.1. The value of ownership similarity is 0 if they are different and 1 if they are not different. The $rdSim$ between class diagrams d_1 and d_2 can be expressed as Eq. (12).

Here, dependency relationship collection RD_1 from class diagram d_1 is compared with dependency relationship collection RD_2 from class diagram d_2.

$$rdSim(d_1, d_2) = \frac{2 \times \left(\sum_{k=1}^{Min(|RD_1|,|RD_2|)} changePivot \left(Max \left(\sum_{i=1}^{|RD_1|} \sum_{j=1}^{|RD_2|} drdSim(rd_i, rd_j) \right) \right) \right)}{|RD_1| + |RD_2|}. \quad (12)$$

$$drdSim(rd_1, rd_2) = \frac{CosSim(src_1, src_2) + CosSim(tgt_1, tgt_2)}{2}. \quad (13)$$

$$rgSim(d_1, d_2) = \frac{2 \times \left(\sum_{k=1}^{Min(|RG_1|,|RG_2|)} changePivot \left(Max \left(\sum_{i=1}^{|RG_1|} \sum_{j=1}^{|RG_2|} drgSim(rg_i, rg_j) \right) \right) \right)}{|RG_1| + |RG_2|}. \quad (14)$$

$$drgSim(rg_1, rg_2) = \frac{CosSim(src_1, src_2) + CosSim(tgt_1, tgt_2)}{2}. \quad (15)$$

The ith-dependency relationship (rd_i) from class diagram d_1 is calculated along with the jth-dependency relationship (rd_j) from class diagram d_2, and *changePivot* is used to change the row and column values in the similarity matrix following Algorithm 1. The dependency relationship consists of the source and target classes. Therefore, we need an advanced dependency relationship similarity (*drdSim*) between dependency relationships rd_1 and rd_2, which can be expressed as Eq. (13).

Class name (source class and target class) similarity uses semantic similarity as shown in Section 2.1. The generalization relationship similarity (*rgSim*) between class diagrams d_1 and d_2 can be expressed as Eq. (14).

Generalization relationship collection RG_1 from class diagram d_1 is compared with generalization relationship collection RG_2 from class diagram d_2. The ith-generalization relationship (rg_i) from class diagram d_1 is calculated along with the jth-generalization relationship (rg_j) from class diagram d_2, and *changePivot* is used to change the row and column values in the similarity matrix following Algorithm 1. Furthermore, the generalization consists of the source and target classes. Therefore, we need an advanced generalization relationship similarity (*drgSim*) between generalization relationships rg_1 and rg_2, which can be expressed as Eq. (15).

The class name similarity includes the source (*src*) and target class names (*tgt*), and class name similarity is calculated along with semantic similarity, as shown in Section 2.1.

3. Structural similarity assessment

Structural similarity in class diagrams can be represented by making a UML common graph (UCG) model of each diagram. Thus, the pair of diagrams for which similarity will be calculated is converted to UCG, and the resulting UCG pair is searched for similarity values.

3.1 Class diagram structural similarity

UCG models were originally intended to build graphs that can accommodate several UML diagrams with the same characters. In this study, we proposed a graph that can provide both class and sequence diagrams. The concept is that the UCG model can calculate structural similarity between two class diagrams and between sequence diagrams, as well as the similarity of class and sequence diagrams. UCG models are currently used to calculate the structural similarity between two class diagrams. As with graphs in general, UCG consists of two elements, namely, vertices and edges. The UCG used in this study is thus an improvement on conventional UCG [21].

A UCG can only be used to create models for class diagrams. In this study, we built a model that can include several UML diagrams. Table 1 shows the designation of the UCG model, which provides for class and sequence diagrams. To this, we added object vertex, class edge, reply message edge, synchronous message edge, and asynchronous message edge components. We propose calculations on sequence diagrams similar to class diagram

Table 1. UML common graph description

| No	Type	Name	Tag
1	Vertex	Class Vertex	vc
2	Vertex	Attribute Vertex	va
3	Vertex	Operation Vertex	vo
4	Vertex	Parameter Vertex	vp
5	**Vertex**	**Object Vertex**	**vob**
6	Edge	Attribute Edge	e_a
7	Edge	Operation Edge	e_o
8	Edge	Parameter Edge	e_p
9	Edge	Association Edge	e_1
10	Edge	Generalization Edge	e_2
11	Edge	Aggregation Edge	e_3
12	Edge	Composition Edge	e_4
13	Edge	Dependency Edge	e_5
14	Edge	Realization Edge	e_6
15	**Edge**	**Class Edge**	**ec**
16	**Edge**	**Reply Message Edge**	**e_7**
17	**Edge**	**Synchronous Message Edge**	**e_8**
18	**Edge**	**Asynchronous Message Edge**	**e_9**
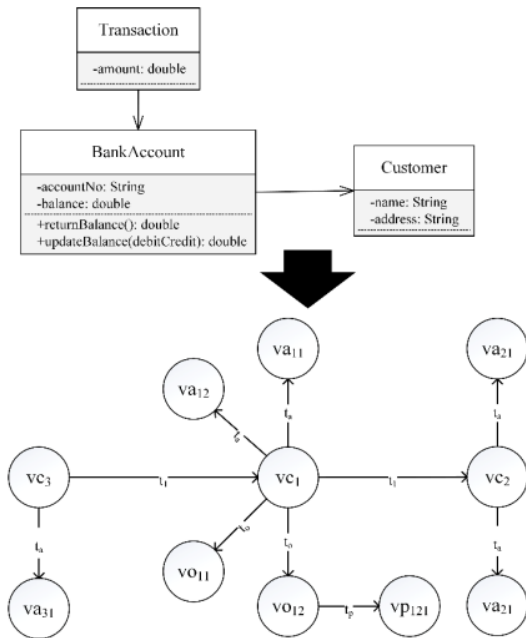
Figure. 4 Transformation of a class diagram into a UML common graph

calculations. These calculations are based on interactions between classes in the class diagram that also occur in the sequence diagram (*intraSim*). The difference is the information in the class (*interSim*).

Following Table 1, we can transform a class diagram to a UCG. Fig. 4 illustrates this process. The graph shows that a class can have zero or more attributes, zero or more operations, and zero or more related classes. Thus, an operation can also have zero or more parameters.

3.2 UCG similarity assessment

We obtained the structural similarity assessment of the class diagram from the UCG similarity assessment. Fig. 5 shows an assessment of structural similarity. Diagrams are divided based on structure and then calculated from the smallest graph. The results of these calculations are then entered into the next calculation.

All proposed structural calculations disregard the lexical information of the class diagram because structural similarity is different from semantic similarity, as explained previously. Structural similarity only considers the types of components in the class diagram. The UCG of class diagrams contains two types of information: *intraStructure* and *interStructure* [21]. *IntraStructure* (*intraSim*) contains the number of attributes and operations of
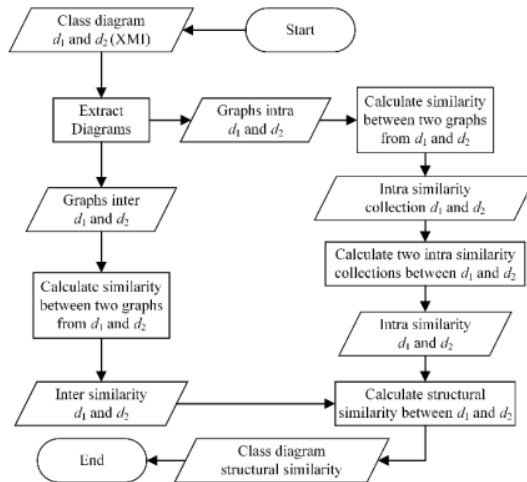
Figure. 5 Flowchart of structural similarity assessment

each class, and *interStructure* (*interSim*) includes the relationship information between the two classes. Consequently, the structural similarity between two UCGs must consider both classes of information, the importance of which is distinguished by ρ_{struc}, which ranges from 0 to 1. Eq. (16) shows the structural assessment between class diagrams d_1 and d_2.

$$classStruc(d_1, d_2) = (1 - \rho_{struc}) \times intraSim(g_1, g_2) + \rho_{struc} \times interSim(g_1, g_2). \tag{16}$$

Graphs g_1 and g_2 are the initial and the comparative UCGs, respectively. Unlike previous studies, we calculated *intraSim* by combining GED [35] using greedy algorithms. Eq. (17) shows how to calculate *intraSim*. The information in each class contains numbers of attributes, operations, and parameters. Thus, a graph is split into several subgraphs based on classes. Here, vc_i is a subgraph of g_1 and g_2. Thus, each class of g_1 is calculated using the GED for all classes of g_2 as Eq. (17).

Here, VC_1 is a collection of subgraphs in g_1, and VC_2 is a collection of subgraphs in g_2. The *changePivot* function is to change the all points row and column values to 0 following Algorithm 1. After determining the similarity of the two subgraphs between g_1 and g_2 using GED (*GED (vci, vcj)*), the greedy algorithm finds the optimal similarity value between the two. This value is then assigned to be the value of *intraSim*. GED calculates the minimum cost to convert g_1 to g_2, and the minimum costs are calculated as Eq. (18).
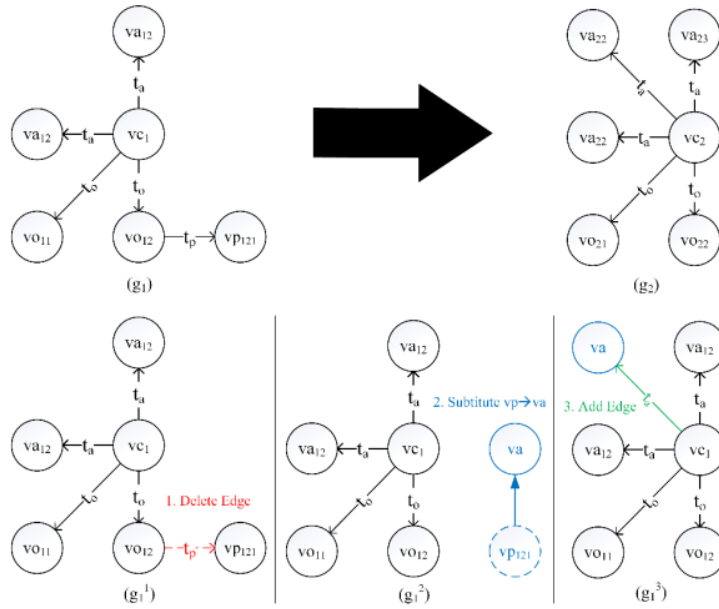
Figure. 6 Subgraph matching using graph edit distance in *intraSim*

$$intraSim(g_1, g_2) = \frac{2 \times \left(\sum_{k=1}^{Min(|VC_1|,|VC_2|)} changePivot\left(Max(\sum_{i=1}^{|VC_1|} \sum_{j=1}^{|VC_2|} GED(vc_i, vc_j)))\right)\right)}{|VC_1| + |VC_2|}. \tag{17}$$

$$d_{\lambda_{min}}(g_1, g_2) = \frac{min}{\lambda \in \gamma(g_1, g_2)} \sum_{e_i \in \lambda} c(e_i). \tag{18}$$

Here, $d_{\lambda_{min}}$ is the lowest cost graph path. GED searches for all possible permutations of the path and calculates the cost of changing g_1 to g_2. The GED we used has three operations: substitution, addition, and deletion. Substitution is accomplished by changing the edges, addition is done by adding edges and vertices according to the destination graph, and deletion is performed by removing edges and vertices according to the destination graph. Here, λ_{min} produces the smallest change cost amount, so it must be normalized into a 0–1 scale. Fig. 6 illustrates how g_1 subgraphs are changed into g_2 subgraphs to calculate *intraSim*. Fig. 6 shows that the minimum cost for converting the first subgraph into the second subgraph is 3. Therefore, the normalized value of similarity for the two subgraphs is $1 - (3/11)$. 11 is obtained by taking the maximum of the sum of vertices and edges in g_1 and g_2.

Unlike previous studies, we used GED [36] to measure *interSim*, which only takes the relationship information between classes from UCGs, meaning that the shape of the graph to be calculated differs from *intraSim*. Fig. 7 shows an example of the form

of *interSim* used to measure the similarity of two class diagrams, CD$_1$ and CD$_2$. CD$_1$ has three classes and two relations, and CD$_2$ has three classes and three relations.

Prior research used a maximum common subgraph that was more suitable for reuse. In this study, we used GED because it matches the similarity assessment that must not see a portion of the graph content. Eq. (19) measures *interSim* with the concept of GED.
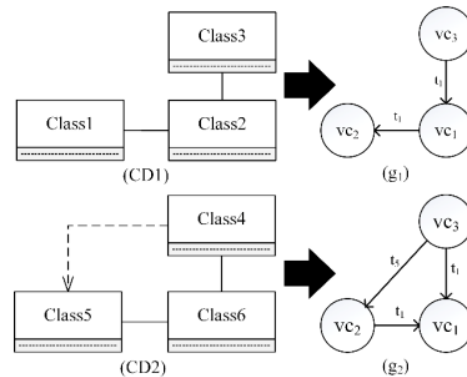

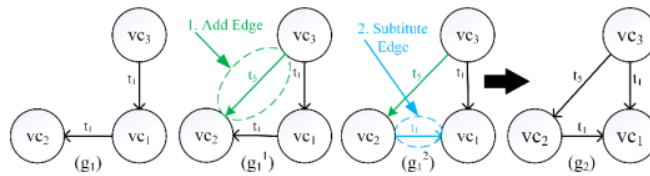Figure. 7 Graph used to measure *interSim*

Figure. 8 Graph matching using graph edit distance in interSim

Table 2. Variable description

Variable	Description	Variable	Description	Variable	Description
$classSem$	Semantic assessment	OP_2	Operation collection of c_2	nm	Name of the relationship
d_1	Firts class diagram	op_i	The i-operation of OP_1	ow	aggregation/composition
d_2	Second class diagram	op_j	The j-operation of OP_2	tgt	Target class name
ρ_{sem}	Distinguishing semantic component importance	$dopSim$	Detailed operation similarity	wra_1	Weight of source class name
$propSim$	Property similarity	mdO_1	Modifier of op_1	wra_2	Weight of relationship name
$relSim$	Relationship similarity	mdO_2	Modifier of op_2	wra_3	Weight of relationship own
C_1	Class collection of d_1	naO_1	Name of operation op_1	wra_4	Weight of target class name
C_2	Class collection of d_2	naO_2	Name of operation op_2	RD_1	Dependency collection of d_1
c_i	The i-class from C_1	tyO_1	Data type of op_1	RD_2	Dependency collection of d_2
c_j	The j-class from C_2	tyO_2	Data type of op_2	rd_i	The i-relationship of RD_1
$cSim$	Class similarity	$prSim$	Parameter similarity	rd_j	The j-relationship of RD_2
$cnSim$	Class Name Similarity	w_{mdO}	Weight of operation modifier similarity	$drdSim$	Detailed dependency similarity
$aSim$	Attribute Similarity	w_{naO}	Weight of operation name similarity	RG_1	Generalization collection of d_1
$opSim$	Operation Similarity	w_{tyO}	Weight of operation data type similarity	RG_2	Generalization collection of d_2
w_{cn}	Weight of class name similarity	w_{parO}	Weight of parameter similarity	rg_i	The i-relationship of RG_1
w_a	Weight of attribute similarity	PR_1	Parameter collection of operation op_1	rg_j	The j-relationship of RG_2
w_{op}	Weight of operation similarity	PR_2	Parameter collection of operation op_2	$drgSim$	Detailed generalization similarity
A_1	Attribute collection of c_1	pr_i	The i-parameter of op_1	$classStruc$	Structural assessment
A_2	Attribute collection of c_2	pr_j	The j-parameter of op_2	ρ_{struc}	Distinguishing structural component importance
a_i	The i-attribute from A_1	$relSim$	Relationship similarity	$intraSim$	Similarity of $intraStructure$
a_j	The j-attribute from A_2	$raSim$	Association similarity	$interSim$	Similarity of $interStructure$
$daSim$	Detailed attribute similarity	$rdSim$	Dependency similarity	g_1	Translated graph of d_1
md_1	Modifier of attribute a_1	$rgSim$	Generalization similarity	g_2	Translated graph of d_2
md_2	Modifier of attribute a_2	w_{ra}	Weight of association similarity	$VC1$	Collection of subgraphs in g_1
na_1	Name of attribute a_1	w_{rd}	Weight of dependency similarity	$VC2$	Collection of subgraphs in g_1
na_2	Name of attribute a_2	w_{rg}	Weight of generalization similarity	vc_i	The i-graph from VC_1
ty_1	Data type of attribute a_1	RA_1	Association collection from d_1	vc_j	The j-graph from VC_2
ty_2	Data type of attribute a_2	RA_2	Association collection from d_2	$d_{\lambda_{min}}$	The lowest cost graph path
w_{md}	Weight of modifier	ra_i	The i-relationship of RA_1	$c(e_i)$	The cost of transforming g_1 into g_2
w_{na}	Weight of attribute name	ra_j	The j-relationship of RA_2	$classSim$	Class diagram similarity assessment
w_{ty}	Weight of data type	$draSim$	Detailed association similarity	ρ	Distinguishing assessment component importance
OP_1	Operation collection of c_1	src	Source class name		

$$interSim(g_1, g_2) = 1 - \frac{d\lambda_{min}(g_1, g_2)}{\max(|g_1|, |g_2|)}. \quad (19)$$

Here, $d\lambda_{min}$ is the minimum cost to convert g_1 into g_2, and $max(|g_1|, |g_2|)$ is the most significant value of the sum of edges and vertices in g_1 or g_2. Fig. 8 shows

graph matching with GED in *interSim* and shows that 2 is the cost of converting g_1 to g_2. There are add vertex and substitute edge operations.

4. Class diagram assessment

The proposed class similarity diagram (*classSim*) assessment combines semantic and structural similarities as a similarity assessment parameter. The similarity of class diagrams d_1 and d_2 is determined using Eq. (20). Semantic similarity (*classSem*) uses all lexical information that appears in the class diagram, and structural similarity (*classStruc*) uses a graph model representing the class diagrams. Structural and semantic similarities have different importance with respect to ρ, the value of which ranges from 0 to 1.

$$classSim = (1 - \rho) \times classSem(d_1, d_2) + \rho \times classStruc(d_1, d_2). \quad (20)$$

The explanation of each variable from Eq. (1) to 20 can be seen in Table 2.

5. Result

5.1 Dataset

The purpose of this study is to develop an automatic assessment method. Therefore, the data are a collection of class diagrams that are answers to student assessments. In the assessments, we gave students three questions related to making class diagrams, and each question contained a different project. Table 3 shows the project details. The projects present information on use case descriptions, and students construct class diagrams based on these use case descriptions. Each project has an answer key that serves as a reference for assessment. The collected class diagram consists of 31 diagrams from three different projects.

5.2 Gold standard

A gold standard serves as a reference for evaluating a proposed method's reliability. Thus, the proposed method's reliability refers to its agreement with a gold standard developed by experts. We measured the agreement using Gwet's AC1 [37].

Table 3. Summary of the collected class diagram

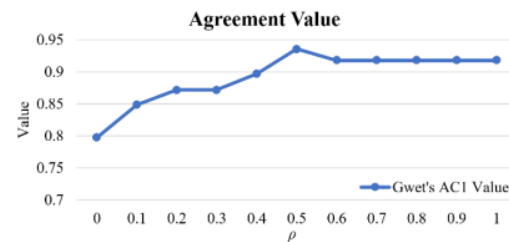| Project	Description	Answers	Class Average	Relationship Average
Outlay	Financial recording	11	9	7
Library	Book rental	10	9	9
QuickBill	Point of sale	10	6	6

Figure. 9 The maximum agreement value of each ρ

The gold standard in this study was built based on expert evaluation questionnaires. Experts were asked to assess student answers to the questions and were given answer keys. The assessment range was 0–100. The experts were software engineering lecturers who have taught class diagrams for two to fourteen years at bachelor degree programs. In this study, we relied on 27 experts who assessed 31 pairs of student and key answers. The nationality of the entire expert is Indonesian. Age of experts ranging from thirty to fifty-two years old. The experts are from nine different universities in Indonesia.

5.3 Similarity assessment evaluation

We evaluated similarity assessment by comparing the proposed method's results with those from the gold standard using Gwet's AC1 to measure inter-rater reliability. We normalized expert assessment values and the proposed method's results on a 1–5 scale, where the scale number represents a range of assessment values with an even distribution (1: <20, 2: <40, 3: <60, 4: <80, and 5: >80). The proposed method is assessed using all combinations of ρ, ρ_{sem}, and ρ_{struc} as differentiators of the importance of assessment components. The experiment was repeated 1331 times with different weight combinations, and the weights were tweaked with the same inputs. The values of ρ, ρ_{sim}, and ρ_{struc} were 0, 0.1, and 0.2 to 1, respectively. Fig. 9 shows the maximum agreement value for each instance of ρ. The maximum agreement value was 0.935 at $\rho = 0.5$, $\rho_{sem} = 0.3$, and $\rho_{struc} = 0$ and 0.1.

6. Discussion

The maximum agreement value resulting from our evaluation indicates that the proposed method can function as reliably as an expert can. This is because experts and the proposed method have an almost perfect level of agreement. We found that experts look at the semantic and structural aspects of class diagrams equally when assessing class diagram similarity. Class diagrams are easy to visually understand using the lexical information they contain.
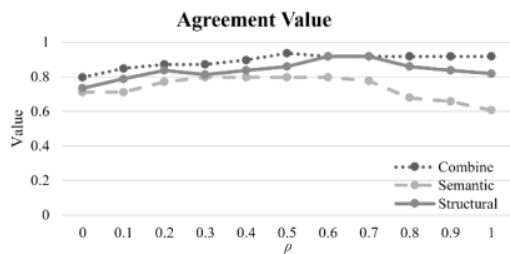
Agreement Value

Figure. 10 Comparison of agreement with previous research in assessment case

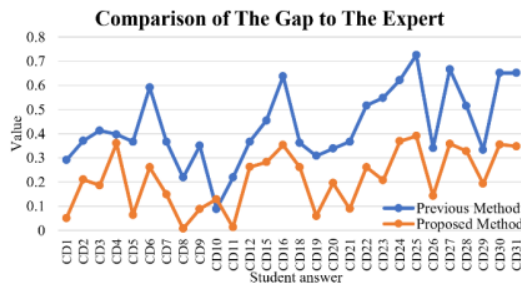**Comparison of The Gap to The Expert**

Figure. 11 Comparison of the gap to the experts

For semantic similarity, experts tend to look at the lexical information from the class separately and tend to look at the property information for each class. Experts do not pay more attention to relationships between classes when making an assessment. For structural similarity, experts look at the structure in the class almost exclusively, rather than considering within-class and between-class structures equally.

We conducted several evaluations of expert judgment before using those judgments as a gold standard. First, we tested the collection of expert answers from each pair of diagrams for correlation using the Pearson correlation coefficient [38]. This stage reduces the three pairs of diagrams, eliminating those that do not have strong correlations or have too many different answers. Second, we conducted an inter-rater reliability evaluation [39] to show the level of agreement among the experts. During this stage, we eliminated five experts to provide more reliable and homogeneous answers. The final test result for data reliability was 0.935 on a 0–1 scale. Therefore, we conclude that the data used re reliable.

We compared research that focused solely on semantic aspects [28] and research that considers only structural aspects [21] with our proposed method that uses both. Fig. 10 shows that using structural and semantic combinations with individual weights can provide better agreement values than separate assessment.

The maximum value of each ρ in semantic similarity tends to be more significant on the left side

of the chart. In line with our proposed method, experts tend to look at the properties of each class rather than between-class relationships when assessing semantic aspects alone. In contrast, the maximum value of each ρ in structural similarity tends to be higher on the right side of the chart. Experts tend to look at relationships between classes rather than within-class structures when assessing structural aspects alone, which differ slightly from our proposed method. The structural similarity tendency still cannot be concluded in general because experts have difficulty accurately measuring the similarity of structures between two class diagrams. We attempted small experiments to measure simple class diagrams and found that experts cannot assess structural similarity as accurately as the proposed method. The proposed method uses GED, which can accurately compare all class relationships. Several factors can influence the accuracy gap between expert assessment and that of the proposed method. These factors include the number of classes, the complexity of each class's property structure, and the number of relationships between classes. Experts find assessment increasingly difficult if the number of these factors increases.

We also compared the results of this study with previous studies. Ali [40] assessed the syntactic similarity of class names, attribute names, and operation names. Fig. 11 shows the results of the comparison. We calculate the gap from the output of our method and Ali's method to the expert assessment. The assessment results use a value range of 0 to 1 where zero is not the same, and one is the same. Based on Fig. 11, 27 of the 28 assessments carried out by our method had a lower gap with the expert assessment. Therefore, our proposed method using semantic and structural assessment provides a closer value to the expert.

The class diagrams used are not for entire projects. We use this limitation to maintain the reliability of the gold standard that is built on expert interpretation. Experts have difficulty assessing the similarity of two diagrams if those diagrams have large and complex shapes. Additionally, experts will be inconsistent when assessing many problems with complex class diagrams.

7. Conclusion

Our proposed automated assessment method can determine the similarity of two class diagrams as reliably as an expert. The level of agreement between the proposed method and experts was 0.935. This means that our proposed method has an almost perfect agreement with expert assessment. Class

diagram similarity is divided into semantic similarity and structural similarity. Combining these two similarities with a specific weight enables the best agreement with experts. Semantic similarity is an aspect experts emphasize in the case of similarity assessment.

This research can be developed further. First, our assessment concept can be used with other UML diagrams because every UML diagram has both semantic and structural aspects. Second, further research is needed to ascertain experts' tendency to assess structural similarity to become an assessment standard that would be used in all assessment cases. Each similarity's weight can be examined for purposes other than assessment such as software reuse and clone detection.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

Conceptualization, Reza Fauzan and Daniel Siahaan; methodology, Reza Fauzan and Daniel Siahaan; software, Reza Fauzan; validation, Reza Fauzan, Daniel Siahaan, and Siti Rochimah; formal analysis, Reza Fauzan, Daniel Siahaan, and Siti Rochimah; investigation, Evi Triandini; resources, Evi Triandini; data curation, Siti Rochimah; writing—original draft preparation, Reza Fauzan; writing—review and editing, Daniel Siahaan and Siti Rochimah; visualization, Reza Fauzan and Evi Triandini; supervision, Daniel Siahaan and Siti Rochimah; project administration, Daniel Siahaan.

Acknowledgments

References

[1] L. J. Gonçales and K. Farias, "Towards a hybrid approach to measure similarity between UML models", In: *Proc. of ACM Int. Conf. Ser.*, 2019.

[2] L. J. Gonçales, K. Farias, and V. Bischoff, "On the effects of developers' intuition on measuring similarity between UML models", In: *Proc. of ACM Int. Conf. Ser.*, 2019.

[3] W. N. Robinson and H. G. Woo, "Finding reusable UML sequence diagrams automatically", *IEEE Softw.*, Vol. 21, No. 5, pp. 60–67, 2004, doi: 10.1109/MS.2004.1331304.

[4] A. Adamu and W. M. N. W. Zainoon, "A Framework for Enhancing the Retrieval of UML Diagrams", In: *Proc. of Software Reuse: Bridging with Social-Awareness 15th International Conference, ICSR 2016*, pp. 384–390, 2016.

[5] S. Singh, "Clone Detection in UML Class Models using Class Metrics", *ACM SIGSOFT Softw. Eng. Notes*, Vol. 39, No. 3, pp. 1–4, 2014.

[6] H. Storrle, "Towards Clone Detection in UML Domain Models", *Softw. Syst. Model.*, Vol. 12, No. 2, pp. 307–329, 2013.

[7] J. Kaur and R. Singh, "Implementation of Model Cloning in Software Models using UML diagrams", *Int. J. Technol. Comput.*, Vol. 2, No. 4, pp. 114–119, 2016.

[8] G. D. Kuh, N. Jankowski, S. O. Ikenberry, and J. Kinzie, *Knowing what students know and can do: The current state of student learning outcomes assessment in US colleges and universities*, No. January. Champaign: National Institute for Learning Outcomes Assessment, 2014.

[9] W. J. Park and D. H. Bae, "A two-stage framework for UML specification matching", *Inf. Softw. Technol.*, Vol. 53, No. 3, pp. 230–244, 2011.

[10] H. O. Salami and M. Ahmed, "A framework for reuse of multi-view UML artifacts", *Int. J. Soft Comput. Softw. Eng.*, Vol. 3, No. 3, pp. 156–162, 2013.

[11] W. K. G. Assuncao and S. R. Vergilio, "Class Diagram Retrieval with Particle Swarm Optimization", In: *Proc. of the 25th Software Engineering on International Conf. & Knowledge Engineering*, pp. 632–637, 2013.

[12] K. Robles, A. Fraga, J. Morato, and J. Llorens, "Towards an ontology-based retrieval of UML class diagrams", *Inf. Softw. Technol.*, Vol. 54, No. 1, pp. 72–86, 2012.

[13] D. H. Qiu, H. Li, and J. L. Sun, "Measuring Software Similarity based on Structure and Property of Class Diagram", In: *Proc. of Sixth International Conf. on Advanced Computational Intelligence*, pp. 75–80, 2013.

[14] M. A.-R. Al-Khiaty and M. Ahmed, "UML Class Diagrams: Similarity Aspects and Matching", *Lect. Notes Softw. Eng.*, Vol. 4, No. 1, pp. 41–47, 2016.

[15] M. A. R. Al-Khiaty and M. Ahmed, "Matching UML class diagrams using a Hybridized Greedy-Genetic algorithm", In: *Proc. of Int. Sci.*

Tech. Conf. Comput. Sci. Inf. Technol., Vol. 1, pp. 161–166, 2017.

[16] M. A.-R. M. Al-Khiaty and M. Ahmed, "Similarity Assessment of UML Class Diagrams using a Greedy Algorithm", In: *Proc. of International Computer Science and Engineering Conf.*, pp. 19–23, 2014.

[17] M. A.-R. Al-Khiaty and M. Ahmed, "Similarity assessment of UML class diagrams using simulated annealing", In: *Proc. of International Conf. on Software Engineering and Service Science*, pp. 19–23, 2014.

[18] A. Adamu and W. M. N. W. Zainon, "Multiview Similarity Assessment Technique of UML Diagrams", *Procedia Comput. Sci.*, Vol. 124, pp. 311–318, 2017.

[19] A. Adamu and W. M. N. W. Zainon, "Similarity Assessment of UML Sequence Diagrams Using Dynamic Programming", In: *Proc. of International Visual Informatics Conf.*, vol. 1, pp. 270–278, 2017.

[20] A. Adamu, W. M. N. Wan Zainon, and S. M. Abdulrahman, "Empirical investigation of UML models matching through different weight calibration", In: *Proc. of ACM Int. Conf. Ser.*, Vol. Part F1479, pp. 167–172, 2019.

[21] Z. Yuan, L. Yan, and Z. Ma, "Structural similarity measure between UML class diagrams based on UCG", *Requir. Eng.*, No. 0123456789, pp. 1–17, 2019.

[22] L. Huang, Y. Duan, X. Sun, Z. Lin, and C. Zhu, "Enhancing UML Class Diagram Abstraction with Knowledge Graph", In: *Proc. of International Conf. on Intelligent Data Engineering and Automated Learning*, pp. 606–616, 2016.

[23] J. Masip, C. Martínez, I. Blandón-Gitlin, N. Sánchez, C. Herrero, and I. Ibabe, "Learning to detect deception from evasive answers and inconsistencies across repeated interviews: A study with lay respondents and police officers", *Front. Psychol.*, Vol. 8, No. JAN, pp. 1–17, 2018.

[24] A. Ylonen, H. Gillespie, and A. Green, "Disciplinary differences and other variations in assessment cultures in higher education: exploring variability and inconsistencies in one university in England", *Assess. Eval. High. Educ.*, Vol. 43, No. 6, pp. 1009–1017, 2018.

[25] J. Moreno and A. F. Pineda, "A Framework for Automated Formative Assessment in Mathematics Courses", *IEEE Access*, Vol. 8, pp. 30152–30159, 2020.

[26] M. Y. Jebli, R., Elbouhdidi, J., & Chkouri, "Automatic Evaluation of UML Class Diagrams Using the XML Schema Matching and the Machine Learning Algorithm", in *Advanced Intelligent Systems for Sustainable Development (AI2SD'2018)*, 2019, Vol. 912, pp. 149–156.

[27] G. Gleb and N. Alexey, "Approach to determining the structural similarity of software projects", *Open Semant. Technol. Des. Intellect. Syst.*, 2018.

[28] R. Fauzan, D. Siahaan, and S. Rochimah, "Class Diagram Similarity Measurement: A Different Approach", In: *Proc. of 2018 3rd International Conf. on Information Technology, Information System and Electrical Engineering (ICITISEE)*, pp. 215–219, 2018.

[29] G. Majumder, P. Pakray, A. Gelbukh, and D. Pinto, "Semantic textual similarity methods, tools, and applications: A survey", *Comput. y Sist.*, Vol. 20, No. 4, pp. 647–665, 2016.

[30] C. D. Manning, J. Bauer, J. Finkel, and S. J. Bethard, "The Stanford CoreNLP Natural Language Processing Toolkit", In: *Proc. of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 55–60, 2014.

[31] L. J. Garcia Castro, R. Berlanga, and A. Garcia, "In the pursuit of a semantic similarity metric based on UMLS annotations for articles in PubMed Central Open Access", *J. Biomed. Inform.*, Vol. 57, pp. 204–218, 2015.

[32] J. Gao, B. Zhang, and X. Chen, "A WordNet-based semantic similarity measurement combining edge-counting and information content theory", *Eng. Appl. Artif. Intell.*, Vol. 39, pp. 80–88, 2015.

[33] Y. Feng, E. Bagheri, F. Ensan, and J. Jovanovic, "The state of the art in semantic relatedness: a framework for comparison", *Knowl. Eng. Rev.*, pp. 1–30, 2017.

[34] R. Fauzan, D. Siahaan, S. Rochimah, and E. Triandini, "Activity diagram similarity measurement: A different approach", in *2018 International Seminar on Research of Information Technology and Intelligent Systems, ISRITI 2018*, pp. 601–605, 2018.

[35] V. Carletti, B. Gaüzère, L. Brun, and M. Vento, "Approximate Graph Edit Distance Computation Combining Bipartite Matching and Exact Neighborhood Substructure Distance", in *International Workshop on Graph-Based Representations in Pattern Recognition*, pp. 188–197, 2015.

[36] K. Riesen and H. Bunke, "Graph Edit Distance - Novel Approximation Algorithms", in *Handbook of Pattern Recognition and Computer Vision*, University of Birmingham, pp. 275–291, 2016.

[37] K. Gwet, "Kappa Statistic is not satisfactory for assessing the extent of agreement between raters", *Stat. Methods Inter-Rater Reliab. Assessmen*, No. 1, pp. 1–5, 2002.

[38] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson Correlation Coefficient", in *Noise reduction in speech ...*, Vol. 2, Springer Berlin Heidelberg, p. 229, 2009.

[39] R. T. Lange, "Inter-rater Reliability", in *Encyclopedia of Clinical Neuropsychology*, Kreutzer, J. S. and DeLuca, J. Caplan, and Bruce, Eds. Springer New York, pp. 1348-1348, 2011.

[40] N. H. Ali, Z. Shukur, and S. Idris, "A Design of an Assessment System for UML Class Diagram Noraida", In: *Proc. of - the 2007 International Conference on Computational Science and its Applications, ICCSA 2007*, 2007, pp. 539–544, doi: 10.1109/ICCSA.2007.31.

Automated Class Diagram Assessment using Semantic and Structural Similarities

2%
SIMILARITY INDEX

PRIMARY SOURCES

1	**vdocument.in** Internet	144 words — **2%**

EXCLUDE QUOTES	ON	EXCLUDE SOURCES	< 2%
EXCLUDE BIBLIOGRAPHY	ON	EXCLUDE MATCHES	OFF