International Journal on Advanced Science Engineering Information Technology

Vol.11 (2021) No. 6
ISSN: 2088-5334

An Automated Statechart Diagram Assessment using Semantic and Structural Similarities

Reza Fauzan [a,*], Daniel Siahaan [b], Siti Rochimah [b], Evi Triandini [c]

[a] Department of Electrical Engineering, Politeknik Negeri Banjarmasin, Banjarmasin, 70124, Indonesia
[b] Department of Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, 60111, Indonesia
[c] Department of Information Systems, Institut Teknologi dan Bisnis STIKOM Bali, Denpasar, 80234, Indonesia
Corresponding author: *reza.fauzan@poliban.ac.id

***Abstract*—The statechart diagram is a behavior diagram in the unified modeling language (UML) diagram. Numerous state chart diagrams are taught in computer science majors. In teaching and learning activities, the assessment process is essential. A teacher is required to be objective in assessing. However, objectivity can be affected by inconsistency and fatigue. Thus, an automatic assessment is very important. Automatic assessments can help teachers save time while assessing answers given by multiple students. By combining semantic and structural similarities, we propose a method to evaluate statechart diagrams automatically. Semantic comparison is conducted based on the lexical information from the states and transitions between the two diagrams. We then use a combination of cosine similarity, Wu palmer, and WordNet to assess the semantic similarity between the two diagrams. The structural assessment is conducted on the basis of the structure of the two diagrams using the greedy graph edit distance. The diagram structure is obtained by translating the diagram into several graphs. The graph is divided into two types of subgraphs, namely intraSim subgraph and interSim subgraph. Further, our results demonstrate that the proposed method agrees well with the state chart diagram assessed by the teacher. The agreement value between the teacher and our proposed method is an almost perfect agreement. In the assessment process, we observe that teachers see the structure of the statechart diagram instead of the lexical of the statechart diagram.**

***Keywords*—Automatic assessment; semantic assessment; statechart diagram; structural assessment; UML assessment.**

Manuscript received 3 Oct. 2020; revised 22 Apr. 2021; accepted 28 May 2021. Date of publication 31 Dec. 2021.

I. INTRODUCTION

The statechart diagram is a diagram in the unified modeling language (UML) diagram. Statechart diagrams describe the behavior of objects and representations of the life cycle of objects in software [1]. They can act as tools for testing functional software [2], [3]. Moreover, the notation on the statechart diagram is the design language used to describe the major software modules [4]. Statechart diagrams are generally taught in tertiary institutions, especially in the field of computer science. In the learning process, assessment is a problem complained about by teachers.

Additionally, the main cause of the teachers' complaints arises from the number of answers that must be assessed [5], [6]. Another cause is that teachers often become inconsistent while assessing student answers [7]. Therefore, an automatic assessment concept can be used to solve this problem. An automatic assessment method can help teachers to correctly evaluate their students' answers with a fixed assessment standard.

Statechart diagrams can be automatically assessed by measuring the similarity between any two statechart diagrams. Previously, automatic assessment has been applied to use case diagrams. Automatic assessment of a use-case diagram can be conducted by comparing its lexical information [8]. The statechart diagram similarity was only measured for reuse and clone detection and not for assessment. Similarity measurements were separately performed. First, the measurement uses lexical information syntactically. Second, the measurement uses state and transition flow structures. Herein, we combined these two pieces of information and detailed the contents of the information in our similarity measurement. Therefore, similarity assessment becomes comprehensive.

In UML diagrams, similarity measurement has been extensively studied [9], [10]. In addition, Storrle [11] syntactically measured the similarity between two statechart

diagrams based on their lexical properties. He aimed to detect clones from software designs. The lexical information of a statechart diagram is based on the names of the components [12]. The similarity in terms of lexical information can be semantically calculated using natural language processing (NLP) [13]–[15]. Semantic similarities are divided into three categories: knowledge-based, corpus-based, and string-based [16], [17]. In the case of knowledge-based semantic similarities, WordNet [18], [19] is generally used as a knowledge base. WordNet-based similarity calculations are performed using additional methods such as Wu Palmer [20], [21]. In the case of corpus-based semantic similarities, a large data corpus and complex data-training process are required. Finally, in the case of string-based semantic similarities, only those strings that are owned are considered irrespective of the meaning of the word.

Some other previously conducted studies [22]–[24] measured the similarity between two statechart diagrams in terms of another perspective, namely, the statechart diagram structure. The statechart diagram was converted into a graph. Moreover, the converted graph comprised states as vertices and transitions as edges. Their study aimed to reuse the software design. The graph similarity could be measured using the graph edit distance (GED) [25]–[27]. The concept of GED is to convert the first graph to the second graph. However, GED suffers from weakness, i.e., high processing time. Therefore, Riesen proposed the approximate GED [28] and greedy GED [29] to expedite the process.

Assessment problems in the computer science course might occur for several reasons. First, how to assess the answers by students without imposing additional burdens on the teacher [30]. Teachers need not spend a significant amount of time to conduct assessments. Second, we need to consider the objectivity of the assessment [31]. Objectivity can be affected by the fatigue and inconsistency of the teacher. Third, many students pose a problem in conducting assessments [32], [33]. Therefore, in the software engineering course, we need an automatic assessment method, the statechart diagram assessment.

This study aims to develop a new method for semantically and structurally assessing the similarity between two-statechart diagrams. Semantic similarity is assessed based on the lexical of statechart diagram. Lexical of statechart diagrams can be semantically compared. Moreover, the structural similarity is assessed based on the structure of the statechart diagram. The statechart diagram structure is translated into a graph. We compared the two graphs from the two-statechart diagrams by utilizing the greedy graph edit distance (GED). Our contribution is to obtain the level of importance of the assessment component so that our proposed method can function as reliably as experts.

II. Materials and Method

A. Statechart Diagram Similarity Assessment Characteristic

The characteristics of the statechart diagram similarity can be analyzed in terms of two types of diagrams, namely, lexical and structural diagrams. For example, Fig. 1 (a) illustrates a pair of diagrams with the same structures and lexically similar. Fig. 1 (b) shows another pair of diagrams with different structures but lexically similar, as shown in. In addition, we have observed that there are lexically similar cases but have a low number of states or transitions.

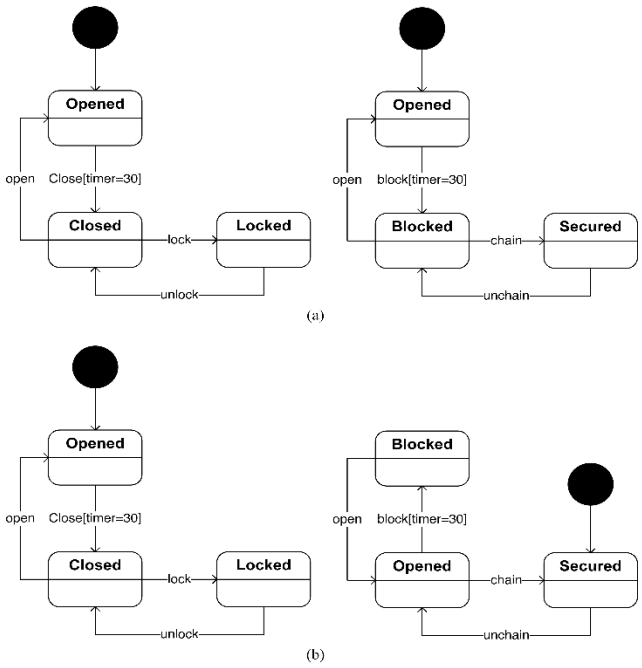

Fig. 1 Example of diagram similarity assessment characteristic (a) Example 1 and (b) Example 2

Furthermore, the answers provided by the students are not necessarily the same as the answer keys provided. Therefore, the answer diagrams may differ on the basis of how different students understand the description of the question. The extent of the vocabulary of the students may also influence the answers by the students.

B. Statechart Diagram Assessment

Statechart diagram similarity (*stdSim*) between statechart diagram d_1 and statechart diagram d_2 is assessed based on the results of semantic similarity (*semStd*) and structural similarity (*strucStd*). Both have their level of importance depending on the way an expert performs the assessment. Moreover, the level of importance is denoted by ρ. The value of ρ is greater than zero or less than equal to one. Statechart diagram similarity can be obtained using Equation 1.

$$stdSim(d_1, d_2) = (1 - \rho) \times semStd(d_1, d_2) + \rho \times strucStd(d_1, d_2) \quad (1)$$

Equation 1 is the main equation used to assess the similarity between two statechart diagrams. The flow of the statechart diagram assessment is shown in Fig. 2. The process begins with two statechart diagrams in the XMI (XML Metadata Interchange) format as input, namely, the answer key diagram and the student answer diagram. The extraction results of each diagram are divided into four main parts: a lexical collection of states, a lexical collection of transitions, a collection of subgraphs based on states and transitions (intra graphs), and a graph that shows the flow between states (inter graphs). The lexical collection is the input to assess the semantic similarity. The collection of subgraphs becomes an input to assess structural similarity. A more detailed explanation of semantic and structural assessment is described in the next section.

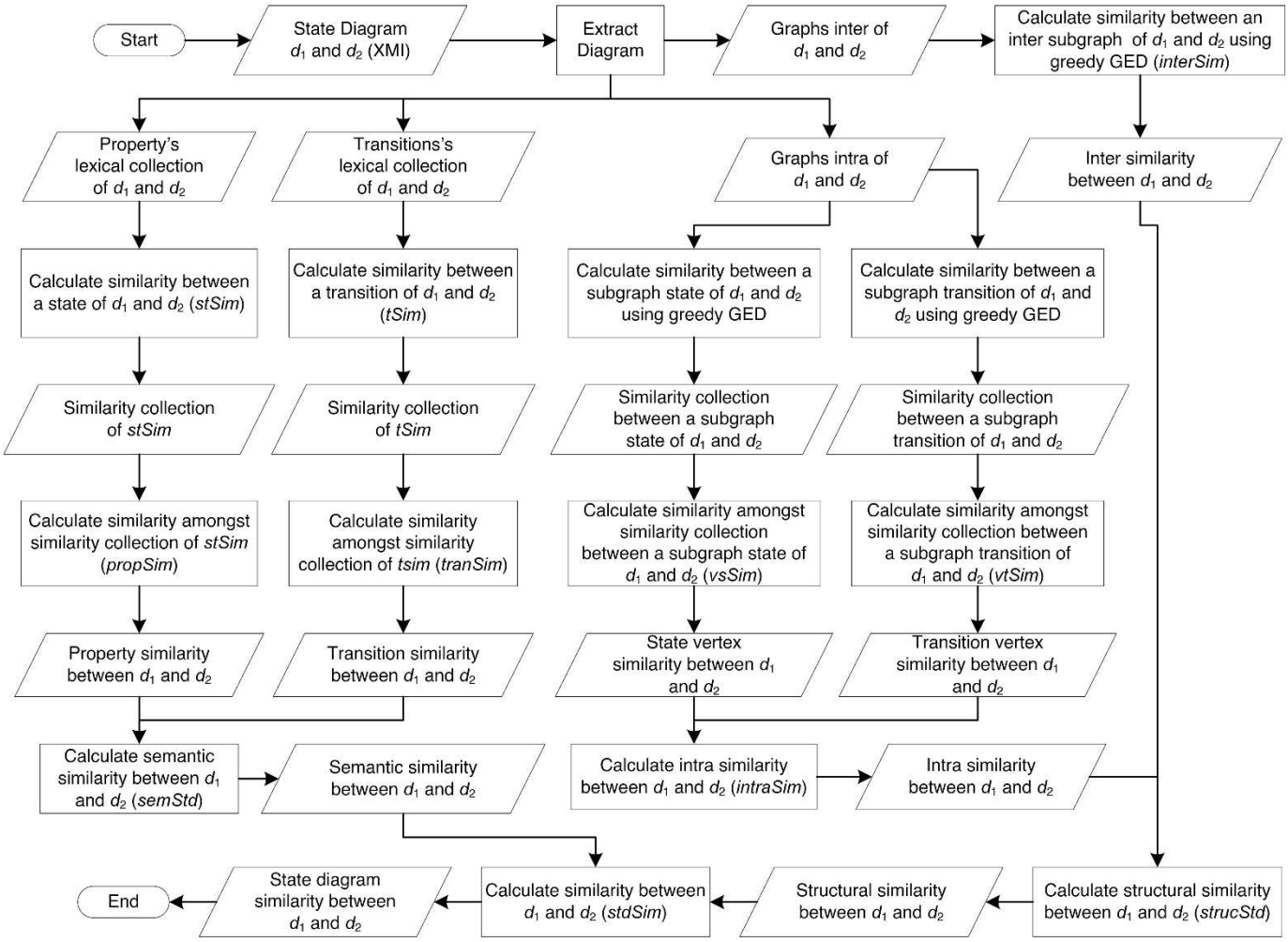

Fig. 2 Flow of state chart diagram assessment

C. *Semantic Similarity Assessment*

The comparison of the semantic similarity is based on the meaning of the statechart diagram. NLP [34] can be used to compare the meaning of each statechart diagram component. We performed a basic NLP on our process based on some previously conducted studies [35]–[40]. Besides, the NLP process applied is tokenization, point of sale (POS) tagging, stopwords removal, lemmatization, and cosine similarity. First, we perform tokenization if the component has a lexical that may comprise more than one word. Second, POS tagging is useful for providing part of speech information from these words. Third, meaningless words are removed. Fourth, the words are transformed into essential words using lemmatization. Fifth, by applying cosine similarity [41], the words of the first diagram component are compared with those in the second diagram component. In cosine similarity, assessment of the similarity between two words is calculated using WordNet and Wu Palmer.

Based on the type of lexical information in the diagram, the semantic similarity assessment on the statechart diagram is calculated. Lexical information is divided into two types: property and transition information. This information sharing can be observed in Fig. 3. Property information comprises entry activity, do an activity, exit activity, and state. Additionally, transition information comprises source state, target state, trigger event, and guard.

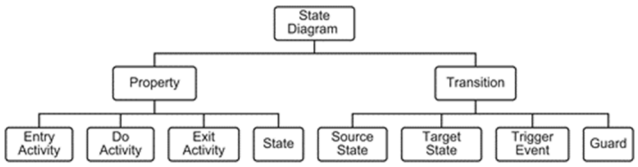

Fig. 3 Lexical information on the state chart diagram

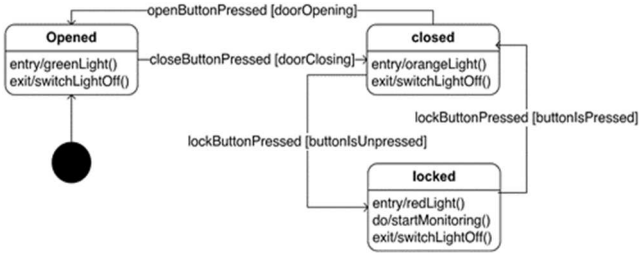

Fig. 4 Example of a state chart diagram

For instance, the lexical information obtained from the statechart diagram in Fig. 4 can be described as follows:

Property
Property-1
State: Opened
Entry Activity: greenLight
Exit Activity: switchLightOff
...
Property-3
State: Locked
Entry Activity: redLight
Do Activity: startMonitoring
Exit Activity: switchLightOff
Transition
Transition-1
Source State: opened
Target State: closed
Trigger Event: closeButtonPressed
Guard: doorClosing
...
Transition-4
Source State: locked
Target State: closed
Trigger Event: lockButtonPressed
Guard: buttonIsPressed.

As illustrated in Fig. 3, the semantic similarity between statechart diagrams d_1 and d_2 (*semStd*) is calculated using Equation 2.

$$semStd(d_1, d_2) = (1 - \rho_{sem}) \times propSim(d_1, d_2) + \rho_{sem} \times tranSim(d_1, d_2) \quad (2)$$

The semantic similarity between statechart diagrams d_1 and d_2 comprises property similarity (*propSim*) and transition similarity (*tranSim*). Each similarity has a different level of importance based on an expert's viewpoint. In Equation 2, the level of importance of the similarity is distinguished from ρ_{sem}. In addition, the value of ρ_{sem} is from zero to one.

$$propSim(d_1, d_2) = \frac{2\times\left(\sum_{k=1}^{Min(|ST_1|,|ST_2|)} changePivot\left(Max(\sum_{i=1}^{|ST_1|}\sum_{j=1}^{|ST_2|} stSim(st_i, st_j))\right)\right)}{|ST_1|+|ST_2|} \quad (3)$$

The property similarity between statechart diagrams d_1 and d_2 is calculated by employing Equation 3. All states in d_1 are semantically compared with those of d_2. ST_1 and ST_2 comprise all states in d_1 and d_2, respectively. Equation 2 describes the greedy algorithm for obtaining the optimal state similarity value of d_1 and d_2. In the greedy algorithm, the *changePivot* is utilized to eliminate the similarity value of state pairs. Algorithm 1 demonstrates the flow of *changePivot*

Algorithm 1. *changePivot*

Input: two dimension matrix and pivot/coordinate maximum value (x,y)

Output: changed matrix

1. Select pivot
2. M(x, :) = 0
3. M(:, y) = 0.

Line 1 obtains the coordinates of the maximum similarity value from the matrix as *x* and *y*. Line 2 turns the value in row *x* into zero. Furthermore, line 3 turns the value in column *y* into zero. As shown in Fig. 3, the property of a state comprises several components. Therefore, a detailed calculation between two states (*stSim*) is required. The *stSim* is given in Equation 4.

$$stSim(st_1, st_2) = \frac{CosSim(en_1,en_2)+CosSim(do_1,do_2)+CosSim(ex_1,ex_2)+CosSim(s_1,s_2)}{Max(|en_1|,|en_2|)+Max(|do_1|,|do_2|)+Max(|ex_1|,|ex_2|)+Max(|s_1|,|s_2|)} \quad (4)$$

Depending on the information established in states st_1 and st_2, state similarity (*stSim*) between them is calculated. Besides, the information is entry activity (*en*), do activity (*do*), exit activity (*ex*), and state (*s*). The semantic similarity of each information is calculated by applying NLP, which was explained at the beginning of Section II.B. Thus, the total similarities of all the information are divided by the number of components that are only available in the two states. For example, if both states only have a do entry and state, then the divisor value is two.

$$tranSim(d_1, d_2) = \frac{2\times\left(\sum_{k=1}^{Min(|T_1|,|T_2|)} changePivot\left(Max(\sum_{i=1}^{|T_1|}\sum_{j=1}^{|T_2|} tSim(t_i, t_j))\right)\right)}{|T_1|+|T_2|} \quad (5)$$

By applying Equation 5, the transition similarity between two statechart diagrams d_1 and d_2 is calculated. All transitions in d_1 are semantically compared with those in d_2. Further, T_1 and T_2 comprise all the transitions in d_1 and d_2, respectively.

Equation 5 describes the greedy algorithm for finding the optimal transition similarity values of d_1 and d_2. As depicted in Algorithm 1, the *changePivot* is utilized to eliminate the similarity value of transition pairs in the greedy algorithm. Based on Fig. 3, the transition of a statechart diagram comprises several components. Therefore, a detailed calculation between two transitions (*tSim*) is required. The *tSim* is presented as Equation 6.

$$tSim(t_1, t_2) = \frac{CosSim(src_1,src_2)+CosSim(tgt_1,tgt_2)+CosSim(trg_1,trg_2)+CosSim(gd_1,gd_2)}{Max(|src_1|,|src_2|)+Max(|tgt_1|,|tgt_2|)+Max(|trg_1|,|trg_2|)+Max(|gd_1|,|gd_2|)} \quad (6)$$

Based on information observed in transitions t_1 and t_2, the transition similarity (*tSim*) between them is calculated. The information is a source state (*src*), target state (*tgt*), trigger (*trg*), and guard (*gd*). The semantic similarity of each information is calculated using the NLP, which was explained at the beginning of section II.B. Moreover, the total similarities of all the information are divided by the number of components that are only available in the two transitions. For example, if both

transitions only have source state, target state, and trigger, then the divisor value is three.

D. Structural Similarity Assessment

By representing the existence diagram into a graph, the structural similarity between two statechart diagrams is calculated. The proposed graph is a directed graph. Unlike the semantic similarity that uses the lexical of the statechart diagram component, structural similarity ignores the lexical in the statechart diagram. Therefore, the structural similarity only considers the structure and type of the statechart diagram component. The element of the proposed graph can be observed as presented in Table I. Each element in the graph has its name and tag. There are two types of elements: vertex and edge. There are thirteen elements used in the graphs from statechart diagrams. Graph vertices comprise *vs*, *vt*, *ven*, *vdo*, *vex*, *vtr*, and *vgr,* while graph edge comprises e_t, e_{en}, e_{do}, e_{ex}, e_{tr}, and e_{gr}. All vertices are connected using the edge.

TABLE I
GRAPH ELEMENTS OF THE STATECHART DIAGRAM

No	Element Type	Name	Tag
1	Vertex	State vertex	*vs*
2	Vertex	Transition vertex	*vt*
3	Vertex	Entry activity vertex	*ven*
4	Vertex	Do activity vertex	*vdo*
5	Vertex	Exit activity vertex	*vex*
6	Vertex	Trigger vertex	*vtr*
7	Vertex	Guard vertex	*vgr*
8	Edge	Transition edge	e_t
9	Edge	Entry activity edge	e_{en}
10	Edge	Do activity edge	e_{do}
11	Edge	Exit activity edge	e_{ex}
12	Edge	Trigger edge	e_{tr}
13	Edge	Guard edge	e_{gr}

Based on Table I, a statechart diagram can be translated into a directed graph. Fig. 5 depicts the translation results of a statechart diagram in Fig.4 into a proposed graph.

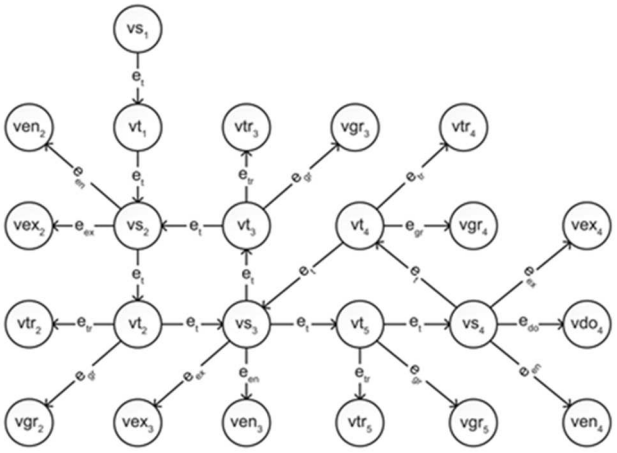

Fig. 5 Statechart diagram translated into a graph

The example statechart diagram comprises four states: start, opened, closed, and locked states; in the proposed graph, the states become vs_1, vs_2, vs_3, and vs_4, respectively. In addition, state vs_2 has an entry vertex ven_2 and an exit vertex vex_2. State vs_3 has an entry vertex ven_3 and an exit vertex vex_3. State vs_4 has an entry vertex ven_4, a do activity vertex vdo_4, and an exit vertex vex_4. The diagram also has five vertex transitions: vt_1, vt_2, vt_3, vt_4, and vt_5. Transition vt_1 connects the start state to the opened state, while transition vt_2 connects opened state to closed state. It has a trigger event vertex vtr_2 and a guard vertex vgr_2. Transition vt_3 connects closed states to opened states. It has a trigger event vertex vtr_3 and a guard vertex vgr_3. Transition vt_4 connects the locked state to the closed state. It has a trigger event vertex vtr_4 and a guard vertex vgr_4. Transition vt_5 connects closed state to locked state. It has a trigger event vertex vtr_5 and a guard vertex vgr_5.

In the structural assessment of the statechart diagram (*strucStd*), we differentiate between two similarities: intra similarity (*intraSim*) and inter similarity (*interSim*). Therefore, structural similarity can be obtained using Equation 7. Intra and inter similarities differ in importance based on the value of ρ_{str}. Thus, the value of ρ_{str} ranges from zero to one. It can be obtained based on an expert's perspective of assessment.

$$strucStd(d_1, d_2) = (1 - \rho_{str}) \times intraSim(d_1, d_2) + \rho_{str} \times interSim(d_1, d_2) \quad (7)$$

Based on the contents of the state and transition vertices, the intra similarity is calculated by considering the subgraphs of graphs that have been built. Fig. 6 (a) illustrates the subgraphs used to assess the similarity of each state. There are four subgraphs used to assess the state of similarity. Fig. 6 (b) demonstrates the subgraphs applied to assess the similarity of transitions. There are five subgraphs employed to assess the similarity of transitions.

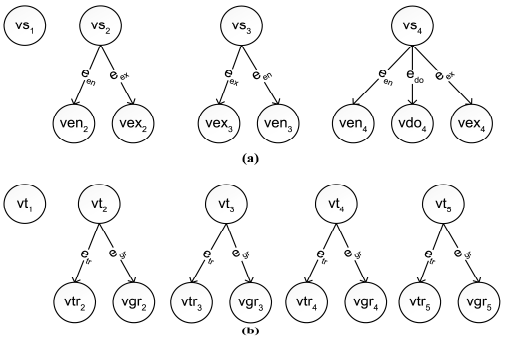

Fig. 6 Subgraphs to assess intraSim: (a) state and (b) transition

Intra similarity, which can be written as Equation 8, considers the subgraphs of states and transitions.

$$intraSim(d_1, d_2) = \frac{vsSim(d_1,d_2)+vtSim(d_1,d_2)}{2} \quad (8)$$

The similarity of state and transition vertices is separately calculated and then equally shared. The similarity of state vertex (*vsSim*) can be obtained by utilizing Equation 9. Furthermore, the similarity of transition vertex (*vtSim*) can be calculated by applying Equation 10. The similarity of state vertices is calculated based on a collection of state vertex subgraphs statechart diagrams d1 and d2. The collection of the subgraphs of state vertex diagrams d_1 and d_2 is SGS_1 and SGS_2,

respectively. All the subgraphs are compared to obtain an optimal similarity value. Moreover, the optimal value search utilizes the greedy algorithm. In Equation 8, *changePivot* plays an important role in eliminating non-optimal values in the greedy algorithm. The flow of *changePivot* can be observed in Algorithm 1. Thus, the similarity assessment between two subgraphs employs a greedy GED. To calculate the cost of the initial state vertex subgraph (sgs_i) to give the final state vertex subgraph (sgs_j), the concept of GED is applied.

$$vsSim(d_1, d_2) = \frac{2\times\left(\sum_{k=1}^{Min(|SGS_1|,|SGS_2|)} changePivot\left(Max(\sum_{i=1}^{|SGS_1|}\sum_{j=1}^{|SGS_2|} GED(sgs_i, sgs_j))\right)\right)}{|SGS_1|+|SGS_2|} \quad (9)$$

$$vtSim(d_1, d_2) = \frac{2\times\left(\sum_{k=1}^{Min(|SGT_1|,|SGT_2|)} changePivot\left(Max(\sum_{i=1}^{|SGT_1|}\sum_{j=1}^{|SGT_2|} GED(sgt_i, sgt_j))\right)\right)}{|SGS_1|+|SGS_2|} \quad (10)$$

Based on a collection of transition vertex subgraphs statechart diagrams d_1 and d_2, the similarity of transition vertex is calculated in Equation 9. The collection of subgraphs of transition vertex diagrams d_1 and d_2 is SGT_1 and SGT_2 in Equation 10, respectively. All the subgraphs are compared to obtain an optimal similarity value. Using *changePivot* in Algorithm 1, the optimal value search applies the greedy algorithm. Then, the similarity assessment between two subgraphs also utilizes a greedy GED. To transform the cost of the initial transition vertex subgraph (sgt_i) to give the final transition vertex subgraph (sgt_j), the concept of GED is employed.

Inter similarity differs from intra similarity. Inter similarity is only taken from the main vertex information. Besides, the main vertex comprises interconnected state and transition vertices. Therefore, as demonstrated in Fig. 5, we do not use all the information obtained from the graph. This is because it only takes the main vertex information, and the resulting subgraphs only have one subgraph. The main vertex comprises state and transition vertices. Fig. 7 depicts a form of the subgraph obtained from the initial graph. Take subgraphs that only contain state and transition vertices.

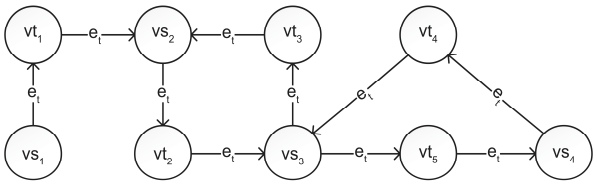

Fig. 7 Subgraphs for assessing interSim

As demonstrated in Fig. 7, *interSim* can be directly assessed using other diagrams. Equation 11 shows how to determine the similarity between subgraphs sg_1 and sg_2 from statechart diagrams d_1 and d_2, respectively. Using the greedy GED, this direct assessment compares the two subgraphs.

$$interSim(d_1, d_2) = GED(sg_1, sg_2) \quad (11)$$

III. Result and Discussion

A. Dataset

The General Description dataset used can be seen in Table II. The assembled statechart diagram is a collection of the students' answers to the questions by the teacher, and the students had directly answered the questions. The questions contained the name of an object and a description of the lifeline of the object. A description regarding the flow of the object was provided so that all the students could have the same mindset toward the object. However, the limitation of this study is that it uses a simple statechart diagram, and we are yet to consider the nested state case.

TABLE II
Dataset Information

Project	Number of Answers	Total State	Total Transition
Door	10	45	52
AC Remote	9	54	74
Counter	10	42	52

For the evaluation process, we set a gold standard for each student's answer. This gold standard is obtained from the average of experts' answers in assessing the similarity of students' answers based on the answer key. Experts are computer science lecturers in the area of software engineering. Furthermore, they must also have taught courses involving UML diagrams. Twenty-four experts contributed to this study.

Moreover, based on the validity of using Pearson [42]–[44] and the reliability of using Cronbach's alpha, the answers from these experts were statistically tested [45], [46]. Fig. 8 shows the correlation results of all expert answers to each assessment made. Of the twenty-nine assessments, eleven assessments had no correlation that met the critical value at the 0.05 level (2-tailed). The critical value is 0.349 based on Pearson's correlation table. Final statistical test results produce eighteen pairs of assessments on the students' answers and answer keys from the eighteen experts. The reliability value of the gold standard used is 0.947. The agreement's value between experts was also tested with inter-rater reliability [47]–[49]. In addition, the average inter-rater reliability value is 0.89.

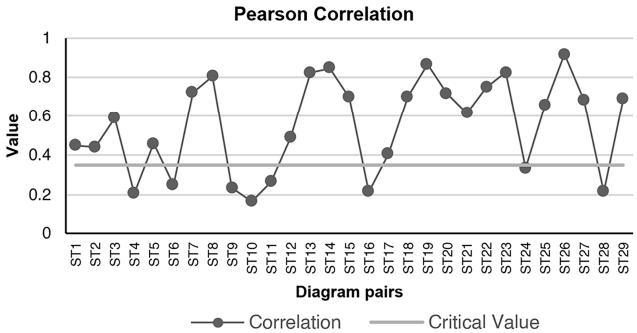

Fig. 8 Pearson correlation on expert assessment based on pairs of student answers and answer keys

B. Evaluation

We conducted experiments and compared the experimental results of the proposed assessment method with those given by experts. If both results significantly agree, then the proposed method is more reliable. Moreover, based on the given answer key, both the expert and our proposed method assess the answer from Table II The agreement value is calculated by employing Gwet's first-order agreement coefficient (AC1) [50]–[52]. To facilitate the assessment of the agreement between our proposed method and the teacher, the similarity value generated by both the proposed method and the teacher is converted to a scale of one to five. One, two, three, four, and five comprise zero to less than twenty, twenty to less than forty, forty to less than sixty, sixty to less than eighty, and eighty to one hundred, respectively. We conducted experiments by combining ρ, ρ_{sem}, and ρ_{str} values between 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, and 1. The term ρ is used to calculate Equation 1 to determine the level of importance between semantic and structural similarity components. The term ρ_{sem} is used to calculate Equation 2 to determine the level of importance between semantic similarity components. The term ρ_{str} is used to calculate Equation 7 to determine the level of importance between structural similarity components. In addition, the experiment was repeated 1331 times with different combinations of ρ, ρ_{sem}, and ρ_{str}. Fig. 9 shows the highest agreement results obtained from each ρ from zero to one. Therefore, the highest agreement value obtained is 0.921 with $\rho = 0.9$, $\rho_{sem} = 0.8$, and $\rho_{str} = 0.6$.

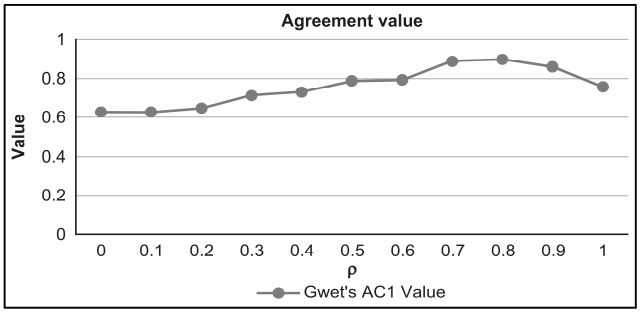

Fig. 9 The maximum agreement value of the proposed method and the expert

Based on the results of the conducted studies, the maximum value of the agreement of the proposed method with an expert is 0.897. According to Landis and Koch [53], this value is included in the almost perfect agreement. Therefore, our proposed method is as reliable as a teacher when assessing statechart diagrams. The maximum agreement value is ρ = 0.9. In assessing two-statechart diagrams, this implies that a teacher tends to look more at the structure of the statechart diagram than the lexical of the statechart diagram. The best ρ_{sem} values obtained are 0. In assessing the lexical of a statechart diagram, a teacher pays more attention to the transition than the state. The best ρ_{str} value found is 0.6; that is, the teacher tends to see the flow structure of the statechart diagram instead of the structure in the state and transition.

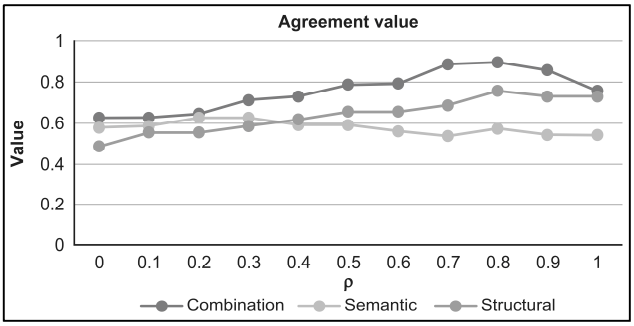

Fig. 10 Comparative experiment among semantic similarity, structural similarity, and combination

Furthermore, we also conducted comparative experiments between semantic use only, structural use, and a combination of both. Fig. 10 depicts the results of the comparison performed. Moreover, it can be observed that the highest agreement value is while combining semantic and structural similarities. The highest agreement values of the semantic and structural similarities are 0.59 and 0.767, respectively. Therefore, the use of one component of similarity alone is insufficient to produce a reliable automated assessment method. From Fig. 8, we can also observe that the structural similarity assessment gives a higher agreement value than that of the semantic similarity. Following the results of this study, the teachers tend to judge the structure of statechart diagrams instead of the lexical of statechart diagrams.

Thus, in assessing statechart diagrams, our proposed method is reliable as an expert. Our investigation shows that similarity measurements of statechart diagrams can also be utilized in clone detection and the reuse of software designs. Our proposed method can be applied to more objectives other than the statechart diagram assessment. Additionally, this study might be used for clone detection and software reuse. The weighting settings for the importance of each combination proposed herein are flexible. Weight values can be changed based on their needs.

IV. Conclusions

This study proposed an automatic assessment method that performed as reliable as a teacher in assessing the similarity of student answer and answer keys. Since semantic and structural similarities can assess the lexical and structural diagrams, they are appropriate components for evaluating the similarity of statechart diagrams. Semantic similarity assessment can compare the lexical based on letters and that based on the meaning of words. Structural similarity assessment can demonstrate the flow statechart diagram and the shape of each component in the statechart diagram. In the assessment process, a teacher sees the structure of the statechart diagram instead of the lexical statechart diagram. The concept of semantic and structural similarities may also be used in other UML diagrams. However, the statechart diagram used herein is simple. In the future research direction, it is necessary to develop an assessment of nested statechart diagrams. It is required to find the best combination of ρ, ρ_{sem}, and ρ_{str} to clone detection and software reuse by involving experts working in the industry.

ACKNOWLEDGMENT

This research was funded by the Ministry of Research and Technology/National Research and Innovation Agency of the Republic of Indonesia. This research is a collaboration amongst Institut Teknologi Sepuluh Nopember, Politeknik Negeri Banjarmasin, and Institut Teknologi dan Bisnis STIKOM Bali.

REFERENCES

[1] B. Rumpe, *Agile modeling with UML: Code generation, testing, refactoring*. 2017.

[2] H. Kaur and A. Sharma, "ANOVA Based Significance Testing of Non-functional Requirements in Software Engineering," *Int. J. Inf. Technol. Proj. Manag.*, vol. 10, no. 4, pp. 100–117, 2019, doi: 10.4018/IJITPM.2019100104.

[3] I. Salman, B. Turhan, and S. Vegas, *A controlled experiment on time pressure and confirmation bias in functional software testing*, vol. 24, no. 4. Empirical Software Engineering, 2019.

[4] P. Chung and B. Gaiman, "Use of State Diagrams to Engineer Communications Software," in *Conference of International Conference on Software Engineering*, 2017, pp. 215–221.

[5] A. Ramadhan and B. Susetyo, "Classification Modelling of Random Forest to Identify the Important Factors in Improving the Quality of Education," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 11, no. 2, pp. 501–507, 2021.

[6] Agusriandi, I. S. Sitanggang, and S. H. Wijaya, "Student Performance Based on Activity Log on Social Network and e-Learning," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 10, no. 6, pp. 2276–2281, 2020, doi: 10.18517/ijaseit.10.6.8753.

[7] J. R. Rico-Juan, A. J. Gallego, and J. Calvo-Zaragoza, "Automatic detection of inconsistencies between numerical scores and textual feedback in peer-assessment processes with machine learning," *Comput. Educ.*, vol. 140, no. February, p. 103609, 2019, doi: 10.1016/j.compedu.2019.103609.

[8] V. Vachharajani and J. Pareek, "Framework to approximate label matching for automatic assessment of use-case diagram," *Int. J. Distance Educ. Technol.*, vol. 17, no. 3, pp. 75–95, 2019, doi: 10.4018/IJDET.2019070105.

[9] R. Fauzan, D. Siahaan, S. Rochimah, and E. Triandini, "Use case diagram similarity measurement: A new approach," in *International Conference on Information and Communication Technology and Systems*, 2019, pp. 3–7.

[10] 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.

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

[12] S. Van Mierlo and H. Vangheluwe, "Introduction to Statecharts Modelling, Simulation, Testing, and Deployment," in *Proceedings of the 2018 Winter Simulation Conference*, 2018, pp. 306–320, doi: 10.1017/CBO9781107415324.004.

[13] C. Li, L. Huang, J. Ge, B. Luo, and V. Ng, "Automatically classifying user requests in crowdsourcing requirements engineering," *J. Syst. Softw.*, vol. 138, pp. 108–123, 2018, doi: 10.1016/j.jss.2017.12.028.

[14] S. Al Tahat and K. Ahmad, "Lexical Disambiguation (CKBD): A tool to identify and resolve semantic conflicts using Context Knowledge," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 9, no. 1, pp. 213–219, 2019, doi: 10.18517/ijaseit.9.1.6387.

[15] G. Mediamer, A. Adiwijaya, and S. Al Faraby, "Development of rule-based feature extraction in multi-label text classification," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 9, no. 4, pp. 1460–1465, 2019, doi: 10.18517/ijaseit.9.4.8894.

[16] P. Sunilkumar and A. P. Shaji, "A Survey on Semantic Similarity," in *2019 6th IEEE International Conference on Advances in Computing, Communication and Control, ICAC3 2019*, 2019, pp. 1–8.

[17] B. Sathiya and T. V. Geetha, "A review on semantic similarity measures for ontology," *J. Intell. Fuzzy Syst.*, vol. 36, no. 4, pp. 3045–3059, 2019, doi: 10.3233/JIFS-18120.

[18] Y. Y. Lee, H. Ke, T. Y. Yen, H. H. Huang, and H. H. Chen, "Combining and learning word embedding with WordNet for semantic relatedness and similarity measurement," *J. Assoc. Inf. Sci. Technol.*, vol. 71, no. 6, pp. 657–670, 2020, doi: 10.1002/asi.24289.

[19] X. Zhang, S. Sun, and K. Zhang, "A new hybrid improved method for measuring concept semantic similarity in wordnet," *Int. Arab J. Inf. Technol.*, vol. 17, no. 4, pp. 433–439, 2020, doi: 10.34028/iajit/17/4/1.

[20] D. D. Prasetya, A. P. Wibawa, and T. Hirashima, "The performance of text similarity algorithms," *Int. J. Adv. Intell. Informatics*, vol. 4, no. 1, pp. 63–69, 2018, doi: 10.26555/ijain.v4i1.152.

[21] S. Likavec, I. Lombardi, and F. Cena, "Sigmoid similarity - a new feature-based similarity measure," *Inf. Sci. (Ny).*, vol. 481, pp. 203–218, 2019, doi: 10.1016/j.ins.2018.12.018.

[22] 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.

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

[24] A. Adamu, W. Mohd, and N. Wan, "Matching and Retrieval of State Machine Diagrams from Software Repositories Using Cuckoo Search Algorithm," in *8th International Conference on Information Technology (ICIT)*, 2017, pp. 187–192.

[25] D. B. Blumenthal and J. Gamper, "On the exact computation of the graph edit distance," *Pattern Recognit. Lett.*, vol. 134, pp. 46–57, 2020, doi: 10.1016/j.patrec.2018.05.002.

[26] D. B. Blumenthal, N. Boria, J. Gamper, S. Bougleux, and L. Brun, "Comparing heuristics for graph edit distance computation," *VLDB J.*, vol. 29, no. 1, pp. 419–458, 2020, doi: 10.1007/s00778-019-00544-1.

[27] D. A. Rachkovskij, "Fast Similarity Search for Graphs by Edit Distance," *Cybern. Syst. Anal.*, vol. 55, no. 6, pp. 1039–1051, 2019, doi: 10.1007/s10559-019-00213-9.

[28] K. Riesen, M. Ferrer, and H. Bunke, "Approximate Graph Edit Distance in Quadratic Time," *IEEE/ACM Trans. Comput. Biol. Bioinforma.*, vol. 13, no. 9, 2015, doi: 10.1109/TCBB.2015.2478463.

[29] K. Riesen, M. Ferrer, R. Dornberger, and H. Bunke, "Greedy Graph Edit Distance," in *Machine Learning and Data Mining in Pattern Recognition*, 2015, vol. 9166, pp. 3–16, doi: 10.1007/978-3-319-21024-7.

[30] S. Ferrán, A. Beghelli, G. Huerta-Cánepa, and F. Jensen, "Correctness assessment of a crowdcoding project in a computer programming introductory course," *Comput. Appl. Eng. Educ.*, vol. 26, no. 1, pp. 162–170, 2018, doi: 10.1002/cae.21868.

[31] F. Restrepo-Calle, J. J. Ramírez Echeverry, and F. A. González, "Continuous assessment in a computer programming course supported by a software tool," *Comput. Appl. Eng. Educ.*, vol. 27, no. 1, pp. 80–89, 2019, doi: 10.1002/cae.22058.

[32] M. Aleyaasin, "Digital assessment of individual engineering assignments in mass courses," *Comput. Appl. Eng. Educ.*, vol. 26, no. 5, pp. 1888–1893, 2018, doi: 10.1002/cae.22014.

[33] D. Galan, R. Heradio, H. Vargas, I. Abad, and J. A. Cerrada, "Automated Assessment of Computer Programming Practices: The 8-Years UNED Experience," *IEEE Access*, vol. 7, pp. 130113–130119, 2019, doi: 10.1109/ACCESS.2019.2938391.

[34] S. Nurhayati and J. Purwanto, "Chatbot Based Applications on Smart Home Use Natural Language Processing," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 11, no. 2, pp. 581–588, 2021.

[35] R. Fauzan, "Use Case Diagram Similarity Measurement : A New," in *2019 12th International Conference on Information & Communication Technology and System (ICTS)*, 2019, pp. 3–7.

[36] E. Triandini, R. Fauzan, D. O. Siahaan, and S. Rochimah, "Sequence Diagram Similarity Measurement: A Different Approach," in *JCSSE 2019 - 16th International Joint Conference on Computer Science and Software Engineering: Knowledge Evolution Towards Singularity of Man-Machine Intelligence*, 2019, pp. 348–351.

[37] 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*, 2018, pp. 601–605.

[38] R. Fauzan, D. Siahaan, S. Rochimah, and E. Triandini, "Automated Class Diagram Assessment using Semantic and Structural Similarities," *Int. J. Intell. Eng. Syst.*, vol. 14, no. 2, 2021.

[39] R. Fauzan, D. Siahaan, S. Rochimah, and E. Triandini, "A Different Approach on Automated Use Case Diagram Semantic Assessment," *Int. J. Intell. Eng. Syst.*, vol. 14, no. 1, pp. 496–505, Feb. 2021, doi: 10.22266/ijies2021.0228.46.

[40] R. Fauzan, D. Siahaan, S. Rochimah, and E. Triandini, "A Novel Approach to Automated Behavioral Diagram Assessment using Label Similarity and Subgraph Edit Distance," *Comput. Sci.*, vol. 22, no. 2, 2021.

[41] N. A. Rakhmawati, A. A. Firmansyah, P. M. Effendi, R. Abdillah, and T. A. Cahyono, "Auto Halal detection products based on euclidian

distance and cosine similarity," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 8, no. 4–2, pp. 1706–1711, 2018, doi: 10.18517/ijaseit.8.4-2.7083.

[42] C. Jebarathinam, D. Home, and U. Sinha, "Pearson correlation coefficient as a measure for certifying and quantifying high-dimensional entanglement," *Phys. Rev. A*, vol. 101, no. 2, pp. 1–18, 2020, doi: 10.1103/PhysRevA.101.022112.

[43] D. Edelmann, T. F. Móri, and G. J. Székely, "On relationships between the Pearson and the distance correlation coefficients," *Stat. Probab. Lett.*, vol. 169, p. 108960, 2021, doi: 10.1016/j.spl.2020.108960.

[44] J. P. B. Mapetu, L. Kong, and Z. Chen, *A dynamic VM consolidation approach based on load balancing using Pearson correlation in cloud computing*, no. 0123456789. Springer US, 2020.

[45] S. Lisawadi, S. E. Ahmed, O. Reangsephet, and M. K. A. Shah, "Simultaneous estimation of Cronbach's alpha coefficients," *Commun. Stat. - Theory Methods*, vol. 48, no. 13, pp. 3236–3257, 2019, doi: 10.1080/03610926.2018.1473882.

[46] K. S. Taber, "The Use of Cronbach's Alpha When Developing and Reporting Research Instruments in Science Education," *Res. Sci. Educ.*, vol. 48, no. 6, pp. 1273–1296, 2018, doi: 10.1007/s11165-016-9602-2.

[47] F. Garcia-Loro, S. Martin, J. A. Ruipérez-Valiente, E. Sancristobal, and M. Castro, "Reviewing and analyzing peer review Inter-Rater Reliability in a MOOC platform," *Comput. Educ.*, vol. 154, no. September 2019, 2020, doi: 10.1016/j.compedu.2020.103894.

[48] J. Jirschitzka, A. Oeberst, R. Göllner, and U. Cress, "Inter-rater reliability and validity of peer reviews in an interdisciplinary field," *Scientometrics*, vol. 113, no. 2, pp. 1059–1092, 2017, doi: 10.1007/s11192-017-2516-6.

[49] K. N. Bromm, I. M. Lang, E. E. Twardzik, C. L. Antonakos, T. Dubowitz, and N. Colabianchi, "Virtual audits of the urban streetscape: Comparing the inter-rater reliability of GigaPan® to Google Street View," *Int. J. Health Geogr.*, vol. 19, no. 1, pp. 1–15, 2020, doi: 10.1186/s12942-020-00226-0.

[50] A. M. Jimenez and S. J. Zepeda, "A Comparison of Gwet's AC1 and Kappa When Calculating Inter-Rater Reliability Coefficients in a Teacher Evaluation Context," *J. Educ. Hum. Resour.*, p. e20190001, 2020, doi: https://doi.org/10.3138/jehr-2019-0001.

[51] T. Ohyama, "Statistical inference of Gwet's AC1 coefficient for multiple raters and binary outcomes," *Commun. Stat. - Theory Methods*, vol. 0, no. 0, pp. 1–9, 2020, doi: 10.1080/03610926.2019.1708397.

[52] A. Karthikayen and S. Selvakumar Raja, "Gwet kappa reliability factor-based selfish node detection technique for ensuring reliable data delivery in mobile adhoc networks," *J. Comput. Theor. Nanosci.*, vol. 16, no. 2, pp. 489–495, 2019, doi: 10.1166/jctn.2019.7756.

[53] J. R. R. Landis and G. G. Koch, "The Measurement of Observer Agreement for Categorical Data," *Biometrics*, vol. 33, no. 1, p. 159, 1977, doi: 10.2307/2529310