ISBN 978-1-5386-7082-8
IEEE Catalog Number CFP18G48-ART

ICITISEE 2018

The 3rd International Conference on Information Technology, Information System and Electrical Engineering

PROCEEDING

"KNOWLEDGE DISCOVERIES IN DATA FOR LEVERAGING INTELLIGENT SCIENCE AND TECHNOLOGY TO GLOBAL CHALLENGE"

November 13-14th, 2018
Yogyakarta, Indonesia

Organized by:

Supported by:

Techical co-sponsored by:

IEEE INDONESIA SECTION

2018 3rd International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE), Yogyakarta, Indonesia

Class Diagram Similarity Measurement: A Different Approach

1st Reza Fauzan, 2nd Daniel Siahaan, 3rd Siti Rochimah
Departement of Informatics, Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
reza.fauzan@poliban.ac.id, daniel@its.ac.id, siti@if.its.ac.id

4th Evi Triandini
Departement of Information System, STMIK STIKOM Bali
Bali, Indonesia
evi@stikom-bali.ac.id

***Abstract*— Unified Modeling Language (UML) is a standard modeling language for specifying, documenting, and building software. One of the problems of designing a model using UML is that it takes a relatively long time to create a model form the scratch. Reusing models can help accelerate the software development process. Previous researches related with measuring similarity of class diagrams were focused on textual and structural similarity between models. They structural similarity ignores the specific characteristics of relations resided in a class diagram and its components. Based on these problems, this study proposed a measurement similarity method of UML class diagrams based on their components and relations. The method improves the previous method by introducing various kind of relations in a class diagram as part of the parameters to calculate the similarity. The initial investigation of this paper shows that all parameters could determine the similarity of models.**

Keywords—class diagram, measurement similarity method, class relations

I. INTRODUCTION

Unified Modeling Language (UML) is a standard modeling language for specifying, documenting, and building a software [1]. UML can also be referred to as a standard language in modeling that is often used by software developers long ago [2], [3]. UML helps designers to model interactions between systems and users, interactions between objects, object behavior, and implementation and logical structure of the system [4].

UML development has several problems. One problem that is often found when making UML is that it takes a long time if it is required to make it from the beginning [5]. Then, reusing UML diagrams appears as a solution to the problem. Reusing UML diagrams can be done using software that was developed before, not from the beginning [6]. Reusing UML diagrams can help accelerate the software development process. In addition, reusing UML can reduce the costs and risks used [7].

Reusing UML diagrams requires a method of calculating similarities between artifacts in UML diagrams. Calculating the similarity between two artifacts UML diagrams is an important and challenging work in Software Engineering [8]. This is challenging because we have to look at UML diagrams with more than one point of view [9]. These many viewpoints make it difficult to calculate the similarities between artifacts in different UML diagrams. In addition, one of the disadvantages of software reuse is an attempt to find and adjust components that can be reused [7], [10].

The determination of similarity is an effort made in maximizing the reuse of UML diagrams. Previous research [11] has attempted to calculate the similarities between UML artifacts in class diagrams. Similarities in UML class diagrams are calculated from the structure of relationships between classes. However, this study does not pay attention to what type of relationship. Although the UML class diagram has different types of relationships, as long as it has a semantically similar class it will still be said to be similar. So that it can be said that the comparison is not equivalent. And, in further research [12], [13] conducted, information obtained from UML class diagrams must be further reproduced. Such information is like data types, methods, parameters, and so on. They have also developed a method of calculation in the next research by considering the type of relation [14]. However, comparing different types of relationships makes unfair comparisons.

Other research [9] do calculations on several UML diagrams. The results obtained are class information and inter-class relations are good indicators in calculating similarities between UML diagrams. However, the study cannot recognize the inconsistencies of words between UML artifact diagrams. Semantic approach is needed as a solution to these problems. The semantic approach is done using natural language processing. Some researches that use natural language processing can help program a more precise word recognition system [15]–[18].

Structural and semantic calculation methods have been carried out in previous research [19]. The study proves that the similarity of structure and semantics in UML diagrams is a good indicator in calculating the similarity of UML diagrams. This paper proposed a measurement similarity method of UML class diagrams based on their components and relations.

II. RESEARCH METHOD

This section describes the methods that are carried out in this paper. The calculation method is adapted from previous studies and then implemented in the UML class diagram. The stage is to prepare a UML class diagram and then calculate the similarity of the UML class diagram.

A. Diagram Preprocessing

Calculation of similarities between 2 UML class diagrams requires preprocessing. This is done to retrieve any information that can be taken from the diagram. The preprocessing diagram produces the metadata from each class diagram into a class diagram metamodel. In this case,

978-1-5386-7082-8/18/$31.00 ©2018 IEEE

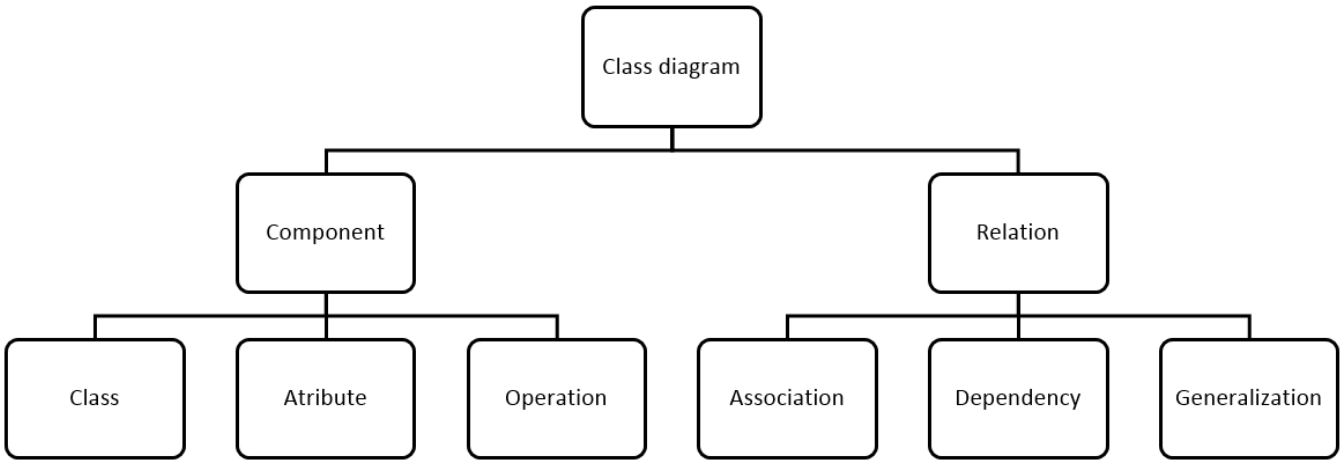

Fig. 1. Metadata of UML Class Diagram

we propose the metadata model for the class diagram as shown in Figure 1. The metamodel is built as a whole and it has several parts in it. The parts can be grouped into two things, namely component and relation. Component consist of classes, attributes, and operations. Classes have names and stereotypes. Attributes have stereotypes, names, and attribute types. And operations have stereotypes, names, types of operations, and parameters. Whereas relations consist of associations, dependencies, and generalizations. The association relation has the initial class, the name of the relation, multiplicity, ownership (aggregation / composition), and the destination class. Dependency has an initial class, relation name, and destination class. Generalization has an initial class and destination class.

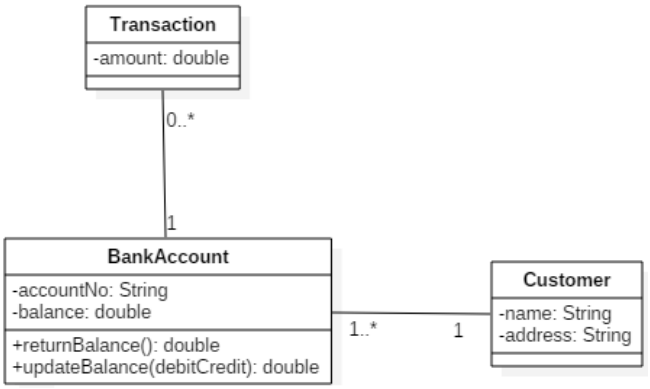

Fig. 2. UML Class Diagram Example 1

Metadata retrieval in UML class diagrams is done using a tool. The tool can convert UML class diagrams into XMI-formats. Based on Fig 2, some of the metadata retrieval results from XMI-format are as follows.

Component

→ *class : BankAccount*

→ *attribute :*

→ *attribute_1 : (private, accountNo, String)*

→ *attribute_2 : (private, balance, double)*

→ *operation :*

→ *operation_1 : (public, returnBalance, double)*

→ *operation_2 : (public, updateBalance, double)*

→ *parameter_1 : (debitCredit)*

Relation

→ *association :*

→ *association_1 : (BankAccount, 1, 1, Customer)*

→ *association_2 : (BankAccount, 0, *, Transaction)*

The BankAccount class has two attributes, namely attribute_1 and atribute_2. Attribute_1 has a private stereotype, accountNo name, and String data type. Attribute_2 has a private stereotype, balance name and double data type. The BankAccount class also has two operations, namely operation_1, and operation_2. Operation_1 has a public stereotype, the returnBalance name, and a double data type. Operation_2 has a public stereotype, name updateBalance, data type double, and parameter parameter_1. Parameter_1 contains a debitCard.

B. Calculation Method

The calculation method used is by adapting and perfecting calculation parameters from the previous method [11], [14], [19]. This paper takes a structural and semantic approach. The structural approach is carried out from the UML class diagram metadata structure. The semantic approach is made from the similarity between words at each end node of the UML class metadata using natural language processing.

As previously explained, the similarities between 2 UML class diagrams can be calculated based on the metadata they have. The metadata has two nodes on the second level, namely component (*comSim*) and relation (*relSim*). Equation 1 describes how to calculate similarity between two class diagrams-, i.e. d_1 and d_2.

$$classSim(d_1,d_2) = w_{com} \times comSim(d_1,d_2) + w_{rel} \times relSim(d_1,d_2) \quad (1)$$

where w_{com} is the weight of the component's resemblance and w_{rel} is the weight of the similarity of the relation. Next, how to calculate from *comSim* is to calculate the similarity of class objects between diagrams. This calculation is shown in Equation 2.

$$comSim(d_1,d_2)=\frac{Max(\sum_{i,j=1}^{Mac(|O_1|,|O_2|} oSim(o_i,o_j))}{|O_1|+|O_2|} \quad (2)$$

where O_1 and O_2 are collections of object classes in the diagrams d_1 and d_2. Then to calculate the semantic similarity of two object classes (*oSim (o_1, o_2)*) using Equation 3.

$$oSim(o_1,o_2)=w_c\times cSim(o_1,o_2)+w_a\times aSim(o_1,o_2)+ w_{op}\times opSim(o_1,o_2) \quad (3)$$

where w_c, w_a, and w_{op} are arbitrary weight assign to class similarity (*cSim*), atribute similarity (*aSim*), and operation similarity (*opSim*), respectively. The similarity of classes measures the lexical similarity of the class names of two class objects. The calculation process is done with cosine similarity as shown in Equation 4.

$$cSim(o_1,o_2)=\frac{Max(\sum_{i,j=1}^{Max(|CP_1|,|CP_2|)} Co\sin eSim(cp_i,cp_j))}{|CP_1|+|CP_2|} \quad (4)$$

where CP_1 and CP_2 are lexical forms of class names between two objects (o_1, o_2). Then, Equation 3 raises the similarity of attributes between two object classes (*aSim (o_1, o_2)*). How to calculate the similarity of attributes in Equation 5.

$$aSim(o_1,o_2)=\frac{Max(\sum_{i,j=1}^{Max(|A_1|,|A_2|)} daSim(a_i,a_j))}{|A_1|+|A_2|} \quad (5)$$

where A_1 and A_2 are a collection of attributes in two object classes. As previously known, attributes consist of stereotypes, attribute names and data types. So we need advanced calculations that contain the three things that are attributes (*daSim(a_1,a_2)*). This is a need to do so that the results obtained are more accurate. The calculation is in Equation 6.

$$daSim(a_1,a_2)=w_{st}\times WuP(st_1,st_2)+w_{na}\times WuP(na_1,na_2)+w_{ty}\times WuP(ty_1,ty_2) \quad (6)$$

where w_{st}, w_{na}, dan w_{ty} are arbitrary weight assign to stereotype similarity, attribute's name similarity, and attribute's data type similarity, respectively. The similarity between two attributes can be calculated from the lexical of each component in the attribute. Then, Equation 3 raises the similarity of attributes between two operations (*opSim (op_1, op_2)*). How to calculate the similarity of operations in Equation 7.

$$opSim(o_1,o_2)=\frac{Max(\sum_{i,j=1}^{Max(|OP_1|,|OP_2|)} dopSim(op_i,op_j))}{|OP_1|+|OP_2|} \quad (7)$$

where OP_1 and OP_2 are a collection of operations in two object classes. As previously known, the operation consists of stereotypes, the operation's names, data types, and parameter. So we need advanced calculations that contain the four things that are operations (*dopSim(op_1,op_2)*). This is a need to do so that the results obtained are more accurate. The calculation is in Equation 8.

$$dopSim(op_1,op_2)=w_{stO}\times WuP(stO_1,stO_2)+ w_{naO}\times WuP(naO_1,naO_2)+w_{tyO}\times WuP(tyO_1,tyO_2)+ w_{parO}\times parSim(op_1,op_2) \quad (8)$$

where w_{stO}, w_{naO}, w_{tyO} and w_{parO} are arbitrary weight assign to stereotype similarity, operation's name similarity, operation's data type similarity, and operation's parameter similarity, respectively. The similarity between the two operations can be calculated from the lexical of each component in the operation. But, the parameter can not be calculated as simple as that. We need to use cosine similarity to calculate as shown in Equation 9.

$$parSim(op_1,op_2)=\frac{Max(\sum_{i,j=1}^{Max(|PAR_1|,|PAR_2|)} Co\sin eSim(par_i,par_j))}{|PAR_1|+|PAR_2|} \quad (9)$$

where PAR_1 and PAR_2 are lexical forms of the parameter between two operations (op_1, op_2). Then, Equation 1 raises the relation similarity between two class diagrams (*relSim (d_1, d_2)*). How to calculate the relation similarity in Equation 10.

$$relSim(d_1,d_2)=w_{ra}\times raSim(d_1,d_2)+ w_{rd}\times rdSim(d_1,d_2)+w_{rg}\times rgSim(d_1,d_2) \quad (10)$$

where w_{ra}, w_{rd}, and w_{rg} are arbitrary weight assign to association relations similarity (*raSim*), dependency similarity (*rdSim*), and generalization relation similarity (*rgSim*), respectively. Then, Equation 10 raises the similarity of association between two UML class diagram (*raSim (d_1, d_2)*). How to calculate the similarity of attributes in Equation 11.

$$raSim(d_1,d_2)=\frac{Max(\sum_{i,j=1}^{Max(|RA_1|,|RA_2|)} draSim(ra_i,ra_j)}{|RA_1|+|RA_2|} \quad (11)$$

where RA_1 and RA_2 are a collection of association relations in two object classes. As previously known, association consist of source class, relation's name, lower level multiplicity, upper level multiplicity, aggregation/composition, and target class. So we need advanced calculations that calculate relation similarity (*draSim(a_1,a_2)*). This is need to do so that the results obtained are more accurate. The calculation is in Equation 12.

$$draSim(ra_1,ra_2)=w_{ra1}\times WuP(src_1,src_2)+ w_{ra2}\times WuP(nm_1,nm_2)+w_{ra3}\times WuP(up_1,up_2)+ w_{ra4}\times WuP(lo_1,lo_2)+w_{ra5}\times WuP(ow_1,ow_2)+ w_{ra6}\times WuP(tgt_1,tgt_2) \quad (12)$$

where w_{ra1}, w_{ra2}, w_{ra3}, w_{ra4}, w_{ra5}, and w_{ra6} are arbitrary weight assign to the similarity of source class, the similarity of relation's name, the similarity of lower level multiplicity, the similarity of upper level multiplicity, the similarity of aggregation/composition, and the similarity of target class, respectively. Then, Equation 10 raises the similarity of dependency relation's similarity (*rdSim* (d_1, d_2)). How to calculate the similarity as shown in Equation 13.

$$rdSim(\mathrm{d}_1, d_2) = \frac{Max(\sum_{i,j=1}^{Max(|RD_1|,|RD_2|)} drdSim(rd_i, rd_j)}{|RD_1| + |RD_2|} \quad (13)$$

where RD_1 and RD_2 are a collection of dependency relations in two object classes. As previously known, dependency consists of source class and target class. So we need advanced calculations that calculate relation similarity (*drdSim*(rd_1,rd_2)). This is a need to do so that the results obtained are more accurate. The calculation is in Equation 14.

$$drdSim(rd_1, rd_2) = w_{rd1} \times WuP(src_1, src_2) + w_{rd2} \times WuP(tgt_1, tgt_2) \quad (14)$$

where w_{rd1} and w_{rd2} are arbitrary weight assign to the similarity of source class and similarity of target class, respectively. The similarity of dependency relation between two class diagrams can be calculated from the lexical of each component in the relation. Then, Equation 10 raises the similarity of generalization relation's similarity (*rgSim* (d_1, d_2)). How to calculate the similarity as shown in Equation 15.

$$rgSim(\mathrm{d}_1, d_2) = \frac{Max(\sum_{i,j=1}^{Max(|RG_1|,|RG_2|)} drgSim(rg_i, rg_j)}{|RG_1| + |RG_2|} \quad (15)$$

where RG_1 and RG_2 are a collection of generalization relations in two object classes. As previously known, generalization consists of source class and target class. So we need advanced calculations that calculate relation similarity (*drgSim*(rg_1,rg_2)). This is a need to do so that the results obtained are more accurate. The calculation is in Equation 16.

$$drgSim(rg_1, rg_2) = w_{rg1} \times WuP(src_1, src_2) + w_{rg2} \times WuP(tgt_1, tgt_2) \quad (16)$$

where w_{rg1} and w_{rg2} are arbitrary weight assign to the similarity of source class and similarity of target class, respectively. The similarity of generalization relation between two class diagrams can be calculated from the lexical of each component in the relation.

III. Empirical Result and Analysis

The main purpose of this paper is to show what parameters are needed and how to calculate the similarity between two UML class diagrams. For example, we show the results of the calculation between the diagrams in Figure 2 (CD_1) and the diagram in Figure 3 (CD_2). Both are UML class diagrams that explain bank transactions. However, the content between classes is largely different. The relationship between the two UML diagrams of this class is also very different. From this example, we look at the similarity values of both.

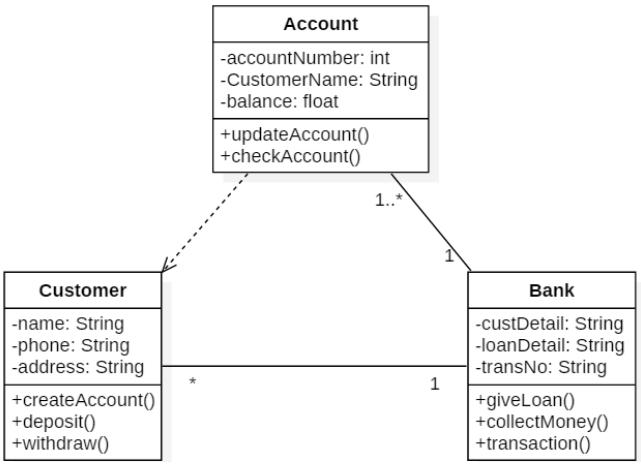

Fig. 3. UML Class Diagram Example 2

Table I shows the results of the calculation of similarities in the UML class diagram's component between CD_1 and CD_2. Calculation using Equation 2. From Table I, we get the highest similarity value in the similarity between o1_1 and o2_3. Then, the similarity between o1_2 and o2_2 becomes the next highest. Next, the similarity between o1_3 and o2_1 becomes the last value.

TABLE I. Component Similarity Between CD_1 dan CD_2

comSim	o2_1	o2_2	o2_3
o1_1	0.458525359	0.437619128	0.557503702
o1_2	0.383783962	0.555750512	0.38828224
o1_3	0.315876795	0.306556741	0.372676284

Table II shows the results of calculating the similarity of the association of UML class diagram between CD_1 and CD_2. Calculation using Equation 11. From Table II, we get the highest similarity value in the similarity between a1_1 and a2_4. Then, the similarity between a1_2 and a2_3 becomes the next highest. Then, the similarity between a1_3 and a2_2 becomes the next highest. Next, the similarity between a1_4 and a2_1 becomes the last value.

TABLE II. Association Similarity Between CD_1 dan CD_2

raSim	a2_1	a2_2	a2_3	a2_4
a1_1	0.18564790	0.28270436	0.31512530	0.38303717
a1_2	0.33270436	0.16064790	0.40803717	0.34012530
a1_3	0.21806438	0.47758478	0.28012834	0.41545365
a1_4	0.52758478	0.16806438	0.41545365	0.28012834

In the case of similarities between CD_1 and CD_2, the calculation of the similarity of relations is only calculated on the association relation. CD_1 does not have generalizations

or dependencies. CD_1 only has association relations. This paper proposes a fair comparison of relations. We will not compare different types of relations.

Having found the results of similar structures and relations, we calculate the weight that has been determined in accordance with Equation 1. The weight of w_{struc} and w_{rel} are set experimentally to 0.6 and 0.4. Equation 1 calculation results give a value of 0.393600838. This result is similar to a direct observation which shows both UML class diagrams are similar to about 30%.

Based on the results, this study shows all parameters that might affect the similarity of UML class diagrams. Complete appearance of all metadata can be done. Then a fair comparison of the relations in the UML class diagram can be done. These two things can be used in improvement from previous research [11], [14]. Then structural and semantic similarity measurement are good parameters in calculating UML diagrams [19].

IV. Conclusion

This paper introduces the method to measure similarity between two models designed as class diagrams. The algorithm uses word similarity methods (WuP and Levensthein Distance) to measure the word similarity between elements of different class, and Greedy Algorithm to find the local optima of similarity values between the two classes and diagrams. The proposed method consists of two parts, namely the semantic similarity of the components and the structural similarity, which considers various class relations, of two models of class diagram The initial investigation of this paper show all parameters could determine the similarity of the two models. Every detail information of UML class diagram can determine UML class diagram similarity. And comparing two relationships by fair comparison could be a good way to enhance the measurement method.

Further research should be carried out to determine using larger dataset. This research should determine a set of weights that can produce the highest measurement accuracy. Thus, it is necessary to look for alternative algorithm that is more accurate than the greedy approach to find the best pairs of component similarity.

Acknowledgment

The result is in cooperation between Institut Teknologi Sepuluh Nopember and STIKOM Bali.

References

[1] M. J. Chonoles, "What is UML?," in *OCUP Certification Guide: UML 2.5 Foundational Exam*, 2018, pp. 17–41.
[2] D. O. Siahaan, *Rekayasa Perangkat Lunak*. Surabaya: Penerbit Andi, 2012.
[3] D. O. Siahaan and F. Irhamni, "Advanced methodology for requirements engineering technique solution (AMRETS)," *Int. J. Adv. Comput. Technol.*, vol. 4, no. 5, pp. 75–80, 2012.
[4] M. Chechik, S. Nejati, and M. Sabetzadeh, "A relationship-based approach to model integration," *Innov. Syst. Softw. Eng.*, vol. 8, no. 1, pp. 3–18, 2012.
[5] W. N. Robinson and H. G. Woo, "Finding reusable UML sequence diagrams automatically," *IEEE Softw.*, vol. 21, no. 5, pp. 60–67, 2004.
[6] W. B. Frakes and K. Kang, "Software reuse research: status and future," *IEEE Trans. Softw. Eng.*, vol. 31, no. 7, pp. 529–536, 2005.
[7] I. Sommerville, *Software Engineering*. 2010.
[8] D. S. Kolovos, D. Di Ruscio, A. Pierantonio, and R. F. Paige, "Different Models for Model Matching: An analysis of approaches to support model differencing," *2nd Work. Comp. Versioning Softw. Model. (CVSM'09), ACM/IEEE Int. Conf. Softw. Eng.*, pp. 1–6, 2009.
[9] A. Adamu and W. M. N. W. Zainon, "Multiview Similarity Assessment Technique of UML Diagrams," *Procedia Comput. Sci.*, vol. 124, pp. 311–318, 2017.
[10] T. C. Lethbridge and R. Laganiere, "Object-Oriented Software Engineering : Practical Software Development Using Uml and Java," *McGraw-Hill Publ. Co.*, p. 561, 2004.
[11] M. A.-R. M. Al-Khiaty and M. Ahmed, "Similarity Assessment of UML Class Diagrams using a Greedy Algorithm," *Softw. Eng. Serv. ...*, pp. 19–23, 2014.
[12] M. Al-Khiaty and M. Ahmed, "Similarity assessment of UML class diagrams using simulated annealing," *Softw. Eng. Serv. ...*, 2014.
[13] A. Sellami, H. Hakim, A. Abran, and H. Ben-Abdallah, "A measurement method for sizing the structure of UML sequence diagrams," *Inf. Softw. Technol.*, vol. 59, pp. 222–232, 2015.
[14] M. A. R. Al-Khiaty and M. Ahmed, "Matching UML class diagrams using a Hybridized Greedy-Genetic algorithm," *Proc. 12th Int. Sci. Tech. Conf. Comput. Sci. Inf. Technol. CSIT 2017*, vol. 1, pp. 161–166, 2017.
[15] X. Yue, G. Di, Y. Yu, W. Wang, and H. Shi, "Analysis of the combination of natural language processing and search engine technology," *Procedia Eng.*, vol. 29, pp. 1636–1639, 2012.
[16] H. Wang, W. Zhang, Q. Zeng, Z. Li, K. Feng, and L. Liu, "Extracting important information from Chinese Operation Notes with natural language processing methods," *J. Biomed. Inform.*, vol. 48, pp. 130–136, 2014.
[17] M. Sevenster, J. Bozeman, A. Cowhy, and W. Trost, "A natural language processing pipeline for pairing measurements uniquely across free-text CT reports," *J. Biomed. Inform.*, vol. 53, pp. 36–48, 2015.
[18] V. Anikushina, V. Taratukhin, and C. von Stutterheim, "Natural Language Oral Communication in Humans Under Stress. Linguistic Cognitive Coping Strategies for Enrichment of Artificial Intelligence," *Procedia Comput. Sci.*, vol. 123, pp. 24–28, 2018.
[19] D. O. Siahaan, Y. Desnelita, Gustientiedina, and Sunarti, "Structural and Semantic Similarity Measurement of UML Sequence Diagrams," in *International Conference on Information & Communication Technology and System (ICTS)*, 2017, pp. 227–234.