Journal of Information & Optimization Sciences
ISSN 0252-2667 (Print), ISSN 2169-0103 (Online)
Vol. 44 (2023), No. 6, pp. 1251–1262
DOI : 10.47974/JIOS-1497

A methodology for model clone detection using statistics of design metrics

G. Shobha *
Rekha Agarwal †
Amity Institute of Information Technology
Amity University
Noida
Uttar Pradesh
India

Vineet Kansal §
Institute of Engineering and Technology
Dr APJ Abdul Kalam Technical University
Lucknow
Uttar Pradesh
India

Sarvesh Tanwar ‡
Amity Institute of Information Technology
Amity University
Noida
Uttar Pradesh
India

Abstract

Model clone detection has predominantly gained momentum in the field of software development process and model driven engineering. Reuse and mutations of the existing models will increase the complexity of managing the software's organisation's internal repositories. Detecting semantic and structural similarities among the models finds various uses like fault prediction, estimation of maintenance and refactoring. Literature in the domain witnesses very few works focussing on model clone detection. This work proposes a model clone detection technique by leveraging the statistical and lexical properties of the UML

* *E-mail:* **shobhaumesh04@gmail.com (Corresponding Author)**
† *E-mail:* **ragarwal@amity.edu**
§ *E-mail:* **vineetkansal@ietlucknow.ac.in**
‡ *E-mail:* **stanwar@amity.edu**

© Taru Publications

diagrams. The primary contribution of the work is the construction of Similarity Measure (SM) by analysing the design metrics from different perspectives namelyi)measuring the statistical variability between the models and ii) estimating the lexical similarity among design metrics. The results of the model indicates that the proposed method was able to detect the semantically similar model clones of the banking use case. Also, the method is very robust and computationally inexpensive, so that it could find its applicability in all fields where model driven engineering is used.

Subject Classification: *Primary 93A30, Secondary 49K15.*

Keywords: *Model clones, Model driven engineering, Similarity measure, Semantic clone, SD metrics, UML diagram, Jaro-Winkler distance.*

1. Introduction

Employing models is perceived to be one of the industry best practices especially during the early stages of software development. These models are found helpful in managing massive projects, integrating multiple schemas in large databases, software process management and even in optimization of project development. The regime of model-driven engineering practices continues even after the emergence new genres of software development process [1]. Many open source model repositories like MDEForge, Lindholmen and industry owned model repositories are available for performing model specific analysis of software components [2, 3]. In addition to this, the industry witnesses increase in the intricacy of software models. Mining of model-driven artifacts is widely gaining popularity as these approaches have become more prominent in the industry to detect software clones. A clone is defined as a part of software development process or even artifact that has more than one occurrence [4]. Detection of these clones are not only confined to codes but also to other artifacts. Tracking cloned models helps to enforce standard and quality of the repositories; helps in software management, refactoring, maintenance and even in development.

A clone to a model element is not exactly same one getting replicated but matching the parent model with more precision .Model clone detection involves isolating similar or identical artifacts, that will be coined as clones. Similarity among the clones is determined by different tools in various genres. All the tools function based on threshold criteria that must be satisfied so that the artifacts under comparison will be considered as a clone.

The model clones are revolving round the industry inherently as the industry fosters reusability, resource constrained working nature, sinister

practices etc [5]. Investigation of clones in large systems resulted in the existence of common patterns of cloning like [6]:

- Forking: Widely used in bootstrap development of more or less similar solutions, with an aim that evolution of model will occur independently, in shorter time span.
- Tinplating: Here the models are copied without any appropriate abstraction mechanisms [8].
- Customization: This method is adopted when existing code does not meet the dynamic requirements.

There are variety of reasons that contribute to the evolution of model clones. The creation of model clones is more specifically the higher level of abstractions and it can be observed that modeling is pivotal industry practice in almost all phases of software development. Augmenting to this, the software models reach a reasonable size, which increases the complexity as well as escalates the chances of duplication [9]. The dearth of studies ascertains that large numbers of techniques are witnessed in the literature for clone detection in UML models.

There model clones are classified as type 1, 2,3 and 4 as given below [10]

- Type 1: They are identical or exact model clones, which are structurally similar. However, they vary in their layout, formatting and positioning.
- Type 2: They are renamed model clones, which are identical but with different names, labels, parts values, types and attributes.
- Type 3: The clone in Type 3 category is termed as near-miss model clones. These clones has model fragments that is replicated from another model.
- Type 4: They include semantically equivalent clones which are the toughest to be detected as they will appear structurally different but is behaviourally and semantically equivalent.

Present day clone detectors use variety of methodologies to detect the all the four types of clones that are developed using modelling languages like UML or state flow models. However, detection of Type 3 and Type 4 clones is challenging as they are structurally different without much functional variations. The present work focusses on detecting model clone

from the UML structure using the software design metrics tool. This tool is very useful in analysing the structural and behavioral aspects of UML models. These metrics are used in the process of software quality assurance by measuring metric like complexity, design size, and coupling degree to detect the similarity between models [11]. Detecting model clones cannot be implied directly to all types of clones as they are not linear text. As there is no standardised way of representing the models, except for XML, UML or any other equivalent tools comparison between them is very difficult. This paves way for evolution of large set of similar model clones with different serializations. However, most of the works of model clone detection relies on UML or XML trees as they have strong resemblance to parse trees. This work also uses the UML diagrams and derives its equivalent XMI to analyse the equivalence or similarity using a hybrid methodology of augmenting statistical analysis. The primary contributions of the present work:

- Generate SD metrics of the UML using appropriate tool.
- Develop a hybrid statistical methodology that assess the design metrics to determine the similarity between the models which are validated using the Jaro-Winkler distance.
- The organisation of the article is as follows: Section 2 briefs some of the important milestones in the field of study. Section 3 describes the proposed hybrid methodology of model clone detection. The empirical results are discussed in section 4 and section 5 concludes the work.

2. Literature Review

It is always harmful to have cloned models as they increase the maintenance costs with high inconsistencies. Florian Deissenboeck et al. presented an extensive study that covers heterogeneous approaches to identify clones in graphical models. The work considers industrial use cases with better scalability and consistency .An index-based clone detection is discussed by. which uses a unique mapping between the normalized statements and their occurrences MD5 hashing is employed for index construction with specula mention to clone retrieval. A comprehensive study of the model clone detection is given by Dhavleesh Rattan et al. [12]. This study focused on both Type 3 and Type 4 clone which are hard to detect.

Martin White et al. proposed a deep learning-based model clone detection technique by linking patterns that are mined in the lexical and syntactic context. The analysis of results across 398 files and 480 method-level indicates the method was able to detect true positives with 93% accuracy [13]. A novel clone detection tool named ModelCD, which was developed using Matlab and Simulink models, were capable of accurately detect the clones in software models . These results of the work indicated that this tool exhibited high degree of scalability, completeness, and accuracy, and scalability.

Matthew Stephan et al. extended the NiCad's code clone detection by creating and identifying the sub units from the grammar of clones. Comparison of these structural sub-units and normalising them helps in the process of detecting the similarities between the clones. An XML tree based clone detection is proposed by Dhavleesh Rattan et al, where the UML models are encoded as XML files [14]. The subtrees in the XML are then compared to find the similarities.

Thus, the brief literature survey indicates that most of the works focus on detecting models using lexical approaches. Not many works are available using statistical approaches. But these computing technologies are less explored in model clone detection. Also, development and deployment of hybrid models are also one of the effective ways on model clone detection.

[15]. [16-17] talks about the cloud computing mechanisms like hadoop and artificial intelligence to be used for providing the better utility of the work .Also in the next generation networks , when we are in the era of developing the protocols for specific applications [18-21] discuss how the machine leaning and IoT are used for making the product more efficient. So in future clone detection may use the AI and cloud computing for efficient performance.

3. Proposed Work

The proposed hybrid methodology for detecting the model clones uses UML diagrams for specific use cases to assess the similarity between the designed models. The work proceeds in three phases. First Creating UML class diagrams for a specific use case and then converts it to its equivalent XML Metadata Interchange (XMI) file. In second phase Use the Software Design (SD) metrics tool to assess the quality of the UML diagrams and generate appropriate quantitative results. Last phase is

Deploy the proposed hybrid clone detection methodology that determines the similarity between models to detect the clones.

3.1 *Computation of Statistical Similarity Measure of models*

SD Metrics features the information obtained from class diagrams, its iterations and sequence. 21 different elucidates the metrics and the features that are used for the statistical assessment.

The UML class diagrams are evaluated based on the 21 different parameters and the statistical features of the diagrams are obtained as the outputs. The method of weighted variance is applied on the output of the SD metrics to arrive the Statistical Variability Measure (SVaM) for each of the above mentioned feature according to the equation 1.

$$SVaM = \frac{\sum_{i=1}^{n} \alpha_1 V_1 + \alpha_2 V_2 + \ldots + \alpha_n V_n}{\sum_{i=1}^{n} V_1^{-1} + V_2^{-1} + \ldots + V_n^{-1}} \quad (1)$$

In the above equation, V_i indicates the variance of the metric values and α_i is the value of the metric. The SVaM used in the proposed work uses the variance as the weighing factor, as the deviation among the classes is an important attribute to isolate the model clones. The estimated SVaM gives the variability among the elements and hence the value should be normalised between [0-1]. The Statistical Similarity Measure (SSM) is obtained from Equation 2.

$$SSM = 1 - SVaM \quad (2)$$

SSM of a particular class diagrams are estimated on the following statistical measures:

- PosDeltas: This is the aggregate value of the positive metric deltas for the metric which signifies the change in metric values upon the addition of new element or on modification of the existing elements in a model.
- NegDeltas: This is the aggregate value of the negative metric deltas for the metric which represents the change in metric values upon the deletion of new element or on modification of the existing elements in a model.
- PosD_na: Aggregate of the positive metric deltas without the added elements. This is a measure of deviation (increase) when an existing design element is modified.

- NegD_nd: Aggregate of the negative metric deltas without the deleted elements. This is a measure of deviation (decrease) when an existing design element is modified.

In addition to the above, the sum, average, association values, percentiles values for the all the design models are also considered for the estimation of SSM.

3.2 *Jaro -Winkler Index*

The proposed method employs Jaro-Winkler Distance metric for estimating the similarity between elements of the model clones [15]. This is a lexical based measure that counts the matching characters among the attributes. This also considers the coincidences (co) and the transpositions (ta) in the lexicons. High value of Jaro-Winkler Index (JWI) indicates more similarity between the compared elements. Estimation of JWI is based on the Equation 2.

$$JWI\left(M1,M2\right)=\frac{1}{3}\left(\frac{co}{|M_1|}+\frac{co}{|M_2|}+\frac{co-ta}{co}\right) \tag{2}$$

This index is found for every element of both the models that are to be compared. The attributes that were considered in this study are class, data types, operations, parameter, model, association, association end, stereotype and tagged values. Association, Indicator that shows whether the instances of the models are linked. Association end, This indicates how two classes are related. However, it also depicts the Ownership, navigability, multiplicity and role.

JWI is a computationally economical way of estimating the similarity between two models as the time complexity is $O(n^2)$.

3.3 *Determining model clone*

After estimating the JWI and SSM between the UML diagrams, the final similarity measure between the model clone is computed as the sum between the two. The complete algorithm of the proposed hybrid model is given in Table 1.

Higher the SM value, more similar the models are. The proposed methodology is much robust that it can be applied to find the similarity between mutated versions of the models which is very difficult. Also, this method is very much helpful in detecting the semantically equivalent and near miss clones as the SSM and JWI operates in different domain spaces

Table 1

Model to detect the model clone

// Input: SD metrics, association values, attribute list

// Output: Similarity Measure (SM)

For all values of αi and Vi of the statistical measures of SD metrics compute the SSM

$$\text{SVaM} = \frac{\sum_{i=1}^{n} \acute{a}_1 V_1 + \acute{a}_2 V_2 + \ldots + \acute{a}_n V_n}{\sum_{i=1}^{n} V_1^{-1} + V_2^{-1} + \ldots + V_n^{-1}}$$

Normalise the SVaM

Compute SSM

SSM=1-SVaM

Compute the JWI between lexical attributes of the models

$$\text{JWI}(\text{M1, M2}) = \frac{1}{3}\left(\frac{\text{co}}{|\text{M}_1|} + \frac{\text{co}}{|\text{M}_2|} + \frac{\text{co} - \text{ta}}{\text{co}} \right)$$

Estimate the overall SM as

SM ← Mean(SSM, JWI (M1, M2))

efficient enough to capture the deviations, modifications and underlying patterns from the UML diagrams and its equivalent XMI files.

4. Result and analysis

This work analyses the performance of the proposed hybrid model by using a clones model of banking use cases from two different banks. Figure 1 gives the UML diagram bank_1 and bank_2.

For further processing the equivalent XMI files are obtained for both the diagrams to compute the SD metrics, SSM and JWI as mentioned in Table 1. The design comparison between the models is envisages three types of values, and they are relative. Zero- Signifies no difference in the corresponding parameter. Positive- Signifies the magnitude as new elements are added to the corresponding parameter. Negative-Signifies the magnitude as elements are deletes from the corresponding parameter.

In addition to this, the JWI and the SSM for various elements of the models are given in Table 2.SSM for the models are estimated from comparison metrics between the models obtained from the SD metrics. The metrices are multiplied by the variance factor and estimated as mentioned in Equation 1. The elements class, attributes , variables, association that are used for estimating the JWI to detect the clone similarity.

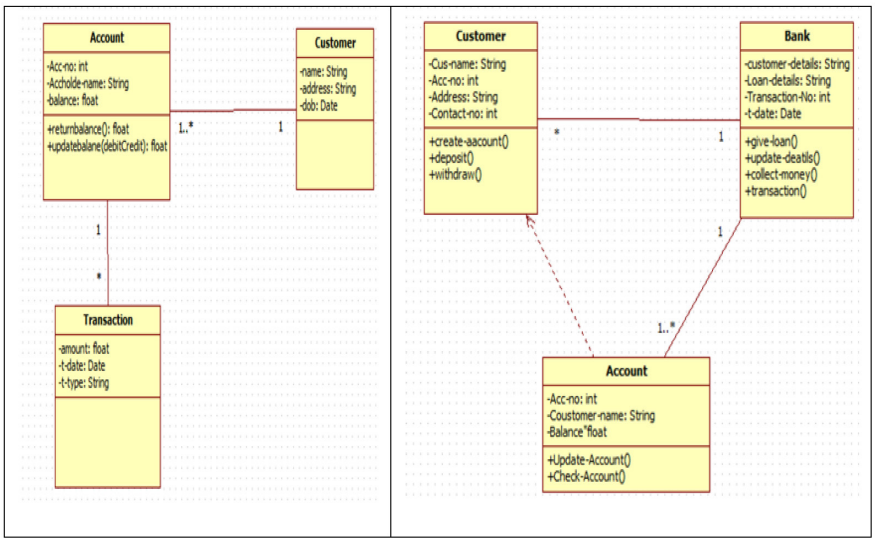

Figure 1

Class diagrams of two different banks (bank_1 and bank_2) drawn in starUML

The JWI similarity index learns the lexicons from the elements listed in the table and compares them to determine the similarity index value according to Equation 2. For instance, the names of two elements of the UML are given as Design Model. Account. updatebalance and Design Model. Account. updatebalance. Debit Credit. Intuitively both refers the same operation expressed in the UML diagrams using different lexical formats indicating a model clone of type 4. The similarity index measured using the JWI between these elements will be very high, though they seem to identify two different entities. The results from the SD metrics for the elements mentioned in Table 2 for the given banking UML diagram gives the similarity index between the range [0. 8767-1], as the given models are clones or mutated versions of the based model. Table 2 summarises the computational results of the given UML diagrams of the banking use case using the proposed method.

Table 2

Estimation of SM from JWI and SSM

JWI	SVaM	SSM	SM
0.8742	0.2444	0.7557	0.8149

The results indicate that the given model are clones as they exhibit a similarity measure of 0.81395, which is a very high similarity value. The proposed method analyses the models in lexical aspects using the JWI index and statistical aspect using the SSM. So, there are high chances that the proposed method will identify clone that belong to all four types.

5. Conclusion and future work

Model clone detection is detrimental in the industry, as the evolution of similar models and their mutated versions must be kept in bay. This work proposes a novel method to detect type 4 clones which are semantically equivalent making its detection very tedious. Two different UML models of the banking use case is considered and the proposed approach analyses the SD metrics of these models in two perspectives: 1) compute SSM by computing the statistical measure of the model elements and 2) estimate the JWI that investigates the SD metrics in lexical manner. The proposed method is straight forward and considers every possibility of the model to be a clone as the methods used explores the data in different way to compute the similarity measure. The empirical analysis shows that the SSM of the given UML models accounts to 0.7557, which is an indication of statistical similarity between the models. The model estimates the JWI as 0.8742. Thus the overall Similarity measure as given by the proposed method is 0.8149, which is a high value, thus claiming that the models are clones. The result indicates that this method can be effectively deployed in models with better scalability and robustness.

References

[1] Whittle, Jon, John Hutchinson, and Mark Rouncefield. “The state of practice inmodel-driven engineering.” IEEE software 31, no. 3 (2013).

[2] Basciani, Francesco, Ludovico Iovino, and Alfonso Pierantonio. “Mdeforge: an extensible web-based modeling platform.” In Proceedings of the 2nd International Workshop on Model-Driven Engineering on and for the Cloud co-located with the 17th International Conference on Model Driven Engineering Languages and Systems, CloudMDE@ MoDELS 2014, Valencia, Spain, September 30, 2014, vol. 1242, pp. 66-75. CEUR-WS, (2014).

[3] Hebig, Regina, Truong Ho Quang, Michel RV Chaudron, Gregorio Robles, and Miguel Angel Fernandez. “The quest for open source

projects that use UML: mining GitHub." In Proceedings of the ACM/ IEEE 19th international conference on model driven engineering languages and systems, pp. 173-183. (2016).

[4] Bird, Christian, Tim Menzies, and Thomas Zimmermann, eds. The art and science of analyzing software data. Elsevier, (2015).

[5] Shobha, G., Ajay Rana, Vineet Kansal, and Sarvesh Tanwar. "Code clone detection—a systematic review." Emerging Technologies in Data Mining and Information Security: Proceedings of IEMIS 2020, Volume 2 (2021).

[6] Kim, Miryung, Lawrence Bergman, Tessa Lau, and David Notkin. "An ethnographic study of copy and paste programming practices in OOPL." In Proceedings. 2004 International Symposium on Empirical Software Engineering, 2004. ISESE'04., pp. 83-92. IEEE, (2004).

[7] Störrle, Harald. "Towards clone detection in UML domain models." In Proceedings of the Fourth European Conference on Software Architecture: Companion Volume, pp. 285-293. (2010).

[8] Stephan, Matthew, and James R. Cordy. "MuMonDE: A framework for evaluating model clone detectors using model mutation analysis." *Software Testing, Verification and Reliability* 29, no. 1-2 (2019).

[9] Alalfi, Manar H., James R. Cordy, Thomas R. Dean, Matthew Stephan, and Andrew Stevenson. "Near-miss model clone detection for Simulink models." In 2012 6th International Workshop on Software Clones (IWSC), pp. 78-79. IEEE,(2012).

[10] Hammad, Muhammad, Önder Babur, Hamid Abdul Basit, and Mark van den Brand. "Deepclone: modeling clones to generate code predictions." In Reuse in Emerging Software Engineering Practices: 19th International Conference on Software and Systems Reuse, ICSR 2020, Hammamet, Tunisia, December 2–4, 2020, Proceedings 19, pp. 135-151. Springer International Publishing (2020).

[11] Shobha, G., Ajay Rana, Vineet Kansal, and Sarvesh Tanwar. "Comparison between Code Clone Detection and Model Clone Detection." In 2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO), pp. 1-5. IEEE (2021).

[12] Rattan, Dhavleesh, Rajesh Bhatia, and Maninder Singh. "Software clone detection: A systematic review." *Information and Software Technology* 55, no. 7 (2013).

[13] White, Martin, Michele Tufano, Christopher Vendome, and Denys Poshyvanyk. "Deep learning code fragments for code clone detection." In Proceedings of the 31st IEEE/ACM international conference on automated software engineering, pp. 87-98 (2016).

[14] Rattan, Dhavleesh, Rajesh Bhatia, and Maninder Singh. "Model clone detection based on tree comparison." In 2012 Annual IEEE India Conference (INDICON), pp. 1041-1046. IEEE (2012).

[15] Leonardo, Brinardi, and Seng Hansun. "Text documents plagiarism detection using Rabin-Karp and Jaro-Winkler distance algorithms." *Indonesian Journal of Electrical Engineering and Computer Science* 5, no. 2 (2017).

[16] Satvik Vats & B. B. Sagar (2019) Performance evaluation of K-means clustering on Hadoop infrastructure, *Journal of Discrete Mathematical Sciences and Cryptography,* 22:8, 1349-1363, DOI: 10.1080/09720529.2019.1692444 .

[17] Satvik Vats, Sunny Singh, Gaurav Kala, Rahul Tarar & Sanjay Dhawan FRCR (2021) iDoc-X: An artificial intelligence model for tuberculosis diagnosis and localization, *Journal of Discrete Mathematical Sciences and Cryptography,* 24:5, 1257-1272, DOI: 10.1080/09720529.2021.1932910.

[18] Kumar, Vishal, Gayatri Sakya, and Chandra Shankar. "WSN and IoT based smart city model using the MQTT protocol." *Journal of Discrete Mathematical Sciences and Cryptography* 22.8 (2019): 1423-1434, https://www.tandfonline.com/doi/abs/10.1080/09720529.2019.1692449.

[19] Gayatri Sakya & Vidushi Sharma, "Performance evaluation of AD-MC-MAC mission critical protocol for wireless sensor networks‖", *Journal of Discrete Mathematical Sciences and Cryptography,* 22:8, 1407-1422, 2019 DOI: 10.1080/09720529.2019.1692448.

[20] Gayatri Sakya, Chhaya Dalela, Laxman Singh & Anuj Jain, –Machine learning based MAC protocol design for pipeline leakage detection in smart city project‖, *Journal of Discrete Mathematical Sciences and Cryptography,* 24:5, 1283-1292, 2021 DOI: 10.1080/09720529.2021.1932915.

[21] Malik, M., Joshi, A. & Sakya, G. Optimized leach protocol for energy management in wireless sensor network. Multimed Tools Appl (2023). https://doi.org/10.1007/s11042-023-16248-2.