2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C) | 978-1-6654-2484-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/MODELS-C53483.2021.00074

A Lightweight Approach for the Automated Classification and Clustering of Metamodels

Riccardo Rubei, Juri Di Rocco, Davide Di Ruscio, Phuong T. Nguyen, and Alfonso Pierantonio
Università degli studi dell'Aquila, 67100 L'Aquila, Italy
riccardo.rubei@graduate.univaq.it, {juri.dirocco, davide.diruscio, phuong.nguyen, alfonso.pierantonio}@univaq.it

***Abstract*—The growing adoption of model-driven engineering raised the need for techniques and tools supporting modeling artifacts' reusability. In this respect, several model repositories have been proposed by academia and industry so that modelers can exploit advanced searching facilities to identify reusable artifacts that might fit the particular problem at hand. Despite the enduring quest for the right ways to search and retrieve modeling artifacts, satisfactory solutions are still missing. This paper investigates the adoption of general-purpose indexing and search features provided by Apache Lucene to support the classification and clustering of metamodel repositories. In particular, we show that Apache Lucene allows us to get accurate results whenever the mandatory requirements of more appropriate techniques, such as hierarchical clustering or neural networks, cannot be met.**

***Index Terms*—Model-Driven Engineering; Classification; Clustering; Apache Lucene**

I. Introduction

Over the last few years, both academia and industry proposed several model repositories to foster the reusability of modelling artefacts [1], [2], [3], [4]. Modelers can interact with such systems by exploiting advanced searching and browsing facilities so that users can identify sets of reusable elements for the problem at hand, instead of e.g., developing new metamodels and model transformation from scratch. Getting insights from software artefacts is a challenging task, which represents 60% of the effort spent on software comprehension [5]. Similarly, in MDE the exploration of datasets consisting of e.g., metamodels and models is challenging because items are typically not labeled and properly organized. For instance, a recent work [6] introduced a dataset consisting of ≈90,000 models, which are made available without any additional information or metadata. A labeled metamodel dataset has been made available by Babur [7], even though its population is limited to 555 metamodels. The reduced availability of labeled artifacts can be explained by the fact that manual annotation is generally tedious and inapplicable in the case of a large amount of data.

To mitigate the problems related to the lack of labeled data on the one hand and to provide model repositories with advanced searching and browsing functionalities, on the other hand, automated classification [8] and clustering [9], [10] techniques have been recently proposed. Even though a good accuracy characterizes such methods, their applications are not always possible since they are enabled only if some constraints are satisfied. In particular, in the case of classification through machine learning techniques, it is mandatory to perform training phases by consuming labeled datasets, which are not always available. Typically, datasets are created by crawling items from heterogeneous sources, and labeling them is a strenuous process, which is not always possible to perform. Concerning clustering approaches, they are shown to suffer performance issues, especially for large datasets: the time needed to automatically group objects according to a given similarity function increases exponentially with the number of elements of the considered dataset.

This paper investigates the adoption of general-purpose indexing and search features provided by Apache Lucene to support the automated classification and clustering of metamodels. The final goal is to examine to what extent it is possible to discard the structural characteristics of metamodels to perform such operations and when instead, it is advisable to use dedicated tools. The proposed approach consists of the following phases: *(i)* Lucene indexes the available unlabelled metamodel dataset; *(ii)* given an input metamodel, a corresponding query is automatically created by exploiting Lucene boosting facilities; *(iii)* the tool automatically retrieves a ranked list of metamodels that are similar to the one given in input. We show that the produced output consists of a group of metamodels, which coincides with a given extent with the cluster, which dedicated clustering techniques would produce. Moreover, we show that the first element of the ranked list can be used to classify the metamodel in input, i.e., to suggest the label be assigned to the metamodel of interest.

We evaluated the accuracy of the proposed approach on a dataset consisting of 551 metamodels by comparing it with two existing baselines [10], [8]. Interestingly, the proposed Lucene-based approach has similar accuracy to the considered baselines. However, it is worth noting that our proposed approach is unsupervised, and consequently, no data preprocessing stages are needed. Moreover, the proposed technique outperforms both approaches in the considered baselines in terms of execution time.

II. Background

Data analysis procedures have the goal of supporting *explorations* of available datasets or *confirming* some hypothesis about phenomena of interest. In any case, data analysis tasks rely on the capability of constructing or recognizing groups of similar objects with respect to given definitions of *similarity* [11]. *Clustering* and *classification* are two important methods, which are used to analyse data under disposal [11]. The former

978-1-6654-2484-4/21/$31.00 ©2021 IEEE
DOI 10.1109/MODELS-C53483.2021.00074

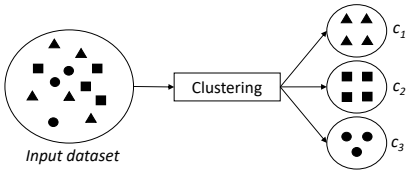


Fig. 1. Explanatory clustering example.

aims to identify similarities among objects in the considered data and thus construct groups of objects with characteristics in common. Thus, employing clustering techniques, the given collection of unlabelled data is automatically organized in different groups. Each group is assigned a distinguishing label. As shown in Fig. 1, the set of labels used to characterize the dataset elements is not given as input since it is part of the outcome of the clustering phase. In the case of classification instead, as shown in Fig. 2, a set of predefined labels is given as input together with the elements to be labelled; the goal is to assign each object in the dataset of interest to one of the available groups.

Different research communities have adopted clustering and classification techniques for grouping unlabelled data. The increasing availability of reusable items like models, metamodels, and model transformations fostered the MDE community to propose techniques and tools to support the exploration and the analysis of extensive collections of reusable modeling artefacts. The reasons behind such efforts are manifold. For instance, in [12], [13] the authors mine metamodel datasets employing dedicated metrics to characterize the complexity of metamodels. Hierarchical clustering approaches have been conceptualized [10], [9] to automatically group metamodels and support the graphical exploration of metamodel repositories by employing different similarity functions, including the complexity similarity of the stored items. AURORA [8] is a neural network to support the automated classification of metamodels. Once being adequately trained, AURORA is able to predict the category to be assigned to a new metamodel given as input with reasonable accuracy.

By focusing on the application of clustering and classification to mine metamodel datasets, appropriate approaches like those presented in [10], [9], [8] cannot always be adopted. In particular, hierarchical clustering techniques have been shown to suffer performance issues: given the dataset of interest, a corresponding similarity matrix must be produced. To this end, it is necessary to calculate the similarity of all metamodel pairs in the dataset. Thus, when a new metamodel has to be added to the considered dataset, the whole similarity matrix must be recalculated by making the approach not appropriate when the content of the considered dataset is supposed to

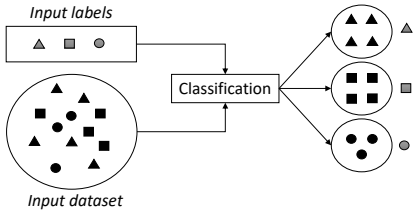


Fig. 2. Explanatory classification example.

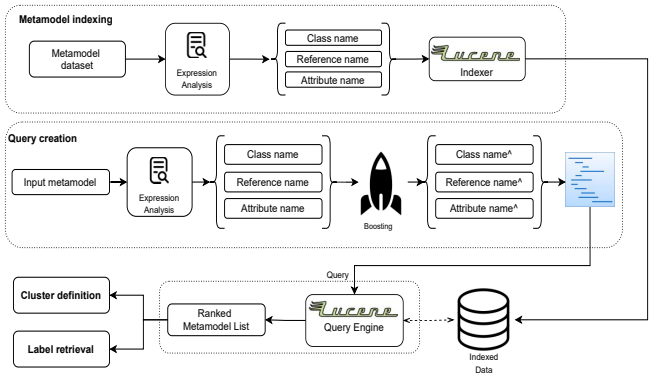


Fig. 3. Overview of the proposed approach.

change frequently. The adoption of AURORA with acceptable prediction accuracy is instead enabled when large and labeled datasets are available. Unfortunately, this is not always the case because labeling datasets is a strenuous and time-consuming task, which cannot always be performed.

III. Proposed Approach

In this section, we present an approach based on general-purpose indexing and searching facilities provided by Apache Lucene [1] to support both the classification and clustering of metamodels. Lucene is a powerful and open-source Java library to add search facilities to any application. It is a general-purpose technology, which can index and search any text. Even though there exist search engines specifically designed for models, e.g., [14], [15], [16], to mention a few, we want to investigate if it is still possible to employ general-purpose technologies to get good clustering and classification accuracies without deep model-specific customizations. Adopting the proposed approach is recommended when it is impossible to satisfy the requirements related to the adoption of appropriate approaches like hierarchical clustering and neural networks. In particular, users can adopt the Lucene-based approach when *(i)* there is no way to label the available metamodels manually; and *(ii)* it is not possible to accept the performance costs related to the calculation of similarity matrixes, especially when the available dataset is supposed to change frequently.

As shown in Fig. 3 the proposed approach relies on four components, i.e., *metamodel indexing*, *query creation*, *cluster definition*, and *label retrieval* as described in the following.[2]

Metamodel indexing: At this stage, the system pre-processes the metamodel dataset under disposal and creates corresponding indexes. It is important to remark that no training activities are needed, and the typical adoption of the proposed technology is intended to feed the system with data collected from different sources without any manual curation. The operation is performed for any newly added metamodel. Only the lexical information presents in the metamodels, i.e., names of the classes, attributes, and references, is used by the indexing; the process does not consider additional structural information

[1] https://lucene.apache.org/

[2] We already made available at https://bit.ly/2Qkxb5J the replication package of the conceived tool to allow for future research.

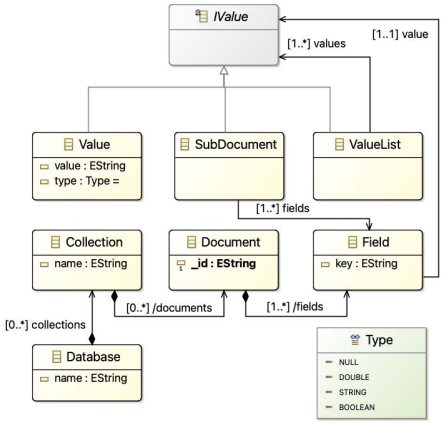


Fig. 4. A simple Ecore metamodel.

like cardinalities, containments, and hierarchies by making the indexing process simple. The metamodels are parsed, and their structural information is excerpted by means of *EMF core*.[3] These metamodels and the extracted elements are essentially considered as textual files. Despite such a simplification, the performed experiments (as shown in the following sections) confirm that the currently considered elements for indexing are enough for achieving a good accuracy of the proposed approach. Figure 4 shows a simple Ecore metamodel defining MongoDB databases. According to the defined metaclasses, each Database consists of Collection elements, which in turn contain Documents. Each Field in the document has a corresponding value, which can be of type Value, SubDocument, or ValueList. Table I shows the corresponding elements that are extracted and filled to the index being created. For each metaclass in the metamodel, the index contains its name, and the name of all the contained structural features.

TABLE I
ELEMENTS THAT ARE EXTRACTED TO INDEX THE METAMODEL IN FIG. 4

Class	Reference	Attribute
Field	value	key
IValue	-	-
ValueList	values	-
Database	collections	name
Value	-	value, type
Collection	documents	name
SubDocument	fields	-
Document	fields	_id

Query creation: Given an input metamodel, the system automatically creates a corresponding query to search the previously created index and get a ranked list of similar metamodels. For explanatory reasons, we consider the same metamodel shown in Fig. 4 to show the corresponding query that would be created. In particular, the query given in Fig. 5 is defined as a sequence of *<elementType:elementName>* pairs where *elementType* is the type of the element of interest that can be of type *class*, *reference*, or *attribute*; *elementName* is a string representing the name of the considered element e.g., the class *Field* or the attribute *value*. All the pairs in the sequence are in OR. Moreover, the boosting mechanism provided by Lucene is exploited to give different importance

[3]The Eclipse Modeling Framework https://www.eclipse.org/modeling/emf/

```
class:Field^2 OR reference:value^1.5 OR attributes:key OR
class:IValue^2 OR class:ValueList^2 OR reference:values^1.5 OR
class:Database^2 OR reference:collections^1.5 OR attributes:name OR
class:Value^2 OR attributes:value OR attributes:type OR
class:Collection^2 OR reference:documents^1.5 OR attributes:name OR
class:SubDocument^2 OR reference:fields^1.5 OR
class:Document^2 OR reference:fields^1.5 OR attributes:_id
```

Fig. 5. Example of a boosted query.

to the elements being queried. As shown in the example, elements of type *class* are of higher importance than reference elements, which in turn, have higher relevance than attributes. Boosting values have been empirically identified to get the best performance, and they are 2 for classes and 1.5 for references. Various studies [17], [18] have shown that the adoption of boosting brings in a better retrieval performance.

Cluster definition: Once the query is executed, Lucene retrieves the top *n* similar metamodels. Such metamodels are always structured as a set, i.e., a group of metamodels being similar to the one given as input. The system needs to know how many results should be retrieved; this number is called *hits*. Among the *hits* obtained item, the system will retrieve the top *n* of them. Since we are interested in providing a cluster of items similar to the one given as input, deciding how many items are part of the same group is crucial. Lucene orders these items by similarity; this means that the similarity decreases progressively from the top similar metamodel with the one given as input.

Label retrieval: As discussed in Sect. II, classification is performed to assign predefined labels to unlabelled objects. Thus, in metamodeling contexts, classification is performed when input metamodels need to be labeled to assign them to predefined groups. For instance, let us imagine that we are keeping organized a dataset of metamodels according to their intended goals e.g., for behavior specification or data definition. When adding new metamodels to the considered dataset, it is essential to keep the wanted organization by assigning the new metamodel to the right group. To this end, the proposed Lucene-based approach can be used by querying the available dataset to retrieve the metamodels that are similar to the one given in input. The first element in the retrieved ranked list is used to retrieve its corresponding group, which is assigned to the metamodel given as input. In this respect, it is necessary to have a dataset manually labeled by humans.

IV. EVALUATION

In this section we evaluate the ability of the system to provide meaningful clusters of metamodels as well as to classify an incoming metamodel, we compare it with two state-of-the-art tools, i.e., HC [10] and AURORA [8]. The former is an unsupervised learning tool that works based on a hierarchical clustering technique. Meanwhile, the latter is a supervised learning approach, and it makes use of a feed-forward neural network to classify metamodels. To compare and evaluate the systems, we rely on three quality metrics, i.e., *success rate*, *precision*, and *execution time*.

Dataset: For our evaluation, we adopted an existing dataset [19], which has been used to evaluate AURORA [8].

TABLE II
DATASET COMPOSITIONS.

Cluster	D_1 (138)	D_2 (276)	D_3 (551)
1	14	28	56
2	2	4	7
3	10	20	37
4	6	12	24
5	25	50	100
6	13	26	54
7	19	38	76
8	40	80	159
9	9	18	38

The dataset consists of 555 metamodels, manually inspected and classified into nine categories [7]. By carefully checking the dataset, we encountered some loading errors with four metamodels; thus, we decided to remove these from the dataset. Table II shows the precise composition of the considered dataset. In order to understand the capability of our approach, we tried different dataset sub-sets. In particular, we randomly excluded elements from the dataset of 551 elements, and we came out eventually with two additional datasets of 138 and 276 metamodels. By creating such datasets, we kept the original cluster distribution.

Methodology: The validation has been assessed by employing the *Leave-one-out* cross-validation technique [20]. Thus, only one element is removed as test and the whole dataset is used as training. The process is repeated *n* times where *n* is the size of the dataset. For instance, in the case of the dataset D_3, the process of indexing and testing occurred 551 times. For each execution, 550 metamodels have been indexed, and only one is used as a test to query the system. For each testing metamodel *m*, we retrieved an ordered list of *H* similar metamodels. We evaluated the performance of the approach by considering different cut-off values, i.e., by considering the first element, the top 5% and the top 10% of the considered dataset size, and the whole result. To evaluate the clustering capability of the approach, for each testing metamodel *m*, we compared the list of H_c retrieved metamodels (up to the cut-off value c of interest) with the *ground-true*, i.e., the content of the real cluster in the dataset under investigation (see Table II), where the input metamodel *m* is supposed to belong. To evaluate the classification capability of the proposed technique, for each testing metamodel *m*, we consider the label of the first element in the retrieved metamodel list. Such a label is assigned to *m* and we check the correctness of such an assignment with the ground-truth data.

Evaluation metrics: To assess the conceived approach, we make use of *Success rate* and *Precision* as they have been widely used to evaluate information retrieval systems [21], and *Execution time*. First, we consider the following definitions:

▷ *True positive (Tp)*: concerning the classification evaluation, *Tp* means that the category for the input metamodel is correctly identified. In the case of the clustering problem, given the calculated cluster related to the input metamodel, a metamodel therein is *Tp* if it belongs to the same cluster of the input metamodel according to the ground-true;

▷ *False positive (Fp)*: in the case of classification, *Fp* means that the retrieved category for the input metamodel is not the correct one. Given the calculated cluster related to the input metamodel, a metamodel therein is *Fp* if it is wrongly calculated as belonging to the same cluster of the input metamodel according to the ground-truth data.

Success rate: Given a set of *H* resulting metamodels, this metric measures the rate at which the proposed approach can return at least 1 correct metamodel belonging to the same cluster of the input metamodel for each metamodel *m* in *H*.

$$Success\ rate = \frac{count_{m \in H}(Tp > 0)}{|H|} \times 100\% \quad (1)$$

where the function *count()* is a Boolean function that returns 1 for each true positive.

Precision: the metric measures the rate of correct items over the entire set of retrieved items:

$$Precision = \frac{Tp}{Tp + Fp} \quad (2)$$

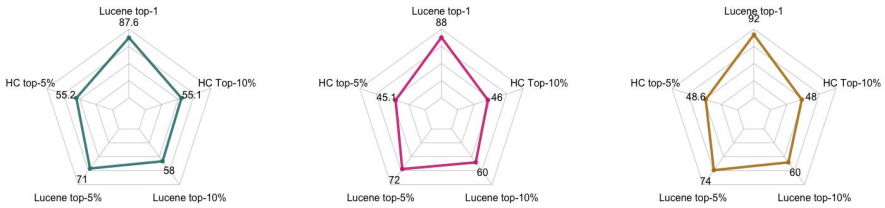
All these metrics are applied for all tests according to the *leave-one-out* method, and then the average values are calculated. As an example, let us assume we use as a query a metamodel belonging to Cluster 1, and the system returns as output a set of five metamodels as follows: three metamodels to Cluster 1, one to Cluster 3, and one to Cluster 7. The proposed cluster has three correctly identified metamodels over 5; the resulting precision is 60%. The total *clustering precision* is calculated as the mean of the precision values obtained for all the metamodels in the dataset. Similarly, the *classification precision* is calculated as the mean of the correctly identified categories over the total number of queried metamodels.

▷ *Recommendation time*: The time needed for the systems to generate predictions is measured using a laptop with Intel i5-8250U CPU @ 1.60GHz, 16GB RAM, and Ubuntu 18.04.

V. EXPERIMENTAL RESULTS

This section discusses the results obtained by the evaluation strategy explained in Sect. IV. Figure 7 shows the classification precision for Lucene and AURORA calculated on three different datasets. As expected, AURORA performs better than Lucene with the increasing size of the considered dataset. This explains the importance of having big datasets to train the underpinning neural network. However, it is worth noting the good precision of Lucene in top-ranking the right metamodel for all three datasets. In particular, for D_1 Lucene gets the same precision of AURORA. For bigger datasets AURORA outperforms Lucene even though for D_3 Lucene gets 92% of precision. This confirms our intuition, i.e., Lucene can support classification tasks when it is not possible to have big labeled datasets. Even in such cases, Lucene can be seen as a lightweight approach to support exploration activities for the datasets at hand. Lucene performs much better to support clustering activities, as shown in Fig. 6. Both Lucene *top-5%* and *top-10%* outperforms hierarchical clustering for all the three considered datasets. For instance, for the smaller dataset (i.e., D_1), Lucene *top-5%* precision is 71%. In contrast, the corresponding HC *top-5%* is 55.2%.[4] For *top-10%* the

[4]It is worth noting that to retrieve the *top-5%* and *top-10%* elements from the unordered sets produced by the hierarchical clustering, we ordered metamodel pairs according to their similarity values, and then retrieved the wanted metamodel subsets by following the induced ranking.



(a) D1 (b) D2 (c) D3

Fig. 6. Clustering Precision for Lucene and HC.

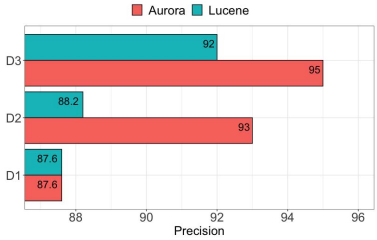


Fig. 7. Classification precision for Lucene and AURORA.

difference between Lucene and HC precision values is smaller, even though the Lucene one is still higher than that of hierarchical clustering, i.e., 58% compared to 55.1%.

Overall, Lucene shows good performance in clustering metamodels independently from the size of the considered datasets. In all the considered configurations, both Lucene precision and recall values are higher than that of the considered hierarchical clustering technique. For classification tasks, Lucene yields a comparable performance with that of AURORA for smaller datasets. Whenever newly labeled metamodels are added, the difference of the precision values increases at the advantage of AURORA; however, with the biggest considered dataset, Lucene can still get good precision (92%) compared with that of AURORA (95%).

Timing performance: We measured the efficiency concerning the time employed to prepare the data and to execute the query. For the most extensive considered dataset, for preparing the data and performing a query, Lucene, AURORA, and HC take ≈30 seconds, ≈67 seconds, and ≈2.8 hours, respectively. The results demonstrate the effectiveness of our approach, i.e., it can provide accurate prediction within short time.

VI. Related work

The problem of grouping metamodels has been investigated over the last few years. Basciani *et al.* [10] proposed an approach based on a hierarchical clustering technique. They explored several similarity measures to establish the correlation between metamodels. Besides the well-known *cosine similarity* and *Dice's coefficient* metrics, the authors conceived two other algorithms which specifically work with metamodels. In a similar work [9], VSM (vector space model) has been used to represent metamodels, which are then clustered with hierarchical clustering. The process makes use also of NLP techniques and weight schemas to augment the employed VSM, and *cosine similarity* was utilized as the distance metric. Strüber *et al.* [22] proposed a clustering technique based on a graph-based representation. The authors generated a meaningful group of sub-metamodels starting from a single monolithic metamodel. In this work, users are expected to interact with the system during the clustering process.

A tool to classify UML stereotypes has been proposed [23], with the ultimate aim of helping modelers design stereotypes, as domain knowledge is typically needed. The contribution is a mechanism that classifies stereotypes according to their capability to alter the syntax and semantics of the base language. The adoption of Machine Learning techniques is a common way to deal with classification. Nguyen *et al.* [24] employed a convolutional neural network (CNN) to classify a set of metamodels. A CNN is a particular neural network developed to overcome the complexity problem related to the connections among perceptrons, leading to an exponential number of weights and biases.

Moogle [16] is a model search engine that supports textual queries and relies on metamodels elements for creating indexes. Moogle can search for models conforming to different languages. The indexing mechanism relies on the Apache SOLR search engine. MAR [14] is a search engine for models based on a query-by-example approach. Apart from searching, MAR was tested to see if it is suitable as a classification tool, achieving good results. However, MAR has not been used for clustering metamodels. At the time of writing this paper, unfortunately, there are no publicly available replication packages and deployable tools that we might use for performing a proper comparison of MAR with the proposed approach. We plan to do this as a short-term plan.

VII. Conclusion and future work

This paper presented a Lucene-based approach to support the clustering and classification of metamodels. The technique has been devised to enable the exploration of metamodel repositories when more specialized techniques cannot be easily adopted due to the lack of labeled data or when the size of the available dataset is so significant to make the adoption of hierarchical clustering techniques prohibitive. Thus, the proposed approach meets demands for metamodel explorations that prefer quick responses and accept some inaccuracies. However, the proposed approach has been compared with HC and AURORA, two existing state-of-the-art techniques for clustering and classifying metamodels, and it has shown promising results, especially for clustering.

As future work, we plan to compare the proposed technique with the MAR search engine. Unfortunately, at the time of writing this paper, MAR cannot be locally deployed to support experiment replications and comparisons. However, our interest is to investigate to what extent it is still possible to get good accuracy without employing model-specific aspects. For instance, in MAR, the structural complexity of models is adequately encoded and indexed. Similarly, in HC, different similarity functions were devised, including those measuring metamodel distances in terms of their structural complexity. As shown in this paper, considering structural metamodel characteristics is not crucial. The Lucene-based approach has shown good precision by considering only the terms of metamodel elements. A proper comparison of the proposed approach with MAR would help continue such an investigation. Last but not least, we are going to test our approach on larger datasets, aiming to study its applicability in real-world settings.

REFERENCES

[1] C. Hein, T. Ritter, and M. Wagner, "Model-driven tool integration with modelbus," in *Workshop Future Trends of Model-Driven Development*, 2009, pp. 50–52.

[2] B. Karasneh and M. R. Chaudron, "Online img2uml repository: An online repository for uml models." in *EESSMOD@ MoDELS*, 2013, pp. 61–66.

[3] M. Koegel and J. Helming, "EMFStore: a model repository for EMF models," in *2010 ACM/IEEE 32nd Int. Conf. on Software Engineering*, vol. 2, 2010, pp. 307–308.

[4] R. Kutsche, N. Milanovic, G. Bauhoff, T. Baum, M. Cartsburg, D. Kumpe, and J. Widiker, "Bizycle: Model-based interoperability platform for software and data integration," *Procs. the MDTPI at ECMDA*, vol. 430, 2008.

[5] P. Bourque, R. E. Fairley, and I. C. Society, *Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0*, 3rd ed. Washington, DC, USA: IEEE Computer Society Press, 2014.

[6] G. Robles, T. Ho-Quang, R. Hebig, M. R. Chaudron, and M. A. Fernandez, "An Extensive Dataset of UML Models in GitHub," in *2017 IEEE/ACM 14th Int. Conf. on Mining Software Repositories (MSR)*. Buenos Aires, Argentina: IEEE, May 2017, pp. 519–522.

[7] Ö. Babur, "A labeled Ecore metamodel dataset for domain clustering," Mar. 2019.

[8] P. T. Nguyen, J. Di Rocco, D. Di Ruscio, A. Pierantonio, and L. Iovino, "Automated classification of metamodel repositories: A machine learning approach," in *2019 ACM/IEEE 22nd Int. Conf. on Model Driven Engineering Languages and Systems (MODELS)*, 2019, pp. 272–282.

[9] Ö. Babur, L. Cleophas, and M. van den Brand, "Hierarchical clustering of metamodels for comparative analysis and visualization," in *Modelling Foundations and Applications*, A. Wasowski and H. Lonn, Eds. Cham: Springer Int. Publishing, 2016, pp. 3–18.

[10] F. Basciani, J. Di Rocco, D. Di Ruscio, L. Iovino, and A. Pierantonio, "Automated clustering of metamodel repositories," in *Advanced Information Systems Engineering*, S. Nurcan, P. Soffer, M. Bajec, and J. Eder, Eds. Cham: Springer Int. Publishing, 2016, pp. 342–358.

[11] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[12] J. D. Rocco, D. D. Ruscio, L. Iovino, and A. Pierantonio, "Mining metrics for understanding metamodel characteristics," in *6th Int. Work. on Modeling in Software Engineering, MiSE 2014, Hyderabad, India, June 2-3, 2014*, J. M. Atlee, V. Kulkarni, T. Clark, R. B. France, and B. Rumpe, Eds. ACM, 2014, pp. 55–60.

[13] J. R. Williams, A. Zolotas, N. D. Matragkas, L. M. Rose, D. S. Kolovos, R. F. Paige, and F. A. C. Polack, "What do metamodels really look like?" in *Procs. the 3rd Int. Work. on Experiences and Empirical Studies in Software Modeling*, vol. 1078. CEUR-WS.org, 2013, pp. 55–60.

[14] J. A. H. López and J. S. Cuadrado, "Mar: A structure-based search engine for models," 2020.

[15] K. Barmpis and D. Kolovos, "Hawk: Towards a scalable model indexing architecture," in *Procs. the Work. on Scalability in Model Driven Engineering*, 2013, pp. 1–9.

[16] D. Lucrédio, R. P. d. M. Fortes, and J. Whittle, "Moogle: A model search engine," in *Int. Conf. on Model Driven Engineering Languages and Systems*. Springer, 2008, pp. 296–310.

[17] M. Borg, P. Runeson, J. Johansson, and M. V. Mäntylä, "A replicated study on duplicate detection: Using apache lucene to search among android defects," in *Procs. the 8th ACM/IEEE Int. Symposium on Empirical Software Engineering and Measurement*, 2014, pp. 1–4.

[18] R. Rubei, C. Di Sipio, P. Nguyen, J. Rocco, and D. Di Ruscio, "Postfinder: Mining stack overflow posts to support software developers," *Information and Software Technology*, vol. 127, p. 106367, 06 2020.

[19] Ö. Babur, L. Cleophas, and M. van den Brand, "Hierarchical clustering of metamodels for comparative analysis and visualization," in *European Conf. on Modelling Foundations and Applications*. Springer, 2016, pp. 3–18.

[20] S. Arlot and A. Celisse, "A survey of cross validation procedures for model selection," *Statistics Surveys*, vol. 4, 07 2009.

[21] M. Robillard, R. Walker, and T. Zimmermann, "Recommendation systems for software engineering," *IEEE Softw.*, vol. 27, no. 4, pp. 80–86, Jul. 2010.

[22] D. Strüber, M. Selter, and G. Taentzer, "Tool support for clustering large meta-models," in *Procs. the Work. on Scalability in Model Driven Engineering*, ser. BigMDE '13. New York, NY, USA: Association for Computing Machinery, 2013.

[23] S. Berner, M. Glinz, and S. Joos, "A classification of stereotypes for object-oriented modeling languages," in *Procs. the 2nd Int. Conf. on The Unified Modeling Language: Beyond the Standard*, ser. UML'99. Berlin, Heidelberg: Springer-Verlag, 1999, p. 249–264.

[24] P. T. Nguyen, D. Di Ruscio, A. Pierantonio, J. Di Rocco, and L. Iovino, "Convolutional neural networks for enhanced classification mechanisms of metamodels," *J. of Systems and Software*, vol. 172, p. 110860, 2021.