Using the ModelSet Dataset to Support Machine Learning in Model-Driven Engineering

José Antonio Hernández López
Universidad de Murcia
Spain
joseantonio.hernandez6@um.es

Javier Luis Cánovas Izquierdo
UOC – IN3
Spain
jcanovasi@uoc.edu

Jesús Sánchez Cuadrado
Universidad de Murcia
Spain
jesusc@um.es

ABSTRACT

The availability of curated collections of models is essential for the application of techniques like Machine Learning (ML) and Data Analytics to MDE as well as to boost research activities. However, many applications of ML to address MDE tasks are currently limited to small datasets. In this demo paper, we will present ModelSet, a dataset composed of 5,466 Ecore models and 5,120 UML models which have been manually labelled to support ML tasks (http://modelset.github.io). ModelSet is built upon the models collected by the MAR search engine (http://mar-search.org), which provides more than 500,000 models of different types. We will describe the structure of the dataset and we will explain how to use the associated library to develop ML applications in Python. Finally, we will describe some applications which can be addressed using ModelSet.

CCS CONCEPTS

• **Software and its engineering → Model-driven software engineering**; *Unified Modeling Language (UML)*; **Software libraries and repositories**; • **Computing methodologies → Machine learning**.

KEYWORDS

Model classification, Model-Driven Engineering, Machine learning

ACM Reference Format:
José Antonio Hernández López, Javier Luis Cánovas Izquierdo, and Jesús Sánchez Cuadrado. 2022. Using the ModelSet Dataset to Support Machine Learning in Model-Driven Engineering. In *ACM/IEEE 25th International Conference on Model Driven Engineering Languages and Systems (MODELS '22 Companion), October 23–28, 2022, Montreal, QC, Canada.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3550356.3559096

1 INTRODUCTION

The use of Machine Learning (ML) techniques to solve Model-Driven Engineering (MDE) problems is becoming widespread. This can be witnessed by the publication of recent works which address different types of problems in MDE. For instance, the task of automatically labelling models stored in model repositories has been successfully addressed using feed-forward neural networks [17]. An LSTM architecture is used to perform model transformation by example [5]. Graph neural networks are used to assess the realism of model generators [14]. Model assistants based on providing recommendations are starting to be built using different techniques [9, 19]. A key element to successfully apply ML techniques is the existence of curated datasets of modelling artefacts. In particular, in this work we will focus on software models. In this sense, some applications are unsupervised and only require the existence of a large number of models. Other applications are supervised and require not only the models but a set of associated labels. The creation of a labelled dataset is typically a manual and tedious process, which makes their creation costly.

In this context, the availability of adequate datasets of models is scarce. Regarding unlabelled modelling datasets, there are a few alternatives. The Lindholmen dataset [18] contains more than 90,000 UML models (although it is not curated and actually many of the models are broken). A dataset of OCL constraints, which also includes associated Ecore models, is presented by Mengerink et al. [16]. An alternative, typically done by many empirical papers is to roll their own GitHub crawler [3]. With regard to labelled datasets, some works have used the dataset created by Babür [2], but it contains only 555 Ecore meta-models thus hindering the potential to obtain reliable conclusions from its use.

To alleviate this issue and to boost research activities of ML techniques applied to MDE, we built the ModelSet dataset [11]. The dataset contains 5,466 Ecore models and 5,120 UML models manually labelled with its category, plus a set of additional tags. ModelSet relies on a subset of the models provided by the MAR search engine [12, 13]. MAR makes available more than 500,000 models[1] of different types, including Ecore, UML and BPMN, among others; extracted from GitHub, GenMyModel and the AtlanMod zoo. We believe that the provision of these two datasets may become key to promote and improve research activities related to the application of ML techniques to address MDE tasks.

In this paper, we will present ModelSet focusing on technical details like its structure and the available support library for Python. Finally, we will describe some ML tasks in MDE which could be facilitated by ModelSet.

Organization. Section 2 describes the features of ModelSet and provides some details about how to use it. Section 3 describes some applications which can be implemented using ModelSet. Finally, Section 4 concludes the paper.

MODELS '22 Companion, October 23–28, 2022, Montreal, QC, Canada
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9467-3/22/10…$15.00
https://doi.org/10.1145/3550356.3559096

[1]http://mar-search.org

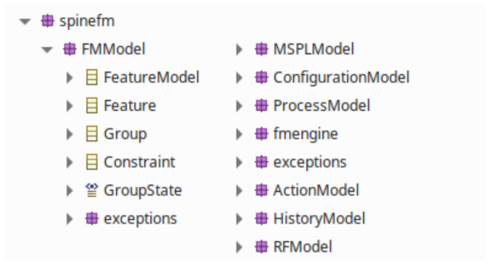


Figure 1: Example Ecore model.

2 MODELSET

ModelSet is a dataset of software models which has been manually labelled and curated in order to foster research in ML and MDE. The project is structured in five separate projects, which are available in the corresponding repositories at https://github.com/modelset. In particular, the repositories are the following:

- modelset-dataset. This repository contains the databases with the manually created labels, some Java code to facilitate the handling of the models and scripts to generate the released package.
- modelset-datasetcreator. An Eclipse plug-in to facilitate the labelling of EMF-based models, like Ecore models and UML models. It implements a method for fast labelling described by Hernández López et al. [11].
- modelset-py. A Python library to facilitate loading ModelSet and using it to train ML algorithms. The library will be described in more detail in Sect. 2.3.
- modelset-apps. This repository contains example projects which uses ModelSet to perform simple ML tasks with software models.
- modelset.github.io. A website available at http://modelset.github.io intended to explore the models and the labels of ModelSet.

2.1 Dataset Description

ModelSet is a collection of labelled software models comprising 5,466 Ecore and 5,120 UML models, making a total of 10,586 models. The models were collected from GitHub and GenMyModel repositories. Especifically, they were reused from the models made available by early versions of MAR. For Ecore, the models were retrieved from GitHub; and for UML, they were collected from GenMyModel. We filtered out duplicated files by computing the MD5 hash of each file.

The models were manually labelled with several types of labels. The labelling was performed by the authors Javier Luis Cánovas and Jesús Sánchez, who have more than 15 years of experience in modelling. To address possible conflicts, the authors met together and each one reviewed a random sample of size 25% of the models labelled by the other author. All disagreement cases were discussed between the authors to reach consensus.

As a running example, Fig. 1 shows one of the meta-models of the dataset. It is a meta-model to represent feature models as part of a software product line. The meta-model contains nine sub-packages but for the sake of the space only the contents of the package FMModel are shown. This model, and the rest of the models in the ModelSet dataset have been manually labelled with the following types of labels:

Table 1: Usage of labels in ModelSet. Taken from [11]

Label	Ecore		UML	
	# values	*coverage*	*# values*	*coverage*
Category	224	100.00%	135	100.00%
Tags	387	117.53%	92	52.99%
Purpose	29	31.16%	–	–
Notation	8	11.09%	–	–
Tool	188	7.78%	–	–
Main-diagram	–	–	6	108.03%

coverage > 100% when label is assigned more than once.

- **Category**. This is the main type of label used in ModelSet. It represents a family of models sharing a similar application domain. In the running example, the model has been labelled as *features* since it is a meta-model to specify feature models. Therefore, all models with characteristics typical of feature models have been labelled with this category.
- **Tags**. This label includes additional keywords providing additional insights about the model, typically specializing the value of the category. For instance, the example model has been labelled with tags *feature-modelling* and *spl*.
- **Purpose**. For some models it was possible to determine its purpose. For instance, a model used in a university course about MDE is assigned the label *assignment* when they were developed by students as part of a course project. Other models were assigned the label *experiment* since they were built for experimental purposes.
- **Notation**. This type of label attempts to indicate whether the meta-model is used as part of a Domain Specific Language (DSL) and there is an associated notation in the same project. For instance, typical labels include *Xtext* for textual languages and *Sirius* for graphical languages.
- **Tool**. This type of label indicates whether the meta-model has been identified as part of a tool. In the example, the meta-model is labelled with the name of the associated tool, *SpineFM*.
- **Main-diagram**. This label is only available in UML models, and identifies the main diagrams included in the model. For instance, if a model contains several diagrams but the most developed or representative diagram is the class diagram, the model is labelled with *class-diagram*.

In total, 28,719 labels were used (15,288 and 13,431 labels in Ecore and UML models, respectively), with an average number of labels per model of 2.71 (2.80 and 2.62 in Ecore and UML, respectively). Table 1 shows the distribution of labels.

2.2 Dataset Structure

The structure of ModelSet is graphically depicted in Fig. 2. ModelSet has been populated by two sets of files, one for Ecore and one for UML. The former were collected from GitHub and the latter from

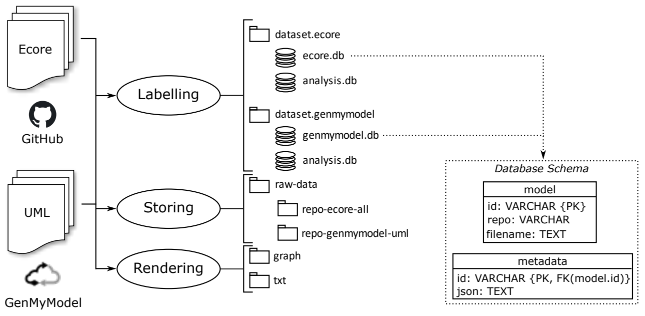


Figure 2: Structure of the dataset.

Listing 1: Example of metadata.

```
{ "category": ["features"],
  "tags":["feature-modelling","spl"],
  "tool":["spinefm"] }
```

GenMyModel. Once the model files were collected, they were the input of the labelling process, in which the models were manually annotated with the labels described above. The actual models are stored in a specific folder, and there is a rendering process, which generates alternative representations of the models (e.g., graph- or text-based). In the following we describe each process.

The labelling process produced a relational database for each set of files. This is the main artefact of the dataset. In particular, we generated two SQLite databases (see dataset.ecore/ecore.db and dataset.genmymodel/genmymodel.db files, for Ecore and UML set of models, respectively). The schema of these databases includes two tables: the table models, which stores the name and reference of the model; and the table metadata, which stores the labels assigned to the model in JSON format. We use JSON to facilitate the integration of external tools with ModelSet. Listing 1 shows an example of JSON metadata. Providing a relational database allows the use of the command-line interface of SQLite to easily query the dataset and perform simple analysis. For instance, Listing 2 shows how to compute the top categories of for Ecore. In particular, the query joins the models and metadata tables to access the JSON document with the labels and then uses json_extract to the stored category[2].

In addition to the manually labelled metadata, for each model a set of statistics are also provided. They are gathered in an associated SQLite database (see dataset.ecore/analysis.db and dataset.genmymodel/analysis.db files, for Ecore and UML, respectively). The statistics include the number of elements of the model, the number of elements per type (e.g., the number of objects of type EClass, EReference, etc. in the case of Ecore) and the number of diagrams of each type in the case of UML. Implementation-wise, these statistics are automatically computed using the model analyser implemented as part of the MAR project (see https://github.com/mar-platform/mar).

The storing process collects all the original files serialized in XMI format in the raw-data folder (see repo-ecore-all and repo-genmymodel-umml subfolders for Ecore and UML, respectively).

[2]SQLite offers a set of functions to interact with JSON data stored in the database. This is very convenient to perform queries over the dataset. See https://www.sqlite.org/json1.html

Listing 2: Querying the dataset.

```
$ sqlite ecore.db
> select json_extract(md.json, '$.category[0]') as category,
        count(*) as total
     from models m join metadata md on m.id = md.id
     group by category
     order by total desc
     limit 7;

category      total
--------      -----
dummy          729
statemachine  392
petrinet      236
library       235
modelling     209
class-diagram 182
gpl           180
```

Listing 4: Path to text-based representation of the model shown in Fig. 1.

```
txt/repo-ecore-all/data/surli/spinefm/
    spinefm-eclipseplugins-root/spinefm-core/
        model/MetamodelSpineFM.ecore/MetamodelSpineFM.txt
```

Finally, during the rendering process all collected models are also stored in other formats suitable to facilitate the usage of the dataset to train ML algorithms. Currently, two formats are available: graph and text (see the so-called folders). The graph version renders the models into the JSON format supported by networkx[3], a popular graph library for Python. Using this JSON-based format it is easy to visualize the graphs for further analysis, and also load and transform them into other formats like COO, as required by PyTorch Geometric. On the other hand, the text version of the models stores the values of the string attributes, one per line. This is useful to easily implement encodings based on text like TF-IDF. Listing 3 shows the text encoding for the previous running example. It is worth noting that if the same term appears more than one (e.g., name), there will be multiple lines containing it. This is needed to weight the relative importance of each term in the models.

Listing 3: Text encoding of the model shown in Fig. 1.

```
spinefm
FMModel
FeatureModel
getStateFT
feature
getFeatureFromName
name
addFeature
name
...
```

To promote efficiency when accessing the models, graph- and text-based model representations are stored with the same relative path as the original model, but using the model file name as a folder to contain the new file. For instance, Listing 4 shows the text-based representation of the model file for the running example. Note that the path after the txt folder name matches with the path of the original model which is repo-ecore-all/data/surli/spinefm/spinefm-eclipseplugins-root/spinefm-core/model/MetamodelSpineFM.ecore.

[3]https://networkx.org/documentation/stable/reference/readwrite/json_graph.html

Listing 5: Using modelset-py.

```
1  import modelset.dataset as ds
2
3  MODELSET_HOME = '/path/to/modelset'
4  dataset = ds.load(MODELSET_HOME,
5                    modeltype='ecore',
6                    selected_analysis=['stats'])
```

Listing 6: Usage example of to_normalized_df method.

```
df = dataset.to_normalized_df(
      min_ocurrences_per_category = 5,
      languages = ['english'],
      remove_categories = ['petrinet',
                           'statemachine'])
```

2.3 Support Library

ModelSet can be used by accessing the SQLite database by means of some existing driver, however, this can be error-prone and relies on the structure of the database, which may change in the future. Therefore, we have developed a companion Python-based support library, named modelset-py, which is available at the PyPi repository[4] and can be easily installed via pip.

The main functionality provided by the modelset-py library is to load the dataset into memory (from the database) and to prepare the data in a manner that is suitable for using well-known Python libraries like Pandas, Scikit-Learn or PyTorch, among others.

Listing 5 shows how to load ModelSet using the library. Once the library is imported (line 1), it is important to load the dataset to a local folder, which must have been downloaded previously[5] (line 3 points to this folder). To load the dataset the load method receives (1) the location of ModelSet package, (2) the type of the models (the modeltype parameter can be ecore or uml), and, optionally, (3) the selected_analysis parameter, which currently accepts the stats value to indicate whether statistics about the models must be included. Note that the selected_analysis parameter may slow down the loading process, thereby its optionality.

The loaded dataset object contains a few methods to query the dataset. In particular, there are two main methods to convert the dataset into a Pandas dataframe: to_df and to_normalized_df. The former converts the complete dataset as it is; while the latter performs the conversion but only considering examples with a minimum number of examples (7 by default), written in the provided languages (English by default) and removing special categories (*dummy* and *unknown*[6]).

For instance, Listing 6 obtains a Pandas dataframe in which any model whose category has less than 5 models is removed, only models written in English are retrieved, and models whose category is petrinet or statemachine are removed. This dataframe provides a good starting point to manipulate the dataset, and to interact with other libraries. Thus, Listing 7 reuses the df variable and shows how to split the dataset into training and test.

[4]https://pypi.org/project/modelset-py/
[5]The lastest version of the ModelSet package can be downloaded from the releases page of the project: https://github.com/modelset/modelset-dataset/releases/.
[6]The *dummy* category represents incomplete models which has little value and *unknown* means that we could not label properly the model.

Listing 7: Creating training and test sets.

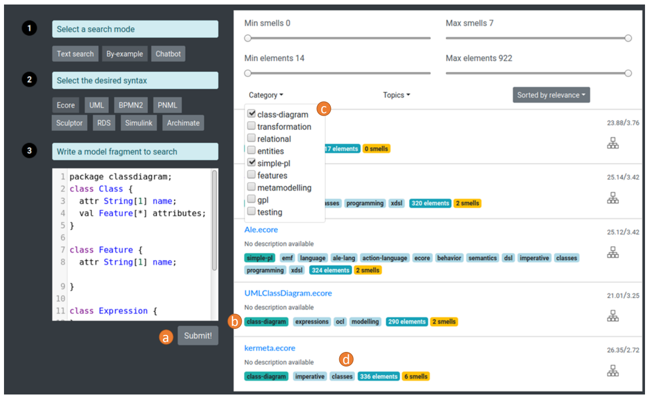
```
from sklearn.model_selection import train_test_split

ids = df['id']
labels = df['category']

train_X, test_X, train_y, test_y =
   train_test_split(ids, labels)
```

Listing 8: Recovering all models in the three available formats.

```
models = [dataset.model_file(id) for id in ids]
texts = [dataset.txt_file(id) for id in ids]
graphs = [dataset.graph_file(id) for id in ids]
```



```
package classdiagram;
class Class {
  attr String[1] name;
  val Feature[*] attributes;
}

class Feature {
  attr String[1] name;

}

class Expression {
```

✓ class-diagram
☐ transformation
☐ relational
☐ entities
✓ simple-pl
☐ features
☐ metamodelling
☐ gpl
☐ testing

23.88/3.76
17 elements 0 smells

25.14/3.42
sses programming xdsl 320 elements 2 smells

Ale.ecore 25.12/3.42
No description available
simple-pl emf language ale-lang action-language ecore behavior semantics dsl imperative classes programming xdsl 324 elements 2 smells

UMLClassDiagram.ecore 21.01/3.25
No description available
class-diagram expressions ocl modelling 290 elements 2 smells

kermeta.ecore 26.35/2.72
No description available
class-diagram imperative classes 336 elements 6 smells

Figure 3: Screenshot of MAR generating labels (from [11])

Finally, the loaded dataset also provides access to the XMI files, the text versions and the graph versions of a model given its id. Listing 8 reuses the variable dataset to illustrate how to obtain all the models of the dataset in the three available formats.

Altogether, we believe the library makes it easy to start an ML project which treats with models since it bridges the gap between the internal structure of the dataset and the Python ecosystem.

3 APPLICATIONS

ModelSet has been previously used to address classification tasks for Ecore and UML models. This type of task can be informally defined as follows: given an unseen model, identify is the most probable label (or labels if it is a multi-label task) for the model.

Although this problem has been addressed before the release of ModelSet, it was done only with small datasets [17]. The usage of ModelSet allows us to derive stronger conclusions, as described in recent work [15]. Moreover, the classification model [11] has been used in practice to implement a faceted search facility in the MAR search engine. Fig. 3 shows an example, in which the models retrieved as a response to a query are automatically provided a category (label **b**) and a set of tags (label **d**). Moreover, it is possible to refine the search using the categories (label **c**).

In the rest of the section we discuss other scenarios in which we believe that ModelSet can be useful. For the sake of completeness, for those scenarios which can be addressed using unsupervised learning techniques, we indicate whether the models provided by MAR can also be used as an equivalent dataset (containing a larger number of models, but not labelled).

Evaluation of clustering methods. ModelSet gives access to a large number of models which can be applied to empirically evaluating clustering methods. In this case, the labels can be interpreted as cluster identifiers and used as ground truth. This is particularly useful to perform a more robust evaluation of previously proposed clustering techniques like SAMOS [3] or [4].

Recommender systems. MAR provides large amounts of models of different types which may allow training neural models, which requires large amounts of models. ModelSet may also be used if one is interested in specific categories of models (e.g., a recommender system to model in the banking domain). Furthermore, the models of MAR have been already used to train and evaluate several recommender systems [1, 19].

Spurious model identification. ModelSet may be used to identify spurious models, which are models with a low quality level (e.g., partial or wrong models). This can be done by focusing on the models labelled as *dummy*.

Label-based stratified evaluation. ModelSet may be used to evaluate ML models using train-test-eval splitting in a stratified fashion using the labels. This is useful to avoid bias in performance estimation as there are several domains that contain much fewer models than others.

Train embeddings. ModelSet may be used to train embeddings of models of a specific category (i.e., for clone detection or recommender systems in a particular domain). For instance, the authors in [6] propose a doc2vec [10] approach over UML class diagrams extracted from code to recommend concepts within a model.

Usage of models in education. The collection of models in ModelSet gives the opportunity to use models for illustrating purposes in educational environments.

Statistical model analysis. ModelSet contains a good amount of samples that can be used to apply several statistical analyses and perform quality assurance. For instance, understanding common characteristics of model through statistical analysis [7, 8] or evaluating the realistic model generators [14].

Altogether, we believe that ModelSet provides a good opportunity to boost research activities related to Machine Learning and Model-Driven Engineering.

4 CONCLUSION AND FUTURE WORK

The application of ML algorithms to address tasks related to MDE is increasingly being researched. However, an important element which may hinder the evolution of this research line is the lack of sufficiently large datasets. In this paper, we have presented the current status of ModelSet, a large labelled dataset of software models composed of 5,466 Ecore meta-models and 5,120 UML models. We have described the structure of the dataset, how to start using the support library to create Python projects, and we have described several applications of the dataset.

As future work, we plan to continue enhancing ModelSet. To this end we aim at associating other types of labels, like textual descriptions of the models (e.g., to generate textual summaries). We also want to improve the modelset-py library with facilities like detection of duplicate models and better integration with other libraries like PyEcore or Scikit-Learn, among others. Also, to facilitate its use we want to allow the automatic downloading of the dataset when the library is first used. Finally, we are looking into ways to integrate the models provided by MAR with the Python library of ModelSet, so that it is possible to use them in unsupervised learning scenarios.

Acknowledgments. Work partially funded by the Spanish government: Ramón y Cajal 2017 grant (RYC-2017-237; co-funded by the European Social Fund) and the LOCOSS project (PID2020-114615RB-I00). José Antonio Hernández López enjoys a FPU grant funded by the Universidad de Murcia.

REFERENCES

[1] Lissette Almonte, Sara Pérez-Soler, Esther Guerra, Iván Cantador, and Juan de Lara. 2021. Automating the synthesis of recommender systems for modelling languages. In *Proceedings of the 14th ACM SIGPLAN International Conference on Software Language Engineering*. 22–35.

[2] Önder Babur. [n. d.]. A Labeled Ecore Metamodel Dataset for Domain Clustering. https://zenodo.org/record/2585456

[3] Önder Babur, Loek Cleophas, and Mark van den Brand. 2019. Metamodel Clone Detection with Samos. *Journal of Computer Languages* 51 (2019), 57–74.

[4] Francesco Basciani, Juri Di Rocco, Davide Di Ruscio, Ludovico Iovino, and Alfonso Pierantonio. 2016. Automated Clustering of Metamodel Repositories. In *Int. Conf. on advanced information systems engineering*. Springer, 342–358.

[5] Loli Burgueno, Jordi Cabot, Shuai Li, and Sébastien Gérard. 2022. A Generic Lstm Neural Network Architecture to Infer Heterogeneous Model Transformations. *Software and Systems Modeling* 21, 1 (2022), 139–156.

[6] Thibaut Capuano, Houari Sahraoui, Benoit Frenay, and Benoit Vanderose. [n. d.]. Learning from Code Repositories to Recommend Model Classes. ([n. d.]).

[7] Juri Di Rocco, Davide Di Ruscio, Ludovico Iovino, and Alfonso Pierantonio. 2014. Mining metrics for understanding metamodel characteristics. In *Proceedings of the 6th International Workshop on Modeling in Software Engineering*. 55–60.

[8] Juri Di Rocco, Davide Di Ruscio, Ludovico Iovino, and Alfonso Pierantonio. 2015. Mining correlations of ATL model transformation and metamodel metrics. In *2015 IEEE/ACM 7th International Workshop on Modeling in Software Engineering*. IEEE, 54–59.

[9] Juri Di Rocco, Claudio Di Sipio, Davide Di Ruscio, and Phuong T Nguyen. 2021. A GNN-based Recommender System to Assist the Specification of Metamodels and Models. In *Int. Conf. on Model Driven Engineering Languages and Systems (MODELS)*. 70–81.

[10] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*. PMLR, 1188–1196.

[11] José Antonio Hernández López, Javier Luis Cánovas Izquierdo, and Jesús Sánchez Cuadrado. 2021. Modelset: a Dataset for Machine Learning in Model-driven Engineering. *Software and Systems Modeling* (2021), 1–20.

[12] José Antonio Hernández López and Jesús Sánchez Cuadrado. 2020. Mar: a Structure-based Search Engine for Models. In *Int. Conf. on model driven engineering languages and systems*. 57–67.

[13] José Antonio Hernández López and Jesús Sánchez Cuadrado. 2021. An Efficient and Scalable Search Engine for Models. *Software and Systems Modeling* (2021), 1–23.

[14] José Antonio Hernández López and Jesús Sánchez Cuadrado. 2021. Towards the Characterization of Realistic Model Generators Using Graph Neural Networks. In *Int. Conf. on Model Driven Engineering Languages and Systems*. 58–69.

[15] José Antonio Hernández López, Riccardo Rubei, Jesús Sánchez Cuadrado, and Davide Di Ruscio. 2022. achine learning methods for model classification: A comparative study. In *Int. Conf. on model driven engineering languages and systems*. To appear.

[16] Josh GM Mengerink, Jeroen Noten, and Alexander Serebrenik. 2019. Empowering Ocl Research: a Large-scale Corpus of Open-source Data from GitHub. *Empirical Software Engineering* 24, 3 (2019), 1574–1609.

[17] Phuong T Nguyen, Juri Di Rocco, Ludovico Iovino, Davide Di Ruscio, and Alfonso Pierantonio. 2021. Evaluation of a Machine Learning Classifier for Metamodels. *Software and Systems Modeling* 20, 6 (2021), 1797–1821.

[18] Gregorio Robles, Truong Ho-Quang, Regina Hebig, Michel RV Chaudron, and Miguel Angel Fernandez. 2017. An Extensive Dataset of Uml Models in GitHub. In *Int. Conf. on Mining Software Repositories*. IEEE, 519–522.

[19] Martin Weyssow, Houari Sahraoui, and Eugene Syriani. 2022. Recommending Metamodel Concepts during Modeling Activities with Pre-trained Language Models. *Software and Systems Modeling* (2022), 1–19.