Steps towards smart modeling tools

Jesús Sánchez Cuadrado (jesusc@um.es)
Universidad de Murcia

(joint work with José Antonio Hernández)

Context

Smart modelling tools

Modelling tools with features to enhance the modellers productivity

Types of smart features

- Provide hints
  - Recommendation. Gives developer hints about how to proceed with her work.
- Automate costly and error-prone tasks automatically
  - Model generation. Could be done manually, but it would be very costly.
- Make approaches scalable (in terms of human resources)
  - Classification. Provide labels automatically for thousands of resources.

Context

https://www.youtube.com/watch?v=Lm_1PHPPZYQ

Context

https://www.youtube.com/watch?v=Lm_1PHPPZYQ

- This is nice, and looks useful, right?
- The question is: **what do we need to get there?**

This talk

Our journey

- A search engine for models: MAR
- A labelled dataset of models: MODELSET
- Applications
  - Using web services
  - Classification
  - Model generation
- New stuff
  - Large scale exploration of MDE artefacts in GitHub
  - Learning the modeling vocabulary
  - Recommender systems

MAR: A search engine for models

Part I: Collecting and processing models

Motivation

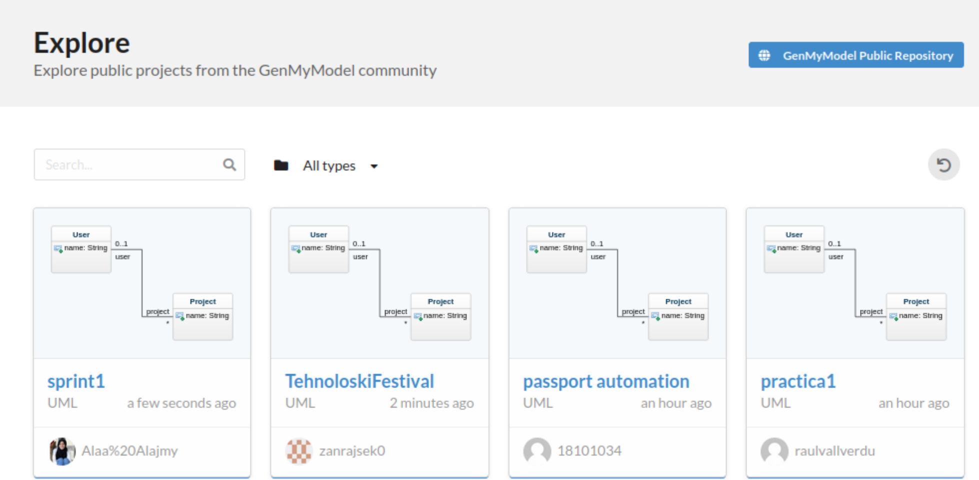
- Models are the primary artifacts in MDE
- Model repositories make models available for reuse and learning
- In practice, models are not typically reused.
- Why? Maybe because it is not easy to find models
  - Limited or no search mechanisms
  - Many models are stored in source code repositories
  - Which are the relevant places to find models?

Motivation

Example. Searching for Ecore meta-models about state machines

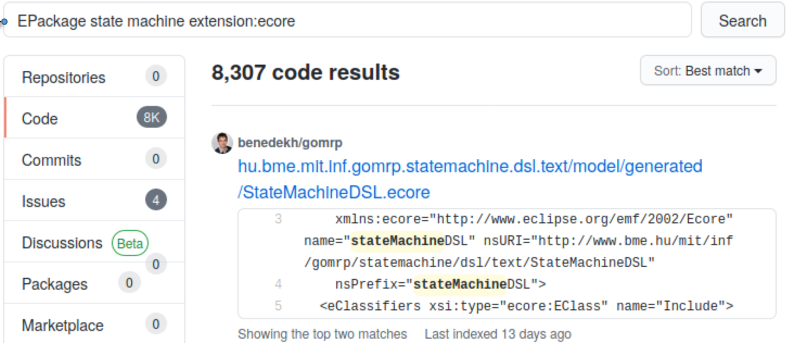
- Models available in diverse repositories like GenMyModel, GitHub, AtlanMod Zoo, etc.
- What can we do to find interesting models?

Motivation


Explore
Explore public projects from the GenMyModel community
GenMyModel Public Repository
Search...
All types
User
name: String
0..1
user
project
Project
name: String
sprint1
UML
a few seconds ago
Alaa%20Alajmy
TehnoloskiFestival
UML
2 minutes ago
zanrajsek0
passport automation
UML
an hour ago
18101034
practica1
UML
an hour ago
raulvallverdu

Motivation

Query string



GitHub search

Motivation

AtlanMod
Meta-model
Zoo

Browser search

Finite State Machine 1.0

date : 2006/07/14

Domain :

Description : This metamodel describes the concepts of a finite state machine.

See : http://repository.escherinstitute.org/Plone/tools/suites/mic/great/

Authors : Youssef Srour (Srour.youssef_NOSPAM <AT> gmail.com)

- source file

state machine

82 EQN 1.0
83 EXPRESS 0.1
84 EXPRESS 0.2
85 EclipseLaunchConfigu
1.0
86 EclipsePlugIn 0.1
87 Edas 1.0
88 Ekaw 1.0
89 Extended UML Core P
90 Family 1.1
91 FeatureDiagrams 1.0
92 Finite Automaton 1.0
93 Finite State Machine 1
94 Flat Signal Flow 1.0

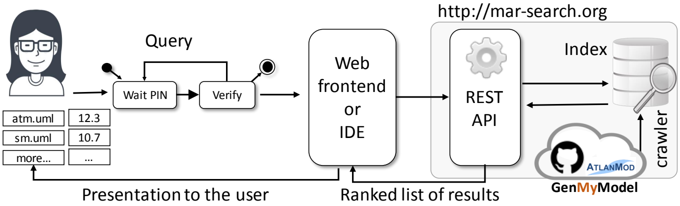
Motivation

Problem

Finding interesting models is a time consuming activity.

- Need to find out where are the models
- Need to search in several places
- Limited search facilities
- Results are not ranked
- Inspecting results is complicated
- No guarantee that the obtained models are valid

Solution



- Query by example and keyword-based queries
- Faceted search and filtering
- REST API + Web
- Inverted index
- Scoring algorithm
- Generic search
- Crawler for GitHub, GenMyModel and AtlanMod Zoo

Demo

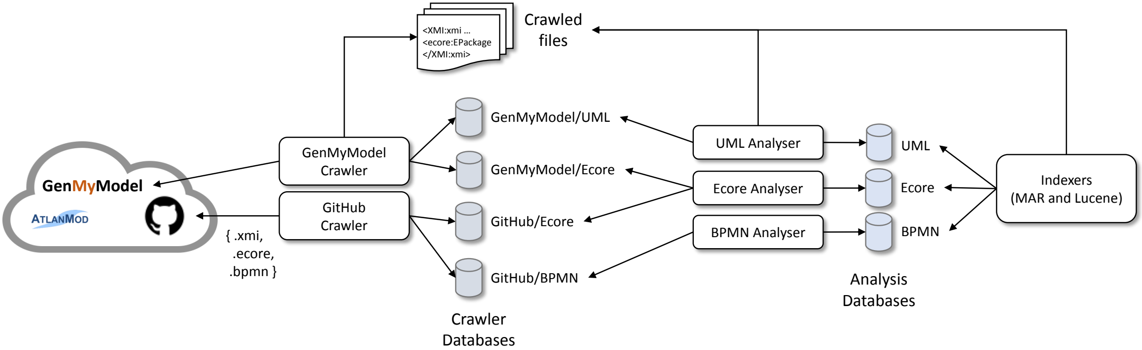
http://mar-search.org

Architecture

Main components

- Crawlers – Discover and collect models
- Analysers – Check validity and compute stats and quality metrics
- Model pre-processing pipeline
- Index
- Query processor
- Scoring algorithm

Crawling and analysis



Crawlers

- Which are sources of models?
- How to extract models from them?
  - GitHub - Rate limit issues
  - GenMyModel - Now public, before Selenium
  - AtlanMod - Webscrapping
- Which metadata is available?
  - Popularity (stars, forks)
  - Creation and update dates
  - Author
  - Topics

Analysers

Phases

1. Open the file (it may blow up the heap, crash, etc)
2. Validate (is it structurally correct?)
3. Analyse quality (e.g., detect smells)
4. Compute statistics (e.g., number of elements)

More difficult than it seems!

- Create an analysis server
- Launch it on demand and communicate via RPC
- If it crashes, the model is invalid

Available models

	Source	Crawled	Duplicates	Failed	Indexed	Observations
Ecore	GitHub	67,322	46,199	341	20,782	
	GenMyModel	3,987	3	27	3,957	
	AtlanMod	304	1	4	299	
UML	GitHub	53,082	7,282	1,699	44,101	Eclipse UML meta-model.
	GenMyModel	352,216	143	23,836	328,237	
BPMN	GenMyModel	21,285	0	200	21,085	EMF BPMN2 meta-model[1].
Archimate	GitHub	496	77	106	313	Archi meta-model[2].
PNML	GitHub	3,291	1,576	1,044	671	PNML framework[3].
Sculptor	GitHub	188	88	0	88	
RDS	GenMyModel	91,411	108	515	90,788	Entity/relationship diagrams.
Simulink	Dataset	200	0	0	200	Massif meta-model
Total	-	593,582	55,477	27,972	510,321	-

[1] https://www.omg.org/spec/BPMN/2.0/
[2] https://github.com/archi-contribs/eclipse-update-site
[3] https://pnml.lip6.fr/

How to query

Keywords

1. Type a few keywords
2. e.g., state machine
3. Simple to implement, less precise
4. Simple to use

Query-by-example

- Provide an example or model fragment (or a complete model!)
- Find models which have partial matches
- More difficult to implement, more precise

REST API

- Described as an OpenAPI spec: http://mar-search.org/openapi
- /search/
  - By keywords
  - By example
- /metadata/
  - Metadata for a given stored model
- /analysis/
  - Smells
  - Metrics
- /ml/
  - Classification

REST APIs – Search with keywords

Example

```
curl -X POST -d "petrinet␣place␣color" http://mar-search.org/search/keyword?max=2
```

REST APIs – Search by example

Example

```
curl -X POST -d "@tournament.ecore" http://mar-search.org/search/example?type=ecore&max
     =100

[
   {
    "id":"github:ecore:/data/Gullskatten/sirius-soccer/no.ntnu.soccer.model/model/soccer.
        ecore"
    "name":"soccer.ecore",
    "modelType":"ecore",
    "url":"https://raw.githubusercontent.com/Gullskatten/sirius-soccer/00
        f8e390fa72a1a85e4d7dd5846852ac41c1c158/no.ntnu.soccer.model/model/soccer.ecore",
    "score":211.54585423692313,
    "metadata":{"smells":{"OverLoadedClassSmell":1},
     "topics":["sirius","␣intellij","␣kaggle","␣soccer"],
     "numElements":115,
     "explicitName":null,"description":null,"category":null},
   }
...
]
```

REST APIs – Smells

- Ecore smells
- http://mar-search.org/analysis/smells

Example

```
$ curl -X GET -d "@relational.ecore" http://mar-search.org/analysis/smells

{
 "IrrelevantClassSmell" : ["//NamedElement"]
}
```

ModelSet

Part II: Building a dataset

Motivation

Apply Machine Learning to Modelling

- We need datasets.
- For some types of problems, datasets need to be labelled.

- In practice: few datasets
- Labelled datasets: small (e.g., 555 models[4])
- Non-labelled datasets: can be large, but not curated (e.g., Lindholmen[5])

[4] https://zenodo.org/record/2585456#.YM5ziSbtb0o
[5] http://models-db.com/oss/

Challenge

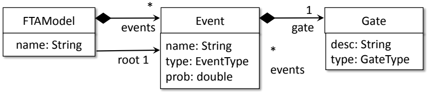
Our goal

Build a large, labelled dataset of software models.

- Inspecting and labelling models is hard.
- Requires modelling expertise and domain knowledge.
- We need to annotate these models, one by one.
- Spend time to figure out a proper label

Challenge – Example

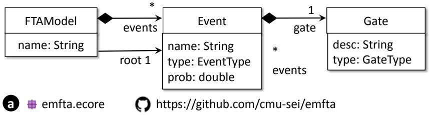
- What is FTA?

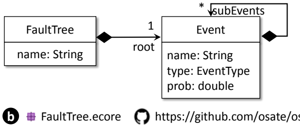


a emfta.ecore https://github.com/cmu-sei/emfta

Challenge – Example

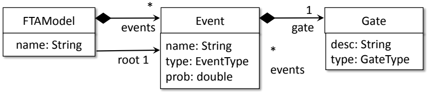
- What is FTA?
- Perhaps you will find out better if you see FaultTree
- But what is a Fault Tree?



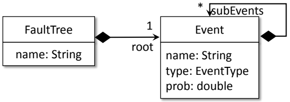


Challenge – Example

- What is FTA?
- Perhaps you will find out better if you see FaultTree
- But what is a Fault Tree?
  - We need context (similar meta-models, GitHub links, look up in Wikipedia).
  - We want to copy-paste once we understand.



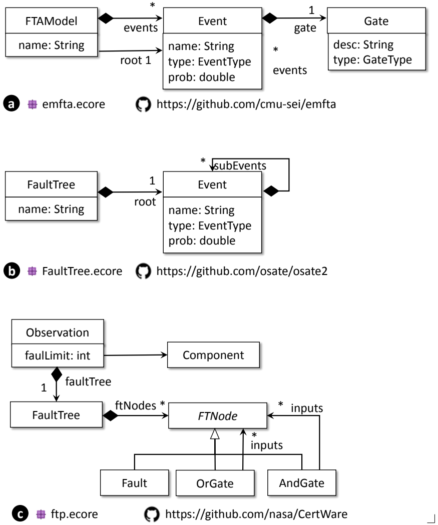
a emfta.ecore https://github.com/cmu-sei/emfta



b FaultTree.ecore https://github.com/osate/osate2

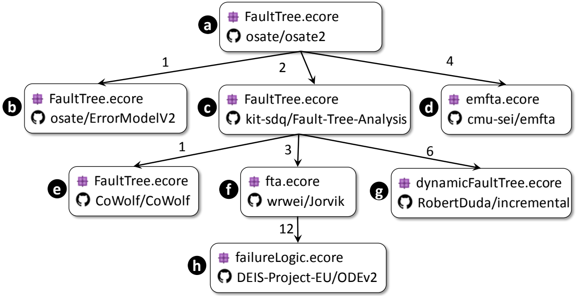
Challenge – Example

- What is FTA?
- Perhaps you will find out better if you see FaultTree
- But what is a Fault Tree?
  - We need context (similar meta-models, GitHub links, look up in Wikipedia).
  - We want to copy-paste once we understand.
- category: fault-tree
- tags: safety, hazard

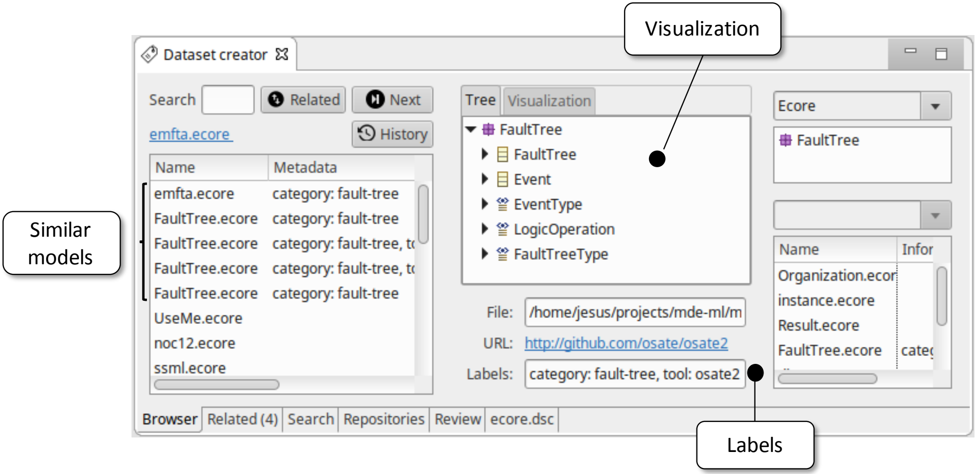


Labelling method

- Need to semi-automate the labelling process
- Interactive labelling algorithm
  1. Dynamic clustering (kind of interactive DB-SCAN)
- Steps:
  1. Pick an unlabelled model *m*
  2. Use MAR to search for similar models
  3. Inspect and label these models together
     - In the background, search models similar to the ones just labelled
  4. Keep labelling the same “streak” of models or go to step 1



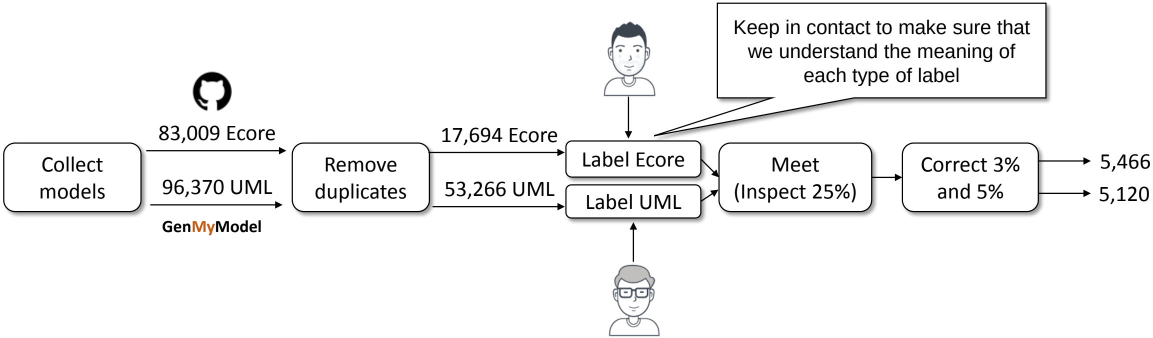
Dataset creator



ModelSet

- 5,466 Ecore models – from GitHub
- 5,120 UML models – from GenMyModel
- 28,719 labels
- Category, tags, purpose, notation, tool
- See http://modelset.github.io.

Process

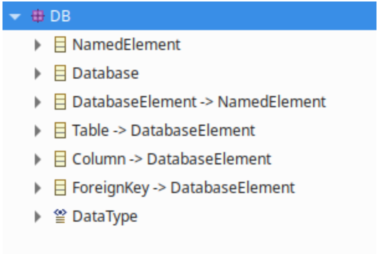


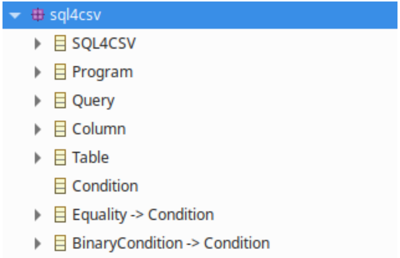
Keep in contact to make sure that we understand the meaning of each type of label
83,009 Ecore
96,370 UML
GenMyModel
Collect models
Remove duplicates
17,694 Ecore
53,266 UML
Label Ecore
Label UML
Meet (Inspect 25%)
Correct 3% and 5%
5,466
5,120

ModelSet – Types of labels

- **Category.** The application domain a model (e.g., petri net)
- **Tags.** Additional insights about a model (e.g., coloured)
- **Purpose.** Is it used for experiments, teaching, etc.?
- **Notation.** Is there an associated notation? (e.g., Sirius, Xtext, etc)
- **Tool.** Is the model used as part of a tool? (e.g., CertWare)
- **Confidence.** Are we sure of the labels? Values can be high, medium or low.

Some examples





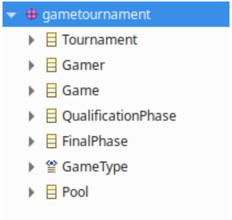
a

category: relational
tags: { ddl }

b

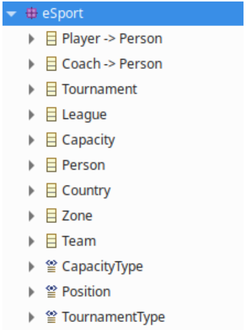
category: relational
tags: { dml }

Some examples



a

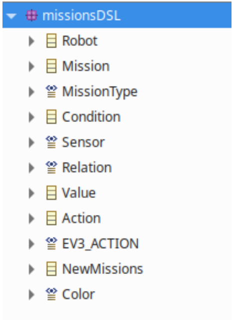
category: tournament
tags: { domain-model }



b

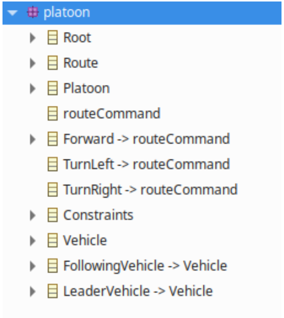
category: tournament
tags: { domain-model, esports }
notation:textual

Some examples



a

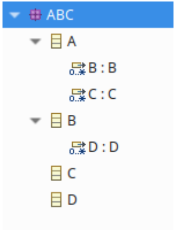
category: robots
tags: { missions, mindstorms }
notation: textual
purpose: assignment



b

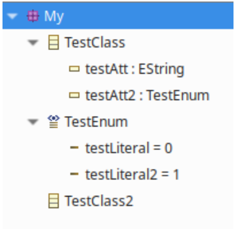
category: robots
tags: { vehicle-coordination }
notation:textual
purpose: assignment

Some examples



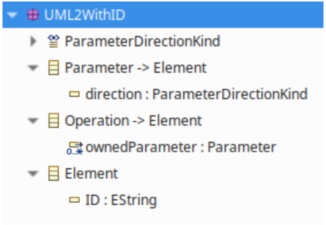
a

category: dummy



b

category: dummy



c

category: dummy

Python library: `modelset-py`

- Automatic downloading of the dataset
- Loading the dataset into a Pandas data frame
- Loading models as text files or graphs
- Computation of duplicates

Applications

Part III: What to do with this stuff?

Applications

Available resources

- Raw models (about 500,000)
- Labelled models (about 10,000)
- Services

Now what?

Applications

- Using services
  - Enhancing modelling tools
  - Avoid re-inventing the wheel
- Using the dataset
  - Category, tags inference (classification)
  - Detecting dummy models (classification)
  - Build embeddings
  - Stratified k-fold
- Using the raw models
  - Recommendation
  - Model analytics

Enhancing modelling tools with services

Example – Enhancing modelling tools

Scenario: Reuse

A developer is creating a DSL in Xtext. It would be desirable not to start from scratch. How one could find similar DSLs?

Example – Enhancing modelling tools

Scenario: Reuse

A developer is creating a DSL in Xtext. It would be desirable not to start from scratch. How one could find similar DSLs?

- See the abstract syntax of the DSL as its interface
- Index Xtext grammars using its abstract syntax
- Search by example

Example – Enhancing modelling tools

- Easily integrated in an Eclipse plug-in

```
Resource r = /* get an EMF resource somehow */
ByteArrayOutputStream bos = new ByteArrayOutputStream();
r.save(bos, null);

HttpResponse<JsonNode> jsonResponse = Unirest.post("http://mar-search.org/search/example?
     type=" + searchType + "&max="+max)
  .multiPartContent()
  .accept("application/json")
  .field("uploaded_file", bos.toString().getBytes(), "model.ecore")
  .asJson();

// [{name: 'relational.xtext', url: 'http://github...', score: 1523.3}, ...]
```

Example – Experiments

Scenario

A researcher is investigating about automatic fixing of meta-models.

- Everything typically starts from scratch
- Manually implement a catalogue of smells
- Need models for doing experiments

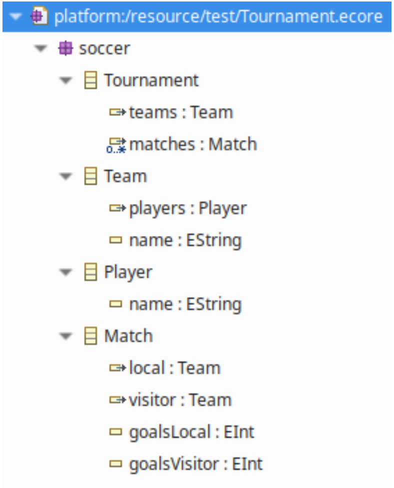
Resources

- Use smells API
- Use the models from ModelSet or MAR

Classification

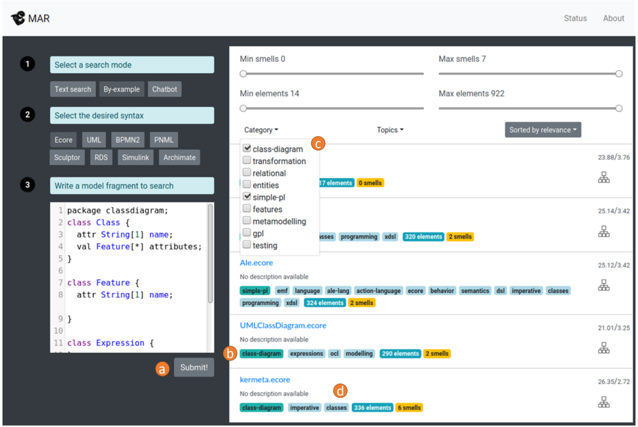
Task

- Given a model, predict its label.
  - For example: *tournament*
- There are variations (binary, multi-label, etc.)
- When is this useful?



Example

- Classifiers
  - Dummy (binary classifier)
  - Category (multi-class)
  - Tags (multi-label, multi-class)
- Integration in MAR
  - http://mar-search.org
  - Support faceted search
  - Infer labels for thousands of unknown models
- Provide facilities for exploring large amounts of models

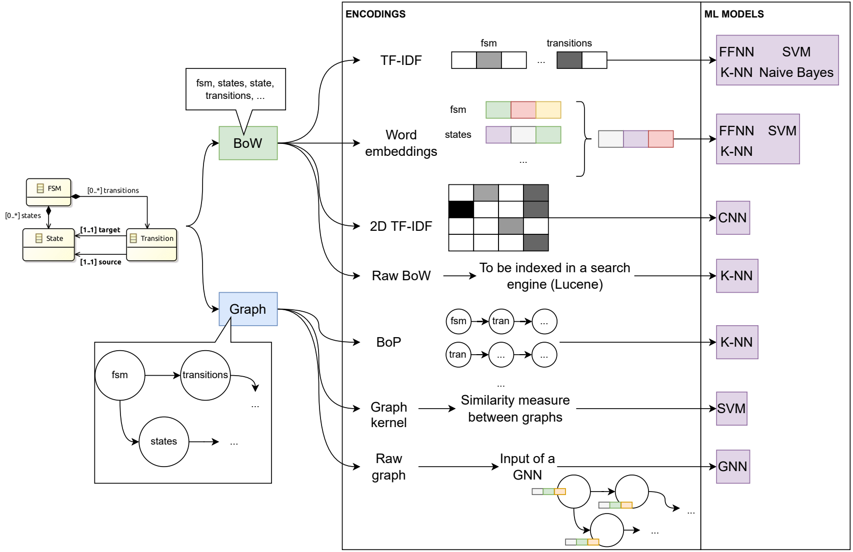


Evaluate classifiers

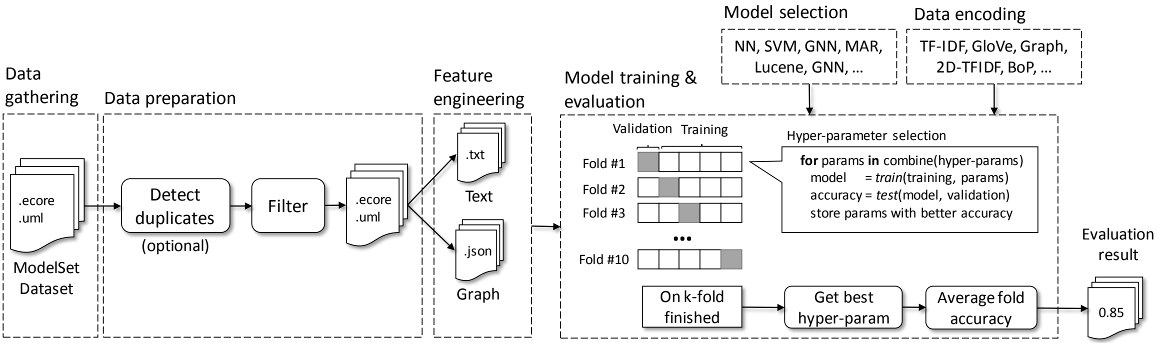
- Choose a ML model
- Choose an encoding
  - We can't use a software model as input!
  - Typically simple structures (e.g., vectors)

- Research questions:
  - **Which is the best combination of ML model and encoding?**
  - **What happens if there are duplicates?**

Models and encodings



Methodology



Results

	Model	Encoding	B. accuracy	Best hyper.
1	FFNN	TF-IDF	0.898511	hidden size = 200
2	SVM	TF-IDF	0.895735	kernel = linear, C = 100
3	GNN	Raw graph	0.888875	–
4	CNN	2D TF-IDF	0.885606	–
5	k-NN	Lucene (BoW)	0.882907	k = 1
6	SVM	WordE	0.880327	kernel = linear, C = 10
7	k-NN	TF-IDF	0.879817	k = 1
8	FFNN	WordE	0.877892	hidden size = 50
9	k-NN	MAR (BoP)	0.873174	k = 1
10	k-NN	WordE	0.852933	k = 1
11	GNB	TF-IDF	0.809371	–
12	SVM	Graph kernel	0.792745	C = 0.1
13	CNB	TF-IDF	0.775844	α = 0.1
14	MNB	TF-IDF	0.754884	α = 0.1

Table 4: Results for Ecore, with duplicate models

	Model	Encoding	B. Accuracy	Best hyper.
1	FFNN	TF-IDF	0.824972	hidden size = 150
2	SVM	TF-IDF	0.815609	kernel = linear, C = 10
3	GNN	Raw graph	0.807656	–
4	SVM	WordE	0.786988	kernel = linear, C = 100
5	k–NN	Lucene (BoW)	0.786793	k = 1
6	CNN	2D TF-IDF	0.778440	–
7	FFNN	WordE	0.777899	hidden size = 150
8	k–NN	MAR (BoP)	0.775339	k = 3
9	k–NN	TFIDF	0.764505	k = 1
10	CNB	TFIDF	0.733788	α = 0.1
11	k–NN	WordE	0.723883	k = 1
12	MNB	TF-IDF	0.716409	α = 0.1
13	GNB	TF-IDF	0.607369	–
14	SVM	Graph kernel	0.593098	C = 0.1

Table 5: Results for Ecore, removing duplicate models

	Model	Encoding	B. Accuracy	Best hyper.
1	SVM	WordE	0.873906	kernel = linear, C = 10
2	FFNN	WordE	0.872578	hidden size = 150
3	SVM	TF-IDF	0.870490	kernel = linear, C = 100
4	FFNN	TF-IDF	0.864192	hidden size = 100
5	k–NN	MAR (BoP)	0.859487	k = 1
6	k–NN	WordE	0.849309	k = 1
7	k–NN	TF-IDF	0.848727	k = 1
8	GNN	Raw graph	0.843559	–
9	GNB	TF-IDF	0.798754	–
10	SVM	Graph kernel	0.769627	C = 0.1
11	CNB	TF-IDF	0.760579	α = 0.1
12	MNB	TF-IDF	0.745817	α = 0.1

Table 6: Results for UML, with duplicate models

	Model	Encoding	B. Accuracy	Best hyper.
1	FFNN	WordE	0.775893	hidden size = 150
2	FFNN	TF-IDF	0.758389	hidden size = 50
3	SVM	WordE	0.756125	kernel = rbf, C = 100
4	SVM	TF-IDF	0.744753	kernel = linear, C = 10
5	k–NN	MAR (BoP)	0.716452	k = 3
6	GNN	Raw graph	0.713418	–
7	CNB	TF-IDF	0.703389	α = 0.1
8	k–NN	WordE	0.694383	k = 1
9	k–NN	TF-IDF	0.686937	k = 1
10	GNB	TF-IDF	0.630084	–
11	MNB	TF-IDF	0.628251	α = 0.1
12	SVM	Graph kernel	0.530019	C = 0.1

Table 7: Results for UML, removing duplicate models

Lessons learned

- FFNN and SVM are the best models.
- Lightweight methods (search engines + K-NN) are competitive.
- Deep learning models perform worse than simpler models.
- Word embeddings work well in UML but not in Ecore.
- The structure of the models is not relevant in this task.
- The performance of all ML models is reduced when (quasi-)duplicates models are removed.

Example – Classifier for categories

```python
import modelset as ms
import pandas as pd

dataset = ms.load('..', modeltype = 'ecore')
df = dataset.to_normalized_df(min_ocurrences_per_category = 7, languages = ['english'])
```

	category	tags	language	id
2661	visualization	graph	english	repo-ecore-all/data/MDEGroup/QMM/validation-su...
1029	statemachine	behaviour	english	repo-ecore-all/data/silverspy/DSL_TP/fr.ut2j.m...
331	library	domainmodel	english	repo-ecore-all/data/prayasb/org.eclipse.emf.te...
3666	behaviourmodelling	statemachine|activities|behaviour	english	repo-ecore-all/data/tue-mdse/ocl-dataset/datas...
4415	simple-pl	imperative|expressions|programming	english	repo-ecore-all/data/Alexandra93/DT/dt.workflow...
324	library	domainmodel	english	repo-ecore-all/data/eclipse/emf/tests/org.ecli...
156	petrinet	behaviour	english	repo-ecore-all/data/tue-mdse/ocl-dataset/datas...
4319	tournament	domainmodel	english	repo-ecore-all/data/dlitvinov/FastEMFStore.oth...
1979	modelling	biology	english	repo-ecore-all/data/rodriguez-facundo/model/ge...
570	families	university|domainmodel	english	repo-ecore-all/data/MDEGroup/QMM/validation-su...

Example – Classifier for categories

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score

all_id = list(df['id'])
all_labels = list(df['category'])

list_train, list_test, y_train, y_test = train_test_split(all_id, all_labels,
    stratify= all_labels, test_size=0.3, random_state=42)

train_corpus = [dataset.as_txt(id_) for id_ in list_train]
test_corpus = [dataset.as_txt(id_) for id_ in list_test]
```

Example – Classifier for categories

```
# Encode as vectors using TF/IDF
vectorizer = TfidfVectorizer(stop_words = None,
 tokenizer = ms.simple_tokenizer, min_df = 2)
X_train = vectorizer.fit_transform(train_corpus)
X_test = vectorizer.transform(test_corpus)

# Train with 100 neurons
n = 100
clf = MLPClassifier(random_state=1,hidden_layer_sizes = (n,), max_iter=1000).fit(X_train,
     y_train)
y_pred = clf.predict(X_test)
score = accuracy_score(y_test, y_pred)
```

Model generation

Model generators

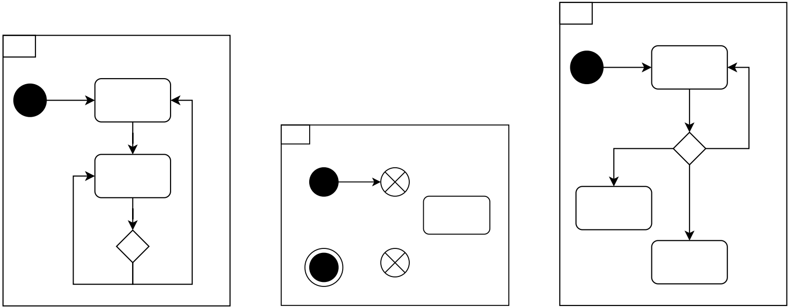
- What is a model generator?
  - A tool that automatically generate software models
  - The models conform to some meta-model and satisfy constraints
  - This is a very hard problem
- When it is useful?
  - Benchmarking
  - Test case generation
  - Overcome intellectual property issues

Properties of model generators

Properties

- **Consistency.** The generated models conform to the meta-model and respect the domain constraints (e.g., OCL constraints).
- **Diversity.** The models contains a wide range of shapes.
- **Scalability.** Ability to produce non-trivial time in reasonable time.
- **Realism.** The generated models cannot be distinguished from real ones.
    - **Structurally realistic.** Look at the typed graph structure (ignore attribute values).

Realism



(a) Model extracted from GitHub

(b) Model generated using VIATRA generator

(c) Model generated using M2

Figure: Example of real and synthetic models.

Realistic model generator

Question

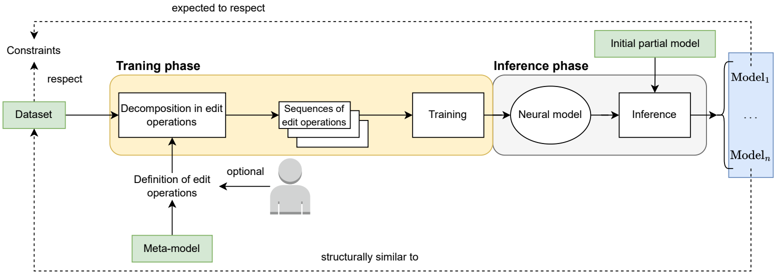
Is it possible to build a model generator focused on the "realism" property which also respect the other properties as much as possible?[a]

[a] We focus on the structurally realistic property

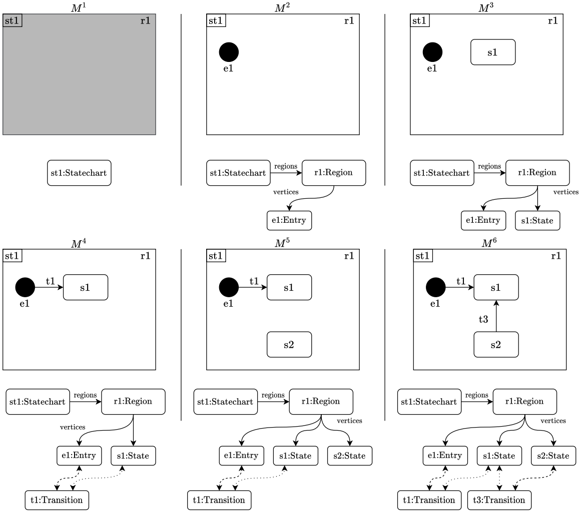
Approach

Two key ideas

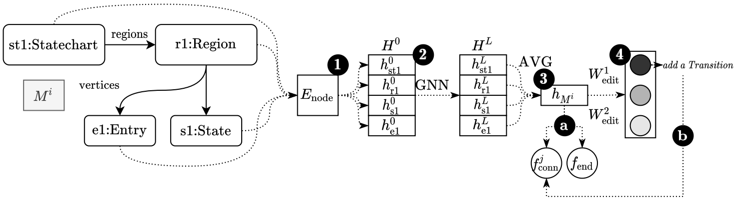
- Use of **edit operations** to decompose a model.
- Use of a **generative model** to learn how to build realistic models by iteratively applying edit operations.

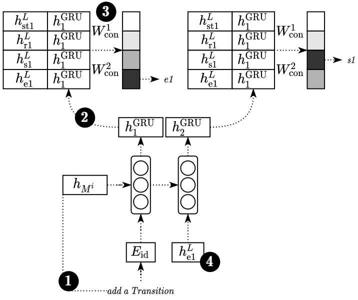


Edit operations



Architecture





Main results

- Tool available (M2) at: https://github.com/Antolin1/M2
- Consistency: Not fully consistent but M2 is almost consistent.
- Novelty: High number novel and unique models.
- Diversity: M2 is as diverse as the dataset from which it is trained.
- Realism: M2 generates realistic models, imitating the structure of the models in the dataset.
- Scalable: Linear scalability (but at the cost of training phase)

New stuff

Part IV: Improvements

Large scale exploration of MDE artefacts in GitHub

Motivation

- MAR collects models with crawlers
- It is possible to collect other artefacts (grammars, transformations)
- We can even search (e.g., an Xtext file based on its metamodel)
- **Can we exploit the relationships between artefacts?**
  - Answer questions of different type: is there reuse? are technologies combined? are MDE projects of good quality?
  - Learn how MDE projects are organised
  - As a side effect, build better modeling tools

Challenges

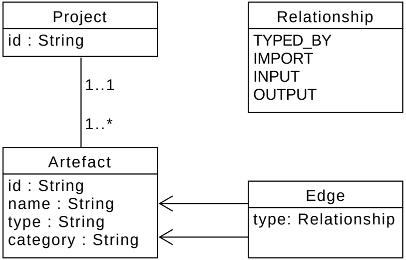
Main challenge

We need to recover a lot of information which is only implicitly described.

- Plethora of technologies
- Loose integration between them (no build system)
- Static vs. dynamic languages
- Broken files
- How to organize the recovered information?

Working solution

- Implement an “inspector” for each type of file
  - Many times we need heuristics
  - Each inspector generates a “mini-graph” per file
- All mini-graphs are merged into a very large graph



Demo

MDE Project exploration environment

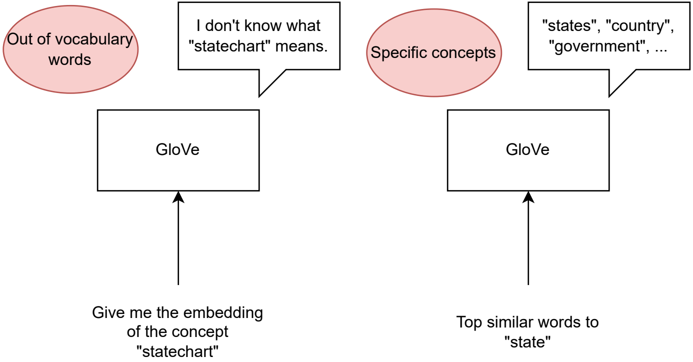
Learning the modeling vocabulary

Motivation

- We need to encode models to vectors to apply ML.
- One way is to use TF-IDF. However, it is not efficient as models are represented by high-dimensional vectors (thousands of dimensions).
- One alternative is to use word embeddings. The idea is to map a word to a low-dimensional vector (up to 300 dimensions).
- The common approach is to use pre-trained word embeddings (GloVe, Word2Vec). Trained by well-known institutions (Stanford, Google) with an extensive corpora of general texts.
- They work well in UML but not in the meta-modelling domain (it struggles for instance in meta-model classification).

Motivation

- Let’s take GloVe word embeddings. It was trained with an extensive corpus of general text by Stanford.



Word2Vec4MDE

- We take a corpus of modelling texts (SoSyM, MODELS, etc).
- We train a Skip-gram model with that corpora to get Word2Vec4MDE.
- These embeddings outperform GloVe and Word2Vec in several meta-modelling tasks:
  1. Meta-model classification
  2. Meta-model clustering
  3. Concept recommendation in the meta-model domain

Recommender systems

Graphical modelling assistant

- The technical problem can be decomposed in two subproblems:
  - **Recommendation of attributes** (relatively easy problem)
  - **Recommendation of edit opertions** (difficult problem)
- What about its integration with editors?
- On-going work.

Recommendation of attributes

- Given edit operations that generate new models elements. Infer attribute values of these new elements based on the context.
- Video https://www.youtube.com/watch?v=Lm_1PHPPZYQ

Recommendation of edit operations

- Given a partial model, what is the most likely edit operation that the user will apply?
- This problem is not trivial as there are exponential number of ways to build a model.
- This makes the training phase difficult to perform.
- The complexity of this problem is similar to the graph generation problem. There is a subfield in Machine Learning that tries to deal with this problem.

Conclusions

Conclusions

Smart modelling tools: are we there yet?

NO!

Conclusions – Further challenges

- Datasets
  - We need more and larger datasets
  - Annotations inside the model
  - Textual summaries of models
  - What about the quality of the models?
  - Comparing to SE, there is less documentation about models
- Benchmarks
  - Define tasks and goals precisely
- Tool integration
  - How to integrate smart features in practice
  - Evaluations with users

Conclusions – Applications

- Automatic model modularity
- Learning to generate realistic models
- Recommender systems
- Learning to rank
- Model summarization
- Architecture recovery of MDE projects
- Model clone detection
- Empirical studies

Resources

- Collecting and searching models:
  - José Antonio Hernández, Jesús Sánchez Cuadrado.
    MAR: A structure-based search engine for models.
    MoDELS'20.
  - José Antonio Hernández, Jesús Sánchez Cuadrado.
    An efficient and scalable search engine for models.
    SoSyM.
  - http://mar-search.org
- Datasets:
  - José Antonio Hernández, Javier Luis Cánovas Izquierdo, Jesús Sánchez Cuadrado.
    ModelSet: A Dataset for Machine Learning in Model-Driven Engineering.
    SoSyM.
  - http://modelset.github.io

Resources

- Applications:
  - José Antonio Hernández, Jesús Sánchez Cuadrado.
    Generating structurally realistic models with deep autoregressive networks.
    IEEE TSE.
  - http://github.com/antolin1/m2
  - José Antonio Hernández, Jesús Sánchez Cuadrado.
    Towards the Characterization of Realistic Model Generators using Graph Neural Networks.
    MoDELS'21.
  - José Antonio Hernández, Riccardo Rubei, Jesús Sánchez Cuadrado, Davide Di Ruscio.
    Machine learning methods for model classification: a comparative study.
    MoDELS'22.

Thanks for your attention! ❤

Any questions?

MODELS & LANGUAGES LAB

http://models-lab.github.io

joseantonio.hernandez6@um.es
@antolin_hl
https://github.com/Antolin1

jesusc@um.es
@sanchezcuadrado
http://github.com/jesusc