COMODI: A Tool for Model Comparison and Diversity Analysis

Adel Ferdjoukh

University of Nantes & LS2N, France
adel.ferdjoukh@univ-nantes.fr

Abstract. Models play the leading role in Model Driven Engineering (MDE). They are used in many contexts: validate Domain Specific Modelling Languages (DMSL), execute a Model Transformation or run an eXecutable DSML. For more usefulness and efficiency, all the previous operations require a large amount of models. Models are either hand-crafted or automatically generated. In both cases, they have to be as different as possible, and they must cover a large scope of their domain. In this paper, we propose COMODI, a tool that analyses sets of models, and ensures the good coverage of the space of possible solutions. COMODI performs model comparison using well-known or custom distance functions. It produces statistics and graphical visualisations, for evaluating the diversity of models.

1 Introduction

Models play the key role in Model Driven Engineering (MDE). They are widely used for many purpose. For example, we can use models in order to validate meta-models and modelling languages, execute and test a model transformation, or run and check the behaviour of an eXecutable DSML. While performing the previous operations, it is more convenient to use a large amount of models. This helps to obtain more accurate results. Models are either hand-crafted (small size) or automatically generated, using model generation tools such as GRIMM [1], PRAMANA [2], EMFtoCSP [3], etc. However, the set of models is not arbitrary. To ensure a reasonable efficiency, it must respect the important criterion of *diversity*. It means that models should be as different as possible. So they cover the biggest area of the space of possible solutions.

In this paper, we propose a method and a tool, called COMODI whom the purpose is to evaluate the *diversity* of a set of models. COMODI compares pairs of models using famous distance functions (hamming, Levenshtein, cosine and centrality). COMODI has three main objectives:

- Provide a graphical representation of the concept of diversity, for a given set of models. The user can then make decisions using this representation.

- Automate the previous step by selecting the most representative models of a given set.
- Automatically detect clone models.
- Provide statistics about the diversity of a set of models: average distance, closest models, most different models, etc.

The rest of the paper is organized as follows: section 2 explains how two models are compared using distance functions. Section 3 gives more details about the functioning and the output of the tool. Section 4 concludes the paper.

2 Comparing Models

The contribution of this paper is based on model comparison or model differences counting. It consists on the comparison of two models by the application of a distance function *d*:

$$\begin{aligned} d: M \times M &\rightarrow \mathbb{R} \\ (m_1, m_2) &\mapsto d(m_1, m_2) \end{aligned}$$

A distance function has three important characteristics:

- $\forall (a,b) \in M^2, d(a,b) = 0 \Rightarrow a = b$ (separation).
- $\forall (a,b) \in M^2, d(a,b) = d(b,a)$ (symmetry).
- $\forall (a,b,c) \in M^3, d(a,c) \leq d(a,b) + d(b,c)$ (triangular inequality).

In this paper we propose 4 different distance functions based on well-known functions and adapted to MDE.

2.1 Hamming distance

Hamming distance compares two vectors. It was introduced by *Richard Hamming* in 1952 [4] and was originally used for fault detection and code correction. Hamming distance counts the number of differing coefficients between two vectors.

The models to compare are transformed into vectors, then we compare the coefficients of vectors to find the distance between both models:

$$\begin{aligned} \text{a} &= (5,\ 4,\ 0,\ 2,\ 4,\ 3,\ 6,\ 1) \\ \text{b} &= \underline{(6,\ 5,\ 3,\ 3,\ 4,\ 7,\ 0,\ 1)} \\ \text{d(a,b)} &= \ 1+\ 1+\ 1+\ 1+\ 0+\ 1+\ 1+\ 0 \\ &= \tfrac{6}{8} \end{aligned}$$

Cosine distance Cosine similarity is a geometric measure of similarity between two vectors, ranging from -1 to 1: two similar vectors have a similarity that equals 1 and two diametrically opposite vectors have a cosine similarity of -1. Cosine similarity of two vectors a and b is given by the following formula:

$$C_S(a,b) = \frac{a.b}{||a||.||b||} = \frac{\sum\limits_{i=1}^{n} a_i.b_i}{\sqrt{\sum\limits_{i=1}^{n} a_i^2}.\sqrt{\sum\limits_{i=1}^{n} b_i^2}}$$

After a vectorization of models, cosine similarity is then used to compute a normalized cosine distance over two vectors [5]:

$$C_D(a,b) = \frac{1 - C_S(a,b)}{2}$$

2.2 Levenshtein distance

Levenshtein distance [6] (*named after Vladimir Levenshtein*) is a string metric used to compare two sequences of characters. To summarise the original idea, a comparison algorithm counts the minimal number of single-character edits needed to jump from a first string to a second one. There exist three character edit operations: addition, deletion and substitution.

Our customized Levenshtein distance is based on the vectorial representation of a model. Each character in original distance is replaced by a class instance of the model. So, we count the minimal cost of class instance edit operations (addition, deletion or substitution) to jump from the first model to the second one.

2.3 Centrality distance

Centrality is a real function that associates a value to each node of a graph [7]. This value indicates how much a node is *central* in this graph, according to a chosen criterion. For example, in a tree, the highest value of centrality is given to the root of the tree, whereas the smallest values are associated to the leaves. A centrality function C is defined by:

$$\begin{aligned} C : \ & E \to \mathbb{R}^+ \\ & v \mapsto C(v) \end{aligned}$$

Many usual centrality functions exist. The simplest one, the *degree centrality*, associates to each node its degree. Among the well-known centrality functions, we can cite: betweenness centrality, closeness centrality, harmonic centrality, etc.

In this paper, we propose a novel centrality function adapted for MDE and based on *eigenvector* centrality. This centrality was also used in the first published version of `PageRank` algorithm of the `Google` search engine [8]. In `PageRank`, eigenvector centrality is used to rank the web pages taken as nodes of the same graph.

More details on the previous distance functions can be found in [9] and [10].

3 Comodi tool

This sections presents our main contribution, COMODI tool (COunting MOdel DIfferences). COMODI is a tool that is used for analysing sets of models. The main objective of the tool is to automatically evaluate the diversity of the models. Given a set of models, COMODI prints *statistics and graphics*, to help the user to select only the representative elements of the set.

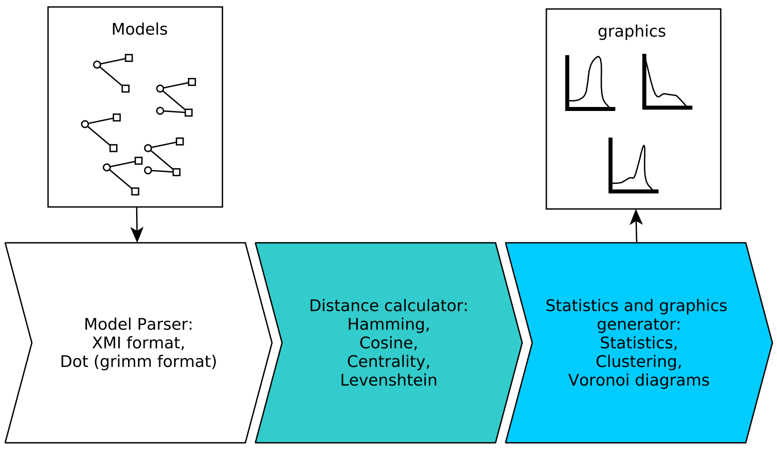

Fig. 1. Steps of process for COMODI tool

As shown in figure 1, the tool consists in two main modules:

1. *Distance calculator* is used to compare models using well-known distance functions. The comparison is first performed for each pair of models. Then a distance matrix is produced.

2. *Statistics and graphics generator* consists in analysing the previous distance matrix in order to evaluate the diversity of the set of models. Diagrams and statistics are automatically generated.

3.1 Distance calculator

Programming language: Java

This module is coded in java and implements the distance functions that are briefly presented in 2. Those functions are adapted versions of well-known functions for comparing vectors, string or graphs. Our tool supports four different distances: *Hamming [4], Levenshtein [6], Cosine [5] and Centrality [7].*

All details about used distance functions, and how they are adapted to models in MDE can be found in [9] and [10].

3.2 Statistics and graphics generator

Programming languages: Bash & R.

This module is written in Bash & R. It uses many R packages [11] to perform matrix clustring and to print result diagrams. The distance matrix that has been produced in the previous step, is used here to print the results.

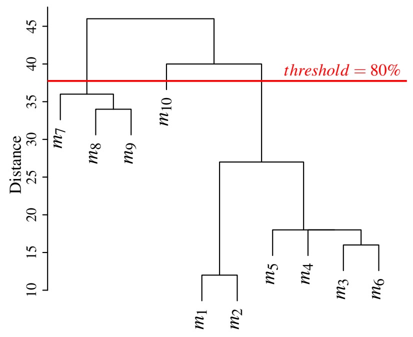

Fig. 2. Clustering tree computed for 10 models using Hamming distance.

Figure 2 shows a clustering tree for a set of 10 models. Using a threshold of 80% of difference, we obtain three clusters. So three representative models are selected by COMODI.

Figures 3 and 4 show two produced voronoi diagrams that show two drastically different sets of 100 models. The first set has a very bad coverage of the

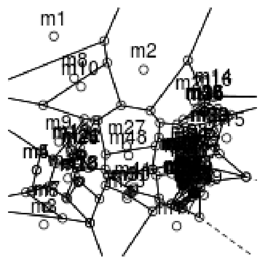
Fig. 3. A voronoi diagram showing a set of 100 models with a bad coverage.

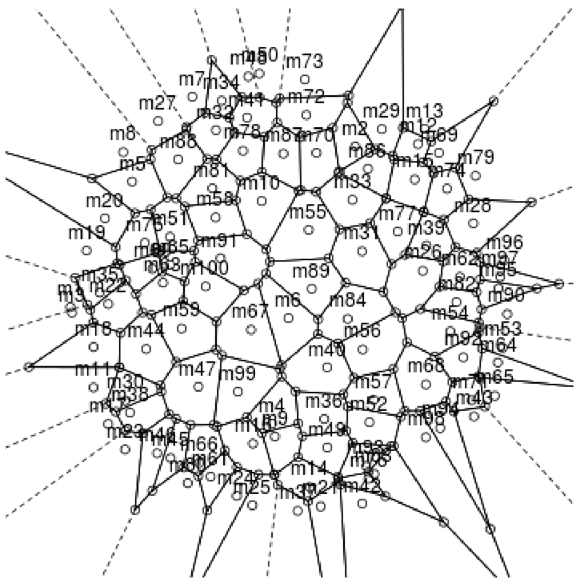
Fig. 4. A voronoi diagram showing a set of 100 models with a good coverage.

space of solutions, whereas the coverage of the second set is almost perfect. Again, using our tool, a user can easily identify a bad set of models and avoid its use.

4 Conclusion

In this paper we presented COMODI, a tool for comparing models in Model Driven Engineering. The main objective of COMODI is to evaluate the diversity of a given set of models. Models are first compared pair by pair using distance functions. In our work, we used famous functions such as: hamming and Levenshtein functions. All the distance functions we used are adapted to MDE. After producing a distance matrix, we apply clustering algorithms in order to produce statistics and graphical representations of the *diversity*. We propose two different graphical representations: voronoi diagrams and hierarchical clustering trees.

COMODI is a free and open source software that can be coupled with any model generation tool. It take two different formats for input models: *xmi* and *dot.grimm* format. It requires the installation of specific R packages for the clustering part. The tool is available for downloading on github[1].

References

1. A. Ferdjoukh, A.-E. Baert, A. Chateau, R. Coletta, and C. Nebut, “A CSP Approach for Metamodel Instantiation,” in IEEE ICTAI, 2013, pp. 1044–1051.
2. S. Sen, B. Baudry, and J.-M. Mottu, “Automatic Model Generation Strategies for Model Transformation Testing,” in ICMT, International Conference on Model Transformation, 2009, pp. 148–164.
3. C. A. González Pérez, F. Buettner, R. Clarisó, and J. Cabot, “EMFtoCSP: A Tool for the Lightweight Verification of EMF Models,” in FormSERA, Formal Methods in Software Engineering, 2012, pp. 44–50.
4. R. W. Hamming, “Error detecting and error correcting codes,” Bell System technical journal, vol. 29, no. 2, 1950, pp. 147–160.
5. A. Singhal, “Modern information retrieval: A brief overview,” IEEE Data Engineering Bulletin, vol. 24, no. 4, 2001, pp. 35–43.
6. V. Levenshtein, “Binary codes capable of correcting deletions, insertions, and reversals,” in Soviet physics doklady, vol. 10, no. 8, 1966, pp. 707–710.
7. G. Kishi, “On centrality functions of a graph,” in Graph Theory and Algorithms. Springer, 1981, pp. 45–52.
8. L. Page, S. Brin, R. Motwani, and T. Winograd, “The PageRank citation ranking: bringing order to the web,” Stanford InfoLab, Tech. Rep., 1999.
9. A. Ferdjoukh, F. Galinier, E. Bourreau, A. Chateau, and C. Nebut, “Measuring Differences To Compare Sets Of Models And Improve Di versity In MDE,” in ICSEA, International Conference on Software Engineering Advances, 2017.

[1] https://github.com/ferdjoukh/comodi

10. ——, “Measurement and generation of diversity and meaningfulness in model driven engineering,” IJAS, International Journal on Advances of Software, vol. 11, no. 12, 2018, p. 15.
11. R Development Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, 2008.