Model Matching for Model Transformation - A Meta-heuristic Approach

Hajer Saada, Marianne Huchard, Clémentine Nebut, Houari Sahraoui

Model Matching for Model Transformation: A meta-heuristic Approach

Hajer Saada[1], Marianne Huchard[1], Clémentine Nebut[1] and Houari Sahraoui[2]
[1]*LIRMM, Université Montpellier 2, CNRS, Montpellier, France*
[2]*Université de Montral, Canada*
{*saada, huchard, nebut*}*@lirmm.fr, sahraouh@iro.umontreal.ca*

Keywords: Model Driven Engineering, Model matching, Model transformation, Meta-heuristic, Multi-Objective Optimization, NSGA-II

Abstract: Model Transformation By Example (MTBE) is a recent approach that derives model transformation rules from a source model, a target model, and matching between models. Building a match between models may be a complex task especially when models have been created or edited manually. In this paper, we propose an automated approach to generate mappings between source and target models. The novetly of our approach consists in the production of *many-to-many* mappings between the elements of the two models.

1 INTRODUCTION

MTBE aims at defining a model transformation according to a set of examples of this transformation. Examples are given in the form of a source model, a target model and a matching between the two models. A matching between two models is a set of correspondences between their elements. Retrieving those correspondences is a complex and time-consuming task, especially when models are created or edited manually. Hence, the transformed elements may be different from the ones of the source model or may use different naming conventions. This task is well-known in different application domains such as schema and ontology integration, e-commerce, data warehouse and semantic web (Rahm and Bernstein, 2001; Shvaiko and Euzenat, 2005). It takes as input two schemas to generate relations between the input schemas entities.

In (Saada et al., 2013), we proposed an approach to recover transformation traces from transformation examples. In this paper, we propose an extension of this approach to deal with the model matching problem. The novelty of this work, compared to other proposals (Lopes et al., 2006; Falleri et al., 2008; Dolques et al., 2011), is to find *many-to many* matching links between the source and target models by associating a set of m source elements to a set of n target elements. The source model is fragmented using the minimal cardinalities of its meta-model and some defined OCL constraints. Then we search for each fragment in the source model the list of candidate corresponding fragments in the target model. A solution to our problem is a set of pairs of source and target fragments, that maximize the lexical and structural similarities between them, and cover the target model to ensure its completeness. Due to the huge number of possible solutions, NSGA-II, a metaheuristic method, is used to solve this problem.

This paper is organized in the following way. Section 2 is dedicated to the problem statement and the overview of our approach. Section 3 introduces the used method and its adaptation to the matching problem. Section 4 presents the experimental evaluation and the obtained results. Section 5 presents the related work. Finally, Section 6 concludes our work and draws some perspectives.

2 OVERVIEW

In this paper, we aim at generating a matching from an input model M_{source}, conforming to a metamodel MM_{source}, and a target model M_{target} conforming to a metamodel MM_{target}. Our approach is based on a fragmentation of M_{source} and M_{target}.

Definition 1. *A fragment F is a set of connected constructs of a model M. A construct $e \in M$ is an instance of a meta-class $C \in MM$ ($\underline{e : C}$).*

We denote by $R\langle C_1 : \underline{R}, \overline{R} : C_2\rangle$, an e-reference of C_1, which has C_2 as a type, and such that $\underline{R}$ (resp. $\overline{R}$) is the minimal (resp. maximal) cardinality of R. For $R\langle C_1 : \underline{R}, \overline{R} : C_2\rangle$, eRe' means that we have $\underline{e : C_1}$, $\underline{e' : C_2}$ and e is connected to e' by R.

Definition 2. *A meaningful fragment MfF of a model M is a fragment that respects the minimum cardinalities of the references defined on the metamodel and its OCL constraints.*

Consequently, a fragment F of a model M which conforms to a metamodel MM (which is provided with a set of OCL constraints) is a MfF iff:
$\forall e : C_1 \in F, (\exists C_2 | R\langle C_1 : \underline{R}, \overline{R} : C_2\rangle \in MM) \Rightarrow |\{\underline{e' : C_2} \in F | eRe'\}| \geq \underline{R}$.

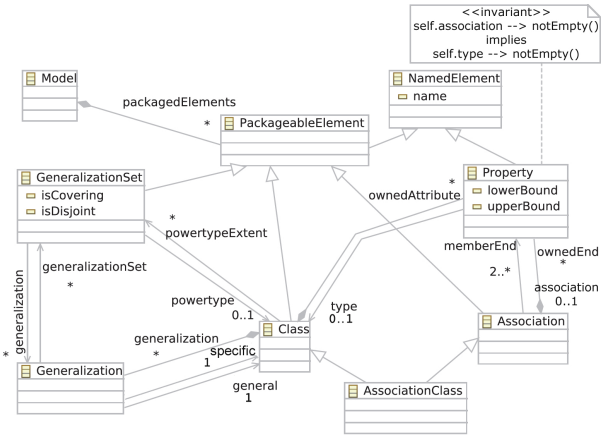

Figure 1: A simplified metamodel for UML class diagrams

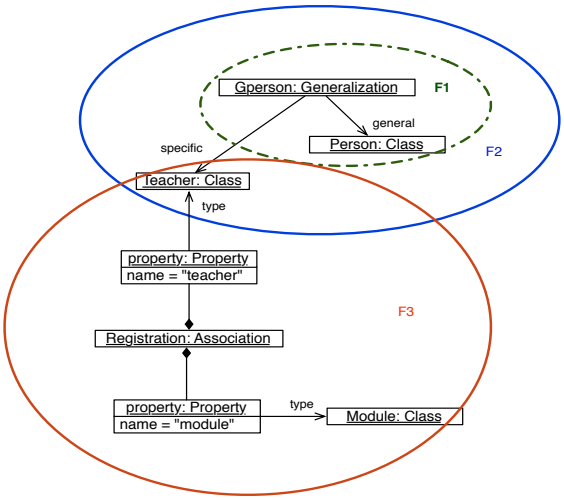

Figure 2: An instance diagram of the class diagram metamodel of Figure 1

Let us consider the instance diagram of Figure 2 that conforms to the simplified UML class diagram metamodel MM of Figure 1. It contains three circled fragments, F_1, F_2, and F_3. F_2 is a meaningful fragment because it respects the minimal cardinalities defined on the generalization meta-class in MM. A generalization consists of a relation between a general class and a specific class. Thus, the generalization between Class *Person* and Class *Teacher* constitutes a meaningful fragment. F_3 is also a meaningful fragment because it satisfies the minimal cardinalities defined on the association meta-class in MM. An association must have two properties, each one having a type class (according to the OCL constraint shown in MM). So, Association *Registration*, Property *teacher* of type *Teacher* and Property *module* of type *Module* form a meaningful fragment.
F_1 is composed of two connected constructs in the instance diagram (the generalization *Gperson* and its general class named *Person*). But in MM, a generalization must have a general class and a specific class. Thus, although F_1 conforms with the definition of fragment, it is not a meaningful fragment, because it violates the cardinality in MM.

Definition 3. *A matching between M_s and M_t is a set of pairs that match a source meaningful fragment in M_s to a target fragment in M_t .*
We denote by $F(M)$ (resp. MfF(M)) the set of all fragments (resp. meaningful fragments) that can be built from a model M.
A specific matching of n pairs takes the following form $\{(MfF_{s_i}, F_{t_i}) \mid i \in \{1..n\}\} \subseteq MfF(M_S) \times F(M_T)$.

The problem is to search a mapping between M_S and M_T. This search needs to find all the matching possibilities ($2^{MfF(M_s)\times F(M_t)}$). To find the best match between each MfF in M_S and F in M_T, using a meta-heuristic search may help.

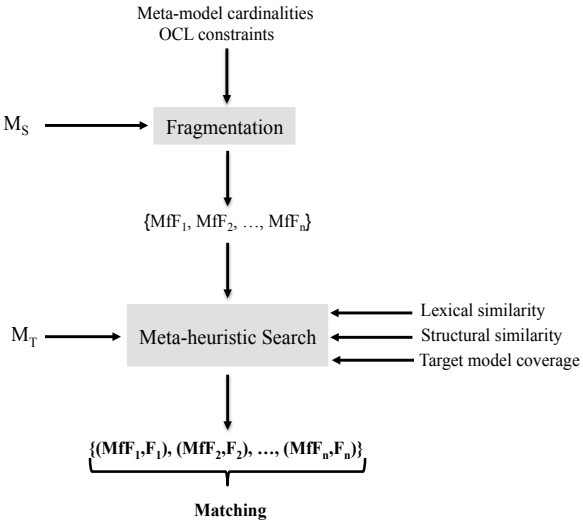

Figure 3: Approach overview

Figure 3 shows an overview of our approach:

- The first step consists in the fragmentation of the source model into meaningful fragments accord-

ing to the minimal cardinalities and the OCL constraints of its meta-model.

- In the next step, a metaheuristic method is used to search for each source meaningful fragment, the corresponding target fragment. This search takes on consideration three factors: 1) Lexical similarity between the fragments in each pair; 2) Structural similarity of the fragments matched by similar meaningful fragments (similar source fragments should be matched to similar target fragments) and 3) Completeness of the target model by the produced mapping.

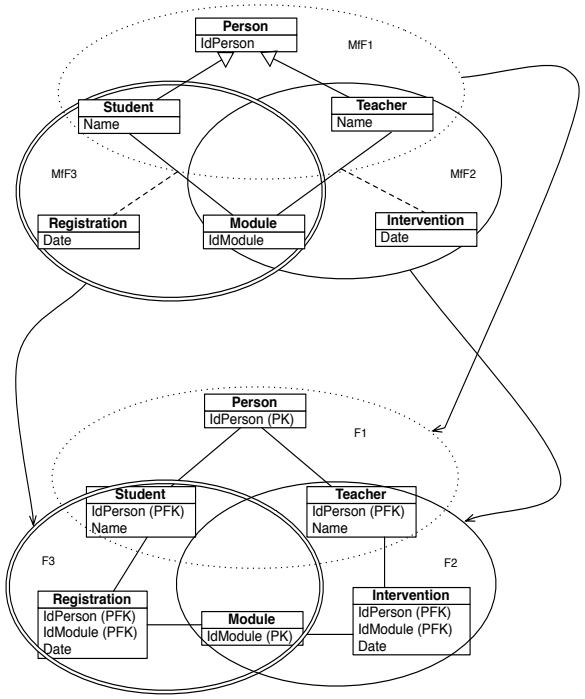

Figure 4: An example of matching between a class diagram and relational schema model

Figure 5 shows an example of matching between an UML class diagram and a relational schema model. The choice of this example is only motivated by clarity considerations. Our approach does not depend on any specific source and target metamodels. The class diagram is decomposed into three meaningful fragments according to the minimal cardinalities and the OCL constraints of the metamodel of Figure 1. In terms of lexical similarity, we note that $MfF1$ matches well $F1$. They contain the same identifiers (Person, Student and Teacher). We observe also a comparable similarity between $MfF2$ and $F2$, and between $MfF3$ and $F3$. In terms of structural similarity, the meaningful fragments $MfF2$ and $MfF3$ which have the same type (an association class between two classes) are matched to the fragments $F2$ and $F3$, which have also the same type (three connected tables).

Lexical similarity, structural similarity and the target model coverage are used in this work to find the best match between two models. Thus, the matching problem can be seen as a multi-objective optimization problem. Hence, we choose NSGA-II algorithm (introduced in the next section) to solve our problem.

3 APPROACH

3.1 The NSGA-II Algorithm

Search-based software engineering (SBSE) is a domain with a growing interest. It aims at solving a variety of software engineering optimization problems using meta-heuristic approaches, including evolutionary algorithms (EAs) (Harman, 2011). Evolutionary algorithms (EAs) develop the metaphor of the biological evolution of a population. They implement different variations on operators that act on the population by selecting individuals, and crossing or mutating them to obtain new individuals.

Specific EAs have been proposed to solve problems with multiple (possibly conflicting) objectives, where it is even more difficult to find a single optimal solution. These algorithms are searching for multiple solutions, non comparable when all objectives are considered in combination, although each one is optimal for an objective, known as Pareto-optimal solutions (Deb et al., 2002). Most famous multi-objective EAs are described in (Horn et al., 1994; Zitzler and Thiele, 1999; Knowles and Corne, 1999; Deb et al., 2002). The non-dominated sorting genetic algorithm (NSGA-II) has been proved to have a better performance than its predecessors in (Deb et al., 2002) and it has been successfully used in the SBSE community (Harman et al., 2012).

NSGA-II procedure. The evolution of the population during an iteration of the NSGA-II procedure is presented in Figure 5 which is taken from the original paper. First, an initial population P_0 of N solutions is created. The individuals of P_0 are sorted based on the non-domination. Non-dominated individuals, corresponding to the best known solutions (with regard to at least one objective) at the current step, are grouped in the first non-dominated front (rank 1). Discarding the individuals of the first front, the current non-dominated individuals form the second non-dominated front (rank 2), and so on. Diversity is preserved thanks to a crowding distance which is calculated for each solution (Laumanns et al., 2002). Fi-

nally, a binary tournament selection operator, which is based on the crowding distance, selects the best solutions. At step t, an offspring population Q_t of size N is created using selection, crossover and mutation operators. Populations P_t and Q_t are combined to form the population R_t. From R_t, the best individuals in terms of non-dominance and diversity are kept to form P_{t+1}. Then those steps are repeated till some termination criteria are satisfied.

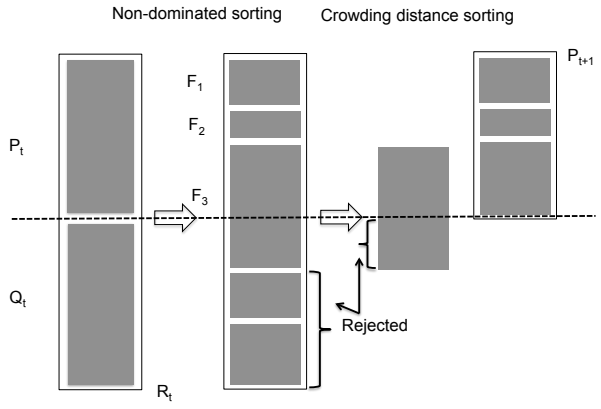

Figure 5: NSGA-II main (Deb et Al, 2002).

Fast non-dominated sorting principle. A solution s_1 *dominates* another solution s_2 if: (i) s_1 is no worse than s_2 in all objectives, and (ii) s_1 is strictly better than s_2 in at least one objective. The first non-dominated front in a population of size N is thus computed as follows. The algorithm calculates first, for each solution p: 1) the domination count n_p, *i.e.* the number of solutions which dominate the solution p and 2) S_p, the set of solutions that the solution p dominates. The solutions p such that $n_p = 0$ are found in the first non-dominated front. Now, to obtain the second non-dominated front, for each p of the first front, each solution $q \in S_p$ is visited, and its domination count is reduced by one. When the domination count of a solution q becomes zero, q is put in a separate list Q which represents the second non-dominated front. Then, the procedure is continued with the members of Q to identify the third front, and so on. If N_{obj} is the number of objectives and N is the size of the population, this algorithm has a time complexity evaluated in $O(N_{obj}N^2)$, which is better than the complexity of the previous algorithms, which require $O(N_{obj}N^3)$.

Diversity preservation. From the parent and offspring populations (each of size N), N best solutions are selected to form the next population. As many non-dominated fronts as possible are included in the next population, by increasing rank, keeping the number of individuals less than N. Let denote k the number of these included fronts. Including the non-dominated front of rank $k+1$ would result in exceeding N. The solutions of the front of rank $k+1$ are thus sorted, and only a part of them is selected to fill the next population and obtain N solutions. This is done by selecting the solutions the less *crowded*, that is, that are maximally apart from their neighbors according to the crowding distance. For a given solution s, this is measured as the average distance of two nearest solutions (neighbors of s) along each of the objectives (in the same non-dominating front). The resulting crowded-Comparison operator helps selecting scattered solutions.

3.2 Adapting NSGA-II to the Model Matching

In this section, we describe the adaptation of NSGA-II to the matching problem. To apply this type of algorithm to a specific problem, we must have a pair of source and target models. We must also define the solution encoding. We also need to specify the fitness functions, one per objective, to evaluate the results, and guide the search process, and the operators to select, crossover and mutate the solutions.

Encoding a mapping between source and target models is an essential element in our approach. In our case, a solution is a set of fragment pairs, $s = \{fp_i, i \in \{1,2,...n_s\}\}$. Each fragment pair fp_i is, in turn, encoded as a pair $fp_i = (MfF_i, F_i)$ where MfF_i is a source meaningful fragment in the source model and F_i is its corresponding fragment in the target model.

As stated before, our approach is based on the fragmentation of source and target models. The source model is divided into three criteria: respecting the minimum cardinalities in the source metamodel, respecting the OCL constraints defined on the source metamodel and ensuring the source model coverage (each construct in the source model must belong to at least one meaningful fragment). The target model is randomly fragmented to match a fragment with each meaningful fragment. We suppose that:

- The corresponding constructs of a source meaningful fragment in the target model need not necessarily to form a meaningful fragment
- The target model may contain constructs which are independent from the ones of source model. Thus, they can have different sizes.

After the fragmentation of the source model into n_s meaningful fragments, the target model is divided randomly into $n_s + x$ fragments such that $-y < x < y$. y is a parameter of our algorithm which is the maximum variation of the number of target fragments with respect to the source ones. We have a maximum of 4 constructs in a target fragment. The size is chosen

randomly for each fragment ($1 < size < 4$). if $size = 4$, we select randomly a construct, call it c, from the target model. Then, if c is connected to other constructs, we extend the fragment by randomly selecting three of them. If c is connected only to two constructs c_1 and c_2 , we can extend the fragment by one of the constructs connected to c_1 or c_2. Then, c is removed from the set of constructs for the next fragments. If c has connections with others constructs, it may still be included in others fragments.

Once the source and target fragments sets are created, we associate each source meaningful fragment MfF, with a target fragment F. A solution is then a vector whose dimensions are the MfF_s and values are the F_s.
$s = \{(MfF_1, F_1), (MfF_2, F_2), (MfF_3, F_3)\}$ corresponds to the matching solution proposed in Figure 5.
For the initial population, we build a set of N solutions (N is a parameter of our approach). Each one represents a matching possibility between the source and target models.

During the evolution process, the fitness functions evaluate the matching solutions. In our case, we have three fitness functions corresponding to three objectives: 1) Lexical similarity in a pair composed of a MfF and a F, 2) Structure similarity: in a solution s, the set of MfF which have the same type must be matched to a set of F in the target model which have the same type and 3) In a solution s, the obtained fragments must cover the target model. The three objectives should be maximized.

- Lexical similarity: to compute the lexical similarity in a solution s, we use: 1) information retrieval methods, which sort documents according to queries by extracting information about the terms' occurrences within document and 2) natural language processing techniques which identify the original forms of the words.
  First, we extract the property value lemmas of MfF_i and F_i in each fp_i using *TreeTagger* (Schmid, 1994; Schmid, 1995), a tool for annotating text with part-of-speech and lemma information. It is used to tag various languages including English, French German, etc. Then, all the distinct lemmas in s are extracted in a list li. li represents the dimensions of vectors associated to each source or target fragment in s. For each fragment and each term, the corresponding dimension is set to 1 if the term exists in the fragment or to 0 otherwise. Then, the similarity is calculated between each pair MfF_i and F_i using the cosine similarity between the two concerned vectors. The resulting lexical similarity ranges from -1, meaning that MfF_i and F_i do not share any term, to 1, meaning that MfF_i and F_i use exactly the same terms. The lexical similarity $LexSim(s)$ of a solution s equals the average of the contained pairs' lexical similarities.
- Structural similarity: to compute structural similarity in a solution s, we proceed in three steps:
  1. We classify the solutions per type of their respective meaningful fragments MfF_i.
  2. We measure for each two pairs of fragments, which have the same type of MfF, the structural similarity of the matched target models. To this end, we use also the cosine similarity, but between vectors whose dimensions are the construct types in the metamodel. Indeed, for each construct type instantiated in the target model, a term is created. Then for each target model fragment, the dimension is set to 1 if it contains a construct of the corresponding type, and to 0 otherwise.
  3. The structural similarity $StrSim(s)$ of a solution s is the average of the target-fragment similarities of the pairs having the same MfF type.
- Target model coverage: the completeness of the target model is very important because it ensures that the obtained matching covers all the target model constructs. It is measured by the number of distinct constructs in the matched target fragments divided by the number of constructs in the target model.

In a metaheuristic method, a population of matching solutions is improved by applying genetic operators (mutation and crossover). Before applying the operators, the solution are selected according to their fitness values. The selection strategy used is the Binary Tournament Selection. It favors the fittest solution for reproduction. The selection criteria are the rank of the containing front and the crowding distance for solutions within the same front.

The crossover operation consists of producing new solutions by crossing the existing ones. It is applied to each pair of selected solutions. After selecting two parent solutions for crossing, two new solutions are created by exchanging parts of the parents. The cut point is decided randomly.

After applying the crossover operation, the obtained solutions may be mutated with a given probability. At each iteration, a solution is selected for mutation. A mutation strategy is also randomly selected. We define two mutations strategies: extending a target fragment with a new construct or deleting a construct from a target fragment.

4 Evaluation

To illustrate the ability of our approach to derive mappings from source and target models, we conducted an experiment on six source and target models coming from several sources on the Internet. The size of models varies between 20 and 40 constructs.

- UML class diagram to Relational Schema model (cl2rs).
- EMF metamodel to Kermeta metamodel (em2ker).
- Kermeta metamodel to EMF metamodel (ker2em).
- UML class diagram to Java code model (cl2jc).
- Ecore metamodel to Jess (Jess, 1997) metamodel (ec2je).
- Book model to publication model (bo2pu).

Source and target models are not obtained by a transformation program and they are not written by the same person. Thus they may have different vocabularies.

As mentioned before, our algorithm uses a set of parameters. For these examples, there are set as follows:

- Crossover probability is set to 0.8.
- Mutation probability is set to 0.35
- The initial population is set to 400 solutions for each example.
- We ran the algorithm with a number of iterations equal to twice the size of the population.
- The maximum variation y of the number of target fragments with respect to the source ones is set to 1. This means that in a solution, we can have a MfF without a corresponding fragment, or an F without an assigned MfF.
- With a metaheuristic method, we can obtain, for the same example, with the same parameters, different results on different executions. Thus, we took the best result from four executions.

Testing the examples consists in generating the mapping from each source and target models and comparing the obtained mapping with those provided by an expert. This comparison allows calculating the precision and the recall for each pair $fp_i = (MfF_i, F_i)$ in the obtained solutions.
The precision of a pair is defined as the number of correctly assigned constructs ($C_{correct}$) among the total number of constructs ($C_{totalNbr}$) (equation1).
The recall of a pair is defined as the number of correctly assigned constructs among the number of expected constructs (C_{expert}) (equation2).

$$Precision(fp_i) = \frac{C_{correct}}{C_{totalNbr}} \quad (1)$$

$$Recall(fp_i) = \frac{C_{correct}}{C_{expert}} \quad (2)$$

The precision (resp. the recall) of a solution is defined as the average precision (resp. recall) of its fragment pairs.

Results and Discussion

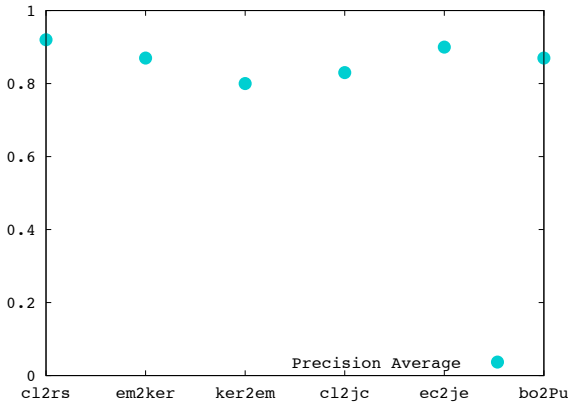

Figure 6: Precision average measured on the six examples

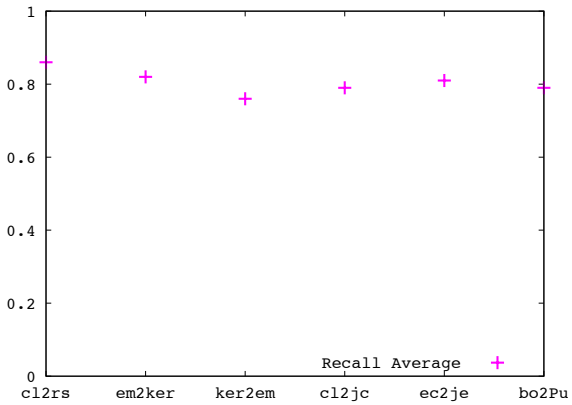

Figure 7: Recall average measured on the six examples

During our experiments, we obtained good results confirmed by the precision and recall averages shown in Figures 6 and 7. The precision scores are all between 0.87 and 0.92 and the recall scores are higher than 0.76 in all cases. The scores of UML class diagram to relational schema model are interesting (0.92 precision and 0.86 recall). This is very encouraging since we used different type of examples.

Results and Discussion

The execution time is very important since we use a metaheuristic method. In our experiments, we used

a simple macBook (2.4 GHz CPU and 2G of RAM). The execution time for generating a mapping between source and target models with a number of iterations up to 800, is less than 90 seconds. We note also that the execution time increases quasi linearly with the models' size.

Threats to validity

The experiment is here conducted on six source and target models. Those models have different size, vocabulary and structure. To help us improving the model matching algorithm, additional experiments have to be conducted, especially to study the two following issues:

- The relatively fixed size of the used examples. Larger models and more examples have to be considered in the future.
- The correctness of the obtained mapping is measured manually by an expert and this is may be a hard task especially when using larger models. An automatic measure may be defined in the future.

5 RELATED WORK

In the following, we give an overview on related work, dealing with model matching.

In database and ontologies domains, this task is called schema matching (Rahm and Bernstein, 2001; Shvaiko and Euzenat, 2005). The basic idea of the main approaches (Do and Rahm, 2002; Madhavan et al., 2001; Ehrig and Staab, 2004; Euzenat et al., 2004; Melnik et al., 2002) is to find semantic correspondences between elements of two schemas. They make the assumption that the relations between the two models being compared are identical. They compute a similarity between the elements using their names. They compute also a structural similarity between the elements. For this, they assume that there is the same kind of relations between the elements in the two compared models.

For model transformation, *Fabro* and *Valduriez* (Fabro and Valduriez, 2009) create links between source and target metamodels by using the similarity flooding technique (Melnik et al., 2002) to construct propagation models which capture the semantics of the relationships between the two models. Then, links are designed by an expert and are used to produce transformation. In (Dolques et al., 2011) a semi-automatic matching approach for discovering links between source and target model is proposed.They assume that the target model results from a transformation from the source model. They extend the Anchor-Prompt approach to discover the pairs of elements for which there is a strong assumption of matching. In (Lopes et al., 2006; Lopes et al., 2009), the authors define an algorithm (SAMT4MDE) that assumes that source and target metamodels are similar in their structure. It finds correspondences using string values of attributes and structure similarity. The contribution of (Falleri et al., 2008) consists to evaluate different parameterizations of the similarity flooding algorithm to compute the mappings.

The approach that we propose does not have any constraint on the used models or metamodels. Target models do not result from an automatic transformation from the source model and they are not written by the same person. Thus, they can have different vocabularies and structures. Furthermore a *many-to-many* matching is obtained from two models.

A mid-term objective of our work is to generate transformation rules from examples. Several approaches (Wimmer et al., 2007; Balogh and Varró, 2009; Kessentini et al., 2008; Saada et al., 2012) use examples to produce rules. Examples consist of a source model, a target model and links between elements or fragments of the models.

6 CONCLUSION

Model Transformation By Example is a novel approach to ease the development of model transformation using examples of source and target models. In this context, model matching is a crucial element to extract links between models elements, or model fragments and learn transformation rules.

The main contribution of this work is a model matching approach, which adapts the NSGA-II algorithm to explore the space of matching possibilities between the source and target model elements. We used TreeTagger, a lexical tool to solve the problem of vocabulary between models.

In order to validate the proposed approach, we performed experiments on six source and targets models and compared, using retrieval information metrics, the obtained matchings to the expected ones. The results are promising. For all the examples, precision average is higher than 0.8 and recall average is higher than 0.76.

To confirm these encouraging results, we plan to improve our work by conducting more experiments to test our approach on other type of examples. We plan also to compare our matching tool to some existing ones. Other techniques can be also explored to improve the lexical similarity between models.

REFERENCES

Balogh, Z. and Varró, D. (2009). Model transformation by example using inductive logic programming. *Software and System Modeling*, 8(3):347–364.

Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T. (2002). A fast and elist multiobjective genetic algorithm: Nsga-II. *IEEE Trans, Evolutionary Computation*, 6(2):182–197.

Do, H.-H. and Rahm, E. (2002). Coma: a system for flexible combination of schema matching approaches. In *Proceedings of the 28th international conference on Very Large Data Bases*, VLDB '02, pages 610–621.

Dolques, X., Dogui, A., Falleri, J.-R., Huchard, M., Nebut, C., and Pfister, F. (2011). Easing model transformation learning with automatically aligned examples. In *7th European Conference, ECMFA 2011*, pages 189–204.

Ehrig, M. and Staab, S. (2004). Qom quick ontology mapping. In *In Proc. 3rd International Semantic Web Conference (ISWC04*, pages 683–697. Springer.

Euzenat, J., Loup, D., Touzani, M., and Valtchev, P. (2004). Ontology alignment with ola. In *In Proceedings of the 3rd EON Workshop, 3rd International Semantic Web Conference*, pages 59–68. CEUR-WS.

Fabro, M. D. D. and Valduriez, P. (2009). Towards the efficient development of model transformations using model weaving and matching transformations. *Software and System Modeling*, 8(3):305–324.

Falleri, J.-R., Huchard, M., Lafourcade, M., and Nebut, C. (2008). Metamodel matching for automatic model transformation generation. In *Proceedings of the 11th international conference on Model Driven Engineering Languages and Systems*, MoDELS '08, pages 326–340.

Harman, M. (2011). Software engineering meets evolutionary computation. *IEEE Computer*, 44(10):31–39.

Harman, M., Mansouri, S. A., and Zhang, Y. (2012). Search-based software engineering: Trends, techniques and applications. *ACM Comput. Surv.*, 45(1):11:1–11:61.

Horn, J., Nafpliotis, N., and Goldberg, D. (1994). A niched pareto genetic algorithm for multiobjective optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 82–87. IEEE.

Jess (1997). Jess rule engine, http://herzberg.ca.sandia.gov/jess.

Kessentini, M., Sahraoui, H., and Boukadoum, M. (2008). Model transformation as an optimization problem. In *Proceedings of the 11th international conference on Model Driven Engineering Languages and Systems*, MoDELS '08, pages 159–173. Springer-Verlag.

Knowles, J. and Corne, D. (1999). The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation. In *Proceedings of the Congress on Evolutionary Computation*, volume 1, pages 98–105. IEEE.

Laumanns, M., Thiele, L., Deb, K., and Zitzler, E. (2002). Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary computation*, 10(3):263–282.

Lopes, D., Hammoudi, S., and Abdelouahab, Z. (2006). Schema matching in the context of model driven engineering: From theory to practice. In *Advances in Systems, Computing Sciences and Software Engineering*, pages 219–227. Springer.

Lopes, D., Hammoudi, S., and Abdelouahab, Z. (2009). A step forward in semi-automatic metamodel matching: Algorithms and tool. In Filipe, J. and Cordeiro, J., editors, *Proceeding of ICEIS 2009*, pages 137–148. Springer.

Madhavan, J., Bernstein, P. A., and Rahm, E. (2001). Generic schema matching with cupid. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, pages 49–58.

Melnik, S., Garcia-Molina, H., and Rahm, E. (2002). Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Proceedings of the 18th International Conference on Data Engineering*, ICDE '02, pages 117–. IEEE Computer Society.

Rahm, E. and Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350.

Saada, H., Dolques, X., Huchard, M., Nebut, C., and Sahraoui, H. A. (2012). Generation of operational transformation rules from examples of model transformations. In *MoDELS 2012*, pages 546–561.

Saada, H., Huchard, M., Nebut, C., and Sahraoui, H. A. (2013). Recovering model transformation traces using multi-objective optimization. In *ASE*, pages 688–693.

Schmid, H. (1994). Probabilistic part-of-speech tagging using decision trees.

Schmid, H. (1995). Improvements in part-of-speech tagging with an application to german. In *In Proceedings of the ACL SIGDAT-Workshop*, pages 47–50.

Shvaiko, P. and Euzenat, J. (2005). A survey of schema-based matching approaches. *Journal on Data Semantics*, 4:146–171.

Wimmer, M., Strommer, M., Kargl, H., and Kramler, G. (2007). Towards model transformation generation by-example. In *Proceedings of the 40th Annual Hawaii International Conference on System Sciences*, HICSS '07, pages 285b–.

Zitzler, E. and Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Trans. Evolutionary Computation*, 3(4):257–271.