Recovering Model Transformation Traces using Multi-Objective Optimization

Hajer Saada, Marianne Huchard, Clémentine Nebut
LIRMM, Université Montpellier 2 et CNRS, Montpellier, France
{saada, huchard, nebut}@lirmm.fr

Houari Sahraoui
DIRO, Université de Montréal, Montréal, Canada
sahraouh@iro.umontreal.ca

***Abstract*—Model Driven Engineering (MDE) is based on a large set of models that are used and manipulated throughout the development cycle. These models are manually or automatically produced and/or exploited using model transformations. To allow engineers to maintain the models and track their changes, recovering transformation traces is essential. In this paper, we propose an automated approach, based on multi-objective optimization, to recover transformation traces between models. Our approach takes as input a source model in the form of a set of fragments (fragments are defined using the source meta-model cardinalities and OCL constraints), and a target model. The recovered transformation traces take the form of many-to-many mappings between the constructs of the two models.**

I. Introduction

MDE involves the construction and manipulation of many models of different kinds in an engineering process [1]. These models cover the whole software-development cycle. They can be manipulated manually or automatically using model transformations. To ensure those models' consistency and maintenance, recovering transformation trace can be essential in an MDE process.

Recovering transformation traces can be useful for different tasks [2]: (1) understanding the system complexity by the navigation on trace links on all the model transformation chains, (2) locating bugs during the execution of transformation programs, and (3) checking the coverage of all input models by a transformation. We can distinguish between two categories of strategies to generate transformation links: the first category depends on the transformation program or engine (e.g., [3], [4], [5]). The corresponding approaches generate trace links through the execution of a model transformation. The second category is independent from a transformation program. In [6], a mechanism is proposed to generate traces from a requirement model conforming to the meta-model i* towards the meta-model UML, thus this approach is specific to two metamodels and cannot be generalized. [2] proposes an approach based on matching techniques to trace links between models. This non-scalable approach generates only *one-to-one* matching links between models.

In this paper, we are interested in the second category. We propose to recover a transformation trace independently from a transformation program. We consider that the transformation program is missing or the transformation was done manually. Our approach takes as input a source model and its corresponding target model. The aim is to find the *many-to-many* matching links between the two models, thus associating a group of m source elements to a group of n target elements. To this end, the source model is fragmented using the minimal cardinalities of its meta-model and the defined OCL constraints. Then, for each source fragment, we search for a list of potential transformed 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. A solution must also cover all of the target model to ensure its transformation completeness. Due to the very large number of possible solutions, a multi-objective metaheuristic method (NSGA-II) is used to solve our problem.

The remainder of this paper is organized as follows. Section II is dedicated to the problem statement and the overview of our approach. While section III details our approach, section IV explains the experimental evaluation. Section V presents the related work.

II. Approach overview

This section shows how recovering a transformation trace between a source model and a target model can be defined as an optimization problem, and presents our approach.

A. Problem Statement

In our context, a metamodel MM is an instance of the Ecore [7] meta-metamodel. A model M is an instance of a metamodel.

Our approach is based on a fragmentation of the source and target models. 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 $Frag(M)$ the set of all fragments that can be built from M. 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. A meaningful fragment MfF of a model M is a fragment that respects the minimum cardinalities of the references defined on the metamodel and the OCL constraints.

Consequently, a fragment F of a model M which conforms to a metamodel MM (with OCL constraints) is a MfF iff: $\forall \underline{e : C_1} \in F, (\exists C_2 | R\langle C_1 : \underline{R}, \overline{R} : C_2\rangle \in MM) \Rightarrow |\{\underline{e' : C_2} \in F | eRe'\}| \geq \underline{R}$. We denote by $MeanFrag(M)$ the set of all meaningful fragments that can be built from M.

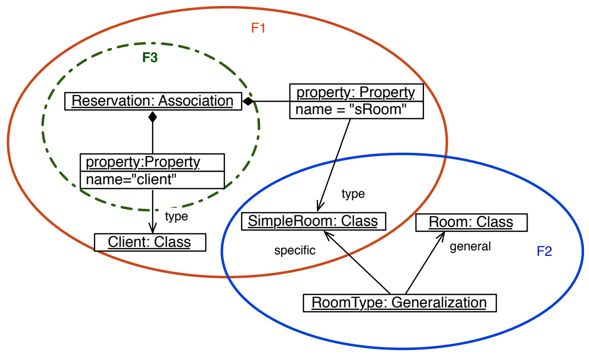

Fig. 1. An instance diagram of the class diagram metamodel of Figure 2

Let us consider the instance diagram of Figure 1 that conforms to the simplified UML class diagram metamodel $CDMM$ of Figure 2. Three fragments are circled, F_1, F_2, and F_3. F_1 (resp. F_2) is a meaningful fragment because it satisfies the minimal cardinalities defined on association (resp. generalization) meta-class in $CDMM$. An association must have two properties of type class. So, association *Reservation*, property *client* of type *Client* class and property *sRoom* of type *SimpleRoom* class form a meaningful fragment (F_1). A generalization consists of a relation between a general class and a specific class. Thus, the generalization between the Class *Room* and the class *SimpleRoom* constitutes a meaningful fragment (F_2). F_3 is composed of two connected constructs in the instance diagram (the association *Reservation* and its property named *client*). It is not meaningful fragment, because the metamodel $CDMM$, an association must have at least two properties.

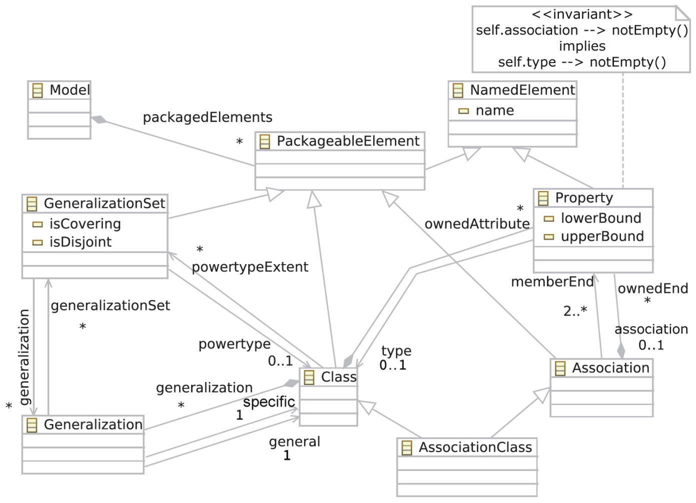

Fig. 2. A meta model for UML class diagrams

Definition 1: A transformation trace between a source model M_s and a target model M_t is a set of pairs connecting a source meaningful fragment to a target fragment. A specific transformation trace of n pairs takes the form $\{(MfF_{s_i}, F_{t_i}) \mid i \in \{1..n\}\} \subseteq MeanFrag(M_S) \times Frag(M_T)$.

The size of the set of possible transformation traces is $2^{MeanFrag(M_s) \times Frag(M_t)}$. Searching this space is hard to perform with an exhaustive search method. This led us to use a metaheuristic search to solve the trace recovery problem.

B. Approach Overview

A first step of our approach consists in the decomposition of the source model into meaningful fragments according to the constraints of the metamodel. Then, a metaheuristic method is used to search for the best match between the identified source meaningful fragments and all the possible target fragments. To evaluate the quality of a match (candidate trace) two factors are considered: 1) Lexical similarity between fragments in each pair and 2) Structural consistency in the mapping of similar source fragments, *i.e.*, similar source fragments should be associated to similar target fragments.

In addition to the lexical and structural factors, to be acceptable, a candidate trace $Trace = \{(MfF_{s_i}, F_{t_i}) \mid i \in \{1..n_{Trace}\}\} \subseteq MeanFrag(M_s) \times Frag(M_t)$ should satisfy the completeness constraints, formalized as follows: $(\bigcup_{i=1}^{n_{Trace}} MfF_{s_i} = M_s) \wedge (\bigcup_{i=1}^{n_{Trace}} F_{t_i} = M_t)$

Figure 3 illustrates an example of transformation trace between a simplified UML class diagram and its corresponding entity-relationship model. Note that the choice of this example is only motivated by clarity considerations. Our approach does not depend on specific source and target metamodels. The class diagram is decomposed into three meaningful fragments according to the constraints of the metamodel of Figure 2. In terms of lexical similarity, MfF_1 matches well F_1 as they both contain the same identifiers. Comparable lexical similarities could be observed respectively between MfF_2 and F_2, and between MfF_3 and F_3. In terms of structural consistency, MfF_2 and MfF_3, which are fragments of the same type (a one-to-many association between two classes) are consistently matched to two fragments F_2 and F_3, which are also of the same type (a relation between two entities).

To find the good transformation trace between a pair of source and target models, we perform the heuristic search guided by the lexical and structural factors as well as by the completeness constraints. Thus, the trace recovery can be seen as a multi-objective optimization problem.

During the past two decades, evolutionary algorithms (EAs) have gained popularity in dealing with software engineering tasks that could be modeled as optimization problems. For problems with multiple (possibly conflicting) objectives, like the one studied in this paper, it is usually difficult to find a single optimal solution. Such kind of problems gives rise to a whole set of solutions, known as Pareto-optimal solutions [8]. In this context, a number of multi-objective EAs have been proposed. The non-dominated sorting genetic algorithm (NSGA-II) [8] is the one that is the most applied in the SBSE community [9]. It allows to easily model the trace recovery as a multiobjective optimization problem.

NSGA-II procedure. First, an initial population P_0 of N solutions is created. This population is sorted based on the

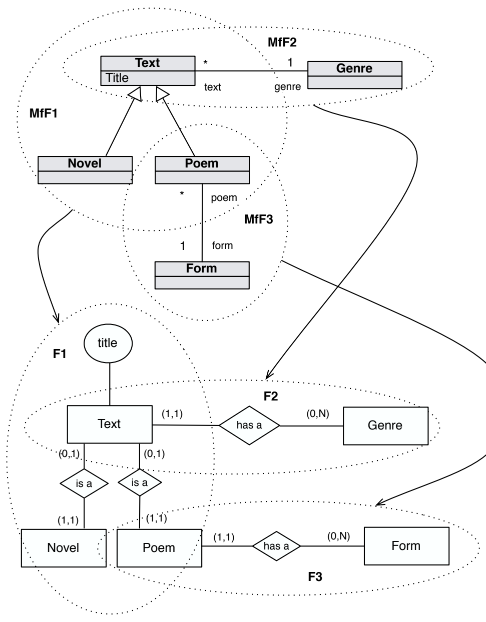

Fig. 3. An example of transformation links

non-domination. The first non-dominating front is assigned a rank of one, the second has a rank of two and so on. Then, a crowding distance is calculated for each solution [10]. A binary tournament selection operator, based on the crowding distance, is used to select the best solutions and an offspring population Q_0 of size N is created by crossover and mutation operators. P_0 and Q_0 are combined to form the population R_0. The best individuals in terms of non-dominance and diversity are chosen from R_0. Then those steps are repeated till the termination criteria are satisfied.

III. Adapting NSGA-II to trace recovery

In this section, we describe the adaptation of NSGA-II to recover transformation traces.

A. Solution representation

A crucial element in our approach is encoding of a transformation trace between the source and target models for each candidate solution. In our case, a solution s, 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 in Section II, the source model is divided into fragments according to three criteria: 1) compliance with the minimum cardinalities defined in the source meta-model, 2) compliance with the OCL constraints specified in the source meta-model and 3) source model coverage. The target model is randomly divided to associate a fragment to each source meaningful fragment. We conjecture that dependent constructs (with cardinality constraints) in the source model have to be transformed together and the corresponding constructs in the target model do not necessarily need to form a meaningful fragment. Additionally, when transforming a source model, some constructs may not have corresponding constructs in the target model, and some new constructs in the target model may be created independently from the source ones.

Thus, after obtaining the n_s source meaningful fragments, an integer x is generated randomly in $[-y, y]$, y being a parameter of our algorithm indicating the maximum variation of the number of target fragments with respect to the source ones. More concretely, for each target-model fragment f, we start by randomly select its size t between 1 and 4. If $t = 3$ for instance, 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 two of them. If c is connected to just one construct c_1, we can extend the fragment by one of the constructs connected to c_1. Then, c is removed from the set of potential starting constructs for the next fragments. Nevertheless, c can still be included in other fragments thanks to its connections with other constructs.

When both source and target fragment sets are created, each source MfF is randomly associated with a target model fragment F. A solution is then a vector whose dimensions are the MfFs and values are the Fs.

To create the initial population of N solutions, our algorithm randomly generates a set of solutions s_i, $i \in \{1, 2, ..N\}$.

B. Solution evaluation

The fitness functions evaluate a candidate trace solution s. We defined three fitness functions corresponding to three objectives:

1) An MfF in a source model corresponds to a F in a target model when MfF and F use similar vocabulary, *i.e.*, are similar in terms of property values of type string.
2) In s, a set of MfF of the same type, *i.e.*, same construct types with the same connections, must be matched to a set of F of the same type in the target model.
3) In s, the obtained fragments must cover the target model.

The two first objectives approximate the semantic equivalence between model fragments belonging to two different metamodels. The third objective ensures that a solution is complete as it recovers all the transformation trace. The three objectives should be maximized.

Lexical similarity. For this fitness function, we take our inspiration from information retrieval methods, which sort documents according to queries by extracting information about the terms' occurrences within documents. The extracted information is used to find the similarity between queries and documents. In our case, the similarity is used to compare the property values of MfF_i and F_i in each fp_i in a solution s. All the terms (distinct property values) in s are extracted in a list l. l defines 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 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.

Structure similarity. In order to measure the structure similarity in a solution s, we start by classifying the set of its fragment pairs per type of their respective meaningful fragments MfF_i. After classifying the solutions per type of their MfF, 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. 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 coverage of the target model is the most important objective because it ensures that the fragments obtained in the solution cover all the target model. The coverage $Cov(s)$ of a solution s is measured by the number of distinct constructs in the matched target fragments divided by the number of constructs in the target model.

C. Operators definition

In NSGA-II, in each iteration, the N solutions selected from the previous generation are used to create new N solutions using genetic operators. This improves the existing solutions by mixing their genetic material (crossover) and/or by creating new material (mutation). Before applying the operators, the solutions are selected according to their fitness values.

In our work, binary tournament selection is used. It consists of choosing some solutions at random in the population, and selecting the fittest two for reproduction. The selection criteria are the rank of the containing front and the crowding distance for solutions within the same front. Several tournaments are run to produce the N needed solutions.

The crossover consists of producing new solutions from the existing ones. When two solutions are selected using the binary tournament method, two offspring solutions are created, with a given crossover probability, by exchanging parts of the parent solutions. This consists in randomly selecting a cut point in the solution vector, and all the target fragments beyond that point in either parent are swapped between the two parents.

After performing the crossover, the obtained solutions could be mutated with a given mutation probability. For our problem, we define two mutation strategies: extending a target fragment with a new construct or deleting a construct from a target fragment. Recall that a transformation-trace solution is a set of fragment pairs; each one contains a source and a target fragment (*MfF* and F). Like a *MfF*, a target fragment contains constructs connected with references. For the first mutation strategy, a pair $fp_i = (MfF_i, F_i)$ is chosen randomly. Two kinds of construct can be added to fp_i: a randomly chosen construct, not already included in F_i, or a construct which has a reference to another one in F_i. The second mutation strategy consists of deleting a construct from a target fragment F_i in a pair $fp_i = (MfF_i, F_i)$ also randomly chosen from a solution. We randomly select a construct c from F_i. Then, we check if c is connected at least to one construct in F_i. If we find that c is not linked to any construct in F_i, it will be deleted.

IV. Evaluation

To evaluate the feasibility of our approach for recovering transformation traces, we conducted an experiment which is reported in this section.

A. Experimental settings

a) Experimental data: Our case study is composed of six existing model transformations collected from the literature or written by the authors in previous projects.

- UML class-diagram to relational schema (Cl2Rs).
- Ecore meta-model to Jess [11] meta-model (Ec2Je).
- Relational schema to Jess model (Rs2Je).
- UML state machine to labeled transition System (St2Lt) [12].
- Abstract syntax examples to graphical syntax examples (As2Gs) [13].
- Application of the design pattern State to a UML model owning at least one class with a StateMachine (St2St).

For each of the above-mentioned model transformations, we wrote a source model. To have realistic models, we decided to set their size to at least 50 constructs. As the selected transformations are implemented, we applied them to the source models to generate their counterpart target models. Afterwards, we used each pair of source/target models as a case to test the trace recovery approach.

b) Experimental protocol: In order to evaluate the relevance of the transformation trace generated by our approach, we also defined, for each pair of models, the actual transformation trace. For the two model transformations Cl2Rs and Rs2Je, we used the transformation rules generated in our previous work [14] which provide for each meaningful fragment its corresponding fragment using the Jess rule engine [11]. For the rest of the model transformations used in this evaluation, a domain expert manually built the actual trace.

Descriptive statistics about the actual (expected) traces are given in Table I. The number of fragment mappings in each trace is given in column Nbr_{trace}. This ranges from 19 for the Cl2Rs transformation to 25 for the St2St one. The size of the source fragments in the actual traces varies in general from 2 (min_F) to 4 (max_F). The variation in size for target fragments is larger, with fragments containing up to 7 constructs (for St2Lt), which makes the mapping recovery more difficult.

For each source and its corresponding target model, we use our algorithm to generate a transformation trace. As we are dealing with multiobjective optimization, usually, many solutions are present in the Pareto front. Usually, a user could look at the proposed solutions and select one. In the case of problems for which the user does not have enough knowledge

Examples	Nbr_{trace}	Source Fragments		Target Fragments	
		(min_F)	(max_F)	(min_F)	(max_F)
Cl2Rs	19	2	4	1	4
Ec2Je	22	1	3	1	2
Rs2Je	21	2	4	1	3
St2Lt	22	3	4	2	7
As2Gs	20	2	4	1	4
St2St	25	2	3	3	5

TABLE I
THE ACTUAL TRACES OF OUR EXAMPLES

to choose a solution, a ranking could be necessary to make a recommendation. For the purpose of this evaluation, as our three objectives are normalized in the interval $[0, 1]$, it is possible to calculate a distance to the optimal solution s_o, *i.e.*, the one having $LexSim(s_o) = 1$, $StrSim(s_o) = 1$, and $Cov(s_o) = 1$. For a candidate solution s, such a distance $d(s)$ could be calculated as follows:

$$\sqrt{(1 - LexSim(s))^2 + (1 - StrSim(s))^2 + (1 - Cov(s))^2} \tag{1}$$

Distance d gives equal importance to all the objectives. However, we believe that consistency and completeness are very important when selecting the final solution. Indeed, having incomplete solutions or similar fragments that are transformed differently could invalidate a solution. Therefore, we propose a variation d_2 of the distance d where only the consistency and completeness objectives are considered. $d_2(s)$ of a candidate solution s is defined as follows:

$$\sqrt{(1 - StrSim(s))^2 + (1 - Cov(s))^2} \tag{2}$$

The solution having the minimal d_2 distance is then evaluated by computing its precision and recall. The precision of a solution s is defined as the average precision of its fragment-pairs. For a pair $fp_i = (MfF_i, F_i) \in s$ and the expected mapping (MfF_i, EF_i), the mapping precision of fp_i is defined as the number of correctly assigned constructs in F_i among the total number of constructs in F_i.

Like for the precision, we evaluate also the pairs' average recall of a solution. For a pair $fp_i = (MfF_i, F_i) \in s$ and the expected mapping (MfF_i, EF_i), the mapping recall of fp_i is defined as the number of correctly assigned constructs in F_i among the total number of constructs in EF_i.

According to Section III, the trace recovery algorithm uses the following parameters:

- Crossover probability is usually high. It is set to 0.8.
- Mutation probability is set to 0.4.
- In each transformation example, a population of 500 solutions was randomly generated, from which we kept only those having a score above a threshold for the three objectives. This ensures an initial decent genetic material.
- The iteration number is equal to twice the size of the population.
- The maximal 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 could have a MfF without assigned F or a F without any MfF.
- As our algorithm is probabilistic by nature, we took the best result from three executions.

B. Results and Discussion

Due to lack of space, we present a summary of the results for the six case studies.

	Distance d_2 from our best solution to the ideal one
S_{Cl2Rs}	0.09
S_{Ec2Je}	0.13
S_{Rs2Je}	0.19
S_{St2Lt}	0.35
S_{As2Gs}	0.21
S_{St2St}	0.23

TABLE II
DISTANCE FROM OUR BEST SOLUTIONS TO THE OPTIMAL ONES

1) Results for the six examples: Table II shows the best obtained solutions. The distances varies from 0.09 for Cl2Rs to 0.35 for St2Lt. The distance results are confirmed by the precision and recall scores reported in Figure 4.

Except for St2Lt, the precision scores are at least equal to 90% and the recall scores are equal to 84% or more. This is very encouraging since the examples involve different types of transformations. The scores of St2Lt are intriguing although both precision (75%) and recall (70%) are interesting. In this transformation, many events are generated and synthetically labeled "i", and the states are indicated by numbers. In this context, our lexical and structural approximations are limited to capture the semantic equivalence between some fragments.

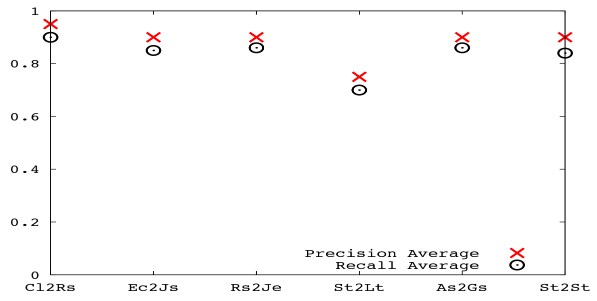

Fig. 4. Evaluation results

2) Performance: The execution time is very important since we use a metaheuristic search to explore a large space. In our experiments, we used a simple MacBook (2.4GHz CPU and 2G of RAM). The execution time for recovering a transformation trace on our examples (about 50 constructs), with a number of iterations up to 1000, is less than 120 seconds. This is very acceptable since the recovery process is not intended to be executed on a short-period basis. With models having different sizes and with the same parameter values (population size and number of iterations), the execution time increases quasi linearly with the models' size.

C. Threats to validity

The experiment is here conducted on six examples of model transformations. The choice of the transformation examples could be a threat to the validity of our evaluation. To circumvent this threat, we carefully chose examples from different transformations (structural/behavioral, exogenous/endogenous, different fragment sizes, etc.). The only possible limitation to the generalizability is the relatively fixed size of the examples (about 50 constructs). We plan to try our approach with models having a larger variation in size in the future. Another threat concerns the fact that our experimental setting is semi-real as all the target models are generated using the known transformation mechanisms, and none was derived manually by an expert. A possible issue is that transformation engines tend to use the vocabulary of the source model whereas a human expert could use derived vocabulary (synonyms, abbreviations, etc.). These situations could be handled by a more subtle way to measure the lexical similarity.

V. Related work

Some model transformation tools provide integrated support for traceability such as QVT [15] and MOFScript [16]. With [17], developers can encode a trace as an output model or attach traceability generation code to ATL program [18]. Grammel et al. [3] propose a generic framework for augmenting arbitrary model transformation approaches with a traceability mechanism. This framework is based on a domain-specific language for traceability. In [19], the authors focus on generated trace relations as part of QVT transformations. In the same context, Amar et al. [4] present an approach to automatically trace imperative model transformation in a Java/EMF environment. Finally, a recent work [5] consists in visualizing traceability in model transformations after adding a trace generator to the transformation engine of ATL. All those solutions generate trace links in parallel with the transformations. They depend on the existence of a transformation engine and could not be applied for trace recovery.

Another category consists in generating transformation trace independently of the transformations. This allows to handle cases where only source and target models are present without a knowledge on how the transformation was performed. Our contribution falls within this category. In [6], the authors present an approach to support generation of bi-directional traceability relations between organizational requirements modeled in i*, and UML use cases and class diagrams. This approach is applied to a specific type of transformation whereas our approach is independent of transformation languages and problems. The work in [2] is probably the one that is the closest to our contribution. It uses graph-based model matching techniques to generate trace links. This approach may have a higher complexity, especially when manipulating large-size models. In addition, it produces *one-to-one* matching links whereas our approach can generate *many-to-many* mappings.

VI. Conclusion

In this paper, we proposed a novel approach for the recovery of a transformation trace for two arbitrary source and target models. Our approach does not require any knowledge of the transformation. We adapted the NSGA-II algorithm to explore the space of mapping possibilities between the two models. We evaluated our approach on six different cases. The obtained results indicate that recovered traces are very similar to the expected ones. Despite the encouraging results, there is still a room for improvement. First, we plan to conduct more experiments to test our approach on other transformation types and compare our results with the ones of the other approaches. From the algorithmic perspective, we will explore other functions to approximate the semantic equivalence between the source and target model.

References

[1] R. F. Paige, N. Drivalos, D. S. Kolovos, K. J. Fernandes, C. Power, G. K. Olsen, and S. Zschaler, "Rigorous identification and encoding of trace-links in model-driven engineering," *Softw. Syst. Model.*, vol. 10, pp. 469–487, Oct. 2011.

[2] B. Grammel, S. Kastenholz, and K. Voigt, "Model matching for trace link generation in model-driven software development," in *Proceedings of the 15th international conference on Model Driven Engineering Languages and Systems*, MODELS'12, pp. 609–625, 2012.

[3] B. Grammel and S. Kastenholz, "A generic traceability framework for facet-based traceability data extraction in model-driven software development," in *Proceedings of the 6th ECMFA Traceability Workshop*, ECMFA-TW '10, pp. 7–14, ACM, 2010.

[4] B. Amar, H. Leblanc, B. Coulette, and C. Nebut, "Using aspect-oriented programming to trace imperative transformations," in *Proceedings of the 2010 14th IEEE International Enterprise Distributed Object Computing Conference*, pp. 143–152, 2010.

[5] M. van Amstel, M. G. J. van den Brand, and A. Serebrenik, "Traceability visualization in model transformations with tracevis," in *ICMT*, pp. 152–159, 2012.

[6] G. A. A. Cysneiros, F. Andrea, and Z. G. Spanoudakis, "Traceability approach for i* and UML models," in *in Proceedings of 2nd International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS03*, 2003.

[7] B. Frank, *Eclipse Modeling Framework (Eclipse Series)*.

[8] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast and elist multiobjective genetic algorithm: Nsga-II," *IEEE Trans, Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[9] M. Harman, S. A. Mansouri, and Y. Zhang, "Search-based software engineering: Trends, techniques and applications," *ACM Comput. Surv.*, vol. 45, no. 1, pp. 11:1–11:61, 2012.

[10] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler, "Combining convergence and diversity in evolutionary multiobjective optimization," *Evolutionary computation*, vol. 10, no. 3, pp. 263–282, 2002.

[11] "Jess rule engine." http://herzberg.ca.sandia.gov/jess.

[12] H.-V. Luong, T. Lambolais, and A.-L. Courbis, "Implementation of the conformance relation for incremental development of behavioural models," in *MoDELS*, pp. 356–370, 2008.

[13] F. Pfister, V. Chapurlat, M. H. Huchard, and C. Nebut, "A proposed tool and process to design domain specific modeling languages," tech. rep., LGI2P, Ecole Des Mines, 2012.

[14] H. Saada, X. Dolques, M. Huchard, C. Nebut, and H. Sahraoui, "Generation of operational transformation rules from examples of model transformations," in *MoDELS*, pp. 546–561, 2012.

[15] "Object management group: Mof 2.0 query view transformation."

[16] "Mofscript, http://www.eclipse.org/gmt/mofscript/."

[17] "Atlas transformation language, http://www.eclipse.org/m2m/atl."

[18] F. Jouault, "Loosely coupled traceability for ATL," in *In Proceedings of the European Conference on Model Driven Architecture (ECMDA) workshop on traceability*, pp. 29–37, 2005.

[19] I. Kurtev, M. Dee, A. Göknil, and K. B. van den, "Traceability-based change management in operational mappings," in *ECMDA Traceability Workshop 2007*, pp. 57–67, 2007.