# Semantic Approach to Automatically Defined Model Transformation

Tiexin Wang, Sebastien Truptil and Frederick Benaben
*Centre Genie Industriel, University de Toulouse - Mines Albi, Campus Jarlard, 81000 Albi, France*

Keywords: Model-Driven Engineering, Automatic Model Transformation, Semantic Check, Ontology.

Abstract: Modelling and model transformation are regarded as two pillars of model-driven engineering; they have been used together to solve practical problems. For instance, since different models (e.g. data model) are used by heterogeneous partners involved in a specific collaborative situation, there is an urgent need for model transformations to exchange information among the heterogeneous partners. To quickly define model transformations, this paper presents an approach, which could replace the users' effort in making mappings during the definition of a model transformation process. This approach is based on model transformation methodology, using syntax and semantic relationship among model elements. For this, a generic meta-meta-model and semantics checking methodology are proposed, before being illustrated by an example.

## 1 INTRODUCTION

Nowadays, different kinds of models have been widely used (store data, simulate industrial processes, manage information service, etc.) by different domains, and modelling and model transformation will play a key role in solving complex, diversity issues. Problems, which exist in collaborative situations, are such issues. As explained in (Rajsiri and *al*, 2010), more and more collaborative situations (domains-crossing) are frequently appearing and disappearing, such as crisis management, supply chain management, and enterprise interoperability (Chen and *al*, 2007), etc. Consequently, to improve the efficiency of the collaborative, fast information exchange among different partners is necessary. Fortunately, model transformation methods can provide a solution which aims at solving such issues. There are several approaches using model transformation methods to solve practical problems, such as mentioned in (Castro and *al*, 2011), and (Grangel and *al*, 2010).

However, the process of model transformation definition could still be improved. Indeed, development of model transformations involves many repetitive tasks, which are often done manually (Del Fabro and Valduriez, 2008). These repetitive tasks are used to define mapping rules between the elements of source and target models. These rules are executed during the transformation process. As the context of model transformation is different, users should make the mappings manually (according to the specificity of domains that models come from, the source and the target models are distinguished). Therefore, generating a solution to automatically define model transformations is a motivating challenge.

Generating automatically model transformation is closely related to the syntax and semantic matching between concepts of the source and target models. The syntax and semantic matching approaches are defined at the abstract level of model (meta-model level). Based on this principle, our main idea consists in automatically defining syntax and semantic mappings between several meta-models. Most syntax and semantic matching approaches cannot be applied to models that conform to different meta-models (Del Fabro and *al*, 2005). For this reason, we define a generic meta-meta-model (we want a generic and simple meta-meta model that can work along with a specific ontlogy which provides the data basis for semantic check). Concerning the syntax matching approach, existing techniques and methodologies could be reused such as explained in (Lano and Kolahdouz-Rahimi, 2013), and (Bollati and *al*, 2013). Although semantic check methods have been used in other research fields as explained in (Ly and *al*, 2006), semantic matching approaches for model transformation are a challenging research goal. To achieve our aim, we try to use ontology (McGuinness and Van Harmelen, 2004) and

Wang T., Truptil S. and Benaben F..
Semantic Approach to Automatically Defined Model Transformation.
DOI: 10.5220/0004713303400347
In *Proceedings of the 2nd International Conference on Model-Driven Engineering and Software Development* (MODELSWARD-2014), pages 340-347
ISBN: 978-989-758-007-9

semantic check rules.

This paper is divided in four parts. In the first section, definitions of model, meta-model and model transformation principles are given. Then an overview of our solution is proposed in the second section. The third section makes a focus on the semantic mapping approach. Before the conclusion, a case study is presented in the fourth section to illustrate our solution.

## 2 MODEL TRANSFORMATION OVERVIEW

With the wide use of model-driven engineering theory in many specific domains, more and more researchers and organizations are becoming interested in finding solutions to effective model transformation.

This section is divided into four sub-sections to give an overview of model transformation (combined with our proposal) in four aspects, respectively. First, the definitions of model and meta-model. Second, the model transformation theories. Third, the model transformation approaches. Fourth, the model transformation techniques.

### 2.1 Model & Meta-Model

Model transformation is based on two basic and crucial concepts: model and meta-model (Bézivin, 2006).

A model could be seen as a picture of a system, depending on a point of view. This picture is a simplification of this system, which highlights its characteristics. A meta-model defines the characteristics of a valid model. A meta-meta-model for deducing meta-models from input models is proposed in this article.

### 2.2 Model Transformation Principles

Figure 1 (Bénaben and *al*, 2010) illustrates the model transformation principles. This idea inspired our work.

The two: "source and target models" are built according to their meta-models (MM).

The key point is that the source MM shares part of its concepts with the target MM (the two spaces, source and target, have to be partially overlapping in order to allow model transformation). As a consequence, the source model embeds a shared part and a specific part. The shared part provides the extracted knowledge, which may be used for the model transformation, while the specific part should be saved as capitalized knowledge in order not to be lost. Then, mapping rules (built based on the overlapping conceptual area of MMs) can be applied onto the extracted knowledge. The transformed knowledge and an additional knowledge (to fill the lack of knowledge concerning the non-shared part of concepts into the target MM) may be finally used to create the shared part and the specific part of the target model.

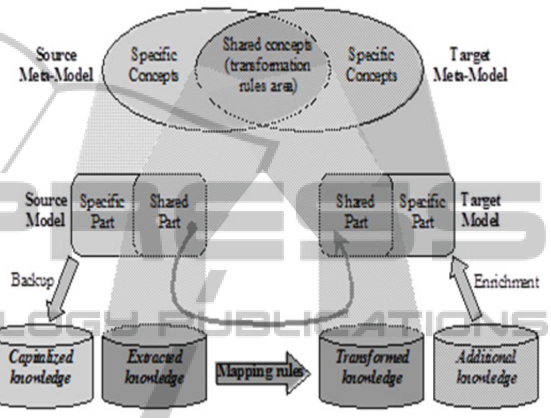


Figure 1: Model transformation framework.

### 2.3 Model Transformation Approaches

In general, according to (Czarnecki and Helsen, 2003), there are two main kinds of model transformation approaches. They are: model-to-code approaches and model-to-model approaches. For the model-to-code approaches, there are two detailed categories:

- Visitor-based approaches
- Template-based approaches

For the model-to-model approaches, there are five detailed categories:

- Direct-Manipulation Approaches
- Relational Approaches
- Graph-Transformation-Based Approaches
- Structure-Driven Approaches
- Hybrid Approaches

The method "automatically define model transformations" could be seen as a complement to the classification of model transformation approaches. Automatic model transformation could also be used in association with other model transformation approaches; it will become a part of the whole process of transformation, or complete a specific function.

## 2.4 Model Transformation Techniques

In practice, a large number of techniques have been developed to perform model transformation. The most prevalent model transformation techniques are: QVT (Query, View, and Transformation language) (OMG, 2002), ATL (Atlas transformation language) (Jouault and *al*, 2007) and some of the graph rewriting based model transformation languages.

QVT is defined by the "Object Management Group (OMG)", and QVT defines three specific model transformation languages.

The ATL model transformation language is defined by the "ATLAS Group, (INRIA & LINA) University of Nantes". ATL architecture provides a set of languages: the ATLAS Model Weaving (AMW), ATL, and the ATL Virtual Machine (ATL VM).

The model transformation languages (based on graph rewriting) describe transformations that operate on a graph by rewriting it. A transformation is performed in steps operating on a current graph. There are several graph rewriting languages, such as "GReAT (Agrawal and *al*, 2003)" and "AGG (Taentzer and *al*, 2009)", etc.

The "ATL" has been choosed to develop the tests for our proposal.

# 3 GENERAL OVERVIEW OF THE SOLUTION

In this section, an overview of the solution will be illustrated.

This section is divided into two subsections. In the first subsection, the main objective of our work is explained. In the second subsection, an overview of our solution is given.

## 3.1 Main Objective

The main objective of this work is to define a process of automatical model transformation.

In order to achive this objective, there are several basic function requirements that should be implemented. They are listed as following:

- Define a generic meta-meta-model.
- Create an ontology based on the meta-meta-model.
- Analysis the input from the users (source model and target model; source model and target meta-model; source meta-model and target meta-model)
- Deduce our source meta-model and target meta-model based on the analysis results and the generic meta-meta-model.
- Apply syntax check rules on the definition of model transformation process
- Apply semantic check rules on the definition of model transformation process.

Here, basic requirements for achieving the final objective are given. In the next sub-section, these functions will be added into the architecture of our solution.

## 3.2 The Architecture of Theoretical Solution

Figure 2 illustrates the architecture of the solution.

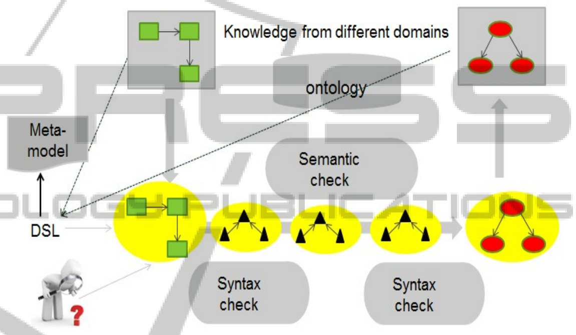


Figure 2: Theoretical solution architecture.

The purpose of this work is to transform a source model to a specific target model automatically. The source and target models could be built in different modelling languages ("UML (Fowler, 2004)", "BPMN (White, 2004)", etc.). In order to ignore the modelling language and use the semantic and syntax check rules on the definition of transforming process, we suppose to develop several intermediary models (building with a specific modeling language). Based on this idea, we also define a meta-meta-model. In order to add semantic check rules to our model transformation methods, we need a specific ontology to provide the data basis. A meta-meta-model that consistent to this ontology will greatly simplify the transformation process (this is the reason that we do not use the existing meta-meta-models, such as: MOF (OMG,2002)). We deduce the meta-models for both source and target models that conform to the meta-meta-model. Then using the semantic and syntax check rules on the meta-model level to build transformation mappings. During the transformation process, the providers of the source models could check the intermediary models.

To be efficient, all the semantic and syntax check rules should be used on the same kind of models (intermediary models). So, we divide the transformation process into three steps: from the source model to the intermediary model, among the intermediary models and from the intermediary model to the target model.

The first and third steps just transform the format of the model (the content and concepts do not change). The second transformation step contains three phases: first, using syntax check to change the syntax part of the source model; next, with the help of the “ontology” (which contains domains-cross knowledge), apply the semantic check rules on the intermediary model to transform the content and concepts; finally, using the syntax check again to transform the intermediary model to its final version. The providers of the source models could check the intermediary models here and valid the process. The syntax and semantic check rules are applied during the transformation process to make the transform mappings, and all the check rules work on the intermediary models.

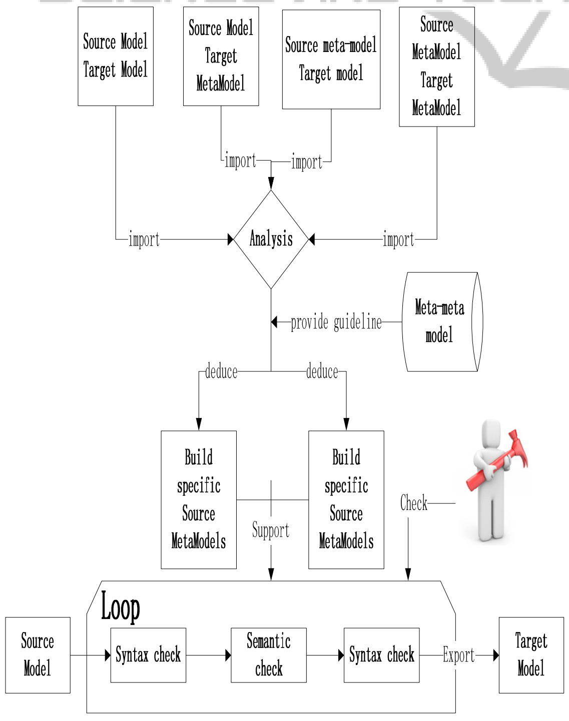


Figure 3: Transformation process.

Figure 3 explains the main steps to define an automatically model transformation process. In order to use the syntax and semantic check rules, the source meta-model and target meta-model should conform to a generic meta-meta-model. We use the combination of source model, target model and their original meta-models, to deduce the meta-models respecting to our meta-meta-model. (In next subsection, we will explain the principles of our meta-meta-model and how can we use it to generate the other meta-models.) Then, semantic and syntax check rules are applying several times, to transform the source model to the target model through intermediary models. At the end of this loop, the provider of the source model could check the intermediary model to see if it is identical to the source model or not.

## 4 KEY ISSUES

This section illustrates two of the key aspects within our solution. They are: the definition of the meta-meta-model and the semantic check rules used on the transformation process.

### 4.1 The Meta-Meta-model

The explanation of the meta-meta-model will be given with the help of figure 4. This meta-meta-model works on the top abstract level of all the other models.

As shown in figure 4, there are ten core elements in this meta-meta-model.

- “Environment”, describes the context of a system such as crisis situation, supply chain, etc. If two “Environments” describe the same context of a specific situation, the relationship between them is “same”; otherwise, the two “Environments” are different.
- “Model” is the core concept in this meta–meta-model. In the context of our solution, every source, target models and their meta-models (deduced or imported) could be regarded as “Model”. Every model contains the component: “Element”.
- “Element” could contain another “Element” (e.g. in BPMN (White, 2004) modelling context, a pool contains a lane). The “Element” has two inheritance classes: “Concept” and “Link”.
- “Concept” stands for an object; it is used to describe a subject that exists in the world.
- “Link” is the relationship between Elements. Every “Link” has two ends (there are two relationships between “Link” and “Element”: “from”, “to”). “Element” contains “Property”.
- “Property” is used to identify and explain the object that contains it. Each “Property” has a “Data Type”.

- "Data Type" should be a "Primitive Type" or an "Enumeration".

The most important part of this meta-meta-model is the element "SemanticRelation". It helps to express the semantics relations between elements. "Environment", "Model", "Element" and "Property" inherit from this abstract class. This means that any items from these class may have "sameAs" or "near" relation with the other items.

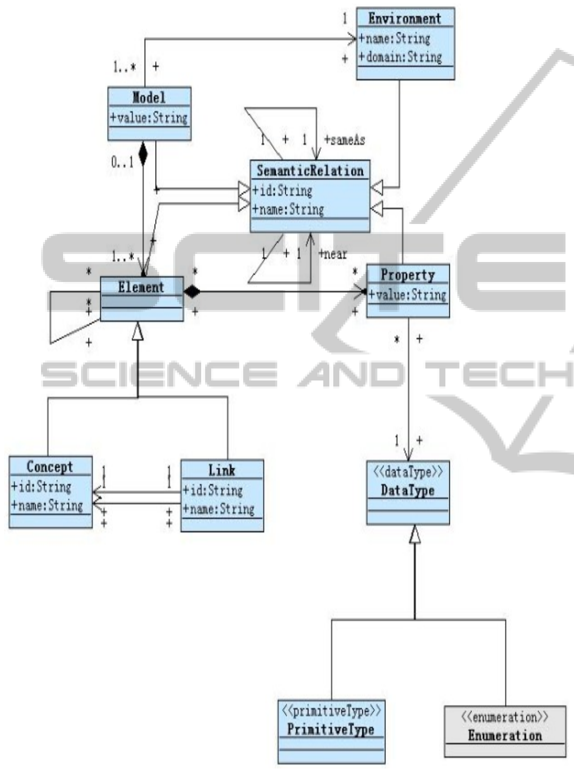


Figure 4: The meta-meta-model overview.

## 4.2 Semantic Mapping Approach

In practice, semantic check principles have been used in many different domains to solve real problems (we explained this point in section 2). For the model transformation domain, semantic check methods can also help.

We rely on the existing semantic check rules defined in (Boissel-Dallier, 2012). Here, only the core idea of the semantic check rules will be shown. The detail of the algorithms and dealing process of the semantic check rules will not be illustrated.

The basic idea is: in order to do semantic matchmaking between models from different domains; a common semantic profile (Boissel-Dallier, 2012) should be defined first. According to the ontology we created, we define this semantic profile. Based on the semantic profile, we can compute the semantic distance measurement between the elements from the source model the elements from the target model. After getting the computed results, we can do the matchmaking between the two models.

Here, we define the algorithms that are used to compute the "sameAs" or "near" relationship for the objects of "semanticRelation" class. In practice, we compute the average semantic relation value between source meta-model and target meta-model within five groups: "Environment", "Model", "Concept", "Property" and "Link". The parameters that we use in these algorithms are assumed for the first test. The details of calculating these average values shown as follow:

1) For the "Environment"

This classification depends on the users who provide the source and target model. The ontology (create based on the structure of the meta-meta model) records all the categories of the imported "Environment". The semantic relation between two "Environments" is "sameAs" or "different".

$$E\_SR\_V = \begin{cases} 1 & \text{if "sameAs"} \\ 0 & \text{if "different"} \end{cases} \tag{1}$$

If the source model and the target model come from the same "Environment", then this value is "1". If not, this value is "0".

2) For the "Model"

The average semantic relation value between two models (deduced source meta-model and deduced target meta-model) could be calculated using the formula.

$$M\_SR\_V = 0.5*E\_SR\_V + 0.4*SR\_Name + 0.1*Num\_Concept \tag{2}$$

The "E_SR_V" is the value calculated from the first formula. The "SR_Name" is the semantic relation between the names of the two models; it can be calculated using existing word recognition algorithm. The "Num_Concept" is the number of "Concept" involved in a model. Using this formula, the semantic relation value, which for the "Model" (defined in the meta-meta-model), is computed.

3) For the "Property"

According to our generic meta-meta-model, "property" is component of "Concept" and "Link". Each "property" (in deduced source meta-model) has a semantic relation value with every "property" in the target meta-model, respectively. The formula for this is:

$$\text{P\_SR\_V} = 0.3*\text{SR\_Name} + 0.2*\text{type} + 0.5*\text{value} \qquad (3)$$

Here, the value of the “SR_Name” is calculated in the same way as explained above. If the “type” of the two “property” is the same, its value is “1”; otherwise, its value is “0”. The same calculation rule is used on “value”.

4) For the “Concept”
“Concept” is the core element in the meta-meta model, the formula for calculating the semantic relation between two “Concepts” is:

$$\text{C\_SR\_V} = 0.3*\text{SR\_Name} + 0.6*\text{SR\_Pro} + 0.1*\text{M\_SR\_V} \qquad (4)$$

In this formula, the “SR_Pro” parameter is calculated using the following formula:

$$\text{SR\_Pro} = \frac{2*\sum \text{Max (P\_SR\_V i)}}{\text{Num\_SP} + \text{Num\_TP}} \qquad (5)$$

In this algorithm, the number of properties of both source model concept “Num_SP” and target model concept “Num_TP” should be calculated first. Then, select the max value of each “P_SR_V” (between the properties of source concept and the target concept), and add them together. For example, to calculate “C_SR_V” between “Concepts A” and “Concept B”; “Concept A” has two properties while “Concept B” has three properties. Each property of “Concept A” has a “P_SR_V” with each property of “Concept B”. The max value of each pair will be selected and added together.

5) For the “Link”
The semantic relation value also computed for the “Link”. The formula for this is:

$$\text{L\_SR\_V} = 0.1*\text{SR\_Name} + 0.35*\text{SR\_FC} + 0.35*\text{SR\_TC} + 0.2*\text{P\_SR\_V} \qquad (6)$$

In this formula, the “SR_SC” stands for the semantic relation value between the two “from concept” (every link has two concepts as two ends). The “SR_TC” means semantic relation value between the two “to concept”.

With the help of these six formulas, the definition of model transformation process (mapping rules) could be automatically generated. To explain this idea more clearly, a case study based on this algorithm will be illustrated in next section.

## 5 CASE STUDY

At this moment, we have defined the meta-meta-model, and illustrated the algorithm used to calculate the semantic relation value (which can provide help to the automatically definition of model transformation process).

In this section, a use case aiming at transforming the “UML (Fowler, 2004)” model (here, we just use a UML class model) to the “OWL (McGuinness and Van Harmelen, 2004)” model, will be shown. This case concerns a part of the whole transformation process: using the input models to deduce the meta-models that conform to the generic meta-meta-model and calculating the semantic relation value between the two models. The process of deducing the meta-models (conform to our meta-meta-model) from the input models is shown in the following tables. Two very simple models are created for this case (using “UML” and “OWL”, respectively).

The “UML” model shows in figure 5.

In this model, there are three classes: “Student”, “Teacher” and “Course”. The relationship between “Student” and “Course” is “to_choose”; the relationship between “Teacher” and Course” is “to_give”. There is a total of eight properties for the three classes.

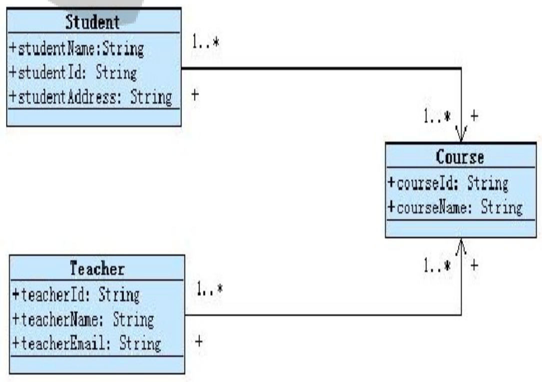
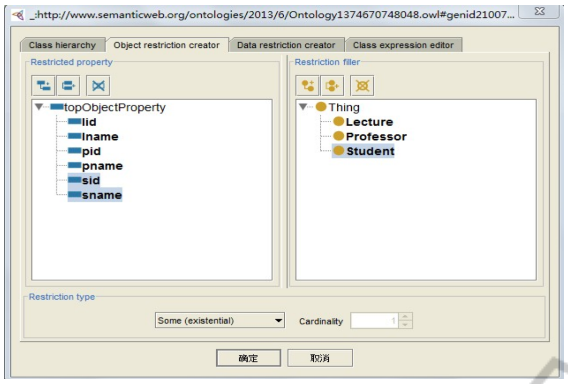


Figure 5: UML class model.

The “OWL” model is shown in figure 6.

According to the meta-meta model, we deduce the meta-models for both the “UML” model and the “OWL” model. Both of them are “Model”; all the classes stand as “Concept”, the properties of these classes could be regarded as “Property” and the “Link” in the meta-meta model replaces the relationships between the classes.

The “Environments” of the two models are similar. Table 1 shows the E_SR_V.



Some (existential)
Cardinality 1

Figure 6: OWL ontology model.

Table 1: E_SR_V of this case.

| Environment | UML |
|---|---|
| OWL | 1 |

Table 1 shows the E_SR_V value between "UML" and "OWL" environment is "0".

After calculating the E_SR_V, the next step is to calculate the M_SR_V, table 2 shows this (the algorithm used is illustrated above).

Table 2: M _SR_V of this case.

| Model | UML |
|---|---|
| OWL | 0.42 |

The most complex step is to calculate the P_SR_V value, table 3 shows this process.

Table 3: P _SR_V of this case.

| Property | lid | lname | pid | pname | sid | sname |
|---|---|---|---|---|---|---|
| courseId | 0.26 | 0.2 | 0.26 | 0.2 | 0.26 | 0.2 |
| courseName | 0.2 | 0.32 | 0.2 | 0.32 | 0.2 | 0.32 |
| studentName | 0.2 | 0.32 | 0.2 | 0.32 | 0.22 | 0.34 |
| studentId | 0.26 | 0.2 | 0.26 | 0.2 | 0.28 | 0.22 |
| studentAdd | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 |
| teacherId | 0.26 | 0.2 | 0.26 | 0.2 | 0.26 | 0.2 |
| teacherName | 0.2 | 0.32 | 0.2 | 0.32 | 0.2 | 0.32 |
| teacherEmail | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 |

In this case study, all the properties' type is "String", and they have no value (just on "class" level, no objects exist) because the use-case should be as simple as possible (just illustrating how to use the algorithm mentioned above). So, the value of P _SR_V for each pair of property has no practical significance.

Based on the P _SR_V value (that known from table 3), the C_SR_V value could be calculated. Table 4 shows the result of this process.

Table 4: C_SR_V of this case.

| Concept | Lecture | Student | Professor |
|---|---|---|---|
| Student | 0.232 | 0.548 | 0.232 |
| Teacher | 0.232 | 0.232 | 0.472 |
| Course | 0.532 | 0.232 | 0.232 |

According to the records of this table, the mapping rules for the model transformation (on class level) could be made. After getting all the C_SR_V values, the final step is to calculate the L_SR_V value. In this case, there are two links (just has a name, they do not have properties).

Table 5: L_SR_V of this case.

| Link | select | teach |
|---|---|---|
| to_choose | 0.428 | 0.267 |
| to_give | 0.267 | 0.36 |

Based on all the values recorded in these five tables above, the model transformation process could be automatically defined. We define the mapping rules between the concepts and links of the source and target models or on their meta-model levels. We can search the recorded table above and select the maximum average semantic value for each pair of concepts and links (from source model and target model, respectively). Then, build the mappings between such kinds of pairs. After getting all the mapping pairs, the model transformation rules are automatically defined.

In practical, when creating these five tables, a specific ontology should provide data basis, and pre-define rules would be used to judge the semantic relation.

## 6 CONCLUSIONS

This paper exposes an approach to automatically define model transformations. Comparing to the existing methodologies and principles of this field, the main contribution of our work is to add the semantic check rules on the transformation process. In order to apply semantic check rules on models, we create a generic meta-meta-model. Furthermore, based on this meta-meta-model, we build ontology to provide the data basis for the semantic check rules.

Automatically defining model transformation process is a big challenge, it will bring great help to solve the complex practical problems (reduce human efforts: avoid the repetitive work) quickly. At this moment, researchers have already been focused on finding solutions to do the semi-automatic model transformations for some specific domains (UML models to Database models, BPMN models to UML models, etc.). Our proposal aims at solving the model transformation more efficient and general (regardless the domains that the models come from and the modelling languages). The idea that we proposed above needs to be improved and it can give some inspirations to the other model transformation methods at the same time.

The further work of our proposal focuses on two aspects: fulfilling the ontology and improving the efficiency of the algorithms (doing the semantic check). The more valuable information stored in the ontology, the more precise transformation mapping rules could be made. More reasonable semantic detection algorithms can also improve the accuracy of the mapping rules.

## REFERENCES

Agrawal, A., Karsai, G., Shi, F., 2003. Graph Transformations on Domain-Specific Models. Under consideration for publication in *the Journal on Software and Systems Modeling*.

Bénaben, F., Mu, W., Truptil, S., Pingaud, H., 2010. Information Systems design for emerging ecosystems. 4th IEEE International Conference on Digital Ecosystems and Technologies (DEST).

Bézivin, J., 2006. Model Driven Engineering: *An Emerging Technical Space. Generative and Transformational Techniques in Software Engineering Lecture Notes in Computer Science* Volume 4143, pp 36-64.

Boissel-Dallier, N., 2012. Réconciliation sémantique des données et des services mis en oeuvre au sein d'une situation collaborative. Les thèses en ligne de l'INP.

Bollati, V., Vara, J. M., Jimenez, Á., Marcos, E., 2013. *Applying MDE to the (semi-)automatic development of model transformations. Information and Software Technology*, Volume 55, Issue 4, Pages 699–718.

Castro, D. V., Maros, E., Vara, J. M., 2011. *Applying CIM-to-PIM model transformations for the service-oriented development of information systems. Information and Software Technology*, Volume 53, Issue 1, Pages 87–105.

Chen, D., Doumeingtsb, G., Vernadatc, F., 2007. *Architectures for enterprise integration and interoperability: Past, present and future. Computers in Industry*, Volume 59, Issue 7.

Czarnecki, K., Helsen, S., 2003. Classification of Model Transformation Approaches. OOPSLA'03 *Workshop on Generative Techniques in the Context of Model-Driven Architecture.*

Del Fabro, M. D., Bézivin, J., Jouault, F., Breton, E., 2005. AMW: *A Generic Model Weaver. 1ère Journées sur l'Ingénierie Dirigée par les Modèles: Paris.*

Del Fabro, M. D., Valduriez, P., 2008. *Towards the efficient development of model transformations using model weaving and matching transformations. Software & Systems Modeling*, July 2009, Volume 8, Issue 3, pp 305-324.

Fowler, M., 2004. *A brief guide to the standard object modelling language*. UML Distilled Third Edition.

Grangel, R., Bigand, M., Bourey, J. P., 2010. Transformation of decisional models into UML: application to GRAI grids. International *Journal of Computer Integrated Manufacturing*, Volume 23, Issue 7.

Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I., 2007. ATL: A model transformation tool. *Science of Computer Programming*, Volume 72, Issues 1–2.

Lano, K., Kolahdouz-Rahimi, S., 2013. Constraint-based specification of model transformations. *Journal of Systems and Software*, Volume 86, Issue 2.

Ly, L. T., Rinderle, S., Dadam, P., 2006. Semantic Correctness in Adaptive Process Management Systems. Business Process Management, Lecture Notes in Computer Science Volume 4102, pp 193-208.

McGuinness, D. L., Van Harmelen, F., 2004. *OWL Web Ontology Language Overview*. W3C Recommendation.

Object Management Group, 2002. MOF 2.0 Query / Views / Transformations RFP. OMG Document.

Rajsiri, V., Lorréa, J. P., Bénaben, F., Pingaud, H., 2010. Knowledge-based system for collaborative process specification. *Computers in Industry*, Volume 61, Issue 2, Pages 161–175.

Taentzer, G., Ehrig, K., Guerra, E., Lara, D. J., Lengyel, L., Levendovsky, T., Prange, U., Varro, D., Varro-Gyapay, S., 2009. Model transformation by graph transformation: A comparative study. *In Proc. Workshop Model Transformation in Practice.*

White, S. A., 2004. Introduction to BPMN. IBM Cooperation.