Applying a Semantic & Syntactic Comparisons Based Automatic Model Transformation Methodology to Serve Information Sharing

Tiexin WANG, Sebastien TRUPTIL, and Frederick BENABEN
Centre Genie Industriel, University de Toulouse - Mines Albi, Albi, France

Abstract - Information sharing, as an aspect of information and knowledge engineering, attracts more and more attention from researchers and practitioners. Since a large amount of cross-domain collaborations are appearing, exchanging and sharing information and knowledge among various domains are inevitable. However, due to the vast quantity and heterogeneous structures of information, it becomes impossible to maintain and share cross-domain information relying mainly on manual effort. To enhance the efficiency of sharing information, this paper presents an automatic model transformation methodology. Comparing to traditional model transformation methodologies, this methodology shields the general weaknesses: low reusability, contain repetitive tasks and involve huge manual effort, etc., by combining semantic and syntactic checking measurements into a refined transformation process. Moreover, the semantic and syntactic checking measurements are supported by software tool; in this way, manual effort is replaced from the information sharing/exchanging process.

Keywords: Information sharing, automatic model transformation, semantic and syntactic checking

1 Introduction

Data, as the basis of information and knowledge, are “symbols that represent properties of objects, events and their environments” [1]. Data are products of observation”; furthermore, “information is contained in descriptions, answers to questions that begin with such words as who, what, where, when and how many”. Information systems generate, store, retrieve and process data” and “information is referred from data”. Fig.1 shows the relationships among data, information, knowledge and wisdom.

Data is raw; it can exist in any form without significance. Information is generated by adding meaning (relational connection) on data, for example: data stored in a relational database could be regarded as information. Knowledge, which exists on a higher understanding level than information, is “a deterministic process” [1].

Based on the context of information sharing, Fig.2 shows the relationships among “nature languages”, “data & information” and “specific domains”.

Fig. 1 Different presentation layers from data to wisdom [1]

Fig. 2 Nature languages, data & information and specific domains

Data and information are presented mainly by nature languages (also with the help of mathematical symbols, diagrams, etc.). Information applied on specific domain, could be regarded as knowledge; while this kind of knowledge may be regarded as information (or data) by other domains.

Due to the Internet, vast amount of heterogeneous data and information appear and they are provided by different sources. Three typical sources are: Internet of things (IoT) [2], people (e.g. experts, skilled workers), and specific domain ontologies. Data and information are maintained and updated on-the-fly by their sources. In order to use data provided by other sources, it seems to be necessary to transform them (at least on format level) to information. For instance, in enterprise engineering domain, as stated in [3], collaborations among various enterprises appear frequently. In some terms, the efficiency of information exchanging (sharing) among heterogeneous partners determines if the collaboration goal could achieve or not. However, this kind of exchanging (sharing) process always involves huge manual work; with the explosion in the amount of data, it is impossible to maintain this process relies mainly on manual work.

Model transformation theories provide a possible solution to share and exchange data/information among
heterogeneous partners. However, there exist several weaknesses in traditional model transformation practices [4]: low reusability, contain repetitive tasks and involve huge manual effort, etc. These weaknesses limit the usage of model transformation theories to serve to cross-domain problems, and also reduce the efficiency of model transformation developing process. For applying model transformation theories to data/information sharing, this paper proposes an automatic model transformation methodology (AMTM) that combines semantic and syntactic checking measurements into model transformation process.

This paper is divided into five sections. The second section presents related work of model transformation domain. Then, a refined model transformation process is stated in the third section. The fourth section presents semantic and syntactic checking measurements in detail. Finally, a conclusion is presented.

2 Related work

This section is divided into two subsections: i) shows the comparisons of several prominent model transformation techniques, and ii) illustrates the category of model transformation practices.

2.1 Model transformation techniques

In this subsection, three popular model transformation techniques are presented and compared briefly.

“ATLAS transformation language (ATL)” [5] is a model transformation language and toolkit. Its architecture composes of three layers: ATLAS Model Weaving (AMW) [6], ATL and ATL Virtual Machine (ATL VM). ATL provides ways to produce a set of target models from a set of source models.

“Query/View/Transformation (QVT)” [7] is a standard set of languages for model transformation defined by the “Object Management Group”; it covers transformations, views and queries together. The QVT standard defines three model transformation languages. All of them operate on models which conform to Meta-Object Facility (MOF) 2.0 [7] meta-models.

“Visual Automated Model Transformations (VIATRA2)” [8] is a unidirectional transformation language based mainly on graph transformation techniques. The language operates on models expressed following the VPM meta-modeling approach [9].

Table 1 shows the comparisons on some characteristics among the three techniques.

<table>
<thead>
<tr>
<th>name</th>
<th>hybrid</th>
<th>rule scheduling</th>
<th>M-to-N</th>
<th>note</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATL</td>
<td>yes</td>
<td>implicit internal explicit</td>
<td>yes</td>
<td>self-executed</td>
</tr>
<tr>
<td>QVT</td>
<td>no</td>
<td>implicit internal explicit</td>
<td>yes</td>
<td>based on MOF 2.0</td>
</tr>
<tr>
<td>VIATRA2</td>
<td>yes</td>
<td>external explicit</td>
<td>yes</td>
<td>graph rewriting</td>
</tr>
</tbody>
</table>

As a short conclusion, many model transformation techniques have been developed. According to the purpose, these techniques could be divided into two groups: serve to “cross-domain” and serve to specific domain.

Normally, domain specific model transformation techniques focus on and provide single solution to particular problematic. The usage of these techniques is limited, and they are not flexible for solving some special cases. On the other hand, cross-domain model transformation techniques provide a wide range of functions, and thus are always complex. So, it needs more time to learn to use this kind of technique properly. The common problem of existing model transformation techniques is: involve huge manual effort and require precondition (e.g. on modeling) to use. To solve this problem, an automatic model transformation methodology that serves to cross-domain, is a possible solution.

2.2 Model transformation category

According to [10], there are two main kinds of model transformation approaches. They are: model-to-code approaches and model-to-model approaches.

As model-to-code approaches, there are two categories: “Visitor-based approaches” and “Template-based approaches”. And for model-to-model approaches, there are five categories: Direct-Manipulation Approaches, Relational Approaches, Graph-Transformation-Based Approaches, Structure-Driven Approaches and Hybrid Approaches.

AMTM is a model-to-model model transformation methodology. Based on AMTM, there are two kinds of model transformation situations. Fig. 3 illustrates the two kinds of situations.

![Fig. 3 Model transformation situations](image)

AMTM is created on the basis of meta-model based model transformation methodology.

In situation (a), the target meta-model is the evolitional version of the source meta-model; so, source models that conform to the source meta-model should be transformed to the new version of models that are conformed to the evolitional target meta-model. For this situation, large amount of research work has been done and different theories and practices have been developed. One of the mature theories is “COPE” [11].

In situation (b), source meta-model and target meta-model are created for different purposes. In order to transform source models to target models (conforming to the source and target meta-models, respectively), model transformation mappings should be built on their meta-model level.

AMTM provides a solution to both situation (a) and situation (b); furthermore, there is no precondition of applying AMTM on both of them.
3 Overview of the methodology

This section presents the detail of the automatic model transformation methodology (AMTM). It is divided into two subsections: first subsection illustrates the basic theories of AMTM and second subsection shows the working mechanism of this methodology.

3.1 Basic theories

The basic theories of AMTM are presented in two parts: the theoretical main framework and the meta-meta-model (MMM) involved in it.

3.1.1 Theoretical main framework

AMTM is created on the basis of a theoretical main framework that shows in Fig. 4.

![Fig. 4 Theoretical main framework](image)

This theoretical main framework is created based on the work stated in [12]. It illustrates the fundamental theories of AMTM.

For the reason “models are conformed to their metamodels [13]”, the potential shared items (between two models) could be traced on meta-model layer. AMTM relies on the meta-model layer (mappings are defined here among shared concepts). 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 between MMs) can be applied onto the extracted knowledge. The transformed knowledge and additional knowledge may be finally used to create the shared part and the specific part of the target model.

In order to automatically generate the model transformation mapping rules, semantic and syntactic checking measurements are combined into transforming process (detecting shared concepts on meta-model layer). The principle of applying S&S on model transformation process is stated in [6]. The mechanism of applying S&S in AMTM is defined in the MMM, which shows at the top of Fig. 4.

3.1.2 The meta-meta model

A meta-meta model defines the rules for meta-modeling; there exists several meta-modeling architectures, for example “MOF: Meta-Object Facility” [7]. However, these architectures aim to serve to general purpose; they define their own semantic and syntax. For this project, a specific meta-meta model serves specific to model transformation domain is more preferred. So, within the theoretical main framework, we define this MMM (adapted from MOF) to serve to AMTM. Fig. 5 shows the detail of this MMM.

![Fig. 5 The meta-meta model](image)

There are ten core elements in this meta-meta-model. As models may come from various domains, a class named “Environment” is defined to stand for these domains. All the model instances are represented by the class “Model”, every model belongs to a specific “Environment”. “Model” is made of “Element”, which has two inheritances: “Node” and “Edge”. “Nodes” are linked by “Edge” based on their “roles”. “Element” has a group of “Property”, the “Property” could identify and explain the “Element”. “Property” has a data type: “Primitive Type” or “Enumeration”; to a certain extent, data type could differentiate “Property”. Another two key items shown in Fig. 5 are: “Semantic Relation” and “Syntactic Relation”. They exist on different kinds of items (e.g. between a pair of elements). Model transformation rules are generated based on these two relations that are existed between different items (i.e. elements and properties).

3.2 AMTM working mechanism

In AMTM, a complete model transformation is regarded as an iterative process: between source model and target model, there could be several intermediate models. An intermediate model could be target model and source model for different transform iterations.

Fig. 6 shows the iterative issue. In each iteration phase, the specific parts from the source meta-model are stored in ontology; the additional knowledge for the specific parts of the target model are enriched by extracting content from the same ontology.
To deal with the granularity issue involved, in each iteration, transformation process is divided into three steps: matching on element level, hybrid matching and auxiliary matching.

### 3.2.1 Matching on element level

According to MMM, meta-models are made of elements. So, model transformation mappings should be defined mainly among elements (nodes and edges); if two elements (come from source model and target model, respectively) stand for the same concept, a mapping should be built between them. The mechanism of defining matching pairs on element level is illustrated by an example shown in Fig. 7.

![Fig. 7 Matching on element's level](image)

The source meta-model has ‘n’ elements and ‘m’ for the target meta-model; the number of comparisons between the two models on element’s level is: “m*n”.

Within each element’s pair, there exists an “Ele_SSV” value. “Ele_SSV” stands for “element’s semantic and syntactic value”; it is calculated based on the elements’ names and their properties. The calculation rule of “Ele_SSV” is shown in (1).

\[
\text{Ele}_\text{SSV} = \text{name_weight} \times \text{S}_\text{SSV} + \text{property_weight} \times \frac{\sum_{i=1}^{x} \text{Max(P}_\text{SSVi})}{x}
\] (1)

In (1), “name_weight” and “property_weight” are two impact factors for parameters “elements’ names” and “elements’ properties”. Both of the two values are between 0 and 1; the sum of them is 1. “S_SSV” stands for “string semantic and syntactic value”; it is calculated based on the words (i.e. element’s name). “P_SSV” stands for “semantic and syntactic value between a pair of properties”. “x” stands for the number of properties of a specific element from source meta-model (e.g. element E A1).

The example shown below is to calculate the “Ele_SSV” value within the element’s pair of “E A1” and “E B1”.

“E A1” has number “x” properties and “E B1” has number “y” properties; within each of the “x*y” pairs of properties, there exists a “P_SSV”. Equation (2) shows the calculating rule of “P_SSV”.

\[
P_\text{SSV} = \text{pn_weight} \times \text{S}_\text{SSV} + \text{pt_weight} \times \text{id_type}
\] (2)

In (2), “pn_weight” and “pt_weight” are two impact factors for the parameters “properties’ names” and “properties’ types”. They play the same role as the impact factors in (1). “S_SSV” is the same as stated in (1); this time, it stands for the semantic and syntactic value between two properties’ names. “id_type” stands for “identify properties type”. If two properties have the same type, this value is 1; otherwise, this value is 0.

Based on (1) and (2), each element (E A1, E A2…) of the source model gets a maximum “Ele_SSV” with a specific target model element (E B1, E B2…). Moreover, a matching pair of two elements requires building mappings among their properties. The mechanism of choosing matching pairs (on both element and property level) will be illustrated later.

### 3.2.2 Hybrid matching

After first matching step, some of the elements are still unmatched; even the matched elements, some of their properties are still unmatched. The hybrid matching step focuses on these unmatched items.

This matching step works on property level, all the matching pairs would be built among properties (come from both the unmatched and matched elements). All the unmatched properties from source model will be compared with all the properties from target model. The mechanism of building such matching pairs is also depending on semantic and syntactic checking measurements (based on properties’ names and types).

In hybrid matching step, all the matching pairs are built on property’s level. This step breaks the constraint: property matching pairs only exists within matched element’s pairs; this constraint is the main granularity issue involved in model transformation process. However, it is also necessary to consider about the influence from element’s level when building mappings in hybrid matching step. The matching mechanism of this step shows in (3).

\[
\text{HM}_\text{P}_\text{SSV} = \text{el_weight} \times \text{S}_\text{SSV} + \text{pl_weight} \times \text{P}_\text{SSV}
\] (3)

In (3), “HM_P_SSV” stands for “hybrid matching property semantic and syntactic value”. “el_weight” and “pl_weight” are two impact factors for the parameters “element level” and “property level”. They perform the same functions as the impact factors in former formulas. “S_SSV” is calculated between two elements’ names (elements contain the two properties). “P_SSV”, as stated in (2), calculates the syntactic and semantic relation between two properties based on their names and types.

### 3.2.3 Auxiliary matching

After the first and second matching steps, all the shared parts between source model and target model are regarded to be found. However, according to the iterative model
transformation process, there are still some specific parts that should be stored as capitalized knowledge and reused as additional knowledge. This matching step focuses on the mechanism of storing and reusing these specific parts.

All the unmatched items from source model, which regarded as specific parts, are stored in ontology (which is called “AMTM_O” within this project). AMTM_O designed with the same structure as the MMM. For a complete model transformation process, the capitalized knowledge from former iterations could be used as the additional knowledge to enrich the target models that are generated in the latter iterations.

3.2.4 Matching pair choosing mechanism

According to the three former sub-subsections, the relation between two elements is represented by a value between 0 and 1, which calculated by semantic and syntactic comparisons. Based on this value, each element from source model could be matched with “zero to several” elements from target model. The mechanism of selecting elements matching pairs depends on the range of this value. Fig. 8 reveals the basic principle.

![Fig. 8 Matching pair choosing mechanism](image)

For choosing element’s matching pairs, two threshold values: 0.5 and 0.8 are assigned. As shown in Fig. 8 situation (a), if two elements have a relation value in region 1 (value between 0.8 and 1), a transformation mapping is built between them; if this value is in region 2, a potential mapping exists between the two elements; else, if this value is in region 3, no mappings will be built between them.

Fig. 8 situation (b) shows the mechanism of choosing matching pairs of two words (i.e. elements’ and properties’ names). Between two words, strong semantic relation means high potential of making mappings. Region 1 stands for two words that have close relationship: could transform to each other. Region 2 stands for two words have strong relationship: potential transform pair. Region 3 means two words have weak relationship: low possibility to transform to each other. Region 4 is special; it stands for words that have close syntactic relation but very weak semantic relation. For example, word pair: common and uncommon, they could not transform to each other. But in some specific domain (e.g. medicine), syntactic relation may be more important than semantic relation.

In this way, an element (or a property) may have several potential matching items. So, from source model to target model, a “many-to-many” (granularity issue solved in this way) matching relationships are built on both element level and property level.

4 Syntactic & semantic checking

Semantic and syntactic checking measurements play a key role in AMTM. They work together to define a relationship (stands by a value between 0 and 1) between two words. As stated in (1) and (2), the “S_SSV” stands for this value; the calculation method of “S_SSV” is shown in (4).

\[
S_{SSV} = sem_{weight}\times S_{SeV} + syn_{weight}\times S_{SyV} \quad (4)
\]

In (4), “sem_weight” and “syn_weight” are two impact factors for semantic value and syntactic value between two words. The sum of them is 1. “S_{SeV}” stands for the semantic value between two words, while “S_{SyV}” stands for the syntactic value.

4.1 Syntactic Checking Measurements

Syntactic checking measurement is used to calculate the syntactic similarity between two words. This kind of checking methodology is based on the alphabets that are contained in the words. Several syntactic checking methodologies have been presented and compared in [14].

The syntactic checking measurements involved in AMTM could be divided into two steps: i) pretreatment: focuses on detecting two words that are in different forms (e.g. tense, morphology) stand for a same word. ii) “Levenshtein Distances” algorithm [15].

For pretreatment, “Porter stemming algorithm” is chosen to be applied in AMTM. “Levenshtein Distances” algorithm is applied between two different words. It calculates the syntactic similarity between two words; the mechanism of using it has been stated in [15].

Equation (5) shows the calculation rule of syntactic relation between two words: word1 and word2, which based on “Levenshtein distances”.

\[
S_{SyV} = 1 - LD / Max (word1.length, word2.length) \quad (5)
\]

In (5), “S_{SyV}” stands for the syntactic similarity value between word1 and word2; “LD” means the “Levenshtein distances” between them. The value of “S_{SyV}” is between 0 and 1; the higher of this value means the higher syntactic similarity between two words.

4.2 Semantic checking measurements

Different to syntactic checking measurement (relies just on the two comparing words); semantic checking measurement relies upon a huge semantic thesaurus.

A huge semantic thesaurus (AMTM_ST) has been created for serving to AMTM, and AMTM_ST is created on the base of “WordNet” [16]. Fig. 9 shows the structure of AMTM_ST.

As shown in Fig. 9, there are three kinds of items stored in AMTM_ST.

- **Word base**: contains normal English words (nouns, verbs and adjectives).
- **Sense base**: contains all the word senses; a word could have “one or several” senses. For example: “star”: it has six senses; as noun, it has four senses; as
• “Synset” base: a group of word senses that own synonym meanings; semantic relations are built among different synsets.

Fig. 9 Structure of AMTM_ST

There are seven kinds of semantic relations that defined among synsets. For each of the semantic relations, a specific value (between 0 and 1) is assigned to it. Table II shows these semantic relations and their corresponding values.

<table>
<thead>
<tr>
<th>Semantic relation</th>
<th>S_ScV</th>
<th>Remark</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>synonym</td>
<td>0.9</td>
<td>words from the same synset</td>
<td>shut &amp; close</td>
</tr>
<tr>
<td>hypernym</td>
<td>0.6</td>
<td>two synsets have this relation</td>
<td>person-close</td>
</tr>
<tr>
<td>hyponym</td>
<td>0.8</td>
<td>two synsets have this relation</td>
<td>creator-person</td>
</tr>
<tr>
<td>similar-to</td>
<td>0.85</td>
<td>only between two adjectives</td>
<td>perfect &amp; ideal</td>
</tr>
<tr>
<td>antonym</td>
<td>0.2</td>
<td>words have opposite meanings</td>
<td>good &amp; bad</td>
</tr>
<tr>
<td>iterative hypernym</td>
<td>0.6(a)</td>
<td>iterative hypernym relation</td>
<td>person-creator-maker-author</td>
</tr>
<tr>
<td>iterative hyponym</td>
<td>0.8(a)</td>
<td>iterative hyponym relation</td>
<td>author-maker-creator-person</td>
</tr>
</tbody>
</table>

In Table II, all the “S_ScV” values are assigned directly (based on experience); these values could be assigned with different values by different application domains.

As a word may have different word senses (furthermore, may belong to different synsets), there might be several semantic relations that exist between two words. So, the number of “S_ScV” values between two particular words is not limited to one. In this project, we focus on finding the maximum “S_ScV” value between two words. Based on this cognition, the process of detecting semantic relations between two words should be serialized.

In order to define the semantic relation between two words, there are several steps to follow:

• First, locating two words (element’s or property’s names) in AMTM_ST.
• Second, finding all the word senses of the two words and grouping these word senses into two sets.
• Third, tracing all the synsets, which the two sets of word senses belong to, and grouping these synsets into two groups.

After getting two synsets groups, the final step is to detect the semantic relations that exist among all the possible synset pairs (one from word1 side, the other from word2 side). For detecting five kinds of semantic relations: synonym, similar-to, hypernym, hyponym and antonym, the basic principle is: search all the synsets that have these five kinds of semantic relations with the synsets in “synset group of word1”, then comparing if there exist one synset in “synset group of word 2”, which is the same as one of the located synsets.

The detecting process of “iterative hypernym” and “iterative hyponym” semantic relations is same. The basic idea is: locating the synsets that have hypernym relation with word1’s synsets iteratively and comparing with word2’s synsets, in order to find two same synsets.

The basic information of doing all these semantic checking measurements is provided by AMTM_ST. So, the content of AMTM_ST should be really huge. For serving AMTM, AMTM_ST contains 147306 words, 206941 word-senses and 114038 synsets.

4.3 Short conclusion

By using syntactic and semantic checking measurements, a “S_SSV” value could be generated between two words. When the words stand for properties’ names, an approximate value between two properties is generated (properties’ types are also considered). When the words stand for elements’ names, an approximate value between two elements is generated (the summary of approximate values on their properties’ level is also considered). Based on all these approximate values, model transformation mappings could be built automatically between source and target models.

5 Conclusions

In this paper, an automatic model transformation methodology (AMTM) is presented. According to the real requirement “exchanging information effectively and efficiently”, model transformation should be done automatically. So, semantic and syntactic checking measurements are combined into model transformation process to replace manual effort.

As theoretical foundation, a main framework is created; within this framework, a meta-meta-model is defined to present the mechanism of combining semantic and syntactic checking measurements into the process of defining model transformation mappings. For syntactic checking, “Porter stemming algorithm” and “Levenshtein distance algorithm” are used. For the semantic checking measurement, a specific thesaurus (AMTM_ST) is built. To deal with the granularity issue, model transformation is regarded as an iterative process and within each iteration phase, the transformation process is divided into three steps. Furthermore, a specific ontology (AMTM_O) is created to support the third matching step: auxiliary matching. This AMTM_O helps to store specific parts from source models and enrich specific parts for the target models.
However, there are some points in this AMTM that needed to be improved in the future.

- The impact factors such as “sem_weight”, “pn_weight” and threshold values for choosing matching pairs: the better way to assign them is “using some mathematic strategy” (e.g. “choquet” integral; one of the usages of “choquet” integral is stated in [17]).

- Semantic checking measurement: only formal English words are stored in the semantic thesaurus; for words that in specific cases or phrases, they cannot be located in AMTM_ST.

- The S_SeV values that defined in table II: they should be assigned differently based on the specific application domains.

- Matching pair choosing mechanism: we aim at finding the strongest semantic relation between two words, but the chosen semantic meaning may not be the exact one that the words conveyed within a specific context.

Furthermore, the usage of AMTM is cross-domain; AMTM aims at transforming and combining rough data to information (with specific structure and format), and then exchanging information among different domains.

Fig. 10 shows the scientific contribution of AMTM: converting rough data to information and exchanging and merging information on the information platform.

![Diagram](Fig. 10 AMMT scientific contribution)

Many data collectors (IoT) such as: sensors, smart equipment, computers, could gather rough data from a particular region or domain. Generally, this kind of data focuses on different purposes and reflects different views of a system. Moreover, different collectors store data in heterogeneous structures. AMTM regards these collected data as many single models, and uses semantic and syntactic checking measurements to detect the intrinsic links among them. Finally, after transforming and combining these data, a huge model (overview of a specific system) is generated. This huge model contains all the useful (not overlap) information. With rules that defined in specific domains, such information could be transformed (exchanging & sharing) to knowledge which serves to domain specific problems.

By combining semantic and syntactic checking measurements into model transformation process, an efficient model transformation methodology “AMTM” is created. With the improvement on some detail aspects, this methodology could serve to information sharing issue for a large number of domains in practice.

6 References