International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:9, No:8, 2015

An Automatic Model Transformation Methodology Based on Semantic and Syntactic Comparisons and the Granularity Issue Involved

Tiexin Wang, Sebastien Truptil, Frederick Benaben

***Abstract*—**Model transformation, as a pivotal aspect of Model-driven engineering, attracts more and more attentions both from researchers and practitioners. Many domains (enterprise engineering, software engineering, knowledge engineering, etc.) use model transformation principles and practices to serve to their domain specific problems; furthermore, model transformation could also be used to fulfill the gap between different domains: by sharing and exchanging knowledge. Since model transformation has been widely used, there comes new requirement on it: effectively and efficiently define the transformation process and reduce manual effort that involved in. This paper presents an automatic model transformation methodology based on semantic and syntactic comparisons, and focuses particularly on granularity issue that existed in transformation process. Comparing to the traditional model transformation methodologies, this methodology serves to a general purpose: cross-domain methodology. Semantic and syntactic checking measurements are combined into a refined transformation process, which solves the granularity issue. Moreover, semantic and syntactic comparisons are supported by software tool; manual effort is replaced in this way.

***Keywords*—**Automatic model transformation, granularity issue, model-driven engineering, semantic and syntactic comparisons.

I. Introduction

NOWADAYS, the theory of model-driven engineering (MDE) [1] has been widely used by numerous domains; furthermore, a large amount of MDE related practices (modeling languages, modeling techniques, model transformation instances, etc.) have been developed to serve to both general purposes and specific domain problems. For example: at this moment, model transformation practices have been widely adopted in enterprise engineering. Enterprises build models to simulate and analyze their producing process, information system, service production, etc. Internally, model transformation helps enterprises to merge different departments and modules by sharing information among them; externally, model transformation helps different companies to cooperate with each other by exchanging data and integrating information. According to [2], the ability of cooperating with other enterprises is becoming one of the core competition factors to an enterprise. It is also stated in [2] that the collaboration among enterprises tends to be global, dynamic and instantaneous. So, it is important to share and exchange data efficiently and effectively among heterogeneous partners that involved in collaborations.

T. Wang, S. Truptil and F. Benaben are with the Centre Genie Industriel, University de Toulouse - Mines Albi, Albi, CO 81000 France, (phone: 0033-0781304573; fax: 0033-0563493185; e-mail: tiexin.wang@mines-albi.fr, sebastien.truptil @mines-albi.fr, frederick.benaben @mines-albi.fr).

Model transformation provides a solution to share and exchange data between heterogeneous partners. However, there exist several weaknesses in traditional model transformation practices [3]: low reusability, contain repetitive tasks and involve huge manual effort, etc. These weaknesses limit the usage of model transformation to serve general purpose and reduce the efficiency of model transformation developing process. Based on this fact, we propose an automatic model transformation methodology (AMTM) that combines semantic and syntactic comparisons into model transformation process. The basic idea of AMTM has been illustrated in our previous work [4]. This paper presents a refined model transformation process and mainly focuses on solving granularity issue. The semantic and syntactic checking (S&S) mechanism has also been detailed. The combination of model transformation process and S&S has been illustrated more clearly compared to our previous work.

This paper is divided into six sections. In the second section, related work to this paper is presented. Then, a refined model transformation process is stated in the third section. The fourth section presents semantic and syntactic checking measurements in detail. The fifth section illustrates the combination of model transformation process and S&S with a use case. The final section presents the conclusion and prospect.

II. Related Work

This section is divided into three subsections: first subsection shows the comparing result of several modern prominent model transformation techniques, second subsection presents the horizontal comparisons among several model transformation practices and the third subsection talks about the category of model transformation. At the end of each subsection, a short conclusion is given to compare the existing achievements with AMTM.

A. Model Transformation Techniques Comparison

In this subsection, four popular model transformation techniques are presented briefly. They are: "ATLAS transformation language (ATL)", "Query/View/ Transformation (QVT)", "Visual Automated Model Transformations (VIATRA2)" and "Graph Rewriting and Transformation Language (GReAT)".

ATL [5] is a model transformation language and toolkit. Its

architecture is composed 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.

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 meta-models. The QVT standard integrates the "Object Constraint Language OCL 2.0" standard and also extends it with imperative features.

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]. VIATRA2 also integrates three sublanguages: Graph pattern language, Graph transformation rules language and Abstract State Machine (ASM) language. All of the three sublanguages may be used in a standalone manner.

GReAT [10] enables the specification of unidirectional translations between sets of models. It mainly uses graphical notation. However, some parts are specified textually: attribute initialization expressions and guards. GReAT is also composed of three sublanguages: Pattern specification language, Graph transformation language and Control flow language.

Table I shows the horizontal comparisons on some characteristics of the four techniques.

TABLE I
MODEL TRANSFORMATION TECHNIQUES COMPARISON

name	hybrid	rule scheduling	M-to-N	note
ATL	yes	implicit internal explicit	yes	self-executed
QVT	no	implicit internal explicit	yes	based on MOF 2.0
VIATRA2	yes	external explicit	yes	graph rewriting
GReAT	yes	external explicit	yes	on UML models

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 general purpose "cross-domain" and serve to specific domain.

Normally, domain specific model transformation techniques focus on and provide single solution to particular problematic. However, the usage of these techniques is limited, and they are not flexible for some special cases. Model transformation techniques, which serve to a general purpose, are always complex and provide a wide range of functions. Comparing with domain specific model transformation techniques, general purpose techniques are complex. So, people needs more time to learn to use general purpose techniques properly. The common problem of existing model transformation techniques is: involved huge manual effort and repetitive tasks. To solve this common problem, an automatic model transformation methodology that serves to general purposes, is presented in this paper.

B. Model Transformation Practices Comparison

Along with the emergence of model transformation techniques, a large number of model transformation practices have been developed. Some of these practices just propose the ideas of doing model transformations (needs tools to support); while the others provide both the theoretical solutions and supportive tools.

Generally, model transformation practices could also be divided into two categories (as model transformation techniques): domain specific practices and general purpose practices.

Table II shows three model transformation practices, the detail of these practices is detailed in [11]-[13].

TABLE II
MODEL TRANSFORMATION PRACTICES COMPARISON

name	technique	domain specific	theoretical or practical	note
Applying CIM-to-PIM model transformations for the service-oriented development of information systems	MDA-based	yes	both	Combining MDA with service-oriented development of information system
Transformation of decisional models into UML: application to GRAI grids	ATL	yes	both	GRAI Grids to UML model
Applying MDE to the (semi-) automatic development of model transformations	MeTAGeM	no	both	applying MDE principles to define model transformation

Many model transformation practices focus on serving to software development process; these practices are created based on different theories and using different techniques. Model transformation practices are also developed for other domains (e.g. enterprise integration). Since the number of modeling techniques is numerous and these techniques are heterogeneous, it is difficult for model transformation practices to serve to a general purpose. A common problem of these practices is: requiring users' effort. During the process of defining model transformation mappings, manual work is indispensable.

Comparing to these existing model transformation practices, AMTM uses S&S as the technique and serves to a general purpose. The basic theories of AMTM are defined in a main framework, which created as a refined meta-model based model transformation methodology. The practical solution of AMTM is implemented as software tools that combining S&S with model transformation process. The final purpose of AMTM is to "automatically define transformation mappings".

C. Model Transformation Category

In general, according to [14], 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 categories:

- Visitor-based approaches: an example of this approach is "Jamda".
- Template-based approaches: e.g., b+m Generator Framework, JET and FUUT–je.

And for the model-to-model approaches, there are five categories:

- Direct-Manipulation Approaches: offering an internal model representation plus some API to manipulate this model.
- Relational Approaches: grouping declarative approaches where the main concept is mathematical relations.
- Graph-Transformation-Based Approaches: e.g., VIATRA, ATOM and GreAT.
- Structure-Driven Approaches: an example is "OptimalJ" model transformation.
- Hybrid Approaches: combining different techniques from the previous categories.

The detail of these approaches (their applicable situations, working mechanism, etc.) and instances for each approach could be consulted in [14].

AMTM is a model-to-model model transformation methodology. Based on AMTM, there are two kinds of model transformation situations. Fig. 1 shows general ideas of the two situations.

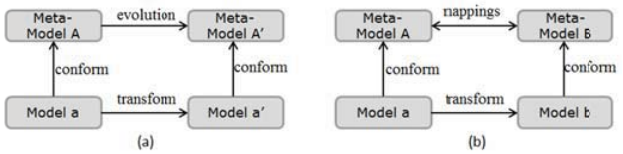

Fig. 1 Model Transformation Situations

As stated above, AMTM is created on the basis of meta-model based model transformation methodology.

In situation (a), target meta-model is the evolutional version of the source meta-model (some new characteristics have been added); source models that conform to the source meta-model should be transformed to target models that conform to the evolutional 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 techniques is "COPE" [15].

In situation (b), source meta-model and target meta-model are created for different purposes (no evolutional relation between them). 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 the meta-model level.

AMTM provides a solution to both situation (a) and situation (b); furthermore, there is no precondition of applying AMTM to both situations.

III. Overview of the Methodology

This section focuses on the detail of the automatic model transformation methodology (AMTM). "Automatic" means the process of making transformation mappings will be done without special user effort.

This section is divided into two subsections: first subsection presents the basic theories of AMTM and second subsection shows the working mechanism of this methodology.

A. Basis Theories

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

1. Theoretical Main Framework

AMTM is created on the basis of a theoretical main framework that shows in Fig. 2.

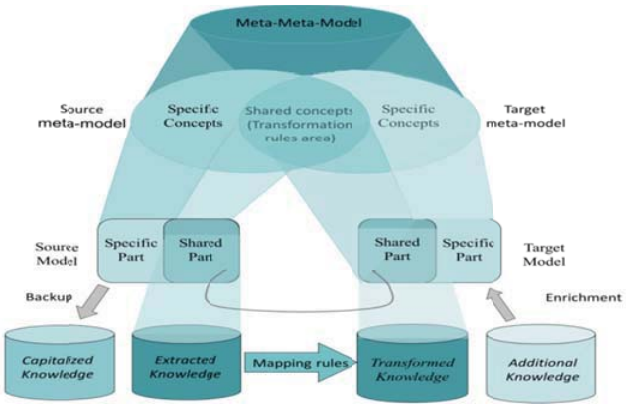

Fig. 2 Theoretical Main Framework

This theoretical main framework is created based on the work stated in [16]. It illustrates the fundamental theories of the whole automatic model transformation methodology.

The significance of doing model transformation could be "sharing knowledge", "exchanging information", etc. The necessary condition of doing model transformation between two models is: source model and target model should have some potential common items (to be detected and found).

For the reason "models are built based on the rules defined in their meta-models [17]", the potential common items could be traced on meta-model layer. AMTM relies on the meta-model layer (mappings are defined here among shared concepts). The source MM shares part of its concepts with the target MM. 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 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 also stated in [6]. The mechanism of applying S&S in AMTM is defined in a MMM, which shows at the top of Fig. 2.

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 serve to general purpose; they define their own semantic and syntax. For our project, these meta-modeling architectures are huge and complex. So, within the theoretical main framework, we define a specific meta-meta-model (MMM) to serve to AMTM. The content of this meta-meta-model determines the matching mechanism that involved in AMTM. Fig. 3 shows the detail of this meta-meta-model.

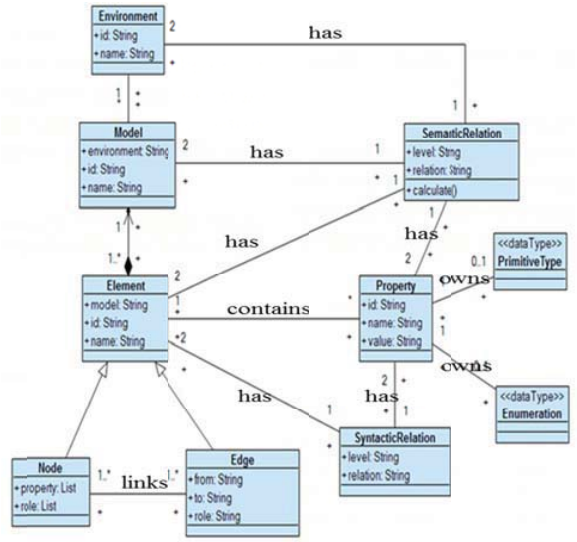

Fig. 3 The Meta-Meta Model

There are ten core elements in this meta-meta-model. They are listed as following:

- "Environment", describes the context of a system that models belong to.
- "Model", stands for all kinds of inputs and outputs.
- "Element", represents all items that could be contained in models (elements are self-contained. The "Element" has two inheritances: "Node" and "Edge".
- "Node", stands for an object or a concept; it is used to describe a subject that exists in the world.
- "Edge", describes the relationship between "Nodes". Every "Edge" links two roles (every node belongs to at least one role).
- "Property", is used to identify and explain the "Element" (node or edge) that contains it. Each "Property" has a "Data Type".
- "Primitive Type" (string, integer, double, Boolean, etc.), identifies property's attribute.
- "Enumeration type" (defined by users), identifies property's attribute.
- "Semantic Relation", exists on "Environment", "Model", "Element", and "Property" levels; it helps to define the transformation mappings automatically.
- "Syntactic Relation", exists only on "Element" and "Property" levels; it works together with semantic checking to define the transformation mappings.

Semantic and syntactic checking measurements are applied between source and target meta-models. The detail of executing process is presented in the following subsection.

B. AMTM Working Mechanism

In AMTM, a complete model transformation is regarded as an iterative process: a target model could also be the source model for the next iteration.

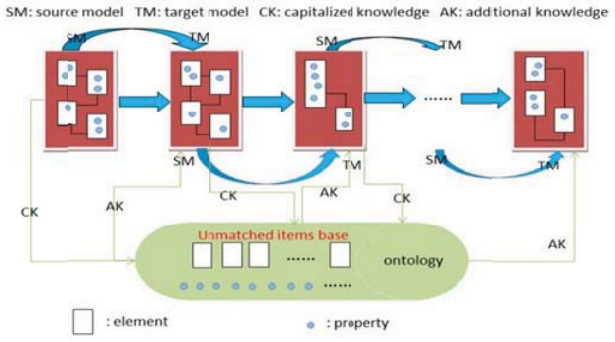

Fig. 4 Iterative Model Transformation Process

Fig. 4 shows the iterative issue. In each iteration phase, the specific part from the source meta-models is stored in ontology (the structure of this ontology is the same as the one of MMM); the additional knowledge for the specific part of the target model is also extracted from this ontology.

To deal with the granularity issue, the transformation process is divided into three steps: Matching on element level, hybrid matching and auxiliary matching for every iteration phases.

1. Matching on Element Level

According to MMM, meta-models are made of elements. So, model transformation mappings should be defined 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. As stated above, semantic and syntactic checking measurements are applied on a pair of elements to detect the relation between them. The mechanism of defining matches on element level is illustrated by an example shown in Fig. 5.

The source meta-model has 'n' elements and 'm' elements for the target meta-model; the number of comparisons between the two models on element's level is: "m*n" (the solid lines indicate parts of all the pairs).

Between each pair of elements (coming from source meta-model and target meta-model, respectively), a specific value, which based on their **names** and **groups of properties**, is generated.

A matrix is created to store all these comparison values. Table III shows this matrix.

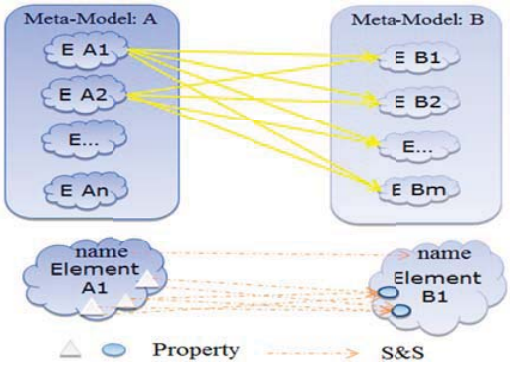

Fig. 5 Matching on Element's Level

TABLE III
ELEMENTS COMPARISON MATRIX

A \ B	E B1	E B2	……	E Bm
E A1	Ele_SSV	Ele_SSV	Ele_SSV	Ele_SSV
E A2	Ele_SSV	Ele_SSV	Ele_SSV	Ele_SSV
……	Ele_SSV	Ele_SSV	Ele_SSV	Ele_SSV
E An	Ele_SSV	Ele_SSV	Ele_SSV	Ele_SSV

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).

$$\text{Ele\_SSV} = \text{name\_weight} * \text{S\_SSV} + \text{property\_weight} * (\sum_{i=1}^{x} \text{Max(P\_SSVi)})/x \quad (1)$$

In (1), "name_weight" and "property_weight" are two impact factors for parameters "elements' names" and "elements' properties", respectively. Both of the value of "name_weight" and "property_weight" 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 (element's name is a word). "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" (focus on their properties' group); Table IV is created for this example.

"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".

$$\text{P\_SSV} = \text{pn\_weight}*\text{S\_SSV} + \text{pt\_weight}*\text{id\_type} \quad (2)$$

In (2), "pn_weight" and "pt_weight" are two impact factors for the parameters "properties' names" and "properties' types", respectively. The sum of "pn_weight" and "pt_weight" is 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.

TABLE IV
PROPERTIES COMPARISON MATRIX

E A1 \ E B1	p1	p2	……	py
P1	P_SSV	P_SSV	P_SSV	P_SSV
P2	P_SSV	P_SSV	P_SSV	P_SSV
……	P_SSV	P_SSV	P_SSV	P_SSV
Px	P_SSV	P_SSV	P_SSV	P_SSV

Based on (1) and (2), Tables III and IV could be fulfilled with calculated values. For each element (E A1, E A2…) of the source model A, it has a maximum "Ele_SSV" with a specific target model element (E B1, E B2…); if this value exceeds a predefined threshold value (e.g. 0.5), a mapping is built between the two elements. Moreover, matching a pair of two elements requires building mappings among their properties; Table IV also provides necessary and sufficient information to build mappings on property level (for a matched pair of elements). The rule of choosing property matching pairs is the same as choosing element matching pairs.

After this matching step: mappings on element level are: "one-to-one" and "many-to-one"; mappings on property level are "one-to-one" and "many-to-one" (constraint with matched element's pairs). In order to solve granularity issue ("many-to-many") on both element's and property's levels, another two matching steps that stated in the following subsections are necessary.

2. Hybrid Matching

After first matching step, some of the elements (both belonging to source and target meta-models) 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 are built among properties (come from both the unmatched and matched elements). Fig. 6 is a simplification of this matching step.

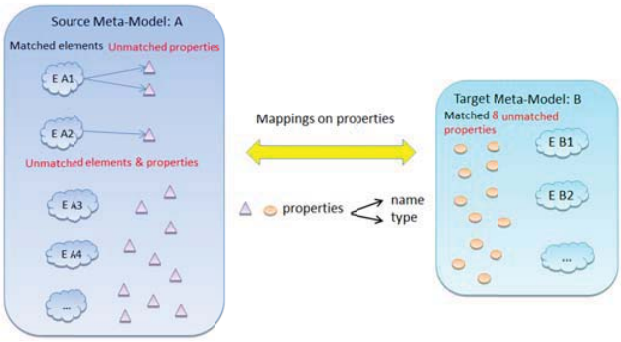

Fig. 6 Hybrid Matching Illustration

All the unmatched properties from source model will be

compared with all the properties from target model. A comparison matrix (similar to Table IV) is created to help complete this step. 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 is shown in (3).

$$HM\_P\_SSV = el\_weight*S\_SSV + pl\_weight*P\_SSV \quad (3)$$

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", respectively. The sum of "el_weight" and "pl_weight" is 1. "S_SSV" is calculated between two elements' names (for source property and target property, respectively). "P_SSV", as stated in (2), calculates the syntactic and semantic relation between two properties based on their names and types.

This step achieves "one to many" matching mechanism on element's level, and on property level matching breaks the matched elements' constraint: properties from one source element could be matched to properties that from several target elements.

With the help of matching pair choosing mechanism (which explained later in this section), on both element's and property's levels the mappings are "many-to-many" after the first two matching steps.

3. Auxiliary Matching

After the first and second matching steps, all the shared parts (presented in the theoretical main framework) between source model and target model are regarded to be found. However, according to the iterative model transformation process that mentioned at the beginning of this subsection, there are still some specific parts should be stored as capitalized knowledge or enriched as additional knowledge. Auxiliary matching step focuses on the mechanism of storing and reusing these specific parts from both source and target models.

As illustrated above, all the unmatched items from source model, which regarded as specific part, are stored in ontology (which is called "AMTM_O" within this project). AMTM_O designed with the same structure as the meta-meta-model that shown as Fig. 3. 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 generated in the latter iterations.

In some cases, the capitalized knowledge within only one iterative process is not enough to generate all the additional knowledge that needed in the same process. So, the capitalized knowledge from other model transformation process is needed to supplement to such cases. All the model transformation practices, which solved with AMTM, keep specific parts that stored in AMTM_O as capitalized knowledge. With the increased number of model transformation practices that solved with AMTM, the content of AMTM_O grows correspondingly. Furthermore, knowledge from other ontologies could also be extracted and stored in AMTM_O (conform to its structure).

Auxiliary matching step is created with the help of AMTM_O, semantic and syntactic checking measurements are the tools used to extract information from AMTM_O in this step.

4. Matching Pair Choosing Mechanism

According to the three former sub-subsections, the relation between two elements (come from source and target models respectively) 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 matching elements pairs depends on the range of this value. Fig. 7 reveals the basic principle.

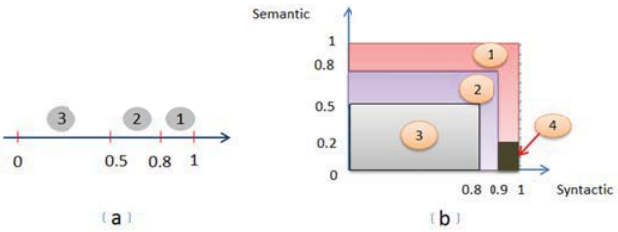

Fig. 7 Matching Pair Choosing Mechanism

For choosing element's matching pairs, two threshold values: 0.5 and 0.8 are assigned. As shown in Fig. 7 (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 (value between 0.5 and 0.8), a potential mapping exists between the two elements; else, if this value is in region 3, no mappings will be built between them.

Fig. 7 (b) shows the mechanism of choosing matching pairs of words: elements' 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 word pairs that have close syntactic relation but very weak semantic relation. For example, word pair: common and uncommon, they could not transform to each other.

On property level, it is following the same choosing mechanism of making mappings.

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" matching relationships are built on both element and property level.

IV. Syntactic and Semantic Checking Measurements

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 the semantic and syntactic value between two words; the calculation method of "S_SSV" is shown in (4).

$$S\_SSV = sem\_weight*S\_SeV + syn\_weight*S\_SyV \quad (4)$$

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.

A. Syntactic Checking Measurement

Syntactic checking measurement is used to calculate the syntactic similarity between two words (elements' and properties' names in our case). There exists several syntactic checking methods; majority of them use classic similarity metrics to calculate the syntactic relations. As stated in reference [18], such methods are: "edit-distance metrics", "fast heuristic string comparators", "token-based distance metrics" and hybrid method.

The syntactic checking measurement involved in AMTM could be divided into two phases:

- Pretreatment: focuses on finding if two words that in different forms (e.g. tense, morphology) stand for a same word.
- "Levenshtein Distances" algorithm [19].

For pretreatment, Table V shows several pairs of the situation: two words that in different forms (e.g. tense, morphology) stand for the same word.

TABLE V
Syntactic Pretreatment Situations

Case	Word 1	Word 2	Example
1		word 1 + 's' at end	son & sons
2	Ends with 's' "sh", "ch", 'x'	word 1 + "es" at end	match & matches
3		word 1 + "ing" at the end	do & doing
4	Ends with 'y'	change 'y' to 'i' + "es"	city & cities
5	……	……	……

For some specific cases (e.g. man and men), which are not followed general transformation rules, pretreatment phase does not consider about them.

If two words are not satisfied the conditions defined in the pretreatment, "Levenshtein Distances" algorithm is applied between them. This algorithm calculates the syntactic similarity between two words; it is based on the occurrences of the letters that involved in two words.

"Levenshtein distances" is equal to the number of operations needed to transform one string to another. There are three kinds of operations: insertions, deletions and substitutions. During the calculation process, a two-dimensional table is created to store the transformation information. The basic theory and concrete executing process of this algorithm is stated in [19].

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

$$S\_SyV = 1 - LD \,/\, Max\,(word1.len, word2.len) \quad (5)$$

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.

B. Semantic Checking Measurement

Different to syntactic checking measurement (relies just on the two comparing words); semantic checking measurement relies upon a huge semantic thesaurus which contains large amount of words, their semantic meanings and semantic relations among these words. A huge semantic thesaurus (AMTM_ST) has been created for serving to AMTM, and AMTM_ST is based on the basis of "WordNet" [20]. Fig. 8 shows the structure of this semantic thesaurus.

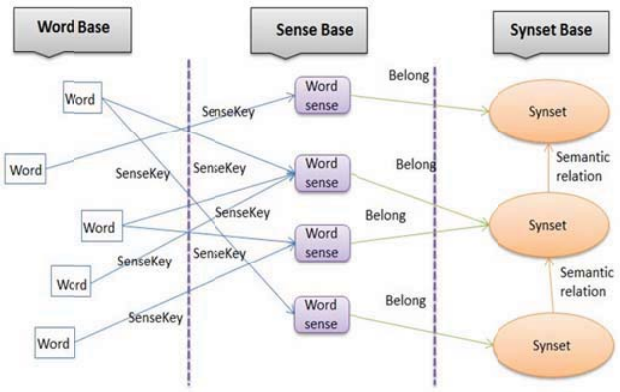

Fig. 8 Structure of AMTM_ST

As shown in Fig. 8, there are three kinds of items in AMTM_ST, they are listed as following:

- 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: word "star": it has six senses; as noun, it has four senses; as verb, it has another two senses
- "Synset" base: a group of word senses that own synonym meanings; semantic relations are built among different synsets.

There are seven kinds of semantic relations that defined among synsets: "synonym", "hypernym", "hyponym" "iterative hypernym", "iterative hyponym", "similar-to" and "antonym". For each of the semantic relations, a specific value

(between 0 and 1) is assigned to it. Table VI shows these semantic relations and their values pairs.

In Table VI, all the "S_SeV" values are assigned directly (based on experience); these values could be assigned with more reasonable methods.

TABLE VI
SEMANTIC RELATIONS AND VALUES IN AMTM_ST

Semantic relation	S_SeV	Remark	Example
synonym	0.9	words from the same synset	shut & close
hypernym	0.6	two synsets have this relation	person-creator
hyponym	0.8	two synsets have this relation	creator-person
similar-to	0.85	only between two adjectives	perfect & ideal
antonym	0.2	words have opposite meanings	good & bad
iterative hypernym	Math.Power (0.6, n)	iterative hypernym relation	person-creator-maker-author
iterative hyponym	Math.Power (0.8, n)	iteritance hyponym relation	author-maker-creator-person

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_SeV" values between two particular words is not limited to one. In our project, we focus on finding the maximum "S_SeV" value between two words (for the purpose of selecting potential shared concepts as matching pairs). 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. However, at this moment, only form English words that in simple case could be located 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.

Fig. 9 is an illustration of the three steps.

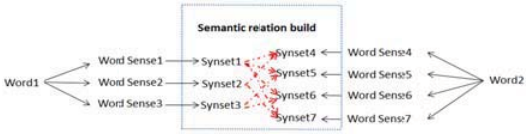

Fig. 9 Synsets Locating

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 part). In Fig. 9, the red dash lines show these possible pairs (not all of them). The detail of detecting process is shown in Fig. 10.

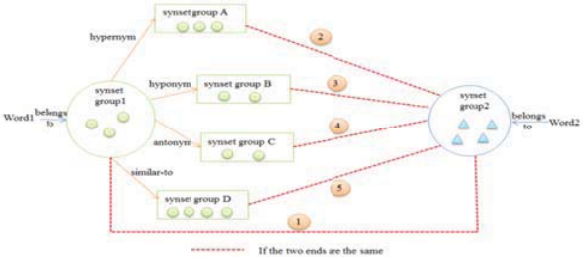

Fig. 10 Semantic Relationship Detecting Process

After locating the two synsets groups, which word1 and word2 belong to, next step is to detect the semantic relations that exist among these synsets. Fig. 10 only shows the mechanism of 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 1", then comparing if there exist one synset in "synset group 2", which is the same as one of the located synsets.

The detecting process of "iterative hypernym" and "iterative hyponym" semantic relations is more complex than these five semantic relations that illustrated in Fig. 10.

The detecting mechanism for "iterative hypernym" and "iterative hyponym" semantic relations is same. Fig. 11 is an illustration of detecting "iterative hypernym" semantic relation between two words.

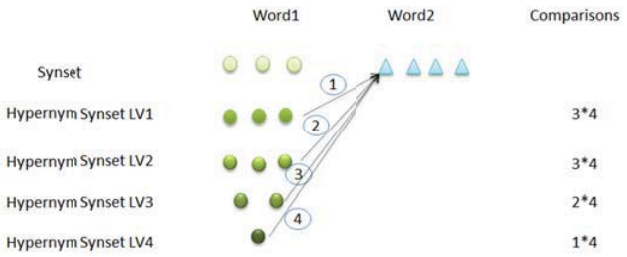

Fig. 11 Iterative Hypernym Relation Detecting

The basic idea of this detecting process 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. Table VII shows the content that stored in AMTM_ST.

As shown in Table VII, large amount of words with their word senses and synsets are stored in AMTM_ST. The semantic relations that built among these synsets have been shown in Table VI.

TABLE VII
AMTM_ST CONTENTS

Item	Number
word	147306
word sense	206941
synset	114038

These synsets could be divided into three groups and each group contains several kinds of semantic relations. Table VIII shows the detail of synsets categories.

TABLE VIII
SEMANTIC RELATIONS IN DIFFERENT SYNSETS

Synset	Number	Semantic relation
noun synset	82115	synonym, hypernym, antonym, hyponym
verb synset	13767	synonym, hypernym, antonym, hyponym
adjective synset	18156	synonym, antonym, similar-to
total	114038	synonym, hypernym, antonym, similar-to, hyponym

Table VIII lists the categories of synsets and the semantic relations that built on each synset category. As stated previously, semantic relations: hypernym and hyponym are iterative; it means the two semantic relations could have several levels of nesting.

C. 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 value on their properties' level are also considered).

Based on all these approximate values that generated on both elements' and properties' levels, transformation mappings could be built between source and target models.

Combining syntactic and semantic checking measurements into process of AMTM (three matching steps that mentioned in the third section), an automatic model transformation process is generated.

V. USE CASE

In this section, a simple use case will be presented to show the working mechanism of AMTM. We test the matching possibility between two elements (each of them has several properties), and show the whole comparison process.

The test use case is shown in Fig. 12.

As stated above, model transformation mappings are built on meta-model level, and focused on elements and their properties. In this test case, there are two elements: "student" and "person". They have four properties and come from source meta-model and target meta-model, respectively. We will apply AMTM on the two elements and calculate the matching possibility between them based on their names and properties' groups.

According to (4), the S&S value between two elements' names could be calculated.

Equation (4) uses the results that shown in Table VI and (5) to calculate semantic value and syntactic value between two names, respectively. At this moment, we consider more about the semantic relation than syntactic relation (assign "sem_weight" as 0.9, "syn_weight" as 0.1 in (4)). So, the calculated "S_SSV" value between two names "student" and "person" is: **0.743**. Next step is to calculate the S&S relation on their property level based on (2) and Table IV. Table IX is created for this test case.

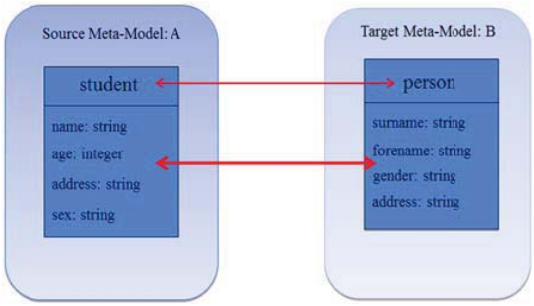

Fig. 12 Test Use Case

When calculating the values in Table IX, both properties' names and their types are taken into consideration. To AMTM, property's name is more important than its type for making mappings. So in (2), we assign "pn_weight" as 0.8 and "pt_weight" as 0.2. With the help of (4) and Table IV, Table IX is fulfilled.

TABLE IX
PROPERTIES COMPARISON IN TEST CASE

student \ person	surname	forename	gender	address
name	0.8936	0.888	0.2136	0.7946
age	0.0229	0.02	0.4856	0.2229
address	0.2	0.21	0.6366	1
sex	0.2114	0.21	0.8616	0.6366

Finally, the final S&S value between the two elements is calculated as: **0.7766**. We treat elements' names and their properties' groups as equal importance (based on (1), both "name_weight" and "property_weight" are assigned 0.5).

According to the matching pair choosing mechanism that illustrated in the third section, the two elements are considered as a potential matching pair.

In this test case, it just shows the process involved in the first matching step: matching on element level. For the other two matching steps, they use the same comparing mechanism (S&S). The only difference is: the hybrid matching focuses on property and the auxiliary matching step relies on AMTM_O. The second and third matching steps could solve the granularity issue involved in model transformation process. On both element level and property level, the matching mechanism is "many-to-many". Furthermore, element and property could be transformed to each other.

VI. CONCLUSION

In this paper, an automatic model transformation methodology (AMTM) is presented. According to the real requirement, model transformation should be done effectively and efficiently, semantic and syntactic checking measurements are combined into model transformation process.

As theoretical foundation, a main framework is created;

within this framework, a meta-meta-model is defined (the mechanism of applying semantic and syntactic checking measurements are defined in this meta-meta-model). For the semantic checking measurement, a specific semantic thesaurus (AMTM_ST) is built. To deal with the granularity issue, model transformation process is regarded as iterative 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. 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 [21]).
- Semantic checking measurement: only formal English words are stored in the semantic thesaurus with semantic meanings; not for words that in specific cases or phrase.
- The S_SeV values that are defined in Table IV: More test cases are needed to modify these values into reasonable scope.
- Matching pair choosing mechanism: we aim at finding the strongest semantic relation between two words, but maybe the semantic meaning is not the exact one that the words conveyed within a specific context.

Furthermore, the usage of AMTM is not limited to one domain; AMTM aims at transforming and combining rough data to information (with specific structure and format), and allows "MDE" theories to serve other engineering domains.

Fig. 13 shows the scientific contribution of AMTM: converting rough data to information.

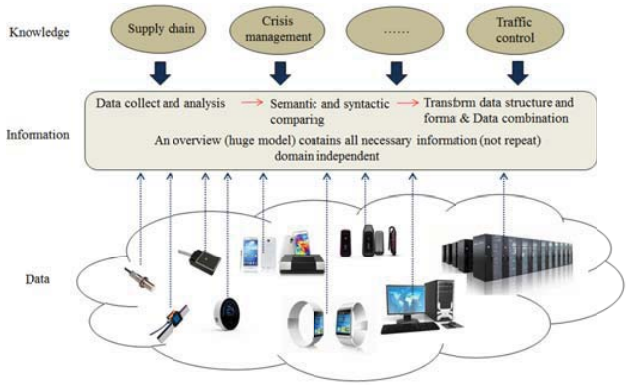

Fig. 13 AMTM Scientific Contribution

Many data collectors 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. So, it is difficult to make use of this kind data as a whole.

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 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 a large number of domains in practice.

REFERENCES

[1] DC. Schmidt, Model-Driven Engineering. IEEE Computer, February 2006 (Vol.39, No. 2) pp. 25-31.
[2] J. Touzi, JP. Lorré, F. Bénaben, Interoperability through Model-based Generation: The Case of the Collaborative Information System (CIS) (J)]. Enterprise Interoperability, 2007: 407.
[3] M.D. Del Fabro, P. Valduriez, Towards the efficient development of model transformations using model weaving and matching transformations. Software & System Modeling, July 2009, Volume 8, Issue 3, pp 305-324.
[4] T. Wang, S. Truptil, F. Bénaben, "Semantic approach to automatically defined model transformation." International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2014), pp. 340-347.
[5] F. Jouault, F. Allilaire, J. Bézivin, I. Kurtev, ATL: A model transformation tool. Science of Computer Programming. 2007, Volume 72, Issues 1–2.
[6] M.D. Del Fabro, J. Bézivin, F. Jouault, E. Breton, AMW: A Generic Model Weaver. 2005, 1ère Journées sur l'Ingénierie Dirigée par les Modèles: Paris.
[7] OMG: QVT. Meta Object Facility (MOF) 2.0 Query/View/ Transformation Specification OMG (2008) http://www.omg.org/spec/QVT/1.0/PDF
[8] D. Varr´o, A. Balogh, The model transformation language of the viatra2 framework. Sci. Comput. Program 68(3), 214–234 (2007).
[9] D. Varr´o, A. Pataricza, VPM: A visual, precise and multilevel meta-modeling framework for describing mathematical domains and UML (the mathematics of metamodeling is metamodeling mathematics), Software. Syst. Model. 2 (3) (2003) 187–210.
[10] G. Karsai, A. Agrawal, F. Shi, J. Sprinkle, "On the use of graph transformation in the formal specification of model interpreters", J. Univ. Comput. Sci. 9 (11) (2003) 1296–1321.
[11] D.V. Castro, E. Maros, J.M. Vara, Applying CIM-to-PIM model transformations for the service-oriented development of information systems. Information and Software Technology, 2011, Volume 53, Issue 1, Pages 87–105.
[12] R. Grangel, M. Bigand, J.P. Bourey, "Transformation of decisional models into UML: application to GRAI grids". International Journal of Computer Integrated Manufacturing, 2010, Volume 23, Issue 7.
[13] VA. Bollati, JM. Vara, A. Jiménez, E. Marcos, Applying MDE to the (semi-)automatic development of model transformations. Information and Software Technology 2013; 55(4):699–718.
[14] K. Czarnecki, S. Helsen, Classification of Model Transformation Approaches. OOPSLA'03, 2003, Workshop on Generative Techniques in the Context of Model-Driven Architecture.
[15] M. Herrmannsdoerfer, S. Benz, E. Juergens, COPE - automating coupled evolution of metamodels and models. In: Drossopoulou, S. (ed.) ECOOP 2009 – Object-Oriented Programming. LNCS, vol. 5653, pp. 52–76. Springer, Heidelberg.
[16] F. Bénaben, W. Mu, S. Truptil, H. Pingaud, "Information Systems design for emerging ecosystems." 2010, 4th IEEE International Conference on Digital Ecosystems and Technologies (DEST).
[17] J. Bezivin, "Model driven engineering: An emerging technical space," in Generative and Transformational Techniques in Software Engineering, International Summer School -GTTSE, 2006, pp. 36–64.

[18] W. C. William, R. Pradeep, E. F. Stephen, "A Comparison of String Metrics for Matching Names and Records." KDD Workshop on Data Cleaning and Object Consolidation, 2003, Vol. 3.
[19] H. Wilbert, "Measuring Dialect Pronunciation Differences using Levenshtein Distance." Ph.D. thesis, 2004, Rijksuniversiteit Groningen.
[20] X. Huang, C. Zhou, "An OWL-based WordNet lexical ontology." Journal of Zhejiang University, 2007, pp. 864-870.
[21] D. Abril, G. Navarro-Arribas, V. Torra, Choquet integral for record linkage Ann. Oper. Res., 195 (1) (2012), pp. 97–110.