A General Model Transformation Methodology to Serve Enterprise Interoperability Data Sharing Problem

Tiexin Wang, Sébastien Truptil, Frederick Benaben

A General Model Transformation Methodology to Serve Enterprise Interoperability Data Sharing Problem

Tiexin WANG, Sebastien TRUPTIL, Frederick BENABEN

Centre Génie Industriel, Université de Toulouse - Mines Albi, Campus Jarlard, 81000 Albi, France
{tiexin.wang, sebastien.truptil, frederick.benaben}@mines-albi.fr

Abstract. Interoperability, as one of the key competition factors for modern enterprises, describes the ability to establish partnership activities in an environment of unstable market. In some terms, interoperability determines the future of enterprises; so, improving enterprises' interoperability turns to be a research focus. "Sharing data among heterogeneous partners" is one of the most basic common interoperability problems, which requires a general methodology to serve. Model transformation, which plays a key role in model-driven engineering, provides a possible solution to data sharing problem. A general model transformation methodology, which could shield traditional model transformation practices' weaknesses: low reusability, contains repetitive tasks, involves huge manual effort, etc., is an ideal solution to data sharing problem. This paper presents a general model transformation methodology "combining semantic check measurement and syntactic check measurement into refined model transformation processes" and the mechanism of using it to serve interoperability's data sharing issue.

Keywords: interoperability; model-driven engineering; model transformation; semantic check; syntactic check

1 Introduction

Nowadays, the world is becoming smaller and smaller. With the advancements of science and technology, more and more collaborations among countries, companies and persons are appeared. Such collaborations appear and disappear within specific periods, with achieving or failing of their goals. Based on this fact, the ability of cooperating with different partners becomes crucial to modern systems and organizations. Furthermore, "interoperability" is proposed specially to describe such ability. There are several definitions for interoperability; one of the initial definitions of interoperability could be referred in [1]. Another two definitions are listed here: as defined in [2], "interoperability is the ability of a system or a product to work with other systems or products without special effort from the user"; a similar definition of interoperability is stated in [3], interoperability is “a measure of the degree to which diverse systems, organizations, and/or individuals are able to work together to achieve a common goal. For computer systems, interoperability is typically defined in terms of syntactic interoperability and semantic interoperability”. Two key issues that stated in the two definitions are: “cooperate without special users’ effort” and “semantic and syntactic” aspects. Although in different domains and from different views of one domain, the definitions of interoperability might be slightly different, the essence reflected by these definitions is similar. Fig. 1 shows the interoperability issue and the data sharing problem of it.

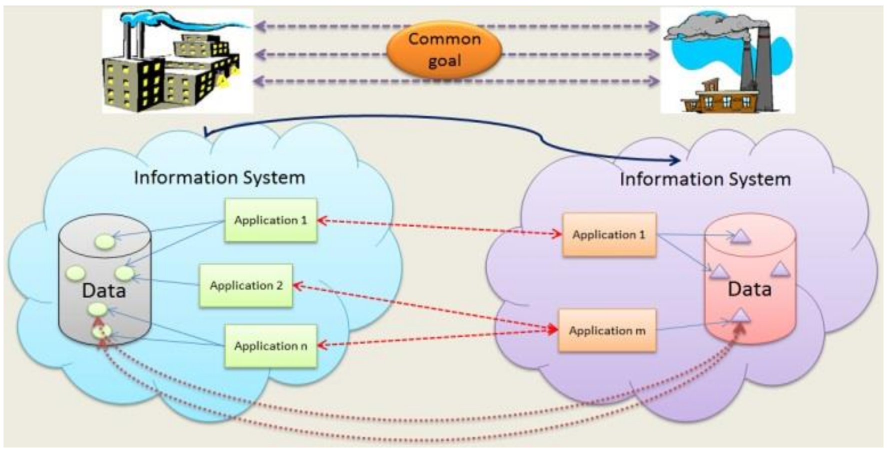

Fig. 1. An illustration of interoperability issue

Fig. 1 shows a collaboration situation between two companies. Modern companies use information systems to manage their business; in some aspects, the cooperation among companies depends on the merge of their information systems. Furthermore, merging information systems relies on the interactions of their applications. So, sharing data among these applications (both within one system and from different systems) is important for enterprise cooperation. However, generally the structures of data are designed for specific applications used by particular enterprises; it is difficult to share data among different applications. Model transformation provides a possible solution to data sharing issue.

“Enterprise Interoperability Framework (EIF)” [4] shows a possible way of combining formally enterprise interoperability and model-driven engineering (especially the model transformation part). However, traditional model transformation practices have their own weakness: low reusability, repetitive tasks, huge manual effort, etc. In order to apply model transformation to solve interoperability problems, a general model transformation methodology (shield these weaknesses) is required. This paper presents such a general model transformation methodology.

This paper is divided into five sections. In the second section, the basic principles of model-driven engineering (MDE) and model transformation are presented. The third section describes the overview of the general methodology. The detail of syntactic and semantic checking measurements is illustrated in the fourth section. Finally, the conclusion is proposed in the fifth section.

2 Basic Background Theories

In this section, the basic background theories of this general model transformation methodology (GMTM) are presented. These theories are divided into two group: theories owned by MDE domain and theories belonging specially to model transformation domain.

2.1 Model-driven Engineering

Model-driven engineering (MDE) [5], which initially referred as model-driven software development, is an important direction in the development of software process. It takes modeling and model transformation as the main means of software development methods. Comparing with other software development methods, the main features of model-driven development approach are paying more attention to construct the abstract description of different areas of knowledge: the domain models; then based on these models to characterize the software system. Through layers of automatic (semi-automatic) conversion of the models, the development from design to achieve the transition to the final completion of the entire system will complete.

At this moment, the principles of "model driven engineering" are applied on many different domains (knowledge engineering, enterprise engineering, etc.); it is not restricted to software development any more.

As an example to broader MDE's vision, "model-driven architecture (MDA)" [6] was launched in 2001 by the Object Management Group (OMG). Fig. 2 shows the basic principles of MDA.

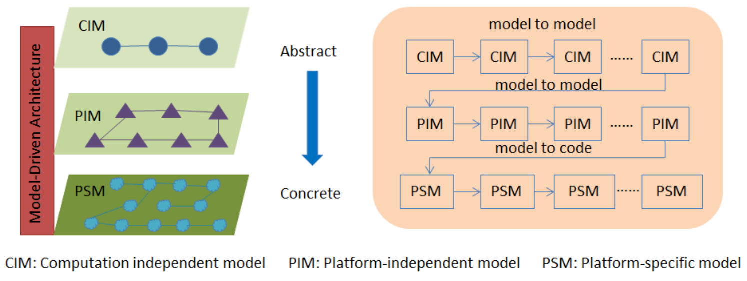
Model-Driven Architecture
CIM
PIM
PSM
Abstract
Concrete
model to model
CIM
CIM
CIM
CIM
model to model
PIM
PIM
PIM
PIM
model to code
PSM
PSM
PSM
PSM
CIM: Computation independent model
PIM: Platform-independent model
PSM: Platform-specific model

Fig. 2. Simple illustration of MDA

In MDA, models could be divided into three groups: "CIM", "PIM" and "PSM". In each of the three groups, large number of models could be built to reflect the characteristics, which based on different point of views, of one system. Models in PIM layer should be generated by transforming the models from CIM layer; the mechanism of building PSM layer's models follows the same principle (generated by transforming models from PIM layer).

In MDE context, everything could be regarded as a model or could be modeled. In simple words, MDE uses models to describe the reality (concerns the modeling techniques) and uses model transformations to solve conversion problems. Modeling, as one key role of MDE, means the activities of building models; model transformation, as another key role of MDE, means the process of taking the source model to generate the target model.

2.2 Model and Meta-model

Model and meta-model are two basic concepts in MDE; Fig. 3 shows the relation between them.

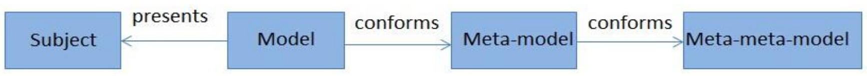

Fig. 3. Relation between model and meta-model

As defined in [7], model is "a simplification of the subject and its purpose is to answer some particular questions aimed towards the subject". Models are built to represent the characteristics of real systems based on specific point views. Meta-models are a specific kind of model; they make statements about what can be expressed in valid models. Meta-models could have several layers; meta-model defines building rules for models that conform to it.

2.3 Model Transformation

Model transformation plays a key role in MDE; it is the nexus among heterogeneous models. With the extensive usage of MDE theory, more and more theories, techniques and tools of model transformation have been created. Large amount of model transformation practices have been developed to serve some specific domain problems using these theories, techniques and tools; two examples are stated in [8] and [9].

In general, according to [10], there are two main kinds of model transformation approaches. They are: model-to-code approaches and model-to-model approaches. For model-to-code approaches (PIM to PSM), there are two categories: "Visitor-based approaches" and "Template-based approaches". For the model-to-model approaches, there are five categories:

- Direct-dManipulation 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.) could be consulted in [10].

However, as mentioned above, traditional model transformation practices have internal weaknesses; these weaknesses limit the scope of model transformation usage. As the inner characteristics and requirement of modern enterprise interoperability (e.g. agility, transient, heterogeneity, complexity), traditional model transformation practices are not a good choice to serve it. So, a general model transformation methodology is required.

3 Overview of the General Methodology

This section presents the detail of GMTM. The main objective of GMTM is "overcoming the shortcomings of traditional model transformation practices and serving to enterprise interoperability". "General" means the use of this methodology is widely, not limited to a specific domain. In order to be general, the process of defining model transformation mappings should be automatic. To achieve this goal, semantic and syntactic checking (S&S) measurements are combined into the traditional model transformation process.

3.1 Theoretical Main Framework of the General Methodology

GMTM is created on the basis of a theoretical main framework, which is based on [11], and shown in Fig. 4.

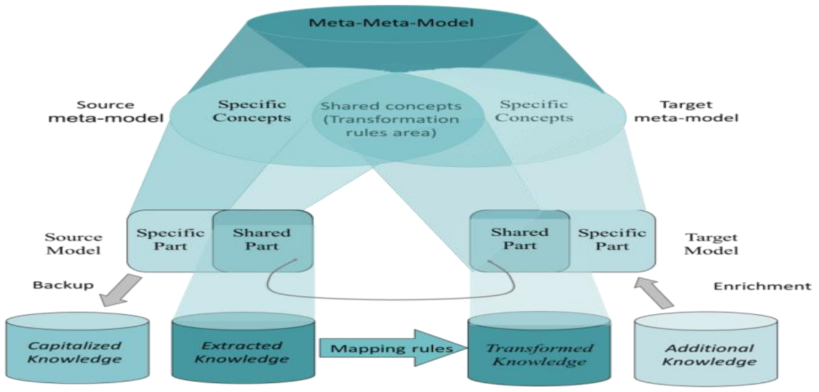

Fig. 4. Theoretical main framework

Fig. 4 illustrates the theoretical basis of GMTM. The significance of doing model transformation could be "sharing knowledge", "exchanging information", etc. The purpose of model transformation practice is: generate the target model based on the source model.

The necessary condition of doing model transformation between two models is: the 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", the potential common items could be traced on meta-model layer.

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 on 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.

3.2 The Meta-meta-model within Main Framework

According to [12], in order to apply semantic checking measurements in the process of defining model transformation mapping rules, some principles should be obeyed. In this GMTM, the mechanism of applying S&S in model transformation process is defined in a meta-meta-model (MMM), which is shown at the top of Fig. 4.

There are several meta-modelling architectures, for example "MOF: Meta-Object Facility" [13]. These architectures define their own semantic and syntax. For GMTM these existing meta-modelling architectures are complex to use. So, based on the context of model transformation, we adapt the idea stated in MOF and generate this MMM. Fig. 5 shows the content of this MMM.

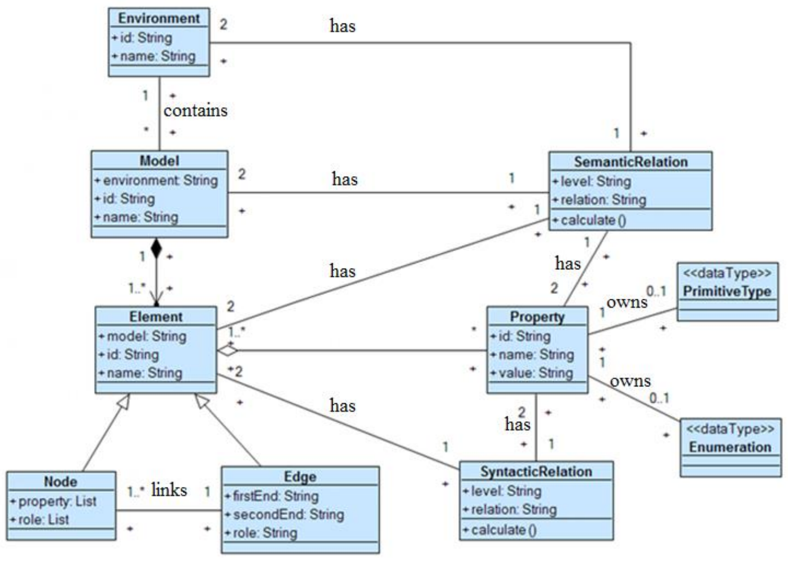

Fig. 5. The class diagram of the meta-meta-model

For GMTM, this MMM works on the top abstract level of all the other models. As this MMM is defined as a common criterion, the meta-models (for both source models and target models) could be built or transformed to the versions that conform to it.

As shown in Fig. 5, there are ten core elements in this meta-meta-model. As models may come from various domains or systems, a class named “Environment” is defined to stand for these domains. All the model instances are standed by the class “Model”, every model belongs to a specific “Environment”. “Model” is made of “Element”, which has two inheritances: “Node” and “Edge”. “Node” 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 extend, data type could differentiate “Property”.

All these items (with the relationships among them), illustrated above, present the standard requirement on specific meta-models. 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.

Generally, model transformation mappings are defined on the element level (node and edge); the mapping rules are usually generated by domain experts. However, applying model transformation practices to serve enterprise interoperability requires model transformation practices to be more flexible and easier (faster) to integration. So, semantic checking and syntactic checking that focused on element and property levels, are introduced to automatically define the mappings (replacing manual efforts). Also, in the MMM, the property and its dada type are highlighted; both of them are used to deduce semantic relation on element level. Furthermore, the inner attribute of element and property: their names, have also been used to define semantic and syntactic relations.

3.3 Matching Mechanism

In GMTM, model transformation is regarded as an iterative process: a target model (generated by one transformation iteration) could be the source model for the next iteration. In each iteration phase, transformation process is divided into three steps: matching on element level (coarse-grained matching), hybrid matching (fine-grained matching) and auxiliary matching (specific parts matching). All these three steps are supported by software tool; experts may only be involved in the validating process.

Iterative Matching Mechanism According to the theoretical main framework, model transformation mappings are built on the potential shared parts between source model and target model. During the transformation process: the specific part of source model is saved as capitalized knowledge and the specific part of target model should be enriched with additional knowledge. So, a question may be put forward: how to deal with the capitalized knowledge and where the additional knowledge comes from? The “iterative matching mechanism” gives a possible answer to this question.

Fig. 6 shows the general idea of this iterative matching mechanism.

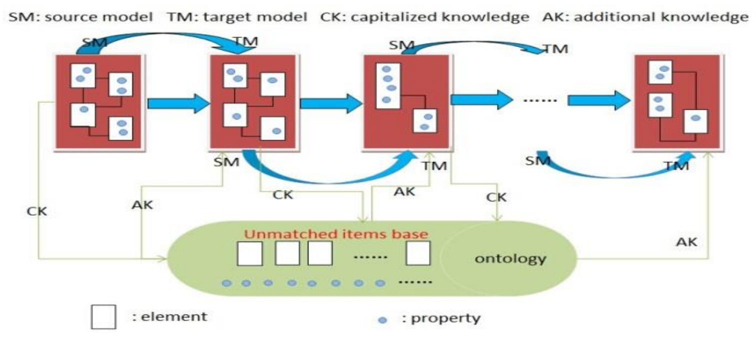
SM: source model TM: target model CK: capitalized knowledge AK: additional knowledge
SM
TM
SM
TM
SM
TM
SM
TM
CK
AK
CK
AK
CK
AK
Unmatched items base
ontology
: element
: property

Fig. 6. Overview of iterative matching mechanism

One complete model transformation process may involve several iterations; each of iterations is an independent model transformation instance. An intermediate model is both the target model of the former iteration and source model of the latter iteration. All the specific parts (unmatched items: properties and elements) from source models are saved into ontology as capitalized knowledge, and the specific parts of target models are enriched with additional knowledge (capitalized knowledge from former iterations) that extracted from the same ontology.

Matching on Element Level Generally, model transformation mappings are defined on element level (nodes and edges); if two elements (come from source model and target model, respectively) stand for the same concept (shared concept between two models), a mapping should be built. 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. 7.

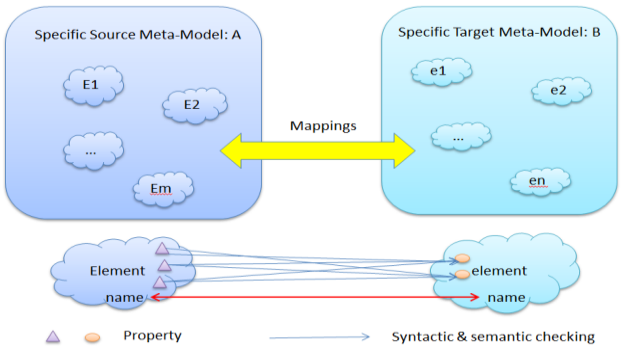
Specific Source Meta-Model: A
E1
E2
...
Em
Mappings
Specific Target Meta-Model: B
e1
e2
...
en
Element
name
element
name
Property
Syntactic & semantic checking

Fig. 7. Example of making matching on element level

The two specific meta-models (marked as A and B) are supposed to be conformed to the MMM. Model A has "m" elements and model B has "n" elements; the mappings should be built within the "m*n" element's pairs. Table 1 shows this comparison matrix. This matrix is built automatically by software tool; based on different inputs (model instances), similar matrix would be generated automatically.

Table 1. Element level selected matrix

A \ B	e1	e2	……	en
E1	Ele_SSV	Ele_SSV	……	Ele_SSV
E2	Ele_SSV	Ele_SSV	……	Ele_SSV
……	Ele_SSV	Ele_SSV	……	Ele_SSV
Em	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. Formula (1) is defined to calculate "Ele_SSV" value.

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

In (1), "name_weight" and "property_weight" are two impact factors for the parameters elements' names and elements' properties, respectively. Both the values 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"; another example which shown below, is used to calculate "P_SSV". "x" stands for the number of properties of a specific element from source meta-model (e.g. element E1).

The example shown below is used to generate the "Ele_SSV" value within the element's pair of E1 and e1 (focuses on their properties' group); table 2 is created for this example. This kind of tables is also built automatically (for different comparing elements' pairs) by software tool.

Table 2. Property level selected matrix

E1 \ e1	p1	p2	……	py
P1	P_SSV	P_SSV	……	P_SSV
P2	P_SSV	P_SSV	……	P_SSV
……	P_SSV	P_SSV	……	P_SSV
Px	P_SSV	P_SSV	……	P_SSV

E1 has "x" properties and e1 has "y" properties; within each of the "x*y" pairs of properties, there exists a "P_SSV". Formula (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.

With the help of table 2 (also needs the "S_SSV" between E1's name and e1's name), the "Ele_SSV" between element "E1" and "e1" could be calculated. In this way, table 1 could be fulfilled with calculated values. For each element (E1, E2…) of the source model A, it has a maximum "Ele_SSV" value with a specific target model element (e1, e2…); if this value exceeds a predefined threshold value (e.g. 0.5), a match is built between the two elements. Moreover, making matching between two elements requires building mappings among their properties; table 2 provides necessary and sufficient information to build mappings on property level. The rule of choosing property matching pairs is same of choosing element matching pairs (set another threshold value). In this way, both on element and property levels, the matches are: "one to one" and "many to one".

At this moment, the impact factors and selecting threshold values are assigned directly by experience.

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 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. A comparison matrix (similar to table 2) 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 shows in (3).

$$\text{HM\_P\_SSV} = \text{el\_weight}*\text{S\_SSV} + \text{pl\_weight}*\text{P\_SSV} \tag{3}$$

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.

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 mentioned at the beginning of this subsection, there are still some specific parts that 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.

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 MMM that shown as Fig. 5.

The syntactic and semantic checking measurements that involved in these three matching steps will be explained in detail respectively in the following section.

4 Syntactic and Semantic Checking Measurements

GMTM requires defining automatically the model transformation mapping rules. So, semantic and syntactic checking measurements (executed by software tool) are involved. As shown in (4), the "S_SSV" stands for the semantic and syntactic value between two strings.

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

"Sem_weight" and "syn_weight" are two impact factors for the parameters semantic value and syntactic value; the sum of them is 1. The two following subsections illustrate the way to calculate "S_SeV" and "S_SyV", respectively.

4.1 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. Some of examples could be referred in [14].

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

1. Pretreatment: focuses on finding if two words that in different forms (e.g. tense, morphology, gender) stand for the same word.
2. "Levenshtein Distances" algorithm [15]: calculates the syntactic similarity between 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. Formula (5) shows the calculation of syntactic relation between two words: word1 and word2 based on "Levenshtein distances".

$$\text{S\_SyV} = 1 - \text{LD} / \text{Max}(\text{word1.length}, \text{word2.length}) \quad (5)$$

In (5), “S_SyV” stands for the syntactic similarity value between “word1” and “word2”; “LD” stands for 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.

4.2 Semantic Checking Measurement

Contrast to syntactic checking measurement (rely just on comparing the two words); semantic checking measurement should rely upon a huge semantic thesaurus which contains large amount of words, their semantic meanings and semantic relations among them. A specific semantic thesaurus has been created for GMTM, and it is based on the basis of “WordNet” [16]. Fig. 8 shows the structure of this semantic thesaurus.

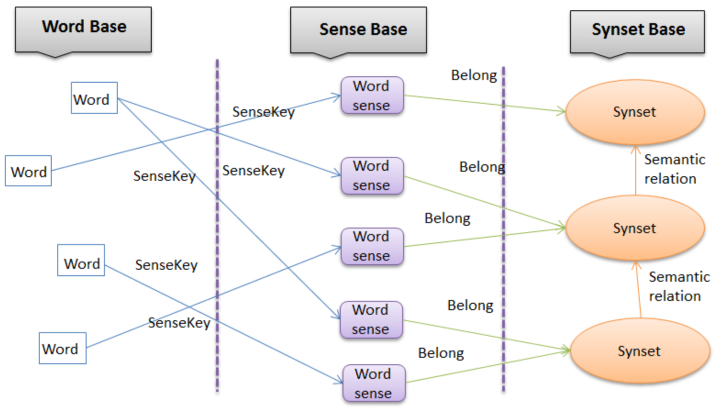

Fig. 8. Structure of the semantic thesaurus

Fig. 8 shows three kinds of elements stored in the semantic thesaurus.

- Word base: normal English words (nouns, verbs and adjectives) are stored here.
- Sense base: contains all the word senses; a word could have “one or several” senses. E.g., word “star”: it has six senses (four as a noun and two as a verb).
- “Synset” base: synonym groups; the word senses are divided into different synonyms groups. Semantic relations are built among different synsets.

Table 3 shows the content stored in this semantic thesaurus and the numbers of each kind of items.

Table 3. Content in semantic thesaurus

Items	Number
words	147306
word senses	206941
synsets	114038

There are five kinds of semantic relations defined among synsets: “synonym”, “hypernym”, “iterative hypernym”, “similar-to” and “antonym”. For each of the semantic relations, a specific value (between 0 and 1) is assigned to it. Table 4 shows these “value and semantic relation” pairs.

Table 4. Relations and values pairs

Semantic relation	S_SeV	Remark
synonym	0.9	words from the same synset
hypernym	0.8	two synsets have this relation
similar-to	0.85	only between two adjectives
antonym	0.2	words have opposite meanings
iterative hypernym	0.8^n	inheritance hypernym relation

In table 4, all the “S_SeV” values are assigned directly (by experience); these values should be assigned with more reasonable methods.

With the huge content stored in the semantic thesaurus (shown in table 3 and table 4), formula (1), (2), (3) and (4) can work for GMTM.

5 Conclusion

In this paper, a general model transformation methodology (GMTM) is presented. This methodology aims at dealing with the data sharing problem of enterprise interoperability. As the inner requirement of interoperability: flexibility, faster exchange information, this general methodology should surmount the traditional model transformation practices’ weaknesses (limited to specific domains).

Some points, which need to be improved in the future, are listed below:

- The impact factors such as: “sem_weight” and “pn_weight” and threshold values: the better way of assigning them is by using some mathematic strategy (“choquet” integral?).
- Semantic checking measurement: only formal English words are stored in the semantic thesaurus with semantic meanings; not for words (in specific cases).
- The S_SeV values that defined in table 4: more test cases are needed to modify these values into reasonable scope.

The usage of GMTM is not limited to the interoperability domain; GMTM allows MDE theories to serve other engineering domains too.

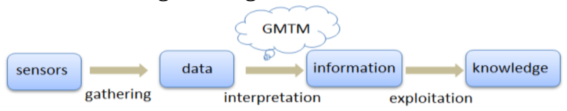

Fig. 9. Position of GMTM usage

Fig. 9 shows the general contribution of GMTM: converting rough data to 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 general model transformation methodology is created. With the improvement on some of the details that involved in this GMTM, this methodology may serve to a large number of domains.

REFERENCES

1. AMICE. CIMOSA: CIM Open System Architecture (2nd ed.). Berlin: Springer-Verlag (1993)
2. Konstantas, D., Bourrières, J.P., Léonard, M., Boudjlida, N.: Interoperability of Enterprise Software and Applications. IESA’05, Springer-Verlag (2005)
3. Nancy, Ide and Pustejovsky, J.: What does interoperability mean, anyway? Toward an Operational Definition of Interoperability. In Proceedings of the Second International Conference on Global Interoperability for Language Resources (ICGL 2010).
4. Chen, D., Dassisti, M., Elvesæter, B., Enterprise Interoperability Framework and Knowledge Corpus - Final report Annex: Knowledge Pieces", Contract no.: IST508 011, Deliverable DI.3 Annex, May 21, 2007.
5. Schmidt, D.C.: Model-Driven Engineering. IEEE Computer, February 2006 (Vol. 39, No. 2) pp. 25-31.
6. Soley, R., and the OMG staff: Model-Driven Architecture, OMG Document, November 2000, http://www.omg.org/mda.
7. Bezivin, J. “Model driven engineering: An emerging technical space,” in Generative and Transformational Techniques in Software Engineering, International Summer School - GTTSE, 2006, pp. 36–64.
8. Castro, D.V., Maros, E., Vara, J.M.: 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.
9. Grangel, R., Bigand, M., Bourey, J.P.: Transformation of decisional models into UML: application to GRAI grids. International Journal of Computer Integrated Manufacturing, 2010, Volume 23, Issue 7.
10. Czarnecki, K., Helsen, S.: Classification of Model Transformation Approaches. OOPSLA’03 Workshop on Generative Techniques in the Context of Model-Driven Architecture (2003)
11. Bénaben, F., Mu, W., Truptil, S., Pingaud, H.: Information Systems design for emerging ecosystems. 4th IEEE International Conference on Digital Ecosystems and Technologies (DEST 2010).
12. Del Fabro, M.D., Bézivin, J., Jouault, F., Breton, E.: AMW: A Generic Model Weaver. 1ère Journées sur l'Ingénierie Dirigée par les Modèles: Paris (2005)
13. Object Management Group, MOF 2.0 Query / Views / Transformations RFP. OMG Document (2002)
14. William, W. C., Pradeep, R., Stephen, E. F., A Comparison of String Metrics for Matching Names and Records. KDD Workshop on Data Cleaning and Object Consolidation, 2003, Vol. 3.
15. Wilbert H.: Measuring Dialect Pronunciation Differences using Levenshtein Distance. Ph.D. thesis, Rijksuniversiteit Groningen (2004)
16. Huang, X., Zhou, C., An OWL-based WordNet lexical ontology. Journal of Zhejiang University, 2007, pp. 864-870.