A Meta-model based Automatic Conceptual Model-to-Model Transformation Methodology

Tiexin Wang[1], Sebastien Truptil[2], Frederick Benaben[2] and Chuanqi Tao[1]
[1]*College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Jiangjun Road, Nanjing City, China*
[2]*Centre Genie Industriel, IMT Mines Albi, Universite de Toulouse, Campus Jarlard, 81000 Albi, France*

Keywords: Conceptual Dissimilarity, Automatic Model Transformation, Semantic Checking Measurements, Meta-model based Transformation Process.

Abstract: Since model-based engineering theories and techniques becoming mature gradually, diverse engineering domains have adopted the idea of employing modelling and model transformations to help simulate and analyze domain specific problems. Consequently, substantial numbers of modelling techniques have been developed. These modelling techniques define specific semantic and syntactic representations. Moreover, models are normally built to represent systems from diverse domains. Both the conceptual dissimilarities between modelling techniques and between diverse systems determine the particularity of models. In model transformation process, distinguishing the conceptual difference from both semantic and syntactic aspects is a time-consuming process relying mainly on manual effort. In order to remove the manual effort from model transformation process, this paper proposes a generic automatic conceptual model-to-model transformation methodology. This methodology employs semantic and syntactic checking measurements to automatically detect the conceptual dissimilarities, and aims to solve both domain specific problems and cross-domain problems. A refined meta-model based model transformation process is defined to better use the two checking measurements.

1 INTRODUCTION

With the gradually mature of the theories and techniques in model based engineering (MBE), more and more engineering domains have adopted MBE principles to solve domain problems. As two of the key concepts in MBE "modelling" and "model transformation" attract attention from both academics and industrials.

Modelling means the activities of building models. For different purposes, substantial numbers of modelling techniques (e.g., UML, BPMN) have been developed by employing specific semantic and syntactic representations. A research of modelling is presented in (Muller et al., 2012).

Models are built to represent systems, and model transformations can simulate the interactions or indicate the connections between systems. Furthermore, for a specific system, concerning different views, many models can be built to represent it. Many defined model, table 1 shows four definitions of model.

Table 1: Four definitions of model.

No.	Definitions
1	"Models provide abstractions of a physical system that allow engineers to reason about that system by ignoring extraneous details while focusing on the relevant ones." (Brown, 2004)
2	"A model is an abstraction of a (real or language based) system allowing predictions or inferences to be made." (Kühne, 2006)
3	"A model of a system is a description or specification of that system and its environment for some certain purpose." (OMG, 2006)
4	"Engineering models aim to reduce risk by helping us better understand both a complex problem and its potential solutions before undertaking the expense and effort of a full implementation." (Selic, 2003)

A model is particular because it is built for a specific purpose (e.g., describing a view of a complex system) and by using a specific modelling technique. Models can be divided into different groups. As stated in (Fowler et al., 1999),

Wang, T., Truptil, S., Benaben, F. and Tao, C.
A Meta-model based Automatic Conceptual Model-to-Model Transformation Methodology .
DOI: 10.5220/0006718105860593
In *Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2018)*, pages 586-593
ISBN: 978-989-758-283-7

depending on the level of precision, models are divided into three levels namely **Conceptual Models**, *Specification Models* and *Implementation Models*. Another similar distinction proposed in (Mellor, 2004), a model can be considered as a *Sketch*, as a *Blueprint*, or as an *Executable*.

In order to build connections between models in the same level and from different levels, model transformation practices are required. However, in model transformation practices, distinguishing the conceptual difference between two models is a time-consuming process which is mainly relied on manual effort.

As stated in (Del Fabro and Valduriez, 2009), in traditional model transformation practices there are several weaknesses: low reusability, contain repetitive tasks and involve huge manual effort, etc. Due to the wide requirement and usage of model transformation practices, it is unacceptable to do model transformation manually. Thus, this paper proposes a generic (domain-cross) automatic conceptual model-to-model transformation methodology (ACMTM), which is built on the base of semantic and syntactic checking measurements (S&S). S&S is used to automatically detect the conceptual similarities and build mapping rules. A refined meta-model based model transformation process is defined to better combine S&S in.

This paper is structured as follows. Section 2 presents the relevant theories to ACMTM. Section 3 shows an overview of ACMTM. A use case is illustrated in Section 4. Finally, a conclusion draws the advantages, potential improvement points and future usage of ACMTM.

2 RELATED WORK

2.1 Model Transformation Definitions

Model transformation is a process, which contains a sequence of activities operating on models. Many propose the definitions about model transformation. Table 2 shows three of them.

Model transformation **is a process of generating target models** based on **source models. The transforming rules** shall be built between **same or similar concepts** that are from the two models, respectively.

Table 2: Three definitions of model transformation.

No.	Definition
1	"model transformation is a program that mutates one model into another" (Tratt, 2005)
2	"the process of converting a model into another model of the same system" (Miller and Mukerji, 2003)
3	"automatic generation of a target model from a source model, according to a transformation description" (Kleppe et al, 2003)

2.2 Model Transformation Category

Generally, model transformation can be divided into three groups: ***Text-to-Model***, **Model-to-Model** and ***Model-to-Text***. The content in models is presented in abstract syntax, while the content in text is presented in concrete syntax.

As defined in (Czarnecki and Helsen, 2003), there are two main model transformation approaches: **model-to-code** and **model-to-model**. For model-to-code category, there are two kinds of approaches: "*visitor-based" approaches* and *"template-based" approaches*. For model-to-model category, there are five approaches: *"direct-manipulation" approaches, "relational" approaches, "graph-transformation-based" approaches, "structure-driven" approaches and "hybrid" approaches*. In **model-to-model** transformation category, there are also some other approaches, such as: *marking and pattern approach, automatic transformation approach,* **meta-model based transformation approach**, *model merging approach, etc*.

ACMTM belongs to **model-to-model transformation category**. It is designed and implemented as **a hybrid approach which is also a meta-model based**.

2.3 Model Transformation Techniques

Focusing on model-to-model transformation category, there are several well-known techniques. Table 3 shows four of these techniques.

ATL and QVT are similar to each other on architecture aspect. Both VIATRA2 and GReAT focus mainly on graph models. Usually, specific model transformation techniques can be only used on models that are built by specific modelling techniques. Also, model transformation techniques integrate (or rely on) other techniques, such as: QVT – OCL, VIATRA2 – graph transformation techniques, etc. Current model transformation techniques lack the ability of automatically detecting

model transformation mappings, and require manual effort to operate them.

Table 3: Model-to-model transformation techniques.

Name	Characteristic	Note
ATL (Jouault et al., 2008)	**Hybrid** (declarative & imperative); three layers architecture	self-executed (provide both transformation language & **toolkit**)
QVT (Omg, 2008)	**Hybrid** three kinds of transformation languages involved	based on **MOF** 2.0 (Omg, 2008) integrated **OCL**
VIATRA2 (Varró and Balogh, 2007)	**Unidirectional** transformation language; based mainly on **graph transformation techniques**	operates on models conformed to **VPM meta-modeling approach**
GReAT (Karsai et al., 2003)	**Visual language** developed using **Generic Modeling Environment**	operates on models conform to **meta-models specified in UML**

Based on these model transformation techniques, numerous model transformation practices have been developed, such as the work stated in (De Castro et al., 2011; Fleurey et al., 2007; García et al., 2013).

Comparing with the existing model transformation techniques and practices, **ACMTM aims to be a generic, automatic conceptual model-to-model transformation methodology**. It provides a theoretical framework and employs semantic and syntactic checking measurements as potential mappings detecting techniques.

3 ACMTM OVERVIEW

ACMTM employs S&S in a refined meta-model based model transformation process. S&S is illustrated first in this section. Then, the refined transformation process is presented.

3.1 Semantic & Syntactic Comparisons

3.1.1 Use of S&S

In ACMTM, semantic checking and syntactic checking measurements are combined as a single function. This function is used between items on meta-model level. Figure 1 shows the relation between them and its usage in ACMTM.

S&S takes two words (strings) as inputs, and its output is the matching possibility between the two strings. For the syntactic checking part, it contains two steps: predefined treatment (**pretreatment**) and employing "Levenshtein distance" algorithm (Hirschberg, 1997; Gilleland, 2009).

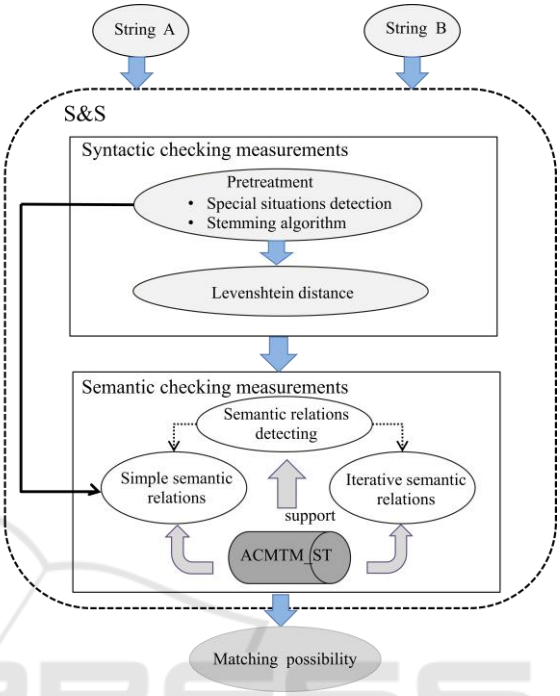

Figure 1: S&S illustration.

"Predefined treatment" also contains two phases: **special forms detection** and applying **stemming algorithms.** Both the two phases aims to discover special semantic relations (e.g., synonym and antonym) between a pair of words. If the pretreatment step fails in discovering such kinds of semantic relations, then the second step employing "Levenshtein distance" algorithm will be executed. This algorithm calculates the syntactic similarity between a pair of words. This syntactic similarity stands by a value ranges between 0 and 1.

In order to detect the potential semantic relations between two comparing words, a semantic thesaurus "ACMTM_ST", is created. Semantic relations stands by a calculating (or assigned) value defined within ACMTM context.

Equation (1) is defined to calculate the S&S relation, between two words (strings). The S&S relation is represented by a value which is the sum of two aspects: semantic and syntactic.

$$S\_SSV = SeV\_weight * S\_SeV + SyV\_weight * S\_SyV \quad (1)$$

"S_SSV" stands for the S&S value between a pair of words. "S_SeV" stands for the semantic

value while "S_SyV" stands for the syntactic value. Two coefficients: "SeV_weight" and "SyV_weight" are defined. Their value range is "0" to "1", and the sum of them is '1'. They are used to determine which aspect is more important in determining S&S value between a pair of words.

3.1.2 Syntactic Checking Measurements

Syntactic checking measurements focus on forms of words (e.g., do-doing, student-students), formats of concepts (date description in different cultures), and units (Celsius, Fahrenheit and Kelvin measuring temperature) used to describe subjects.

In the first checking phase, inspired by the research work stated in (Benaben et al., 2013), a profile is created and used to detect the different formats and units standing for the same or similar concepts. For words in different forms (also words belong to the same semantic group: concerning the stemming issue), a special algorithm "**word forms detecting: WF_D**" is developed to detect these situations. **WF_D** adopts parts of the "porter stemming" (Porter, 1980) algorithms.

The second phase employs "Levenshtein Distances" algorithm which is a string metric for measuring the difference between two alphabet sequences. Informally, the "Levenshtein distance" between two words is the **minimum number** of single-character edits (i.e. **insertions, deletions** or **substitutions**) required to change one word into the other.

Mathematically, the Levenshtein distance between two strings: string a and string b with the length $|a|$ and $|b|$, respectively) is given by "$Lev_{a,b}(|a|, |b|)$".In order to use this value, equation (2) is defined.

$$S\_SyV = 1 - Lev_{a,b}(|a|, |b|) / max(|a|, |b|) \quad (2)$$

The value of "S_SyV", which first appears in equation (1), shall always be within the range of 0 to 1. The higher of this value means the higher syntactic similarity between two comparing words.

3.1.3 Semantic Checking Measurements

Semantic checking measurements focus on the semantic meanings. Between a pair of words, one syntactic similarity value always exists, while several or no semantic relations (with different semantic values) can exist.

To support semantic checking, **ACMTM _ST**, which contains large amount of words, semantic meanings and semantic relations, is particularly created to support ACMTM. It adopts parts of the content stored in "WordNet" (Fellbaum, 1998). Figure 2 shows the structure of ACMTM _ST.

Three kinds of items are stored in ACMTM _ST.

- **Word Base:** contains 147306 English words (i.e., nouns, verbs and adjectives).
- **Word-sense Base:** contains 206941 senses that owned by the words stored in "Word Base".
- **Synset Base:** contains 114038 synsets. A synset contains a group of word senses, which own synonym meanings; semantic relations are built among different synsets.

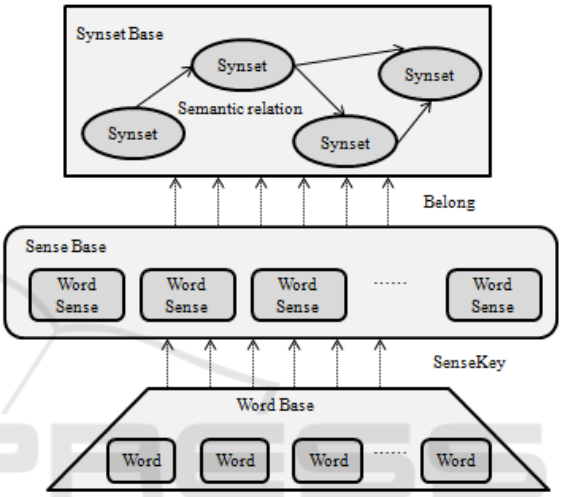

Figure 2: ACMTM_ST structure.

The relation between word and word senses is "**one-to-several**", and the relation between word senses to synset is "**several-to-one**". Eleven kinds of semantic relations (adopted from WordNet) are maintained among synsets in ACMTM_ST. Table 4 shows these semantic relations and their values pairs.

Table 4: Semantic relations maintained in ACMTM_ST.

Semantic relation	S_SeV	Example
synonym	0.9	shut & close
hyponym	0.6	person-creator
hypernym	0.8	creator-person
similar-to	0.85	perfect & ideal
partmeronym	0.7	tire & car
partholonym	0.55	car & tire
membermeronym	0.65	car & traffic jam
memberholonym	0.45	traffic jam & car
Antonym	0.1	good & bad
iterative hyponym	0.6^n	person-creator-maker
iterative hypernym	0.8^n	maker-creator-person

The "S_SeV" (first introduced in equation (1)) stands for the semantic similarity between a pair of comparing words. The higher of this value means the closer of the two words in semantic aspect. All these "S_SeV" values are assigned directly (based on experience).

Both the calculating rules for "S_SyV" and "S_SeV" are illustrated. The "S_SSV" between any pair of comparing words is computable.

On the basis of semantic and syntactic checking measurements, a "S_SSV" value can be calculated. This "S_SSV" value means the possibility of matching two words. The determination mechanism is shown in Figure 3.

Figure 3: Matching pair chosen mechanism.

According to the range of S_SSV, three regions are divided. If two words have a S_SSV in Region 1, the two words have a high matching possibility. While this value in Region 2, the two words have a medium match degree. If this value is in region 3, no matching can be made between the two words.

3.2 ACMTM Theories & Process

The S&S illustrated above are used between word pairs, while ACMTM focuses on transforming models. So, a refined meta-model based model transformation process is created.

3.2.1 ACMTM-MMM

Meta-model is a special kind of model which defines the rules of building models. Meta-models can exist in several levels.

In a model transformation process, a model is regarded as two parts: **shared part (transformable)** and **specific part (non-transformable)**. Both shared and specific part on model layer can be traced on meta-model layer as shared and special concepts. In this way, identifying the shared part on model layer becomes detecting the shared concepts on meta-model layer. In ACMTM, the mechanism of applying S&S is defined in a meta-meta-model.

There are several meta-modelling architectures, two of them are: "MOF: Meta-Object Facility" (Omg, 2008) and "ISO/IEC 24744" (Henderson-Sellers and Gonzalez-Perez, 2008). These are general-purpose architectures. For supporting particularly to model transformation field, a specific meta-meta-model "ACMTM-MMM" is created.

As shown in Figure 4, there are nine core elements in this meta-meta-model. "**Model**" stand for all the model instances. "**Model**" is made of "**Element**", which has two inheritances: "**Node**" (concepts) and "**Edge**" (relations). "Element" is self-contained. "Node" are linked by "Edge" based on their "**roles**". "**Element**" has a group of "**Property**", "**Property**" can identify and explain the "Element". "**Semantic Relation**" and "**Syntactic Relation**" exist between different kinds of items (i.e. between element's pairs, between property's pairs, between models pairs and between environment's pairs). Potential model transformation mappings are built based on them.

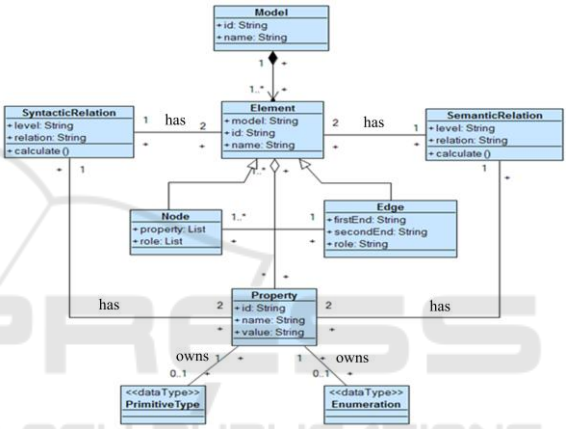

Figure 4: The structure of ACMTM-MMM.

3.2.2 Iterative Transformation Process

Model transformation is regarded as an iterative process in ACMTM. Between the original source model and final target model, several intermediate models can be generated. The target model of former iteration becomes the source model of latter iteration.

In each iteration phase, the specific part of source model shall be stored in ontology named "ACMTM_O". Also, the specific part of target model shall be enriched by additional knowledge stored in ACMTM_O.

As shown in Figure 5, the content stored in ACMTM_O comes from both the specific part of source models and other knowledge base (e.g., domain ontologies).

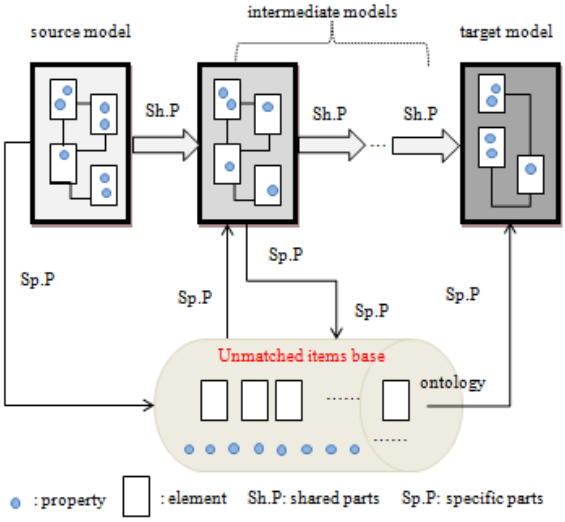

Figure 5: Iterative transformation process illustration.

3.2.3 Four Matching Steps

To apply S&S to build potential mappings between meta-models, four matching steps are divided. These four matching steps aim to solve the inherent **granularity issue** in model transformation domain: M-to-N matching and cross-level (element-property) matching. Figure 6 shows an overview of the four matching steps.

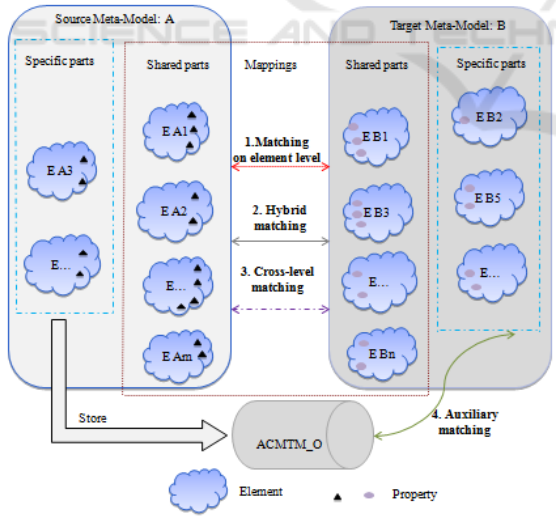

Figure 6: Four matching steps.

The first step "**matching on element level**": aims to build mappings between *element's pairs* (considering **elements' names** and their **property groups**) and between *property's pairs* (considering **properties' names** and **types**) which are within the matched element pairs. Two equations: (3) and (4) are defined to do this matching step.

$$Ele\_SSV = name\_weight * S\_SSV + property\_weight * \left(\sum_{i=1}^{x} \max(P\_SSVi)\right)/x \quad (3)$$

$$P\_SSV = pn\_weight * S\_SSV + pt\_weight * Id\_type \quad (4)$$

The second step "**hybrid matching**" focuses on properties (property-to-property matching), which are unmatched after executing the first matching step. Equation (5) is defined for this matching step.

$$HM\_SSV = en\_weight * S\_SSV + pl\_weight * P\_SSV \quad (5)$$

The third step "**cross-level matching**" concerns making mappings between properties and elements. This step focuses on the unmatched elements and properties after executing the two former matching steps. S&S are applied between **elements' names** and **properties' names**. Equation (6) is defined to work for this step.

$$CM\_SSV = sem\_weight * S\_SeV + syn\_weigh * S\_SyV \quad (6)$$

"**Ele_SSV**" stands for the semantic and syntactic value between an element's pair, while "**P_SSV**" stands for this value between property's pairs. "**HM_SSV**" stands for the value of hybrid matching and "**CM_SSV**" for cross-level matching value. All of the four values are the sum of two variables. In each of the equations, two impact factors (e.g. name_weight & property_weight), the sum of them is 1, are defined to determine which of the two variables plays a more important role in deciding the final equation value.

All the three matching steps aim to define mappings within the shared part. For the specific parts, the fourth step "auxiliary matching" can be used.

"A**uxiliary matching**" focuses on enriching the specific parts of target models by extracting additional knowledge from ACMTM_O. It **reuses the three former matching steps to detect potential model transformation mappings, while taking ACMTM_O as the source meta-model.**

4 USE CASE

To explain and test the working mechanism of ACMTM, a simple use case is illustrated in this section. This use case concerns the process of comparing two "Elements". The two elements are shown in Figure 7.

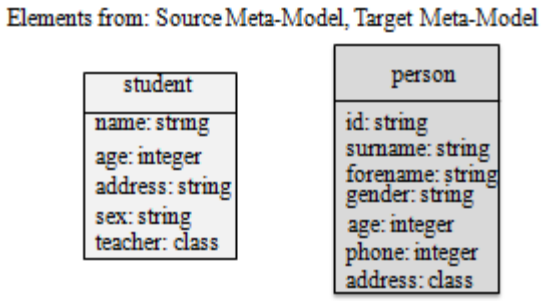

Figure 7: Iterative transformation process illustration.

Two elements "student" (with five properties) and "person" (with seven properties) are taken as inputs. The outputs are potential mappings between them. Before executing the detecting process, concrete values are assigned to the parameters used in equation (1), (3), and (4). Table 5 shows the assigning value pairs.

Table 5: Assigning values to parameters.

No. Equation	Parameter	value
1	SeV_weight, SyV_weight	0.9, 0.1
3	name_weight, property_weight	0.5, 0.5
4	pn_weight, pt_weight	0.8, 0.2

Taking the calculation process of "Ele_SSV" between two elements: "student" and "person" as an example; equation (3) is used to do this step. Figure 8 is the screenshot of calculating the "S_SSV" value between elements' names: "student" and "person".

```
The student has 2 senseKeys
The student has sense of: student_NN_1
The student has sense of: student_NN_2
The student_NN_1 belongs to the synset of WN30-110665698
The student_NN_2 belongs to the synset of WN30-110557854
The person has 3 senseKeys
The person has sense of: person_NN_1
The person has sense of: person_NN_2
The person has sense of: person_NN_3
The person_NN_1 belongs to the synset of WN30-100007846
The person_NN_2 belongs to the synset of WN30-105217688
The person_NN_3 belongs to the synset of WN30-106326797
The iterative hypernym semantic relation comes from the synset: WN30-100
The Syntactic relation value between the two comparing words is: 0.14285
The Semantic relation value between the two comparing words is: 0.640000
The S&S between the two comparing words is: 0.5902857142857145
```

Figure 8: S&S comparisons between elements' names.

The word "student" has two semantic meanings, and the word "person" has three semantic meanings. The semantic relation between the two words is "iterative hypernym", and the semantic value between them is "0.64". The syntactic similarity value between them is: 0.1428. In this use case, semantic relation is assumed more important than syntactic relation, so two coefficients: "SeV_weight" and "SyV_weight" in equation (1) are assigned with values as 0.9 and 0.1, respectively. The final S&S value between the two words is: **0.5903**.

The S&S comparisons between the two elements' properties groups are calculated by using equation (4). Table 6 is created to store these comparison values. Between each pair of properties, a "P_SSV" can be calculated. The two parameters "pn_weight" and "pt_weight" are assigned with values 0.8 and 0.2. This means property name is more important than property type when making mappings.

When calculating Ele_SSV, the two parameters in Equation (3) are assigned as 0.5 and 0.5. This means element name and property group have the same weight in deciding element matching pairs.

The "Ele_SSV" calculated between "student" and "person" is: **0.695**. According to the matching pair chosen mechanism, there is a medium potential mapping exist between them.

5 CONCLUSION

This paper presents an automatic **conceptual model-to-model** transformation methodology: ACMTM. Comparing with the existing model transformation methodologies, two main characteristics of ACMTM are: generic and automatic.

ACMTM combines semantic and syntactic checking measurements into a refined meta-model based model transformation process. Also, ACMTM takes model transformation as an iterative process and four matching steps are divided within each iteration phase. To better use S&S, five equations have been defined to use in different matching steps.

Some potential improvements in ACMTM are as follows.

- A validation and evaluation process of the automatic generated model transformation mappings is required.
- Strengthen semantic checking measurements by extending ACMTM_ST with more content from specific domains (e.g., ontology).
- A better way to assign values to coefficients defined in equations (3), (4), (5) and (6) (e.g., mathematical, statistical analysis).

Table 6: S&S comparisons for property groups.

person \ student	id	surname	…	age	phone	address
id	1	-	…	-	-	-
name	0.2	0.6777	…	0.04	0.016	0.011
age	0	0.0229	…	1	-	-
address	0.21	0.2	…	0.02	0.011	0.8
sex	0.2	0.2114	…	0	0	0.011
teacher	0	0	…	0.02	0.011	0.2

ACKNOWLEDGEMENTS

The authors would like to acknowledge the financial support from European Commission C2Net project (H2020-FoF-1-2014/636909), Chinese Scholarship Council, National Natural Science Foundation of China (61502231) and Natural Science Foundation of Jiangsu Province (BK20150753).

REFERENCES

Benaben, F., Boissel-Dallier, N., Pingaud, H., Lorre, J. P. 2013. Semantic issues in model-driven management of information system interoperability. International Journal of Computer Integrated Manufacturing,26(11), 1042-1053.

Brown, A.W., 2004. Model driven architecture: principles and practice. SoSyM 3(3), 314–327.

Czarnecki K, Helsen S., 2003. Classification of model transformation approaches[C]//Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture, 45(3): 1-17.

De Castro V, Marcos E, Vara J M., 2011. Applying CIM-to-PIM model transformations for the service-oriented development of information systems [J]. Information and Software Technology, 53(1): 87-105.

Del Fabro, M. D., & Valduriez, P., 2009. Towards the efficient development of model transformations using model weaving and matching transformations. Software & Systems Modeling, 8(3), 305-324.

Fellbaum, C., 1998. WordNet. Blackwell Publishing Ltd.

Fleurey, F., Baudry, B., France, R., Ghosh, S., 2007. A generic approach for automatic model composition. In Models in Software Engineering (pp. 7-15). Springer Berlin Heidelberg.

Fowler, M., Scott, K., Booch, G., 1999. UML distilled, Object Oriented series, 179 p.

García, J., Diaz, O., Azanza, M., 2013. Model transformation co-evolution: A semi-automatic approach. Software Language Engineering,7745, 144-163.

Gilleland, M., 2009. Levenshtein distance, in three flavors. Merriam Park Software: http:// www.merriampark.com/ld.htm.

Henderson-Sellers B, Gonzalez-Perez C., 2008. Standardizing methodology metamodelling and notation: an ISO exemplar [M]. Springer Berlin Heidelberg.

Hirschberg, D., 1997. Serial computations of Levenshtein distances.

Jouault, F., Allilaire, F., Bézivin, J., & Kurtev, I., 2008. ATL: A model transformation tool. Science of computer programming, 72(1), 31-39.

Karsai, G., Agrawal, A., Shi, F., & Sprinkle, J., 2003. On the use of graph transformation in the formal specification of model interpreters. J. UCS, 9(11), 1296-1321.

Kleppe, A. G., Warmer, J. B., Bast, W., 2003. MDA explained: the model driven architecture: practice and promise. Addison-Wesley Professional.

Kühne, T., 2006. Matters of (meta-) modeling. SoSyM, 5(4).

Mellor, S. J., 2004. MDA distilled: principles of model-driven architecture. Addison-Wesley Professional.

Miller, J., Mukerji, J., 2003. MDA Guide Version 1.0. 1

Muller, P. A., Fondement, F., Baudry, B., & Combemale, B., 2012. Modeling modeling modeling. Software & Systems Modeling, 11(3), 347-359.

OMG 2006: Model Driven Architecture. http:// www.omg.org/mda/.

Omg, 2008. Meta object facility (mof) 2.0 query/view/transformation specification. Final Adopted Specification.

Porter, M. F., 1980. An algorithm for suffix stripping. Program, 14(3), 130-137.

Selic, B., 2003. The pragmatics of model-driven development. IEEE Softw. 20(5), 19–25.

Tratt, L., 2005. Model transformations and tool integration. Software & Systems Modeling, 4(2), 112-122.

Varró, D., & Balogh, A., 2007. The model transformation language of the VIATRA2 framework. Science of Computer Programming, 68(3), 214-234.