A study to define an automatic model transformation approach based on semantic and syntactic comparisons

Tiexin Wang

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

École Nationale Supérieure des Mines d'Albi-Carmaux conjointement avec l'INSA de Toulouse

---

Présentée et soutenue par :
Tiexin WANG
le Jeudi 10 décembre 2015

Titre :
A study to define an automatic model transformation approach based on semantic and syntactic comparisons

---

École doctorale et discipline ou spécialité :
EDSYS : Génie Industriel 4200046

Unité de recherche :
Centre Génie Industriel - École des Mines d'Albi-Carmaux

Directeur/trice(s) de Thèse :

Frédérick Bénaben

Jury :

David CHEN	Président	Professeur, Université Bordeaux
Chihab HANACHI	Examinateur	Professeur, Université Paul Sabatier Toulouse
Jean-Pierre BOUREY	Rapporteur	Professeur, Ecole Centrale de Lille
Jordi CABOT	Rapporteur	Professeur, Univ. Oberta de Catalunya
Frédérick BENABEN	Directeur de thèse	Maître assistant HDR, Mines Albi
Sébastien TRUPTIL	Co-directeur	Maître assistant, Mines Albi

Acknowledgements

Time flies!
It has been more than four years since the first time that I stepped on the ground of France; at that time, I was a student pursuing a master degree. Now, my postdoctoral contract has been carried out for three months. During the last four years, for me, there are too many moments of both happiness and bitterness. Also, I grew up from a teenager to a man. I should say thank you to many people who helped me become a doctor and enjoy the life in France.
My deepest gratitude goes first and foremost to the director of my thesis Professor Frederick Benaben and my supervisor Sebastien Truptil, for their constant encouragement and guidance. To me, they are not only my colleagues (help me with work) but also my friends (take care of my life). Without their consistent and illuminating instruction, my thesis could not have reached its present form; without their concern and care, my life in France would be totally different (becoming a tough one).
Second, I would like to express my heartfelt gratitude to all of my colleagues from the laboratory "CGI" – EMAC; they are so kind and enthusiastic. "CGI" is a family and all the colleagues are always ready to help each other. We had group activities (e.g. the buffet, badminton match and mountain biking) and we shared the happiness. The friendship is enhanced through such kind of activities. Especially, I want to mention two colleagues "Aurélie Montarnal" and "Isabelle Fournier", they helped me a lot with the official issues (as a foreigner, the official issues are really miscellaneous and distracting in France).
Next, I am also greatly indebted to my friends here. They did sports with me and gave me pleasant leisure in my spare time. Because of them, my life here is more rich and colorful. Good restaurants, special foods, visiting attractions, etc. they are everywhere within my good memories. I built international friendship with friends from America, Montenegro, Costa Rica, etc. I really cherish the hard-won friendship. Many good people did me good things and left good memories to me. One of them is Laurent Steffan, he is the first one in France that I regarded as my true foreign friend. No matter where I am in the future, I will always remember him.
Finally and most importantly, my thanks would go to my beloved family for their loving considerations and great confidence in me all through these years. Without the support from my parents, I could nerve image that I can study abroad and get PhD diploma in a developed country. To be a son and the only child in my family, words could not convey all my feelings to my parents. Hope everything goes well with them.

Contents

List of figures

General Introduction

Chapter I Problem statement

Chapter II Literature Review

Chapter III AMTM Overview

Chapter IV S&S mechanism

Chapter V Semantic Checking Measurements

Chapter VI Syntactic Checking Measurements

Chapter VII AMTM-SS Implementation & Use Case

General Conclusion

List of tables

General Introduction

To describe or measure a subject, different words and units might be used. For instance, Celsius, Fahrenheit and Kelvin could be used to describe temperature; however, for a same temperature (e.g. 15 ℃), the three measurements use different values to describe it as: 15℃, 59 ℉ and 288.15°K. Another example, to describe a date (e.g. 15th of September, 2015), people from different countries use different formats to represent it, such as: 15/09/2015 in France, 09/15/2015 in United State of America and 2015-09-15 in China, etc. The two instances reveal the difference in ***syntactic aspect*** of describing a same subject. Furthermore, the difference also exists in ***semantic aspect*** of describing a same subject. For example, within the same language (e.g. English), different words could have similar meanings, for instance: fine, glorious and happy. Also, there are different relationships between different words, such as: inclusion (automobile - van), composition (university - faculty) and inheritance (father - son), etc. In different languages, same subjects stand by different words, student (English) to étudiant (French).

Therefore, bridging the gap between syntactic and semantic differences could help people coming from different cultures (e.g. Muslim and Chinese) or different regions (England and Australia) to work together. Such a working environment (heterogeneous partners involved) could be regarded as collaboration. Nowadays, the rapid development of science and technology in various engineering domains makes the world become "smaller and smaller". In some aspects, this trend leads changes in people's lifestyles and ways of working; furthermore, it also leads the appearance of more and more collaborations among countries, organizations and persons, who are seeking to an optimal solution (e.g. focusing on political factor, interest factor, and emotional factor).

Comparing to the traditional collaborations, the new coming collaborations own characteristics such as: last short period, dynamic combination of partners and globally cooperation (Touzi et al, 2007). These collaborations are created to achieve specific goals; for achieving a common goal, partners coming from different domains might be involved in. Furthermore, the involved partners might be heterogeneous: speaking different languages, belonging to different systems, applying various tools (especially on information technique aspect) and obeying different cultures. Thus, collaboration could be seen as a "system of systems (SoS)"; as stated in (Maier, 1998), there are five criteria to define a SoS: (i) **Operational independence of the systems**, (ii) **Managerial independence of the systems**, (iii) **Evolutionary development of the system of systems**, (iv) **Emergent behavior of the system of systems**, and (v) **Geographical distribution of elements**. Considering the context of SoS, each partner involved in collaboration is a system and all partners compose the system of systems. Focusing on the particular partners (systems), two criteria mentioned above: **operational and managerial** independences are important. Indeed, they imply that each partner has to work with its own means and vocabulary, thus both of the two criteria concern about the semantic and syntactic aspects conveyed by the partners (systems).

The characteristics of new collaborations bring new problems, which need to be solved. A common problem of new collaborations is: **how to build efficiently and effectively collaboration among the partners (systems)**? Generally, the main obstacles in these collaborations exist on the different *syntactic and semantic representations* used by partners. Moreover, these obstacles could be divided into two levels: *human and technique*; the idea of this division is adopted from (Chen et al, 2008). Figure GI-1: Illustration of collaborative situations and its obstacles. Figure GI-1 shows a simple illustration of collaborations and their obstacles.

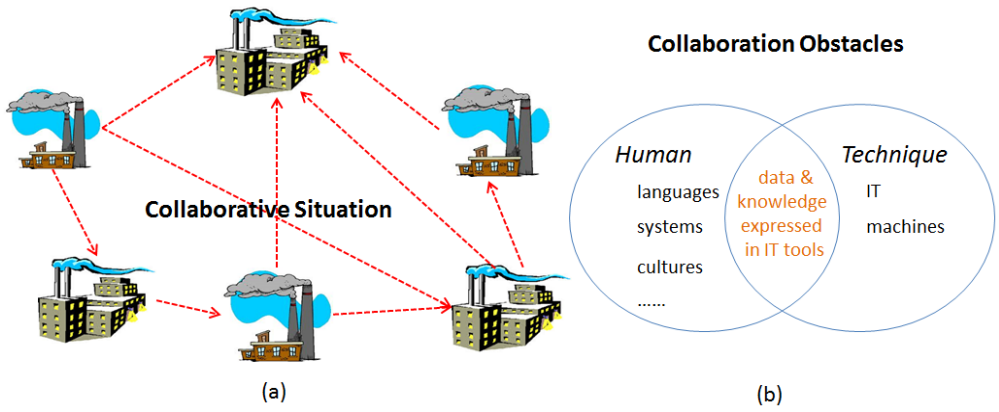


Figure GI-1: Illustration of collaborative situations and its obstacles.

Figure GI-1 part (a) is a simple illustration of a collaborative situation, which contains six partners. Based on the concept of System of Systems, each partner could be seen as a heterogeneous and autonomous partner, which has to exchange information with other partner during all the lifecycle of the collaboration. These interactions meet barriers from two levels shown in Figure GI-1 part (b): **human** and **technique**. On human level, the obstacles are: different speaking and writing languages, different working systems and different embracing cultures; on technique level, the main obstacles are mainly in the technical tools: production machines and information technology (IT) tools (e.g. information systems: IS). There exists an overlap region between the two levels: data and knowledge from human level that are stored and expressed in IT ways. As stated in (Alessandro et al, 2008), a computer Information System (IS) is a system composed of people and computers that processes or interprets information. Moreover, the distinction between human and IT is also underline in the definition of Information System given by (Morley, 2002). Morley defines Information System as a system composed of two sub-systems, which are: Information treatment system and Informatics system.

Adopted the idea expressed in (Chen et al, 2007) "barriers in enterprise interoperability", one of the difficult points in collaboration on human level is communication. The communication difficulties come from two aspects: languages (e.g. speaking and writing) and working systems (e.g. organizational, management).

- **Languages**: communication (with a shared understandable language) among partners is a preferable solution; but different partners may use their own languages, which have specific: vocabulary, grammar, special expressions, etc. The structure of one language could be similar to the structure of another language (e.g. French and Italian) or totally different to each other (e.g. Chinese and English).
- **Working Systems**: normally, different countries and organizations apply different organizational structures to manage daily business. Moreover, people work in different departments and at different levels that are defined within these organizational structures. In order to improve the efficiency of collaborations, interactions should take place between same or similar departments and between people at same or similar working level.

In order to overcome the obstacles on human level, it is necessary to recognize and identify the differences in languages and in organizational structures that are used by different partners involved in

collaborations. Based on the technical point of view, the difficult points of collaboration are how to allow interaction between software and heterogeneous production machines. Nowadays, based on the growth of data, it is not realistic to think that the human level is enough to achieve the shared goals of collaboration. Thus, collaboration at the IT level is also required.

The aim of the research work, which is presented in this PhD, is to allow heterogeneous partners to work together in a hurry while keeping their own characteristics. In order to achieve this aim, here we mainly focus on the obstacles of data transfer. As explained before, these obstacles could have two kinds of sources. (i) People express and record data (knowledge) based on their languages and cultures. Normally, this kind of knowledge is domain-specific; leaving the specific context, the knowledge might become rough data or information without special meaning. (ii) Data & Knowledge expressed in IT tools: at this moment, large institutions and organizations use many kinds of IT tools (e.g. Microsoft office software, database, simulation software) to manage their daily business. The data and knowledge are heterogeneous; furthermore, these kinds of IT tools are also heterogeneous: build within different software environments, develop by different programming techniques, etc. Vast amounts of data and knowledge are stored in these tools. In order to overcome the obstacles of collaboration on technique level, one important aspect is to use and share these data and knowledge stored in different IT tools.

Moreover, the new coming collaborations are complex due to (i) the heterogeneity of involving partners, and (ii) the interactions between them are complex due to their heterogeneity. To analyze complex systems and simulate complex processes, models could be built to represent the systems and simulate the processes. For the reason "models could reflect characteristics of systems depending on some specific point of views" (Bézivin, 2006), different models could be built to represent one system to highlight different characteristics of it. As an example, the "Computer Integrated Manufacturing Open System Architecture": CIMOSA (Kosanke et al, 1999) is a framework, which defines the mechanism of building models for enterprise. Figure GI-2 shows the relation between a system and its models.

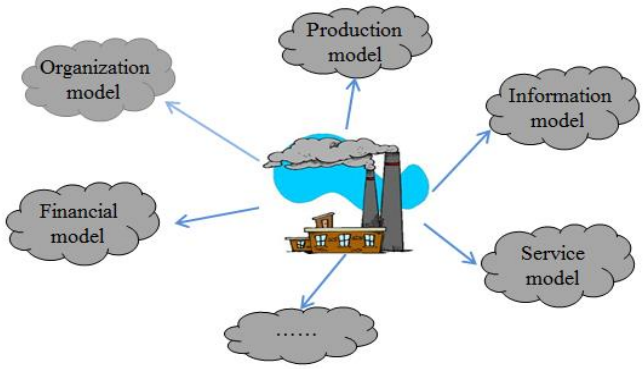


Figure GI-2: Relation between real systems and its models.

In the modeling world, model has a specific meaning; many definitions have been provided for it. Adopted from (Bézivin, 2006), a model is an abstraction representation of a subject (or part of a subject) in the real world and it is built with a specific representation language. As another key concept, **model transformation means the process of taking in a source model and generating a target model**; it could be used to build connections between different models. The two concepts: model and model transformation will be illustrated in detail in the first chapter. In modeling world, everything is a model or could be modeled and different models could be connected by model transformations. In this way, models

and model transformations could be used together to do simulation, verification and validation of complex systems and processes.

Based on the fact "**systems (e.g. partners in collaboration) could be presented by models" and models could be connected by model transformations**, we could assume that modeling and model transformation practices could be used to help collaboration to shared information without lot of human efforts. Indeed, within collaboration, each of the involved partners could be represented by a group of models; to simulate the connections among these partners, model transformations are applied. This thesis focuses on how to build these connections (comparing, sharing and exchanging knowledge among these partners). Since model transformations provide a proper solution to this kind of problems, **an automatic model transformation methodology: AMTM** is proposed in this thesis. Figure GI-3 shows the idea of using models and model transformations to serve collaboration.

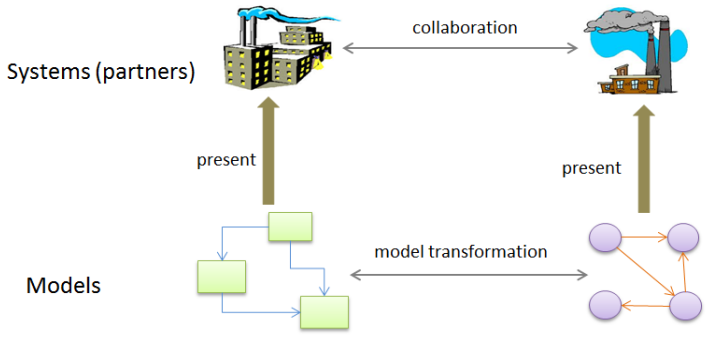


Figure GI-3: Using model and model transformations to serve to collaboration.

As a conclusion, one of the main problems of the new coming collaborations is the complexity of the heterogeneous partners and building connections (sharing and exchanging data, information and knowledge) among them. However, as the heterogeneous of the system themselves, the models built to present them are also heterogeneous. Moreover, the number of modeling techniques is numerous, models could also be heterogeneous. Due to the complexity and heterogeneity of these models, making transformations among them is also a challenge.

Large amount of model transformation principles and techniques have been developed; based on them, a large number of model transformation applications have been created for serving both domain specific problems and cross-domain problems. However, both of the two kinds of model transformation applications have their own weakness.

- **Domain specific applications**: focus on particular problematic; they have constraints on modeling techniques and applicable domains (e.g. some model transformation applications could only serve to enterprise engineering or software engineering). So, **the reusability and portability of them are poor, and thus lot of user efforts are asking to achieve it**. On the other hand, this kind of applications has many advantages: as they focus on particular problematic, they are designed **to be automatically or semi-automatically executed, easy to use**, etc.
- **Cross-domain applications**: serve broad purpose and consider about as many as possible the transformation situations. The weaknesses of them: complex to use, huge manual efforts have to

be involved, etc. In some cases, this kind of applications has also constraints on modeling techniques and applicable domains.

The new coming collaborations own specific characteristics: heterogeneous partners from various domains, transient occurrence and specific period of duration for each partner, dynamic combination of partners. Based on the fact that many powerful modeling techniques have been developed and many engineering domains have already adopted one or several of these techniques to serve their problems, an efficient model transformation methodology is needed. Such a model transformation methodology should contain the advantages: **easy to use**, **serve broad purpose** (ignore application domains) and **high efficient** (automatic executing in order to reduce manual efforts and repetitive tasks)

In order to develop a high efficient model transformation methodology, the existing model transformation principles and practices should be considered and compared to extract useful ideas; furthermore, techniques and principles from other research and engineering domains might also be involved in this methodology. Nevertheless, it is important to solve a specific problem to be able to define an easy to use methodology without lot of human effort. This problem is the exchange of heterogeneous data due to a syntactic or semantic difference between partners. The whole structure of the proposed methodology AMTM is shown in Figure GI-4. AMTM is manly based on involving semantic and syntactic checking measurements into a refined model transformation process.

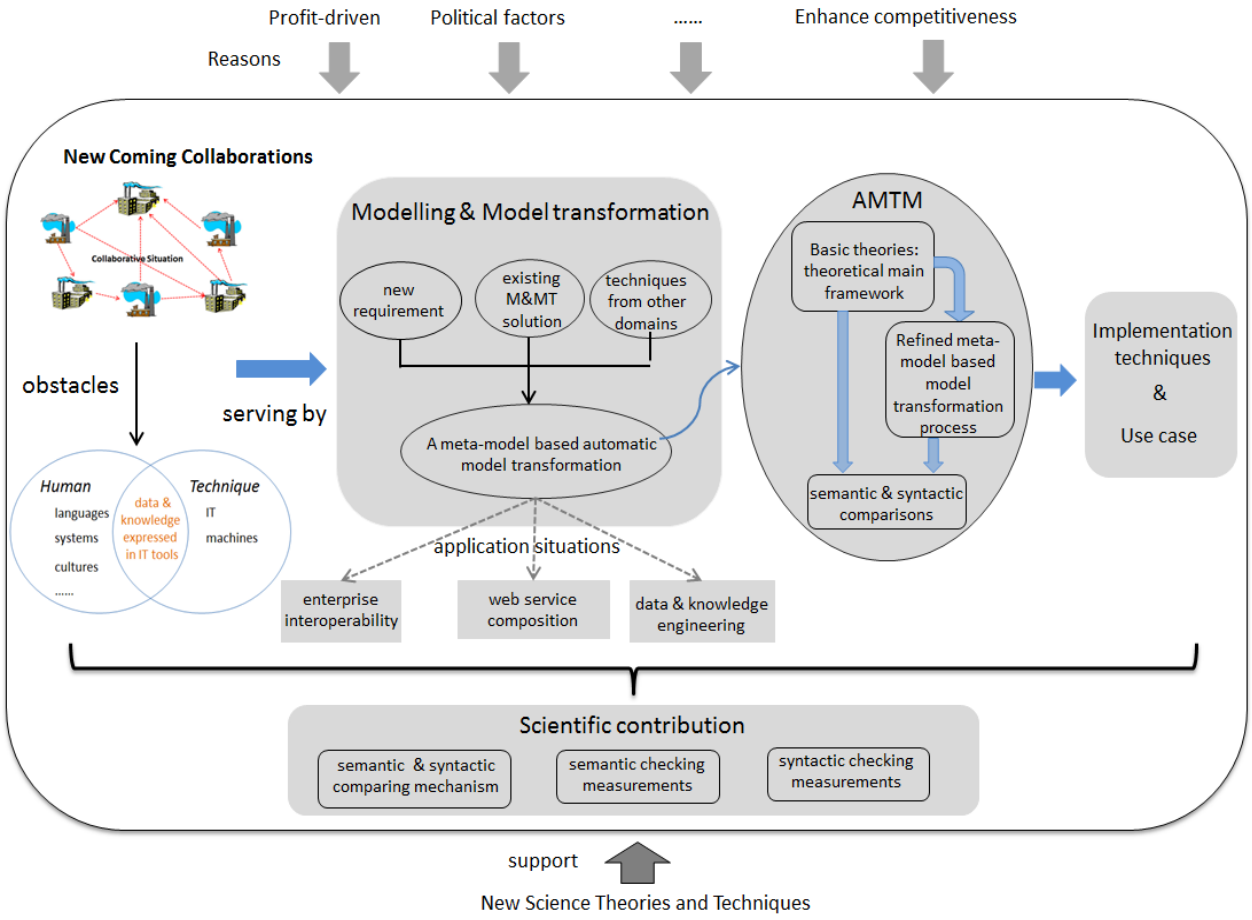


Figure GI-4: Structure and main parts of this thesis.

On the basis of the relevant research work, the detail of AMTM is illustrated. Moreover, to test the feasibility and correctness of the theories defined in AMTM, a software tool is developed and a model transformation use case is executed with this software. The new ideas proposed are presented separately from the detail of AMTM. These new ideas are regarded as the scientific contribution proposed by this thesis. Therefore, this thesis is divided in eight main parts.

- **The first chapter: problem statement**. This chapter focuses on two kinds of problems: (i) problems in new coming collaborative situations, and (ii) problems of using model transformation methodologies to solve the first kind of problems. Three concrete examples of new coming collaborative situations are presented: enterprise interoperability, web service composition and data & knowledge engineering. The main and common problems in these situations are how to connect heterogeneous partners involved in and how to share the information (data) among these partners. By adapting MDE theories: building models to simulate partners and doing model transformation to share information among them, both of the two problems could be solved. As the inner requirement of these collaborative situations "information sharing and exchanging should be done within short period", the model transformation methodology involved should have high efficiency. In order to improve the efficiency of traditional model transformation methodologies, the syntactic and semantic checking methods should be used to replace the manual effort involved before.
- **The second chapter: literature review about AMTM**. The content in this chapter could also be divided into two groups: review the research work that relates to model transformation domain, and the other techniques involved in AMTM. The review work in first group contains: a brief introduction to model-driven engineering and model driven architecture, several existing prominent model transformation techniques and model transformation practices, model transformation methods category, etc. The review work in the second group contains: introduction and real applications of syntactic checking methods, semantic checking methods and ontology, etc.
- **The third chapter: overview of AMTM**. The content in this chapter could also be divided into two dimensions. The first dimension concerns the theories involved in AMTM; a theoretical main framework and the meta-meta-model involved in this framework are presented and explained. The theoretical main framework reveals the method of doing model transformation in AMTM: a refined meta-model based model transformation process. The meta-meta-model defines the mechanism of doing syntactic and semantic checking between meta-models. The second dimension focuses on the execution process of AMTM. AMTM uses the refined meta-model based model transformation process iteratively. In each iteration phase, the model transformation process is divided into four steps: matching on element level, hybrid matching, cross-level matching and auxiliary matching. Each matching step focuses on different potential matching pairs and uses different principles to do the comparisons; but, the basic comparing techniques are always the same: semantic and syntactic checking measurements.
- **The fourth chapter: syntactic and semantic checking mechanism in AMTM**. This chapter provided as the first scientific contribution of this thesis. It presents the idea of using syntactic checking and semantic checking as a whole "S&S" and details the idea of using "S&S" in model transformation process. Since semantic checking is a time-consuming process and syntactic checking could detect several semantic relations, merging the two checking measurements as one brings advantage to AMTM. Furthermore, this chapter also presents the mechanism of choosing

S&S relation pairs; this mechanism could be regarded as one part of the bridge, which crosses the gap between S&S and model transformation domain.

- **The fifth chapter: semantic checking measurements**. This chapter presents the semantic checking methods involved in AMTM. The semantic thesaurus "AMTM_ST" is adopted from "WordNet". The scientific contribution is the creation of "AMTM_ST", which is suitable to serve to model transformation domain and also serves with good efficiency. Moreover, to use "AMTM_ST" in AMTM, a set of semantic relations and semantic values pairs are defined in a table. Thus, the concrete semantic relations become values (range from 0 to 1), which are used as one of the factors to compare different potential matching pairs.
- **The sixth chapter: syntactic checking measurements**. Similar to the chapter of semantic checking measurements, this chapter is also regarded as scientific contribution provided by this thesis. The syntactic checking measurements involved in AMTM are divided as two phases: pretreatment and "Levensthein distance". The pretreatment phase tries to detect several potential semantic relations between a comparing word pairs and the "Levensthein distance" algorithm could calculate the syntactic similarity between any pair of strings. In some terms, syntactic checking could replace the semantic checking, which takes more resource and time to execute. This replacing idea and the idea of using it in AMTM are the main contribution of AMTM.
- **The seventh chapter: software implementation and use case**. As the final functional part of this thesis, this chapter tries to validate the feasibility and correctness of AMTM. A software system "AMTM-SS" is developed to implement the theories in AMTM and simulate the matching processes defined in AMTM. The first part illustrates the main developing process of AMTM-SS; the second part shows a huge test case to verify the whole theoretical basis of AMTM and working performance of AMTM-SS.
- **General conclusion**. As the end of this thesis, this part shows a conclusion of this thesis. Also, this part illustrates the working direction for the future of AMTM. Several points in both AMTM and AMTM-SS are needed to be improved. Finally, a prospect of using AMTM to serve to real social problems is shown.

All these eight main parts compose this thesis. Focusing on a specific social problem "sharing and exchanging data (information) among heterogeneous partners involved in collaboration", an automatic model transformation methodology is proposed. Considering the content involved in this methodology, we present the relevant techniques and principles. To implement the theoretical solution of AMTM, a software tool AMTM-SS is developed. Moreover, a use case is carried out to test the theories defined in AMTM and the performance of AMTM-SS.

Chapter I: Problem Statement

I.1 Introduction

As illustrated in the general introduction, in some aspects, model transformation provides a possible solution to share and exchange data (information) between heterogeneous partners involved in collaborations. However, to meet the requirement (the typical one problem) demanded by new collaborations, a more effective model transformation methodology is required.

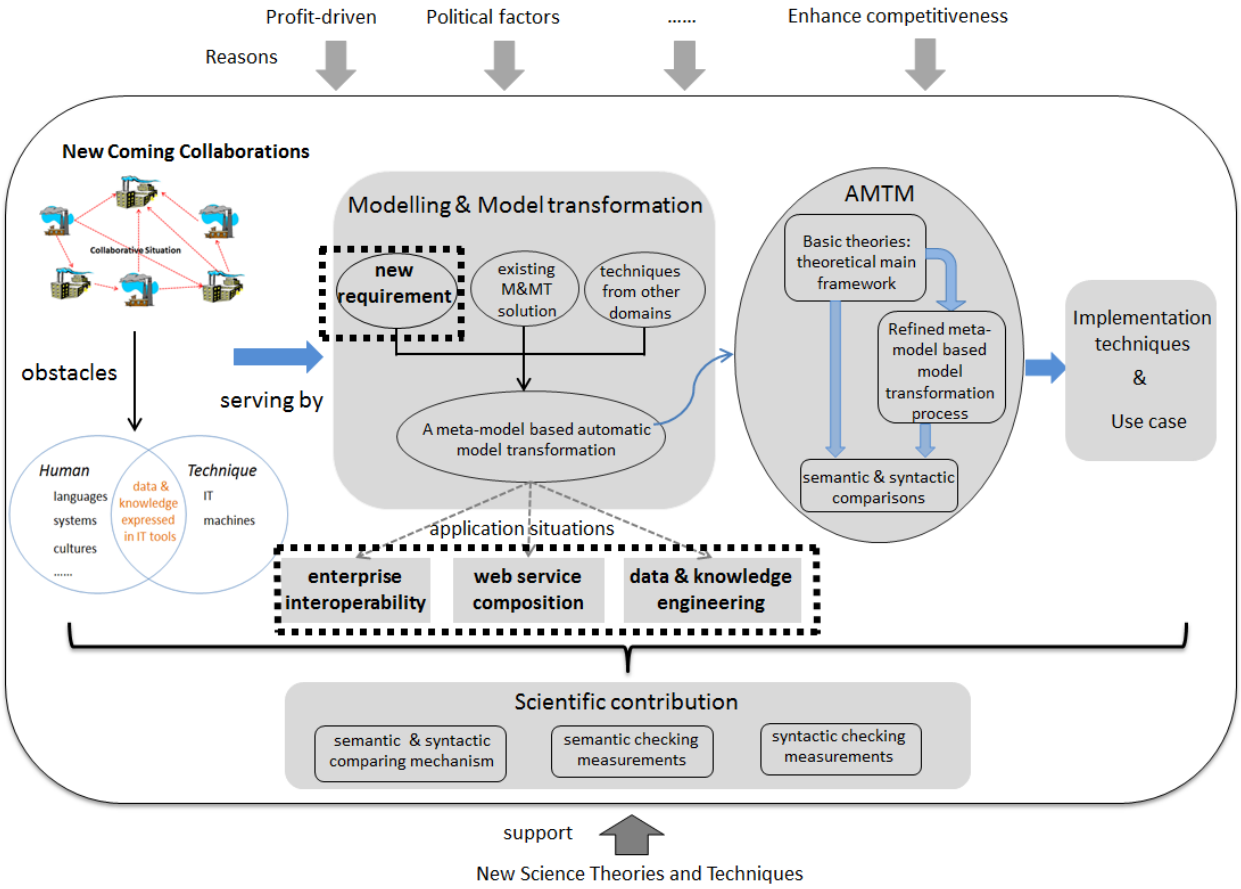


Figure I-1: Position of Chapter one in the thesis.

The position of this chapter in the thesis is shown in Figure I-1 (the square marked by dash lines). This chapter shows three concrete situations and indicates the information sharing problems involved in them. Then, by summarizing the characteristics of the information sharing (exchanging) problems, the new requirement on model transformation domain is proposed.

This chapter contains five main sections: (i) gives the concepts of modeling and model transformation, (ii) describes an example to show concretely the definition and usage of model transformation, and then summaries the characteristics and weaknesses of traditional model transformation practices, (iii) presents three kinds of new coming collaborations and reveals the mechanism of using model transformations to serve information sharing problems involved, (iv) introduces the origin of the project of the thesis: serving to the huge research project "Mediation Information System Engineering (MISE) 2.0", and finally (v) a short conclusion of this chapter is given.

I.2 Modeling & Model transformation

Model transformation is a process of actions performed on models. So, before talking about model transformation, it is necessary to give some knowledge about models and modeling. Thus, both model and modeling play important roles in the context of model transformation research field.

It is important to firstly define the concept of "subject" before talking about model. A subject is something (e.g. a complex system) people want to reason about. It can be something from the real world (an apple tree, a boat, etc.), or something imaginary (e.g. saucer man, devil). The act of reasoning is typically geared around one specific problem or question. The answer to the question typically only concerns a small subset of all the characteristics of the subject (Merrill, 1916). In general, to solve a problem people construct simplifications of the subject that are called models. As defined in (Bézivin, 2006), "model is a simplification of the subject and its purpose is to answer some particular questions aimed towards the subject. As a model captures only a part of the complete subject, many models could be built to represent the same subject but capture different variables of the subject". The other definitions of model could be consulted in (Vernadat, 1999) and (Terrasse et al, 2005). These definitions are similar to each other.

To reason about a model, it is necessary to know which exact concepts it offers. In other words, we need to know the structure of the model. This information is expressed in the "meta-model", where a meta-model is a model that makes statements about what can be expressed in valid models (Bézivin, 2006). Several of the other definitions of meta-model are provided by (OMG, 2008), (Chapron, 2006) and (Bataille et al, 2001). The basic meaning of meta-model is "a specific model that defines the rules to build other models which are conformed to it".

As stated in (Bézivin, 2006), a model shows a simplification of a subject; a meta-model offers the vocabulary for formulating reasoning on top of a given model. Figure I-2 shows the relationships between subject, model and meta-model; this figure is adopted from (OMG, 2008).

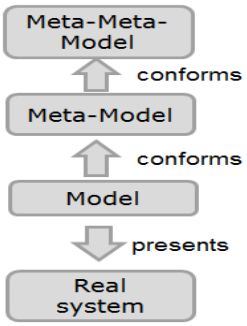


Figure I-2: Relationships between subject, model and meta-model (OMG, 2008).

Meta-model defines the rules of building models; they could exist on several layers: the higher layer, the more abstract the meta-model would be (e.g. a meta-meta-model defines the building rules for meta-models that are conformed to it). Normally, for a specific application domain, the meta-model on the highest layer should be self-defined and self-explained. The activities of building models are called modeling. Figure I-3 shows a simple illustration of the modeling process.

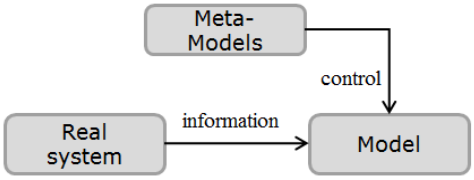


Figure I-3: Illustration of modeling process.

A model is built particularly to present a specific system (subject). Several models could represent one system; one model for each point of the view of the system. Based on previous definitions, several meta-models could be used to describe a system. Therefore, the same information could be in different models. In this case, model transformation is useful.

There are many definitions provided by different research groups for model transformation. Such as: in (Tratt, 2005) "model transformation is a program that mutates one model into another". The Object Management Group (OMG) defines model transformation in the context of the model-driven architecture (MDA) as "the process of converting a model into another model of the same system" (Miller et al, 2003). In (Kleppe et al, 2003), model transformation is defined as the "automatic generation of a target model from a source model, according to a transformation description". Considering all these definitions, **model transformation is a process of generating a target model based on a source model; the transforming rules should be built between same or similar concepts that are from the two models, respectively**.

I.3 Model transformation usage & weaknesses

As stated in (Biehl, 2010), "model transformation is a central concept in model-driven development approaches" and since the theories of model-driven development approaches have been widely adopted by various engineering domains to solve their domain specific problems, a large number of model transformation practices have been developed and used by engineering domains.

I.3.1 The usage of model transformations in software engineering

This subsection takes the usage of model transformation practices in software engineering domain as an example; by analyzing and comparing the model transformation practices involved in, a conclusion about the characteristics of model transformation is presented.

Software engineering, as one of the engineering domains that could be served by model-driven development approaches, adopts modeling and model transformation methodologies in software developing process. Generally, according to (Davis et al, 1988), software developing process contains four main steps: requirement analysis (business level), system design (functional level), coding (implement & technical level) and testing (validation & verification). By adapting model-driven development approaches theories "using models and model transformations to simulate working process", all of the four steps could be implemented. The final results of each developing step are represented by one or a set of models. For connecting different developing steps, model transformations are applied to

generate the new models based on the existing models (e.g. transforming a set of requirement analysis models to system design models).

Figure I-4 presents the basic idea of using modeling and model transformation practices in software developing. Software tools serve to various domains and different domains have different requirements on these software tools. Thus, software tools are developed to serve to different purposes. Normally, the process of developing software tools contains four main steps. The purpose of this example is to show: what are model-driven development approaches and how to use these approaches to serve to real engineering domains.

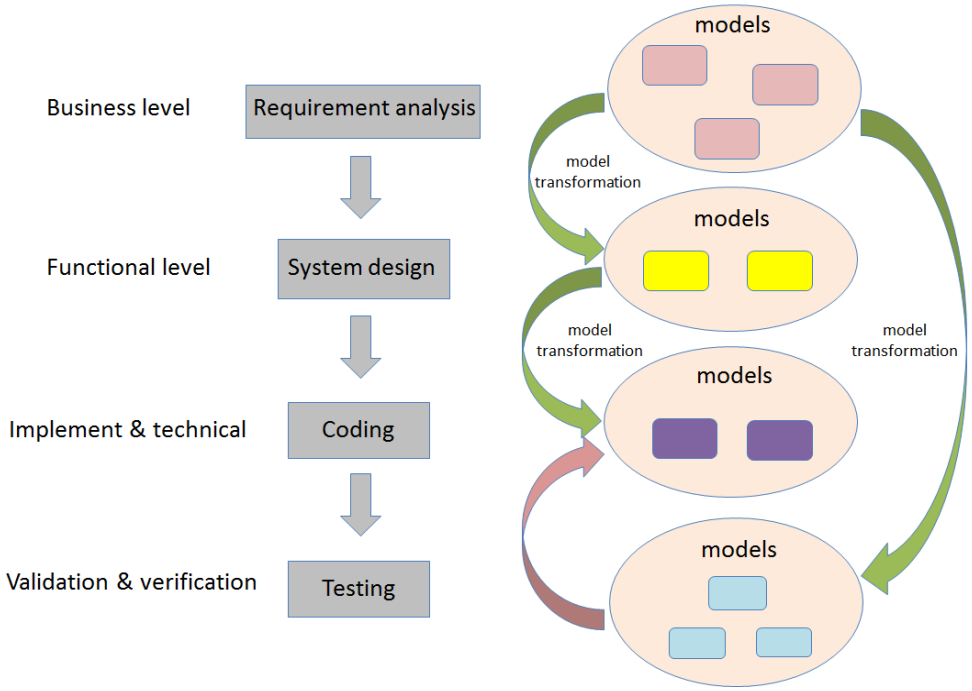


Figure I-4: Usage of modeling and model transformations in software developing process.

For developing complex software tools, it is usual that people (without IT background), who come from specific domains, implement the requirement analysis step; then based on the analysis results, some professional people finish the second design step. Next, software developers do the coding part on the base of system design. Finally, according to both the requirement analysis and coding implementation, the test cases are created (by software test department or final users) to validate and verify the software tool. According to (Davis et al, 1988), since "different working groups (people with different knowledge background and focus on different tasks) might be involved in the developing process", the whole process of developing is difficult to manage. Especially the communication parts (three main gaps) between different working groups are difficult to overcome.

According to (France et al, 2007), model-driven development approaches provide the solution to shield the differences between different working groups and enhance the efficiency of software developing process. Model-driven development approaches suggest that people come from different working groups use models instead of textual documents and formulas to express their ideas, and these models are used as tools to connect different working groups. To improve the efficiency of developing process, model

transformation practices are used to connect these models (models from the same developing step or sequential steps).

I.3.2 Model transformation weaknesses

As explained above and the definitions presented, model transformations could be used to connect different models; it is a process of generating new models (target models) based on existing models (source models). The target models and source models should have some shared (similar) concepts. Model transformation practices aim at detecting these shared (similar) concepts. However, these shared (similar) concepts are difficult to find due to several reasons. Since the difficulty in finding same or similar concepts in models, as stated in (Del Fabro et al, 2009), there are several typical weaknesses in model transformation practices: **low re-usability, involves repetitive tasks, and requires huge users' effort.** These difficulties are originally from the models.

- **The complexity of models**. Models are always built to present systems; some systems could be really huge and complex, thus the models themselves might become complex (even only reflect a point of view of the system). To analysis complex models is a tough job.
- **Contents conveyed by models**. Models represent systems from different domains; these systems could be far away from each other, thus detecting the shared or similar concepts (if such concepts exist between two systems) between these systems is also a tough job.
- **Heterogeneous modeling techniques**. Models are built to represent systems, at the same time they are built based on particular modeling techniques. Such techniques could be Uniform Modeling Language: UML, System Modeling Language: SysML (Friedenthal, 2014), Integrated Definition Modeling: IDEF (Cheng-Leng et al, 1999), Business Process Model and Notation: BPMN, etc. Based on the usages, various modeling techniques have been developed for serving to different domains. These modeling techniques define their own semantic and syntactic representations; so, even the same or similar concepts might be expressed differently with different modeling techniques.

For both the complexity of the systems that are represented by models and heterogeneous modeling techniques, model transformations become difficult to implement.

I.4 New requirement on model transformation from engineering domains

Based on our research work, model transformation practices could be used or adopted by several engineering domains. This section lists three kinds of using mechanism of model transformations in three particular situations. Moreover, the mechanism of using model transformation theories to serve the information sharing problem involved in each of these situations is also illustrated.

I.4.1 Enterprise interoperability

Nowadays, in order to achieve a common goal, enterprises should cooperate with each other; in this way, enterprises could focus on their specific strengths and use the strengths of other enterprises. As stated in (Touzi et al, 2007), more and more collaborative situations are frequently appearing and disappearing within various enterprises. Based on this fact, the ability of cooperating with different partners becomes crucial to modern enterprises. Furthermore, "interoperability" is proposed specially to describe such ability. Enterprise interoperability is now considered as a key factor for successful collaborations (Touzi et al, 2007). It affects the behavior of the enterprise at different levels, ranging from decision-making to operation via information management.

There are several definitions for interoperability. As defined in (Konstantas et al, 2005), "interoperability is the ability of a system or a product to work with other systems or products without special effort from the user"; another definition of interoperability is stated in (Ide, 2010), 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".

Therefore the general idea of interoperability is: "**cooperate without special users' effort thanks to the system semantic and syntactic interoperability**". Although in different domains and from different views of one domain, the definitions of interoperability might be slightly different (e.g. the definitions in (Konstantas et al, 2005) and (Ide, 2010)), the essence reflected by these definitions is similar.

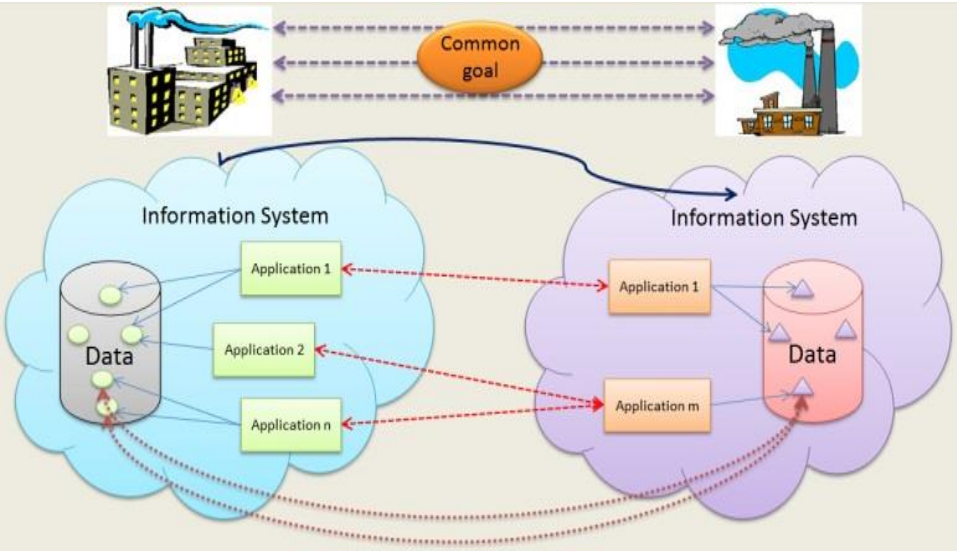


Figure I-5: An illustration of enterprise interoperability issue (Wang et al, 2015a).

Figure I-5 shows a collaborative situation between two enterprises. Enterprises 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 partly on the interactions of the applications contained in the information systems. So, sharing data among these applications is important for enterprise collaboration. However, generally the structures of data are designed for specific applications used by particular enterprises; it is difficult to share data among different applications. This

difficulty lies in the format and completeness of the data. For example, one application needs data inputs: time (hour: minute: second) and date (year: month: day), another application could only provide the data as: date (day: month: year). The two kinds of data are not exactly the same, so the two applications could not be merged directly. However, the two data are similar (some concepts are overlapped); they could be transformed to each other (some special parts should be added or reduced). In the modeling world, these kinds of data could all be regarded as particular models; model transformation provides a possible solution to transform models by detecting similar or same concepts between them. So, model transformations could be used to serve to data sharing problem in merging different information technical applications.

"Enterprise Interoperability Framework (EIF)" (Chen et al, 2007) shows a possible way of combining formally enterprise interoperability and model-driven approaches (especially the model transformation part). Figure I-6 is a representation of EIF.

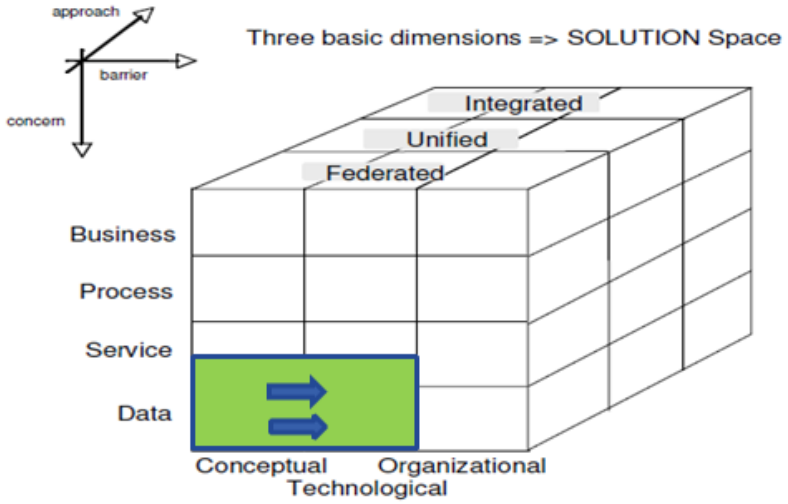


Figure I-6: EIF structure and data sharing interoperability involved.

"EIF" defines four main concerns and three kinds of barriers of interoperability. Furthermore, focusing on these concerns and the barriers, there are three approaches defined within "EIF": "**Integrated**", "**Unified**" and "**Federated**".

Here, we focus on the interoperability on data level. In "**Integrated approach**", all the partners involved use **the same common data format**, so information and data from these partners could be shared very easily but each partner has to use exactly the same data format. This assumption is only available in a long-term collaboration. In "**Unified approach**", all the partners use their own data formats but they have to transform their data to a **common shared format** before sharing them. In "**Federated approach**", there is **no common format**, all the partners use their particular data formats and data are on the fly transformed.

The "integrated" approach is the easiest solution to solve the interoperability problem. However, in the short-term collaborations, it is difficult for partners to meet the requirement of using "integrated" approach (This approach asks large users' effort before turning collaboration to application). The last two approaches imply that each partner keep its own particular data format, thus no deep modification for a

specific collaboration but they require users' effort to solve the interoperability problems, especially on data sharing part.

As explained in (Wang et al, 2014), model transformation theories could be adopted and used in "**Unified approach**" and "**Federated approach**" to serve enterprise interoperability. Since lots of research works about enterprise modeling and simulating have been done, many enterprise modeling techniques and theories have been defined and developed. These modeling techniques and theories concern different kinds of enterprises and take every aspect within an enterprise into account. So, enterprise could be represented by a set of models. In this way, model transformation aims at building connections among models (by detecting same or similar concepts conveyed by models). In "**Unified approach**" and "**Federated approach**", the models built for enterprises are heterogeneous; a high efficient model transformation with few users' effort is required to build connections among them.

I.4.2 Web service composition

Web service composition could be regarded as a situation that many particular partners (atomic web service) would be involved in; this subsection focuses on illustrating this situation and presenting the mechanism of using model transformation methodology to serve it.

Web service composition, as one of the key aspects in web service domain, has attracted more and more research attentions. Generally, in order to provide a capable function to a specific problematic, several web services should be combined and work together. The process of building such a composition of web services is regarded as web service composition. There are two main difficulties in web service composition: selecting web services and making interactions among these web services. Normally, a web service works as a functional black box; it takes in inputs and generates outputs. As stated in (Curbera et al, 2003), (Papazoglou et al, 2003), and (Lee et al, 2011), it is widely accepted that combining multiple web services into a composite service is more beneficial to users than finding a complex and preparatory atomic service that satisfy a special request. The resulting composite services can be used as atomic services themselves in other service compositions.

As explain in (Wang et al, 2015b), despite standards, such as Web Services Description Language: WSDL (Christensen et al, 2001), Simple Object Access Protocol: SOAP (Box et al, 2000), and Universal Description, Discovery and Integration: UDDI (Richards, 2006), are defined with the purpose to standardize web service description, discovery and invocation, **these standards, in their current form, suffer from a lack of semantic representation leaving the promise of automatic integration of applications written to web services standards unfulfilled**. Therefore, lots of works, as stated in (Rao et al, 2005) and (Hwang et al, 2008), define methods of selecting and sequencing web services in order to create composite service. The principle of selecting web services is based on web services description.

A web service is described by three essential attributes. Figure I-7 shows the three identifiers of web service.

- **Signature**: contains service's inputs, outputs and exceptions.
- **State**: is specified by precondition and post condition.
- **Non-functional values**: are used to evaluating the services.

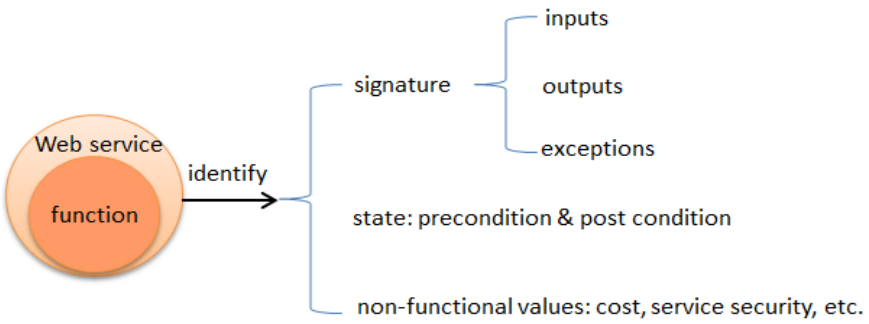


Figure I-7: Web service identifiers (Wang et al, 2015b).

According to (Rao et al, 2005), the principle of selecting web services depends on their "pre-condition" and "post-condition" from the description category "state".

In web service composition, after defining the section and the sequence of web services, the interactions among web services "**especially the data exchanging aspect**" have to be defined. These interactions among web services are complex because of two main reasons:

- **Inputs and outputs of web services** could be complex because they are described by XSD in WSDL. Therefore the inputs and outputs could be regarded as models that are conformed to the building rules of XSD files.
- In order to produce the input of a web service, several web services should be executed before and their outputs have to be combined and transformed to the format of the input.

Since the lack of semantic representations in standards, it is difficult to combine several outputs to one input. XML Metadata Interchange: XMI (OMG, 2000) tries to solve this issue. **Unfortunately, XMI is not enough used today and it is assumed that each web service describes their inputs and outputs with their own semantic that could be different from each other.**

Based on the facts that "the resulting composite services can be used as atomic services themselves in other service compositions" and "web services are created and updated on the fly and a web service composition may involve large number of web services", the inputs and outputs of web services cannot exactly satisfy other inputs or outputs because of:

- **Semantic**: each web service could use its own semantic.
- **Granularity**: an output's element could become an input's property, e.g. a characteristic of an element.
- **Composition**: an input could be defined base on several outputs.

Indeed, the last problem "composition" is due to the three situations of transforming outputs to inputs.

- One web service's outputs could be transformed just as a part of another web service's inputs.
- Only a part of one web service's outputs could be transformed as inputs of another web service.
- A part of one web service's outputs transform to a part of another web service's inputs.

For all three situations, the parts of outputs that could be transformed (e.g. data formats, structure) as inputs should be located and transformed easily. The problems are: **how to specify these specific parts and define the transformation rules based on the description of inputs and outputs**.

Generally, web services' inputs and outputs are defined by "XML Schema Definition: XSD" (W3C, 2000) part in WSDL. The XSD part of WSDL defines the structure and the name of the elements of inputs and outputs. Therefore, it could be seen as a meta-model and the inputs and outputs of a web service could be seen as models that are conformed to this meta-model (XSD).

Based on this assumption, it is possible to apply model transformation principles in order to transform outputs into inputs for web services.

As illustrated by Figure I-8 (dash lines), all combination between inputs and outputs of web services has to be tested. Therefore, manually defining model transformation is not compliant with the automatic composition of web service.

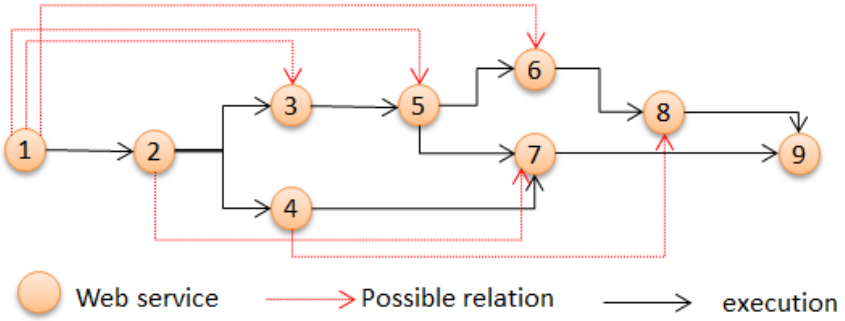


Figure I-8: Web service composition situation (Wang et al, 2015b).

One of the main problems of web service composition is: **how to specify transformable parts of outputs and define the transformation rules in order to generate inputs for other web services**. The composition of web services could be regarded as a process of selecting and sequencing several web services as one. Moreover, each atomic web service could be regarded as a model (the outputs are source models and the inputs are target model) that conforms to the XSD meta-model. So, the interactions among these selected web services models could be implemented by model transformations.

I.4.3 Data & knowledge engineering

As explained in (Wang et al, 2015c), it is possible to use model transformation to serve data and knowledge engineering.

Information sharing, as an aspect of data and knowledge engineering, attracts more and more attention from researchers and practitioners. Since large amount of cross-domain collaborations are appearing, exchanging and sharing information and knowledge among various domains are inevitable. However, due to vast amount and heterogeneous structure of information, it becomes impossible to maintain and share cross-domain information relying mainly on manual effort.

Before talking about information and knowledge, it is necessary to have a quick look on their basis: data. According to (Ackoff, 2010), "Data are symbols that represent properties of objects, events and their environments. They are products of observation"; furthermore, "Information is contained in descriptions, answers to questions that begin with such words as who, what, where, when and how many". The relation between data and information is: "information systems generate, store, retrieve and process data" and

"information is referred from data". Figure I-9 shows the relationships among data, information, knowledge and wisdom.

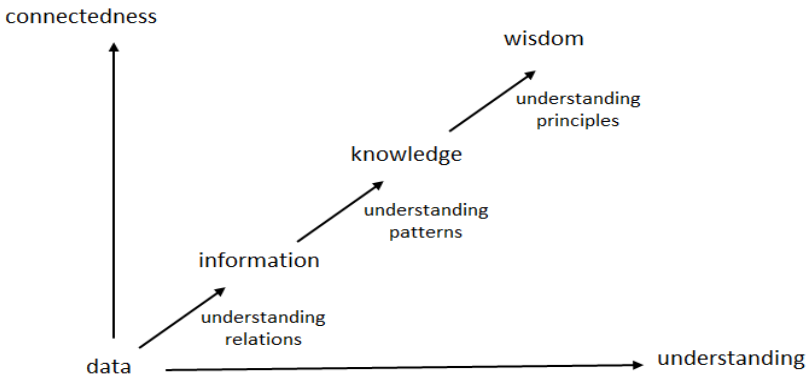


Figure I-9: Different presentation layers from data to wisdom (Ackoff, 2010).

Data is raw; it can exist in any form without significance. Information is generated by adding meaning (relational connection) on data, for example: the data stored in a relational database could be regarded as information. Knowledge, which exists on a higher understanding level than information, is "a deterministic process" (Ackoff, 2010). Knowledge becomes wisdom when add more understanding principles to it; compare to knowledge, wisdom pays more attention to the connectedness issue.

Based on the context of information sharing, Figure I-10 shows the relationships among "natural languages", "data & information" and "specific domains".

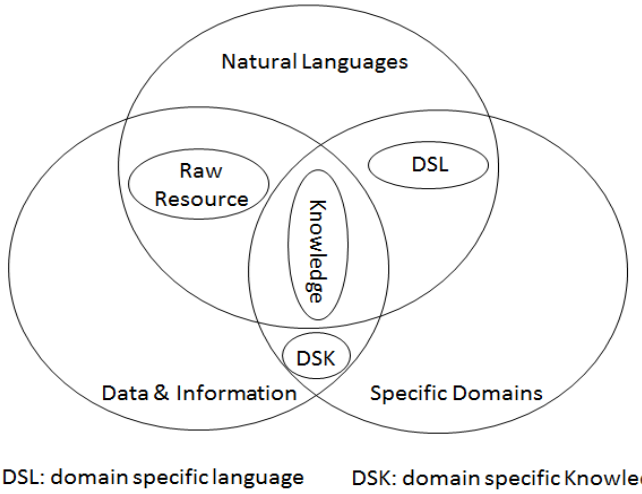


Figure I-10: Relations among natural languages, data, information and specific domains (Wang et al, 2015c).

Data and information are created mainly using natural languages (also with the help of mathematical symbols, diagrams, etc.). Information applied on specific domain, could be regarded as knowledge; while this kind of knowledge may be regarded as information (or data) by other domains. Information that leaves its context is just rough data. In each domain, there are specific expressions for particular items; these expressions are regarded as domain specific languages: DSL (Fowler, 2010). Furthermore, the useful information expressed in DSL is regarded as domain specific knowledge: DSK (Carey et al, 1994).

Due to the development of new technologies, vast amount of heterogeneous data and information appear, and they are provided by different sources. Indeed, amounts and variety of data arose from the development of domain ontologies and the development of Internet of Event (IoE). According to (Van der Aalst, 2014a) and (Van der Aalst, 2014b), IoE could be divided in four sources.

- **Internet of Things**: all physical objects connected to the network. This includes all things that have a unique id and a presence in an Internet-like structure. Things may have an Internet connection or tagged using Radio-Frequency Identification (RFID), Near Field Communication (NFC), etc.
- **Internet of Location** refers to all data that have a spatial dimension. With the uptake of mobile devices (e.g. smartphones) more and more events have geospatial attributes.
- **Internet of People**: all data related to social interaction. This includes e-mails, Facebook, Twitter, forums, LinkedIn, etc.
- **Internet of Content**: all information created by humans to increase knowledge on particular subjects. This includes traditional web pages, articles, encyclopedia like Wikipedia, YouTube, etc.

Data and information are maintained and updated on-the-fly by their sources. In order to use directly the data provided by other sources, some modifications (transformation) should be made. However, this transforming (translating) process always involves huge manual work; with the explosion in the amount of data, it is impossible to maintain such a process relies mainly on manual work.

Model transformation theories provide a possible solution to transform and translate information that comes from heterogeneous partners. Each kind of data or data sets could be regarded as a model that respects some specific building rules. Different data sets might have shared concepts (transformable parts) with each other. So, model transformation could help to detect the potential shared parts in these data sets models.

I.4.4 Summary of the three situations

All these three situations: enterprise interoperability, web service composition, information sharing in data & knowledge engineering are kinds of new coming collaborations. Furthermore, they have common requirement of exchanging and sharing data in an efficient and rapid way. The mechanism of using modeling and model transformation to serve them is also illustrated briefly.

However, the traditional model transformation practices are not suitable to serve these problems because of their weaknesses (Del Fabro et al, 2009**): low reusability, contain repetitive tasks and involve huge manual effort, etc**. These weaknesses limit the usage of model transformation theories to serve to heterogeneity semantic representation (e.g. cross-domain, cross-organization) problems, and also reduce the efficiency of model transformation developing process. Considering about this fact, AMTM is proposed.

I.5 Origin of the thesis

The research work presented in this thesis is a part of an internal research project "Mediation Information System Engineering: MISE", which lasts more than ten years and contains several PhDs' research work. For solving some particular problems involved in "MISE", this research work was proposed in 2012. This section focuses on illustrating the relation between this research work and "MISE"; it is divided into three subsections.

I.5.1 MISE 2.0 introduction

The MISE project "Mediation Information System Engineering" was launched in 2004 and is dedicated to providing an approach (and the associated tools) for Mediation Information System (MIS) design, following the mediation principle as described in (Wiederhold, 1992). The overall objective is to meet both the expectations regarding collaborative situations and the preponderant role of the information system (IS). The approach aims at defining a mediation system able to connect the whole set of partners' ISs in a way that is (i) coherent with the business objectives of the network (effective) and (ii) easy and fast to deploy (efficient). Furthermore, the MIS thus obtained should ensure the interoperability functions (translation of data, sharing of services and orchestration of workflows) in an agile manner.

In reality, collaborations are very unstable situations requiring adaptation: contexts can change (new opportunities, modification of objectives, etc.), networks of partners can change (withdrawal or arrival of partner, lack of resources, etc.) or dysfunction during the collaborative behavior can occur (even if context and partners are still the same, something may not happen as expected). Therefore, the MIS should remain well adopted to the potentially changing needs of the collaboration. The general approach of the MISE project is based on three levels:

- Business level: definition of the appropriate collaborative behavior that fits with the business issues of the network.
- Technical level: design of the Service-oriented architecture (SOA) system and deployment on an enterprise service bus (ESB) so that it can assume the role of mediator between all partners' ISs.
- Agility level: management of evolutions and changes required for the MIS. Figure I-11 shows the three levels.

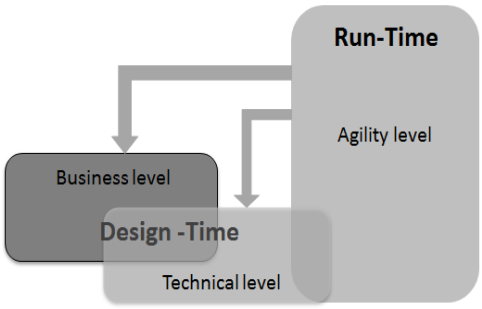


Figure I-11: Three levels defined in MISE (Benaben et al, 2015).

Two iterations of the MISE project have already been performed. MISE 1.0 is presented in (Touzi et al, 2009), (Rajsiri et al, 2010), (Truptil, 2011) and (Benaben et al, 2010). The first iteration ran from 2004 to

2010 (including three doctoral works and the two funded national projects ISyCri and JOnES). MISE 2.0 is the second iteration of MISE project started in 2009 and ended in 2013. MISE 2.0 includes four PhD and three funded projects PLAY (European), SocEDA and ISTA3 (French project). The third iteration MISE 3.0 started in 2011 and is currently on-going. It is also based on several doctoral works (currently five) and on funded projects (currently three: SIM-PeTra, OpenPaaS and DRIVER). The main objective of this research project is to provide any emerging collaborative situation with methods and tools to deploy a Mediation Information System (MIS). MISE 2.0 aims at defining and designing a service-based platform, dedicated to initiating and supporting the interoperability of collaborative situations among potential partners.

I.5.2 MISE 2.0 & Model transformation

The MISE 2.0 platform implements a model driven engineering approach to the design of a service-oriented MIS dedicated to supporting the collaborative situation. This approach is structured in three layers (Benaben et al, 2015), each providing their own key innovative points:

- **Gathering of individual and collaborative knowledge** to provide appropriate collaborative business behavior (key point: knowledge management, including semantics, exploitation and capitalization): *business level*.
- **Deployment of a mediation information system able to computerize the previously deduced collaborative processes** (key point: the automatic generation of collaborative workflows, including connection with existing devices or services): *technical level*.
- **Management of the agility of the obtained collaborative network of organizations** (key point: supervision of collaborative situations and relevant exploitation of the gathered data): *agility level*.

The overall MISE design approach might be seen as a dive into abstraction layers based on model-driven engineering: MDE (Schmidt, 2006). As stated in (Benaben et al, 2015), the general principle of the MISE Enterprise Information Systems approach (whatever the iteration considered) is structured according to four steps: two at the business level and two at the technical level:

- **Design of collaboration model**: this level concerns the gathering of knowledge about the considered collaborative situation to instantiate concepts of the so-called collaborative meta-model (concerning mainly the environment of the collaboration, the objectives of the collaboration, and the partners and services of the collaboration).
- **Deduction of collaborative behavior model**: the second step deals with the automated deduction of collaborative processes, based on the knowledge collected at the previous level. Schematically, the aim is to select and organize partners' services according to the objectives and environment of the collaboration.
- **Design of collaborative workflows**: the previously deduced business behavior (processes) is translated into a technical behavior (workflows) to be implemented. The goal is mainly to match services with activities and data with information.
- **Deployment and orchestration of the MIS**: the previously obtained workflows are integrated in a workflow engine to be executed on an ESB. All available web services of the partners are connected on the same ESB (in case of necessity, specific interfaces are also deployed to connect

other service or even human tasks). The collaborative behavior is consequently performed on this middleware among partners' services.

Furthermore, these four steps are used in an agile framework, which deals with **detection of evolution and adaptation of behavior**. The agility of the MIS is based on **event analysis** (according to the received event, is the situation in line with what is expected?) and on **behavior adaptation** (by re-invoking step 1, step 2, step 3 or step 4, depending on the nature of the event analysis). From a technical point of view, the MISE project is based on a Service-Oriented-Architecture (SOA) paradigm and MISE tools are also deployed as web services on the same ESB as the partners' web services. Even if there are some differences and specific features, each of the three iterations of the MISE project is structured according to the four steps presented above, and the associated agile framework. Furthermore, from a technical point of view, these iterations are all centered on SOA principles and on web services.

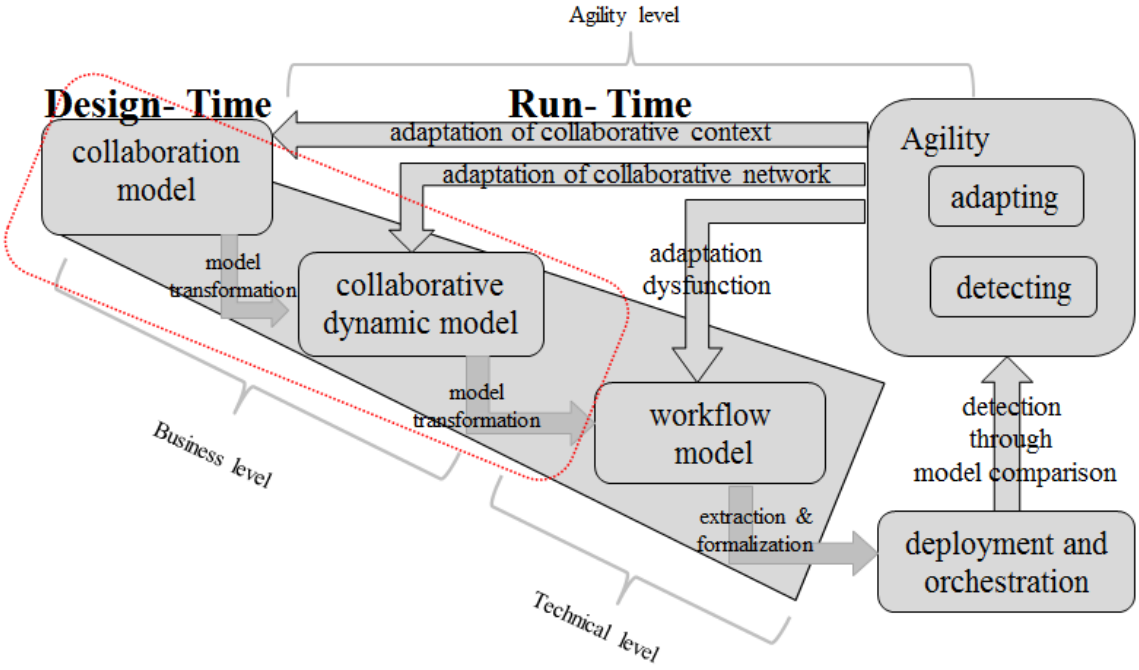


Figure I-12: Four steps in the lifecycle of MISE (Benaben et al, 2015).

Figure I-12 is a simplification about the four steps. Model transformation is a part of the whole project; it fulfills the gap between the first step to the second step and from the second step to the third step. Also, within the first and second steps, large number of models will be built to represent the static and dynamic collaborative situations. There are some potential relations among the models that are built on the same layer, model transformation practices could also be used to detect the relationships among these models.

I.5.3 Specific requirement on model transformation of MISE

In some aspects, MISE is a mode-driven engineering related platform. Modeling and model transformation theories and techniques are used in part of the life cycle of MISE.

MISE divides four main layers to deal with collaborative information; each of the layers could be regarded as a specific domain. The gap between adjacent layers is difficult to fulfill. Furthermore, within

each layer, information and knowledge expressed in models are heterogeneous. As the inner requirements of MISE, model transformation practices involved should solve cross-domain problems with high efficiency. In other words, model transformation practices should work as bridge to cross the gaps between different layers defined in MISE; also within each layer, model transformation practices should build connections among heterogeneous models. Considering the purpose of MISE, all the model transformation practices (detect same or similar concepts and thus sharing data) used in MISE should be created on the basis of a high efficient model transformation methodology.

I.6 Conclusion

This chapter illustrates the problem statements of the research work presented in this thesis. First, the usage of model transformation practices is explained with a concrete example of applying "model-driven development approach" in software development process. Then, three situations of new collaborations in different engineering domains have been presented. The specific problems involved and the solving mechanism by applying model transformation to these problems are illustrated. Finally, the original of this research work is presented. The initial aim and purpose of developing this project is clear.

The focusing problems of this thesis could be divided on two levels: **business level** and **technical level**.

- **Business level:** sharing knowledge between different domains, and exchange and share data (information) that are owned by heterogeneous partners. Since the appearing of new coming collaborations, knowledge sharing and data (information) exchanging should be done efficiently: fast and accurate. However, this kind of tasks is used to be done mainly relying on manual work, which is inefficient and error-prone. To improve the efficient issue, new theories and techniques are required. This thesis proposes a possible solution "AMTM" to solve the problems on business level; the proposal of "AMTM" brings the problems on technical level.
- **Technical level**: AMTM is a methodology of doing automatically model transformation; the main problem of it is the detection of same or similar concepts conveyed by different models. As illustrated above, models are heterogeneous (i.e. heterogeneous modeling techniques, heterogeneous content presented). The concepts conveyed in models are always expressed with different syntactic and semantic representations.

As a short conclusion, the main difficulty of defining a high efficient model transformation methodology to serve collaborations comes from three aspects: semantic checking, syntactic checking and granularity issue involved. Table I-1 shows the three aspects.

Table I-1: Three difficulties of defining high efficient model transformation methodology.

Origin \ Difficulty	semantic detecting	syntactic detecting	granularity issue
application domains	√	√	
modeling techniques	√	√	√
model transformation domain			√

Different engineering domains use their own semantic and syntactic representations to define concepts; different modeling techniques also define different semantic and syntactic representations (for their

elements) to build models. So, models, which are built to simulate and analyze collaborations, might have different semantic and syntactic representations. Furthermore, both modeling and model transformation domain involve granularity issue "to differentiate the range of concepts, different elements are defined" (this part will be illustrated in the third chapter with examples). So, using modeling and model transformation theories to serve new coming collaborations also needs to overcome the problems brought by them.

Focusing on the problems presented above, an automatic model transformation methodology (AMTM) created on the basis of semantic and syntactic checking measurements, is proposed in this thesis.

Chapter II: Literature review: concurrent and component issues to this research work

II.1 Introduction

In order to tackle the problem of data sharing and exchanging among heterogeneous partners involved in collaborations, an effective automatic model transformation methodology is proposed in this thesis. Here, "effective" means high performance (i.e. involving less manual work, fast implementation, with high accuracy). To be "effective", one of the key issues is to define the model transformation process automatically.

This thesis presents AMTM. For developing AMTM, a large amount of research works have been studied. The research works can be divided into two categories: concurrent issues "related works to model transformation domain" and component issues "theoretical and technical issues, which belong to other research domains, adopted in AMTM".

The concurrent issues focus mainly on the origin and context of model transformation; the component issues concern about the technical and theoretical issues that are developed in other research domains and have been involved in AMTM. Both the two kinds of issues are regarded as literature to this thesis and being reviewed and presented in this chapter. The position of this chapter in the thesis is shown in Figure II-1; the part that is marked with square dash lines).

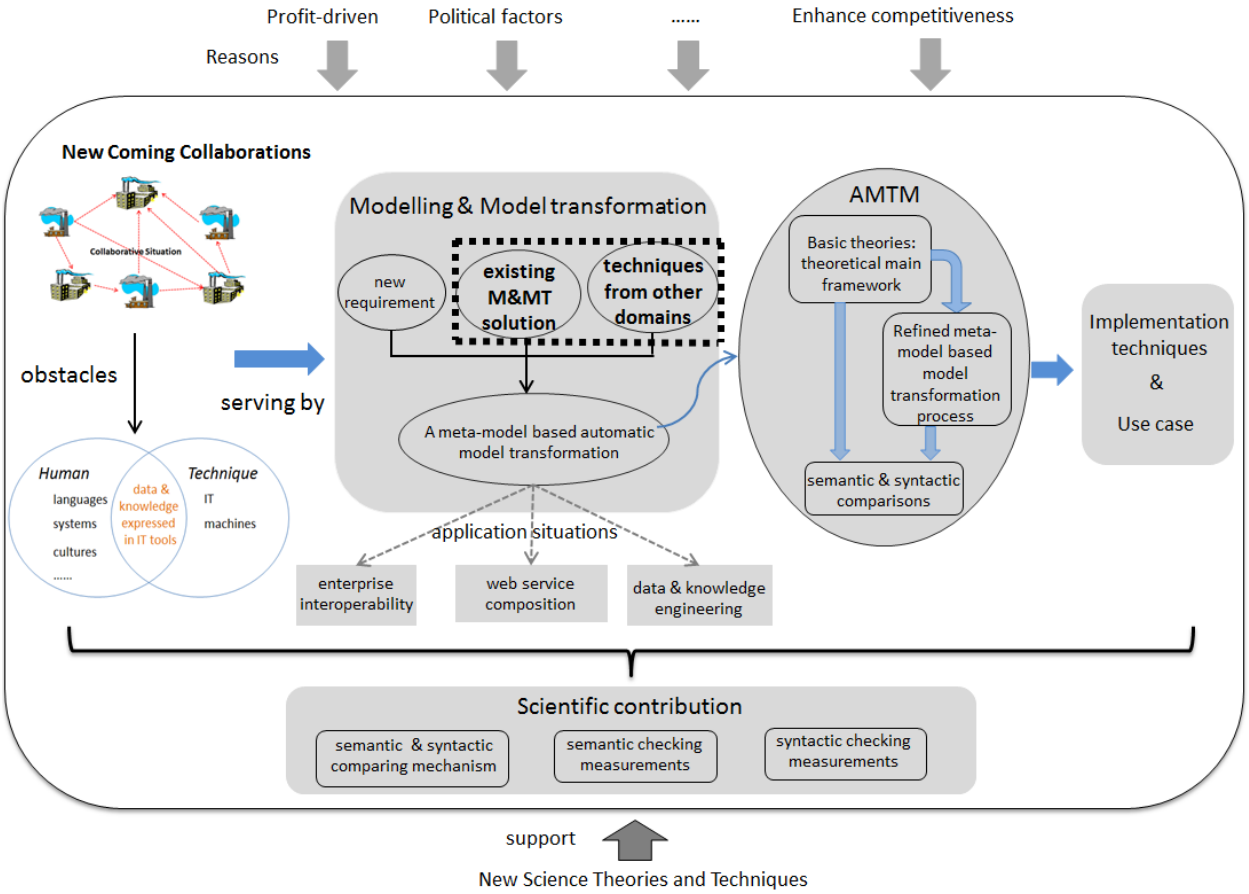


Figure II-1: Position of chapter two in the thesis.

As shown in Figure II-1, this chapter works as the foundation to the other chapters in the thesis. Since large amount of research work is presented in this chapter, it is divided into eight sections to illustrate different research aspects. The relation between each section (research aspect) involved in this chapter is shown in Figure II-2.

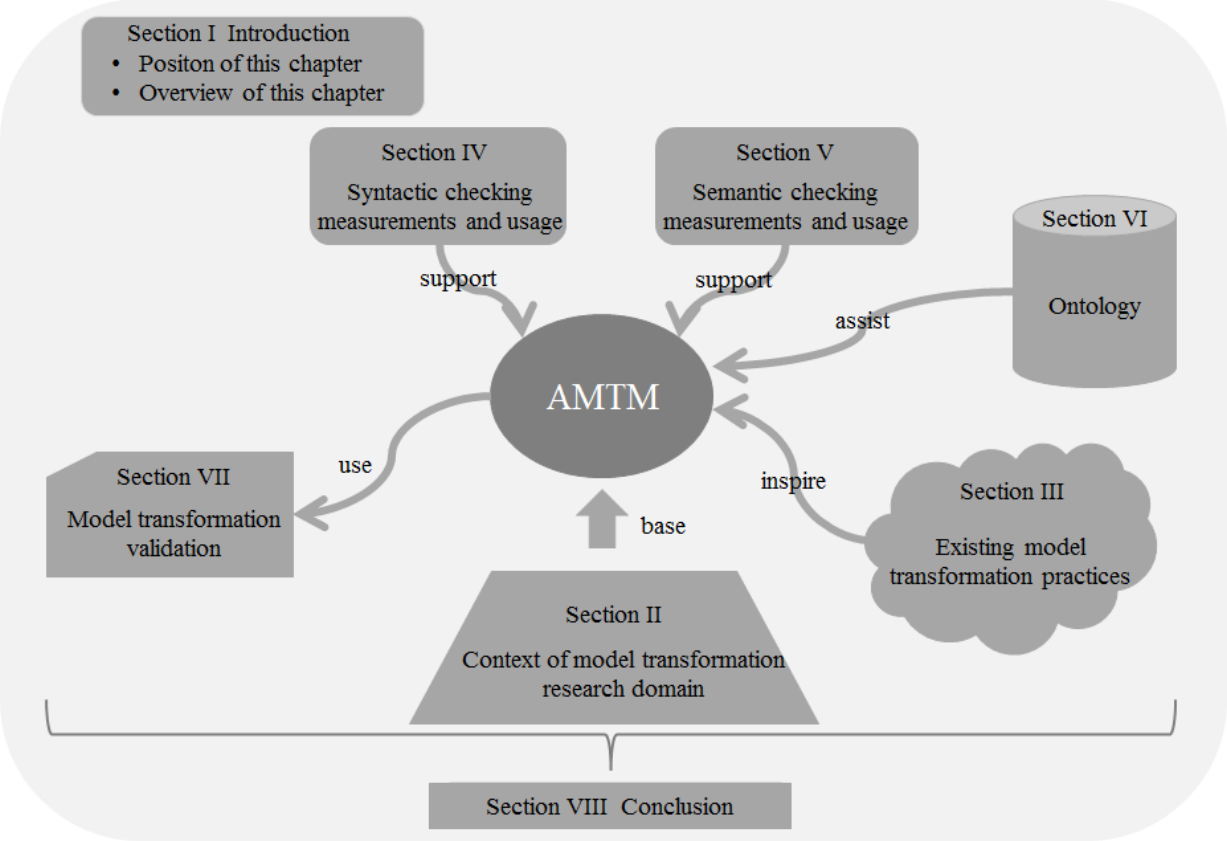


Figure II-2: Map of sections of chapter two with regards to the global objective.

All sections present content that centers on the core "AMTM".

- The second section illustrates the context of model transformation domain; it contains four subsections.
- The third section presents four model transformation instances; all of the four instances inspire the development process of AMTM in some aspects.
- The fourth section focuses on the syntactic checking measurements and its usage; a comparison and a conclusion of several syntactic checking measurements are given at the end of this section.
- Similarly to the fourth section, the fifth section focuses on semantic checking measurements.
- The sixth section is a simple illustration about ontology; and the usage of ontology in engineering domains is also presented with simple examples.
- The seventh section talks about the model transformation validation issue. At last, a conclusion of this chapter is given.

II.2 Context of model transformation research domain

This section is the state-of-the-art of model transformation domain. It aims at presenting the origin and context of model transformation. Three subsections contained in this section. (i) Model-driven architecture and model transformation, (ii) model transformation techniques presentation and comparisons, and (iii) model transformation practices category.

II.2.1 Model-driven architecture and model transformation

Model-driven architecture (MDA) is a software design approach for the development of software systems. It provides a set of guidelines for the structuring of specifications, which are expressed as models. Model-driven architecture is a kind of domain engineering, and supports model-driven engineering of software systems. It was launched by the Object Management Group (OMG) in 2001. According to figure II-3, the relation between MDA and MDD is "MDA can be regarded as a subset of MDD", for the reason MDA is the OMG's particular vision of MDD and thus relies on the use of OMG standards (Brambilla et al, 2012).

The MDA model is related to multiple standards, including the Unified Modeling Language (UML), the Meta-Object Facility (MOF), XML Metadata Interchange (XMI), Enterprise Distributed Object Computing (EDOC), the Software Process Engineering Meta-model (SPEM), and the Common Warehouse Meta-model (CWM). The term "architecture" in Model-driven architecture refers to the architecture of the various standards and model forms that serve as the technology basis for MDA. As stated in (OMG, 2014), "MDA provides an approach for deriving value from models and architecture in support of the full life cycle of physical, organizational and IT systems. The MDA approach represents and supports everything from requirements to business modeling to technology implementations. By using MDA models, we are able to better deal with the complexity of large systems and the interaction and collaboration between organizations, people, hardware, software." Figure II-3 shows the general overview of MDA.

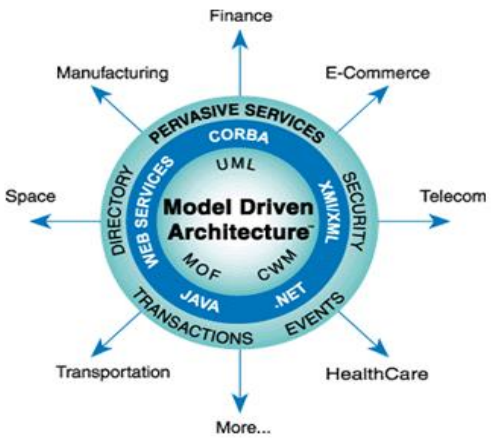


Figure II-3: General understanding of MDA (OMG, 2014).

As can be seen from Figure II-3, MDA can be used in many specific fields (Finance, Manufacturing, E-Commerce, Healthy, etc.). MDA has three levels; for each level, the basic implementing techniques and tools are shown clearly and the standards of each level are also illustrated in the figure.

MDA has a 4-layer meta-modeling architecture (level M0, M1, M2 and M3): "M0: real things (the "subject" that is mentioned in the first chapter); M1: UML models; M2: UML meta-model; M3: Meta-Object Facility (Meta-meta-model level model)"; this 4-layer meta-modeling architecture extends the idea presented in Figure I-2 by assigning specific modeling techniques for each layer. At the same time, MDA defines three main viewpoints: "computation independent viewpoint", "platform independent viewpoint" and "platform specific viewpoint". The three main viewpoints form the main framework (structure) of MDA, and for each viewpoint several models are defined.

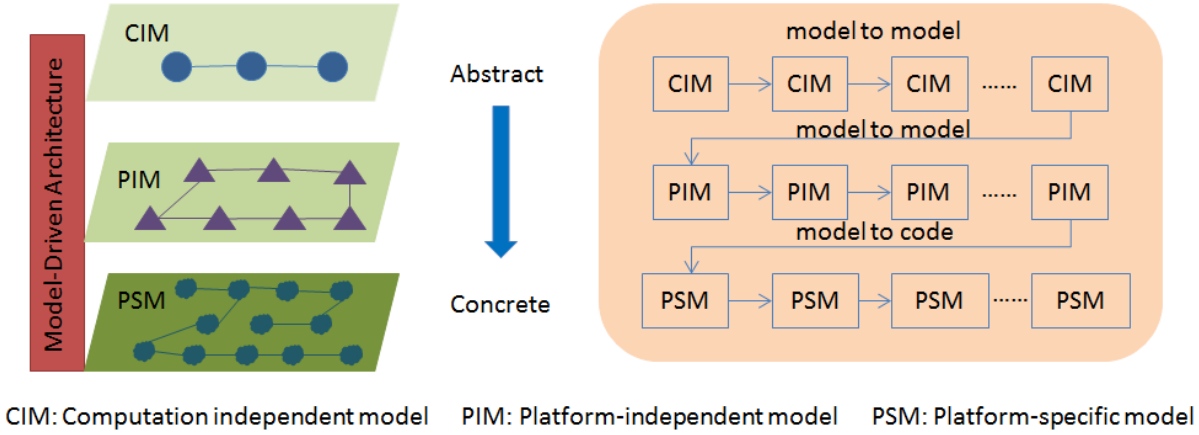


Figure II-4: MDA three viewpoints (Wang, 2015c).

MDA approach pays attention to the "computation independent viewpoint", this viewpoint is more concerned with the domain experts, the models belonging to this viewpoint are "computation independent models: CIMs" that are free from programming issues; MDA approach defines system functionality using the platform-independent models: PIMs, which belong to the platform independent viewpoint, that are created with domain-specific language: DSL (Fowler, 2010). Then, a platform model corresponding to CORBA, .NET, the Web, etc. is given, and the PIM is translated into one or more platform-specific models: PSMs, which belong to platform specific viewpoint and could be executed. This requires mappings and transformations and should be modeled too. The PSM may use different DSLs, or a General Purpose Language: GPL as Java, C#, PHP, Python, etc. Model transformation is also used as a tool in MDA; it connects models belonging to the same viewpoint or cross-viewpoint. The final purpose is to use model transformations to generate all the other models (i.e. PIMs and PSMs) based on the CIMs.

The OMG organization provides rough specifications rather than implementations, often as answers to Requests for Proposals (RFPs). Implementations always come from private companies or open source groups. So, as a short and brief conclusion, *model driven architecture provides just a specification which can separate the concerns (related to the development process of software system based on MDE)*. It does not provide the implementation based on its own theory. MDA principles can be also applied to other domains, such as business process modeling (BPM) where the CIMs are related to business process simulation.

As a conclusion of this part, both of MDE and MDA focus on the theories and standards of using models and model transformations to solve practical problems in real engineering domains.

II.2.2 Model transformation techniques

In practice, a large number of techniques for model transformation have been developed. Some of these techniques are mature and have been used by various research organizations (or companies) to serve to their domain specific problems; while some of the techniques are still at the stage of developing, only a few of researchers are involved in and use them to solve particularly problems. At the same time, some of the techniques are supported very well. Many relevant tools and execution engines have been published out to support them; many integrated development environments (IDE) provide the developing plug-ins for specific model transformation techniques.

II 2.2.1 QVT

Query/View/Transformation: QVT (OMG, 2008) is a standard set of languages for model transformation defined by "OMG"; it covers transformations, views and queries together. The QVT standard defines three model transformation languages and integrates the "Object Constraint Language: OCL 2.0" (Cabot et al, 2012) standard and also extends it with imperative features. The operational context of QVT is shown in Figure II-5.

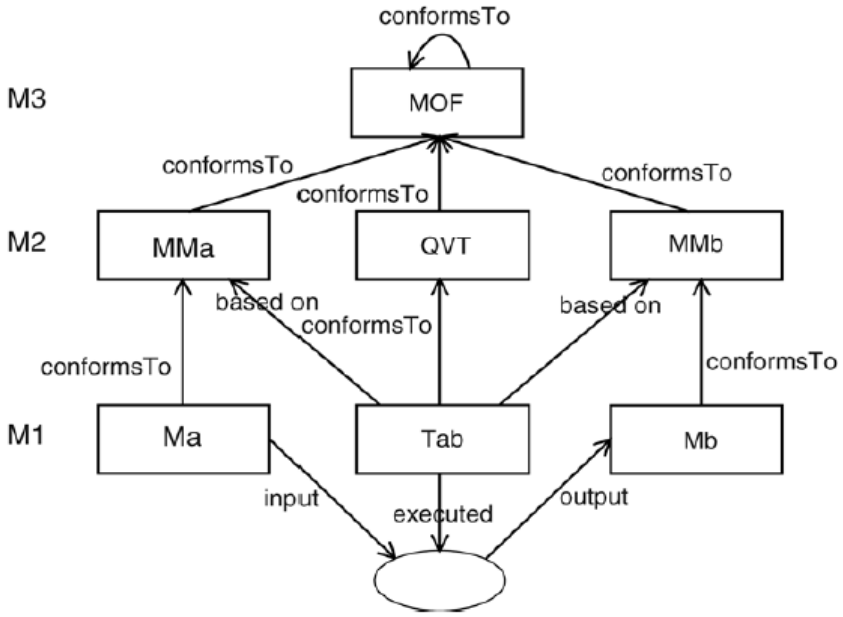


Figure II-5: Operational context of QVT (OMG, 2008).

In the operational context of QVT, models are categorized in four layers: subject, model (M1), meta-model (M2), meta-meta-model (M3); the idea of this category is illustrated in the first chapter. QVT provides its own meta-model that defines its abstract syntax. The abstract syntax of QVT is defined as a meta-meta-model using Meta-Object Facility (MOF) 2.0. This meta-model defines three sublanguages for transforming models. The three QVT languages collectively form a hybrid transformation language with

declarative and imperative constructs. The languages are named "Relations", "Core" and "Operational Mappings". These three languages are organized in a layered architecture shown in Figure II-6.

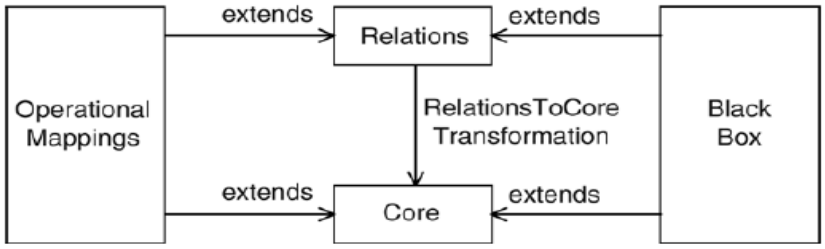


Figure II-6: QVT languages layered architecture (OMG, 2008).

The languages "Relations" and "Core" are declarative languages at two different levels of abstraction. The specification document defines their concrete textual syntax and abstract syntax. In addition, "Relations" language has a graphical syntax. "Operational Mappings" is an imperative language that extends "Relations" and "Core languages".

The Black Box mechanism allows the plugging-in and execution of external code during the transformation execution. This mechanism allows complex algorithms to be implemented in any programming language and enables reuse of already existing libraries.

II 2.2.2 ATL

ATLAS transformation language: ATL (Jouault et al, 2008) is a model transformation language and toolkit. It provides ways to produce a set of target models from a set of source models. ATL operates in the same context as QVT: the four layers of models' category. ATL architecture is composed of three layers, as shown in Figure II-7. They are: (in decreasing abstraction level order) the ATLAS Model Weaving (AMW) (Del Fabro et al, 2005), ATL, and the ATL Virtual Machine (ATL VM).

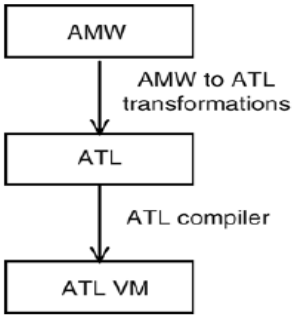


Figure II-7: ATL layered architecture (Jouault et al, 2008).

ATL provides both declarative and imperative constructs, and is therefore a hybrid model transformation language. ATL VM compiles ATL programs using a model-oriented instruction set. AMW may optionally be used as a higher abstraction level transformation specification language.

The declarative part of ATL is based on the notion of matched rule. ATL offers two imperative constructs: "called rule" and "action block". A "called rule" is explicitly called, like a procedure, but its body may be composed of a declarative target pattern. "Action blocks" are sequences of imperative instructions that

can be used in either matched or called rules. Transformation programs written in ATL are inherently unidirectional.

The current ATL execution engine is based on virtual machine architecture. The VM is implemented on top of two model handlers: the "Eclipse Modeling Framework: EMF" (Steinberg et al, 2008) and NetBeans "Meta-Data Repository: MDR" (Matula, 2003). Figure II-8 shows the positon of VM in ATL. The ATL compiler works on top of the ATL VM and generates ATL programs capable of running on top of it too.

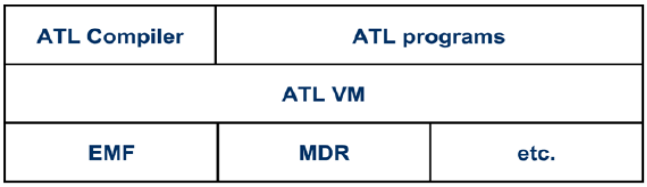


Figure II-8: ATL VM architecture (Jouault et al, 2008).

ATL VM provides a basic set of constructs, which are sufficient to perform automatic operations on models. While ATL provides a higher level language for transformation definition, AMW provides solutions to express transformations in even more abstract terms. Model weaving is about establishing typed links between model elements. Links themselves form a model, and link types are therefore defined in a meta-model. Weaving links are more abstract than ATL rules.

II 2.2.3 VIATRA2

Visual Automated Model Transformations: VIATRA2 (Varró et al, 2007) is a unidirectional transformation language based mainly on graph transformation techniques. The language operates on models expressed following the VPM meta-modeling approach (Varró et al, 2003). VIATRA2 also integrates three sublanguages: Graph pattern language, Graph transformation rules language and "Abstract State Machine: ASM" (Börger et al, 2012) language.

- **Graph pattern language**, expresses patterns that are matched to select elements in the current graph. Patterns may reuse other patterns and may form recursions.
- **Graph transformation rules language**, expresses graph rewriting rules. Every rule has a left and a right hand side (form a pattern). Rules are unidirectional. Following the classical scheme of graph transformations, a rule may delete matched elements, create new elements and preserve existing elements.
- **ASM language, expresses order of execution of the transformation rules**. The order is specified by using ASM constructs. It provides a set of control flow structures: sequencing, rule call, conditionals, fixed-point iteration, etc. In some terms, VIATRA2 could be regarded as a hybrid language. ASM language also allows code generation by using code templates.

VIATRA2 focuses only on model-to-model transformations; it is possible to define generic template rules in which the classes are parameters that may be substituted via template instantiation. The instantiation is achieved by a meta-transformation in the terms of VIATRA2 (Varró et al, 2004). The instantiation executes a higher-order transformation (HOT) that manipulates the generic transformation. The execution environment provides both an interpreter and a compiler. For the purposes of testing and simulation, the transformation programs may be interpreted. In addition, transformations may be compiled to a given

target environment (Balogh et al, 2006). It is possible to invoke native functions implemented in Java. This is similar to the Black Box mechanism in QVT.

II 2.2.4 GReAT

Graph Rewriting and Transformation Language: GReAT (Karsai et al, 2003) is a visual language developed using Generic Modeling Environment (GME). It enables the specification of unidirectional translations between sets of models. These models conform to meta-models specified in UML. Each meta-model is called a domain, and traceability information conforms to a user-specified cross-domain meta-model. This meta-model may also be used to extend source or target domains with transformation-specific elements. GReAT contains three sublanguages:

- **Pattern specification language**. A pattern is a graph that is recognized by pattern matching over the host graph. The GReAT pattern matcher does not work on the whole host graph but starts from a set of already matched elements, which is more efficient. The corresponding algorithm only works on connected graphs. The pattern specification language supports cardinalities and the grouping of sub-patterns.
- **Graph transformation language**. Transformation rules are specified in the graph transformation language, which is an extension of the pattern specification language. Each element of a pattern is associated to a role depending on the existence of the element before and after the application of the rule. The bind role corresponds to elements existing before and after. The delete role is used for elements that exist before, but not after (i.e., they are deleted). The new role specifies elements that exist only after (i.e., they are created). Guards can be specified as Boolean C++ expressions. Attribute mappings written in C++ are used to initialize the attributes of new elements.
- **Control flow language**. Rule application order is specified using a dedicated imperative language, which provides iterating and conditional constructs. Moreover, partial matches (called packets) are passed from rule to rule in order to set the initial bindings. This implies a specific sequencing of execution. Depending on how the rules and the control flow are specified, a given transformation may be non-deterministic (i.e., have different targets for the same source).

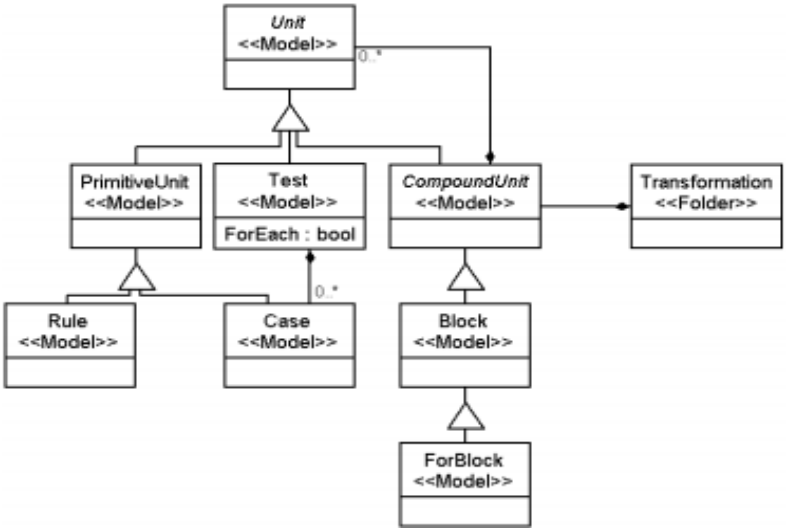


Figure II-9: The object hierarchy of GReAT (Karsai et al, 2003).

Figure II-9 shows the conceptual hierarchy of GReAT. The "CompoundUnit" is used for Hierarchy and recursion. "Test" and "Case" are conditional branching. Sequencing and non-deterministic execution are achieved with the help of sequencing connections (not shown in figure). These connections (implicitly) specify the order of execution for the rules.

II 2.2.5 Summary of the four model transformation techniques

Four model transformation techniques are presented briefly in the former four subsections, respectively. ATL and QVT are two general model transformation techniques; they aim at serving all the model transformation situations. Otherwise, VIATRA2 and GReAT are two model transformation techniques focusing on graph transformation (rewriting).

Many model transformation techniques have been developed. A comparison among the four techniques illustrated above is shown in Table II-1. All of the four techniques are mature and complex. They provide powerful functions to achieve model transformation processes. However, some predefined conditions should be obeyed in order to use them.

Table II-1: Model transformation techniques comparisons

name	hybrid	rule scheduling	M-to-N	note
ATL	yes	implicit internal explicit	yes	self-executed
QVT	yes	implicit internal explicit	yes	based on MOF 2.0
VIATRA2	yes	external explicit	yes	based on VPM
GReAT	yes	external explicit	yes	on UML models

According to the developing purpose, model transformation techniques could be divided into two groups: serve general purpose (cross-domain) and serve specific domain.

Normally, domain specific model transformation techniques focus on particular problematic. Therefore, the use of these techniques is limited, and they are not flexible for some special cases; on the other hand, this kind of techniques could be executed automatic or semi-automatic in some aspects. Model transformation techniques, which serve cross-domain, are always complex and provide a wide range of

functions. So, people need more time to learn to use these techniques properly. One of the common problems of existing model transformation techniques is: **involved huge manual effort and repetitive tasks**. Without solving such kind of problem, the usage of model transformation would be always limited.

II.2.3 Model transformation category

Since model transformations could be used to solve real engineering problems, more and more attentions have been paid to this domain. For developing effectively model transformation process, it is necessary to differentiate model transformation situations. So, different model transformation theories and techniques could be created based on different application situations.

II.2.3.1 Model transformation situations

Generally, model transformation could be divided into three groups. The summary of these three groups is shown in Table II-2. The content presented in models is descripted in abstract syntax, while the content presented in text is descripted in concrete syntax.

Table II-2: Category of model transformation situations

Category	Content
Text to Model	concrete syntax to abstract syntax
Model to Model	abstract syntax to abstract syntax
Model to Text	abstract syntax to concrete syntax

All of the three categories have their own use instances: "Text to Model" – java code to UML class diagram (model), "Model to Model" – UML class diagram (model) to UML activity diagram (model) and "Model to Text" – UML class diagram (model) to formal specifications, one of this research works about this situation is presented in (Bruel et al , 1998). Different theories and techniques are developed focusing on serving to them, respectively. In this research work, we focus on the model to model transformation methodology.

II.2.3.2 Classification of model transformation methods

According to (Czarnecki et al, 2003), 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, there are two categories:

- **Visitor-based approaches**: provides some visitor mechanism to traverse the internal representation of a model and write code to a text stream. An example of this approach is "Jamda". Jamda is an object-oriented framework providing a set of classes to represent UML models, an API for manipulating models, and a visitor mechanism to generate code.
- **Template-based approaches**: templates lend themselves to iterative development as they can be easily derived from examples. Several instances: e.g., b+m Generator Framework, JET and FUUT–je. An introduction to template-based code generation is stated in (Cleaveland et al, 2001).

When bridging large abstraction gaps between platform independent models: PIMs and platform special models: PSMs, it is easier to generate intermediate models rather than go straight to the target PSM. Also,

to connect different domains model-to-model transformations are needed. For model-to-model approaches, there are five categories:

- **Direct-Manipulation Approaches**: offering an internal model representation plus some API to manipulate this model. They are usually implemented as an object-oriented framework.
- **Relational Approaches**: grouping declarative approaches where the main concept is mathematical relations. The basic idea is to state the source and target element type of a relation and specify it using constraints. Two examples are: (Akehurst et al, 2002) and (OMG, 2008), declarative approaches in (Gerber et al, 2002), and mapping rules in (OMG, 2008).
- **Graph-Transformation-Based Approaches**: operate on typed, attributed, and labeled graphs (Andries et al, 1999), which is a kind of graphs specifically designed to represent UML-like models. Some examples of these approaches are: VIATRA, ATOM and GReAT.
- **Structure-Driven Approaches**: two distinct phases contained in this category. First, creating the hierarchical structure of the target model; second, setting the attributes and references in the target. The overall framework determines the scheduling and application strategy; users are only concerned with providing the transformation rules. One examples of this approach is "OptimalJ" model transformation.
- **Hybrid Approaches**: combining different techniques from the previous categories. Such approaches are QVT and ATL.

The detail of these approaches (their applicable situations, working mechanism, etc.) and instances for each approach could be consulted in (Czarnecki et al, 2003). This reference also presents a comparison among these approaches. Other two model-to-model transformation approaches are: **Common Warehouse Meta-model** (CWM) Specification (Poole et al, 2002) and transformation implemented using **XSLT** (Clark, 1999).

Focusing at other viewpoints of model transformation, the classification of model transformation methods could be different; model transformation methods are also classified as four kinds: marking and pattern, automatic transformation, meta-model based transformation and model merging.

- **Marking and pattern**. Mark: a concept of the target model applied to a source model element to indicate how to transform it. Then, the marked elements of the source are transformed according to the pattern to produce the target. A simple explanation of this method is shown in Figure II-10 (Based on the context of MDA).

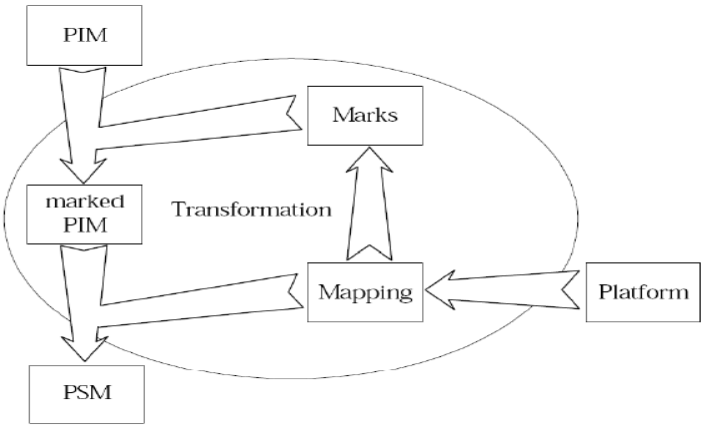


Figure II-10: Marking and pattern model transformation method.

A model from PIM level is regarded as the source model, and mappings are made according to the special platform. One mapping connects a concept in source model to an element in target model. All the mapping consist the pattern. The source model changes to "marked source model" after adding the marks. According to the pattern, target model could be generated.

- **Automatic transformation**. This method means: no additional information is needed to produce the target model from source model. Actually, in practice semi-automatic transformation is more used than the totally automatic transformation method, for the reason the automatic transformation results are usually not explicit or acceptable at this moment.
- **Meta-model based transformation**. Models are built based on their meta-models then mappings could be made on the meta-model level; next, generate the transformation rules according to the mappings, and finally execute the transformation rules on model level to generate the target model.
- **Model merging**. This method takes several models as the source models; then, based on the pre-defined rules to merge the source models and generate the target model. Figure II-11 illustrates the simple principle of this method (using the example PIM-PSM model transformation).

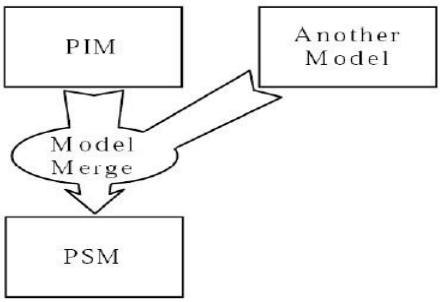


Figure II-11: Models merge.

As shown in Figure II-11, the source model is a PIM, and the target model is a PSM. The transformation process is "merging another model with the source model to generate the target model".

II.2.3.3 Meta-model based model transformation

The automatic model transformation methodology "AMTM" presented in this thesis is designed and implemented as a meta-model based model-to-model transformation methodology. So, a detail illustration about meta-model based model-to-model transformation methodology is given here. Generally, this kind

of transformations could be divided into two categories. Figure II-12 shows the principles of the two categories.

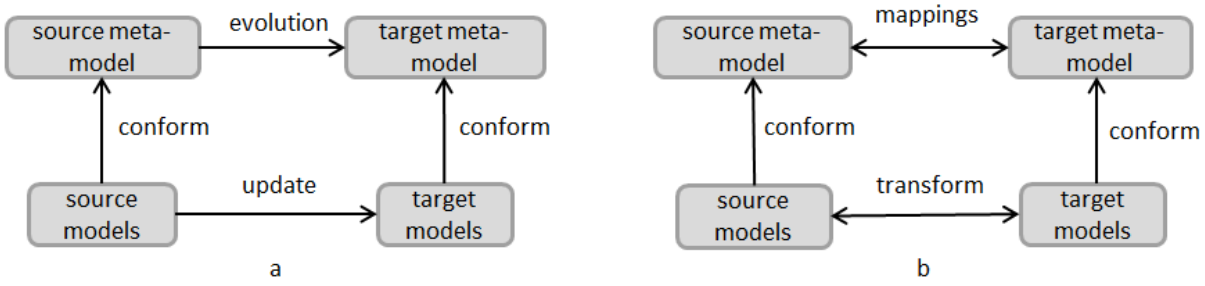


Figure II-12: Meta-model based model transformation category (Wang, 2015c).

In category (a), target meta-model is generated by evolving (add new characteristics to) the source meta-model; source models conformed to the source meta-model should be updated to target models that are conformed to the target meta-model. Large amount of research work focusing on this category has been done; moreover, different theories and practices working on this category have been defined and applied. For instance, a methodology named "COPE", which defined in (Herrmannsdoerfer et al, 2009), is a mature framework dealing with category (a). In category (b), the source meta-model and the target meta-model are two different models; there is no relationship (e.g. evolutional relation) between them. In order to transform source models to target models, model transformation mappings should be built on the meta-model level and use on the model level.

II.3 Model transformation practices

This section focuses on the model transformation practices that are developed. According to the methodologies of these model transformation practices, this section could be divided into two parts: general model transformation practices and model transformation practices applying semantic detecting, which are similar to this research work.

II.3.1 General practices of model transformation

A large number of model transformation practices have been developed along with the emergence of model transformation techniques. These practices could be classified in different categories, and similar to model transformation techniques, model transformation practices could be also divided into two categories: serve to specific-purpose practices and serve to general-purpose practices.

Concerning only about the category of model-to-model transformations, Table II-3 presents three particular model transformation practices and simple comparisons among the main characteristics of them. The research work of the three model transformation methodologies are listed as references; the details of them could be consulted in (De Castro et al, 2011), (Grangel et al, 2010) and (Bollati et al, 2013).

Table II-3: Comparisons among three model-to-model model transformation practices

name	technique	domain specific	note
Applying CIM-to-PIM model transformations for the service-oriented development of information systems (De Castro et al, 2011)	MDA-based	yes	Combining MDA with service-oriented development of information system
Transformation of decisional models into UML: application to GRAI grids (Grangel et al, 2010)	ATL	yes	GRAI Grids to UML model
Applying MDE to the (semi-) automatic development of model transformations (Bollati et al, 2013)	MeTAGeM (Bollati et al, 2011)	no	applying MDE principles to define model transformation

Different model transformation theories and techniques are used to create model transformation practices that serve various purposes. These various purposes could be enterprise integration, software development, etc. More and more model transformation practices focus on particular problems and provide solution to a set of special problems since the huge differences between model transformation situations. However, to serve new collaborative situations, high efficient general-purpose model transformation methodology and practices are required.

II.3.2 Model transformation practices applying semantic detecting

As stated in the general introduction, semantic difference is an issue to be solved in collaboration. This subsection, we focus on the usage of semantic detecting in model transformation domain.

The research work presented in (Kappel et al, 2006) is similar to this research work presented in this thesis in some terms. The research work (Kappel et al, 2006) concerns the usage of model transformation in software development domain. It proposes a process lifts semi-automatically meta-models into ontologies by making implicit concepts in the meta-model explicit in the ontology. In this way, a shift of focus from the implementation of a certain modeling language towards the explicit reification of the concepts covered by this language is made. In (Kappel et al, 2006) three steps are defined to lift meta-models to ontologies.

- **Changing of formalism** "a meta-model is transformed into ontology", the transformation is given by a mapping between the model engineering space and the ontology engineering space, namely a mapping from a meta-meta-model to ontology meta-model.
- **Unfolding typically hidden concepts** in meta-models that should better be represented as explicit concepts in ontology.
- **Enriching ontologies** being extracted from modeling languages' meta-models with axioms and putting in relation with other ontologies representing a shared vocabulary about a certain domain.

The basic idea and principle of the research work stated in (Kappel et al, 2006) is intuitive; however, it contains three main weaknesses.

- The first step of "mapping between the model engineering space and ontology engineering" **depends on manual work**; the mappings rules for this step should be **made previously concerning about the specific modeling techniques**.

- The second step "unfolding typically hidden concepts" also **depends mainly on manual effort**.
- The third step "enriching ontologies being extracted" needs ontologies from certain domains; in this research work, **no concrete ontologies are used and no huge semantic thesaurus is created to support the theoretical solution**.

Other similar research works presented in (Kappel et al, 2007), (Bruel et al, 2000) and (Dolques, 2011) also adopt semantic detecting in model transformation process. There are several common weaknesses in these methodologies: **serve specific domains**, **leave the granularity diversity involved in model transformation process unsolved**, **no specific semantic thesaurus created to support automatically semantic relations detecting**, etc. Besides, in database community, semantic checking measurements are also used to automatically define rules of schema matching. The research work focusing on schema matching could be referred in (Rahm et Bernstein, 2001) and (Bernstein et al, 2011).

II.4 Syntactic checking and its usage

Syntactic checking measurements are used to calculate the syntactic similarity between two words; they could be used for matching names and records.

A comparison of string metrics based syntactic checking measurements is illustrated in (Cohen et al, 2003). Syntactic checking focuses on the occurrences of the letters involved in words; it should also consider about the stemming, prefixes and suffixes of the words.

String metrics are always created while comparing syntactic similarity between two words. There is several string metrics based syntactic checking methodologies, such as edit-distance metrics, fast heuristic string comparators, token-based distance metrics, and hybrid methods.

The real usage of syntactic checking in engineering domains could be "matching entity names". As stated in (Cohen et al, 2003), the task of matching entity names has been explored by a number of communities, including: 1) statistics, 2) databases, and 3) artificial intelligence. The basic goal is to classify entity pairs as matching or nonmatching.

- **In statistics**, several proposals adopted by subsequent researchers, often with elaborations of the underlying statistical model (e.g., (Jaro et al, 1989), (Jaro, 1995), and (Winkler, 1999)). These methods have been used to match individuals and/or families between samples and censuses.
- **In the database community**, some work on record matching has been based on knowledge-intensive approaches (e.g., (Hernández et al, 1995) and (Galhardas et al, 1999)). The use of string-edit distances as a general-purpose record matching scheme was proposed in (Monge et al, 1996) and (Monge et al, 1997).
- **In the AI community**, supervised learning has been used for learning the parameters of string-edit distance metrics (e.g., (Ristad et al, 1998) and (Bilenko et al, 2002)) and combining the results of different distance functions (e.g., (Tejada et al, 2001) and (Cohen et al, 2002)).

As stated in (Pasula et al, 2002), probabilistic object identification methods have been adopted to matching tasks. In these communities there has been more emphasis on developing autonomous matching

techniques which require little or no configuration for a new task, and less emphasis on developing "toolkits" of methods that can be applied to new tasks by experts as explained in (Cohen et al, 2003).

II.5 Semantic checking and its usage

Semantic is the study of meaning. It focuses on the relation between signifiers, like words, phrases, signs, and symbols, and what they stand for; their denotation. Linguistic semantics is the study of meaning that is used for understanding human expression through language. Other forms of semantics include the semantics of programming languages, formal logics, and semiotics.

Relevant studies about semantic are: semantic mapper, semantic network, semantic mapping and semantic analysis, etc. For each of these studies, it may provide usages based on different application domains. Taking "semantic analysis" as an example, it could serve machine learning, knowledge representation, and linguistics, etc. For different usages, semantic analysis might be defined in different ways.

This thesis focuses on **semantic checking measurements**; (Kim et al, 2012) presents an example of using these measurements to serve data communication protocol. Protocol translation is a method for transforming pieces of information from a source protocol into relevant target protocol formats in order to communicate between heterogeneous legacy systems in interoperability environments. The work in (Kim et al, 2012) proposes a method that translates messages to semantically equivalent messages. Other semantic checking applications that are dealing with different domains are listed in the references as (Feldmann et al, 2014), (Gábor et al, 2013), and (Nikolova et al, 2012).

Semantic checking is complex; furthermore, the use of it will be really different based on different application domains. So, this section just presents a general introduction to it and gives several research examples about it (without implementation details). Since semantic checking measurements play a key role in AMTM, the basic rules and relevant content of it will be detailed in chapter five.

II.6 Ontology

Ontology is the philosophical study of the nature of being, becoming, existence or reality as well as the basic categories of being and their relations. It is a "philosophical" term; people have used it for many years. As time goes on, people give this word some new meaning according to their own research fields. There are many definitions of ontology; some of them have been defined in recent years. One of the definitions of ontology is stated in (Crowston, 1994): ontology defines the basic terms and relations comprising the vocabulary of a topic area as well as the rules for combining terms, and relations to define extensions to the vocabulary. As stated in (Neches et al, 1991), the most quoted definition of ontology by the ontology community is: "a formal explicit specification of a shared conceptualization for a domain of interest."

Comparing to database (DB), which is also used to manage huge amount of data and maintain the potential relations exist among the data, ontology has its own focus and characteristics. DB schema focuses mainly on data themselves, while ontology focuses on the meanings of data and sharing understanding. The core purpose of DB is "structuring instances for efficient storage and querying", and the purpose of ontology is human communication, interoperability, etc. In DB, the meanings of data are missing and the instances are important. Conversely, instances are optional to ontology. Furthermore, DB schema pays minimum focus on formal semantics, while ontology pays strong focus on formal semantics. Since AMTM adopts semantic checking and focuses on semantic meanings, ontology is a more suitable option than database for AMTM.

Nowadays, ontology has been widely used in many domains, such as: knowledge engineering; computer science; the semantic web, etc. Ontology offers a semantic way to electronic information management and exchange (Neches et al, 1991). It is a way to organize the knowledge and change the view of it to a standard form.

At this moment, there exists a set of AI-based ontology implementation languages. Two of the famous ones are: Resource Description Framework Schema: RDFS (Brickley et al, 2000) and Web Ontology Language: OWL (McGuinness et al, 2004).

A list of several existing ontology is shown in Table II-4. Different ontologies have been developed for different purposes as required during the recent years. The ontology covers different domains such as medicine, tourism, business process, etc.

Concerning the context of usage, different ontologies have been created in different domains. Normally, ontologies belonging to different domain contain different concepts (it is possible to have some overlapping concepts but with different semantic meanings); the ontologies belonging to the same domain may contain similar concepts (overlap parts) but the structures of each ontology could be totally distinguished from one to one.

Table II-4: Examples of existing ontologies

Ontology	Working domain
AIAI Enterprise Ontology (Uschold et al, 1998)	Enterprise modeling
Toronto Virtual Enterprise Ontology (Fox et al, 1998)	Enterprise engineering
The business Process Management Ontology (Jenz, 2003)	Business process & IT
Process Specification Language Ontology (Gruninger, 2000)	Manufacturing applications
MIT Process Handbook Ontology (Malone et al, 2003)	cross-domain

II.7 Model transformation validation

Model transformation validation could be used to **test if the generated model transformation mappings and rules are good or not**; so it is an important aspect of model transformation domain. AMTM also involves model transformation validation aspect as one of its component; this section focuses on this aspect. One of the research works about doing verification and validation of model to model transformations is presented in (Cabot et al, 2010).

Normally, model transformation validation contains two dimensions: **model validation** and **transformation traceability**. Model validation is put forward after the concept of "model driven engineering" becoming more and more familiar to people. Many theories and methods have been proposed concerning about this research field. In the context of model transformation, model validation is an important dimension. Before executing the transformation rules between two models, **the source model must be validated to prove it is reasonable**; after generating the target model, it is also necessary to prove that **the target model can be accepted**.

Normally, **domain knowledge takes the role of reference in model validation**, and it varies from measured data of real system to qualitative experience of experts. Knowledge-based system is implemented based on domain knowledge and validation techniques proposed. In practice, **experience of domain expert** is often used for model validation when there is not enough measured data about real system. The idea of knowledge-based validation roots from this fact. Besides measured data and experiential knowledge, the classical theorem and formula about the dynamic of real system can also be utilized for the validation of simulation models. All these information, termed as domain knowledge takes the role of reference for validity judgment. In other words, domain knowledge describes the valid characterization of simulation output, i.e. what kind of dynamic features it should take. Three model validation methodologies are listed in the reference (Min et al, 2010), (Romero et al, 2011) and (Brinkman et al, 2013). **In AMTM, involving manual effort to validate the target model is one of the solutions to test the automatically generated model transformation mappings and rules.**

Another dimension of model transformation validation concerns about model transformation traceability. During the model transformation process, traceability of models modification is very important.

In (Radatz et al, 1990), traceability is defined as: "*the degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor–successor or master–subordinate relationship to one another*; for example, the degree to which the requirements and design of a given software component match. Traceability implies keeping track of the relationships between requirements, design artifacts, source code, test cases, etc. As stated in (Santiago et al, 2012), "*in some terms, traceability information becomes obsolete very quickly and sometimes it is completely omitted. However, the advent of Model-Driven Engineering (MDE), which principles are to enhance the role of models and modeling activities and to increase the level of automation all along the development process, can drastically change this landscape*."

The key role of models in any MDE development process can decisively help to facilitate trace maintenance. Therefore, trace maintenance can be seen mainly as links between the elements of those models. Furthermore, the traces could be collected in other models and, therefore, processed using any model processing technique, such as model transformation, model matching or model merging. Moreover, if the models considered in the development process are connected by a model transformation and the language used to develop the transformation provides support to keep the trace information, such information can be generated automatically (Tratt, 2005). Thus, if an element from a source model is modified, this modification could be propagated to the corresponding elements in the target model. This scenario is represented by a very simplistic example in Figure II-13: two given models (Ma and Mb) are connected by a model transformation (MMa2MMb).

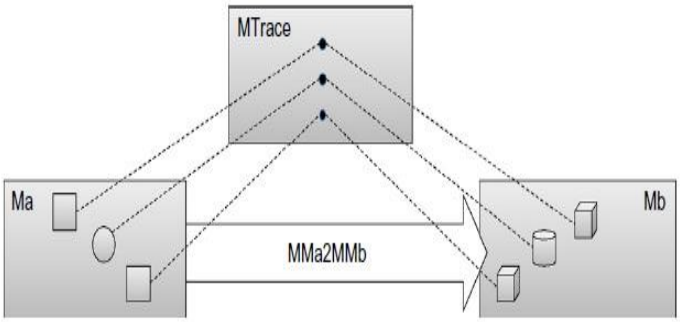


Figure II-13: Example of the trace links collected in a MDE scenario (Santiago et al, 2012).

The transformation maps squares and circles from the source model (Ma) into cubes and cylinders in the target model (Mb). To keep track of these relationships after the transformation has been executed, it would be desirable to have at one's disposal an ''extra'' model of trace objects (MTrace). MTrace records the mappings pairs and it could be used to reverse the model transformation process. Two of the other model transformation traceability methodologies are listed in the reference as (Van Amstel et al, 2012) and (Yu et al, 2012).

II.8 Conclusion

This chapter presents the literature review about the research work presented in the thesis. The content in this chapter could be divided into two main groups. First group content is about the relevant domain knowledge of model transformation; this part concerns the second, third and seventh sections. The second group content is about the research works from other domains that are adopted by AMTM; this part concerns the fourth, fifth and sixth sections.

As illustrated at the end of the first chapter, there are three main difficulties: semantic checking, syntactic checking and granularity issue involved, in defining a high efficient model transformation methodology that aims at serving collaborations. The research works presented in this chapter could solve parts or several of the three difficulties. The relations between the research works and the difficulties are shown in Table II-5.

Table II-5: Difficulties and solutions of defining automatic model transformation methodology

Research works \ Difficulty	semantic detecting	syntactic detecting	granularity issue
MDA			√
Model transformation category			√
Model transformation techniques	√		√
Model transformation instances	√		√
Semantic checking	√		
Syntactic checking	√	√	
Ontology	√		
Model transformation validation	√	√	√

As shown in Table II-5, six of the research works are associated partly with the **semantic detecting issue**; the **syntactic detecting issue** could be served by *syntactic checking* and *model transformation validation*; **the granularity issue** could be partly solved by: *MDA* (by spreading the focuses in model transformation process), *model transformation category* (by differentiating model transformation situations), *model transformation techniques and instances* (several of the above mentioned items concern the granularity issue) and *model transformation validation* (by testing the transformation results).

Chapter III: Automatic model transformation methodology (AMTM) overview

III.1 Introduction

This chapter presents an overview of the automatic model transformation methodology AMTM. As explained in the former chapters, AMTM is a component to the research work "MISE". **It focuses on the problematic of sharing and exchanging data** (information & knowledge) **among heterogeneous partners involved in the new coming collaborations**. To define such an automatic methodology, the main obstacles come from three aspects.

- Detecting semantic relations conveyed by input model sets.
- Detecting syntactic relations conveyed by input model sets.
- Solving the granularity issue involved in model transformation process.

This chapter presents the solutions, which are provided in AMTM, to all the three main kinds of obstacles. The position of this chapter to the thesis is shown in Figure III-1 (the square marked by dash lines).

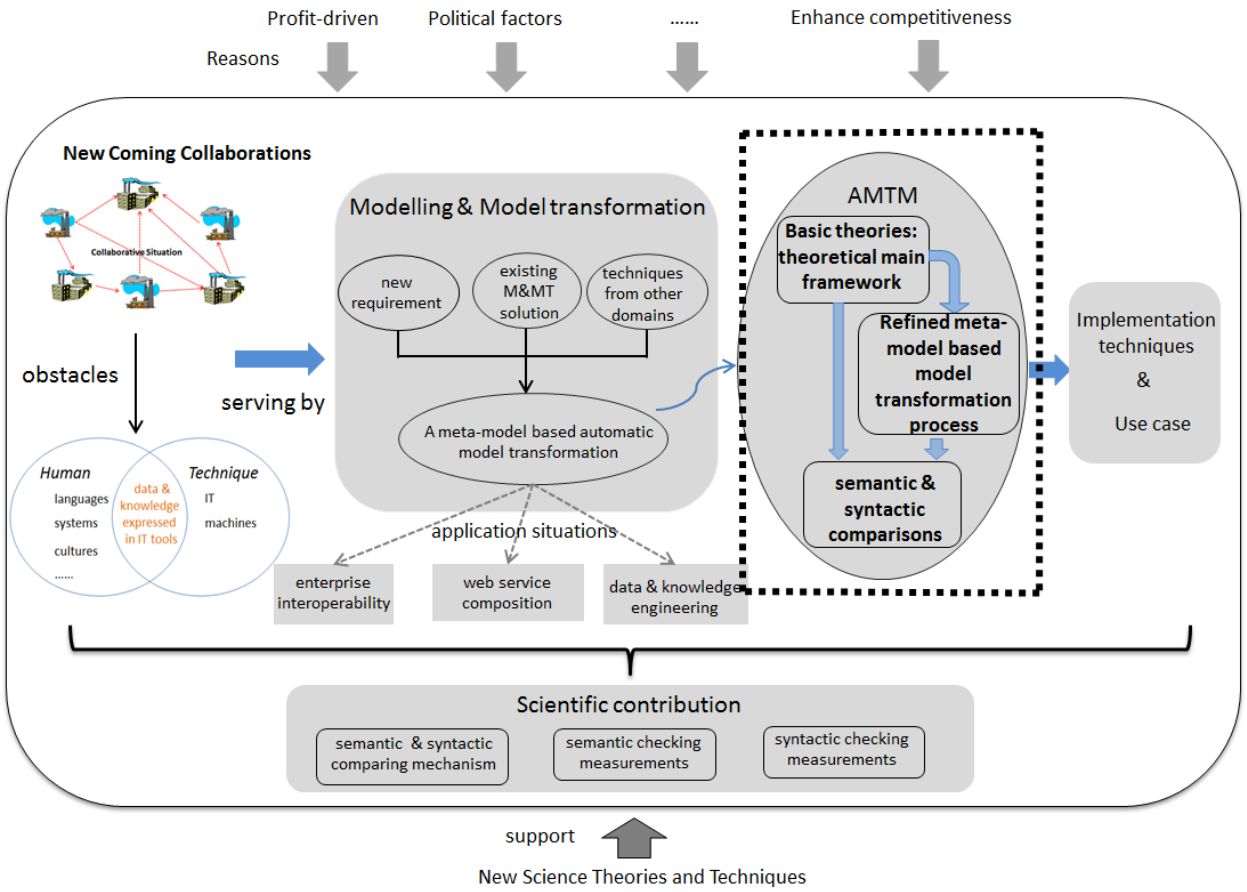


Figure III-1: The position of chapter three to the thesis.

As shown in Figure III-1, this chapter presents the whole AMTM methodology; it contains three main parts.

- The fundamental theories of building AMTM.

- The process of doing AMTM.
- The idea of combining the main technical issues "semantic and syntactic checking (S&S)" measurements into model transformation process.

Concerning to the three main obstacles "**detecting semantic relations**", "**detecting syntactic relations**" and "**solving granularity issue**", the third part focuses on the first two obstacles and the second part focuses on the third obstacle. The first part shows an overview of AMTM, and the second and third part details particular issues.

III.2 Fundamental theories of building AMTM

This section focuses on the theoretical foundation of building AMTM; it aims at giving the formal answers to the following questions.

- What is model transformation?
- Why doing model transformation?
- How to do model transformation?
- How to involve semantic and syntactic checking measurements into model transformation process?

This section contains two main subsections: the theoretical main framework that gives the answers to the three first questions, and the meta-meta-model (MMM) involved in this main framework that answers the fourth question.

III.2.1 Theoretical main framework defined in AMTM

In the first chapter, the relationship between modeling and model transformation in model-driven development domain has been illustrated. Furthermore, several definitions of model transformation, which are defined in (Tratt, 2005), (Miller et al, 2003) and (Kleppe et al, 2003), have been presented. In AMTM, model transformation is regarded as a process of taking in source models and generating target models.

Based on model transformation principles and the mechanism of detecting the potential relationship between source models and target models compose the theoretical foundation of AMTM. This foundation has passed several iterative versions presented in (e.g. (Wang et al, 2014b)). This theoretical foundation is created as a theoretical main framework that is shown in Figure III-2.

This theoretical main framework is created based on the work stated in (Bénaben et al, 2010). This framework gives answers to the questions "why doing model transformation" and "How to do model transformation" with the context of AMTM.

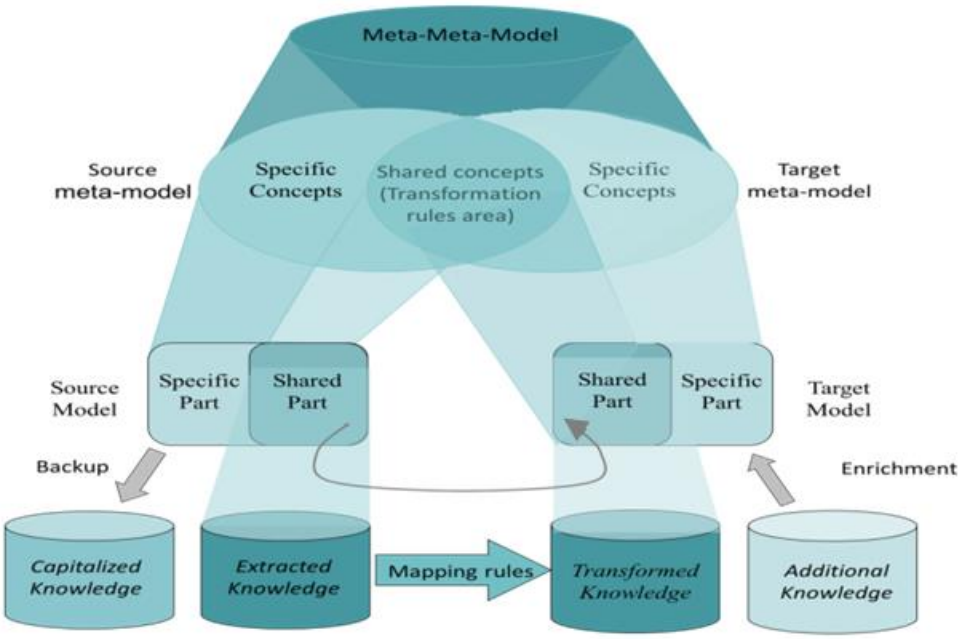


Figure III-2: The theoretical main framework created for AMTM.

The significance of doing model transformation could be "sharing knowledge", "exchanging information", etc. As stated in (Bézivin, 2006), "models are built based on the rules defined in their meta-models", such potential common items on model layer could be traced on meta-model (MM) layer (regarded as shared concepts here). Thus, for both source model and target model, they could be regarded as composing by two parts: shared part and specific part. The shared part provides the extracted knowledge, which may be used for model transformation, while the specific part should be saved as capitalized knowledge in order not to be lost. The transformed knowledge and additional knowledge may be finally used to create the shared part and the specific part of the target model, respectively.

Here, we show an example focusing on shared concepts between two meta-models. In the context of relational database management system, a meta-model should at least contain two elements: "table" and "column"; the relation between the two elements is: "table" contains "column". In the context of object-oriented programming domain, a meta-model should at least contain four elements: "class", "interface", "attribute" and "method". The shared (or similar) concepts between the two meta-models are: "table-to-class", "table-to-interface", and "column-to-attribute"; they are potential matching pairs on meta-model level and should be detected.

AMTM is 'a meta-model based' model transformation methodology; this means that model transformation rules should be built on meta-model layer. In order to discover automatically the model transformation mapping rules on meta-model layer, semantic and syntactic checking measurements should be used to detect shared (same or similar) concepts on this layer. The principle of applying S&S on model transformation process is stated in (Del Fabro et al, 2005), transformation mappings should be built between two models that are conformed to the same meta-meta-model. In this theoretical main framework, a meta-meta-model (MMM) is defined at a high abstract level. The detail of this MMM is illustrated in the next subsection.

III.2.2 The meta-meta-model

In AMTM, the mechanism of applying S&S on meta-model layer is defined in a MMM, which is shown at the top of Figure III-2. Normally, a meta-meta-model defines the rules for meta-modeling; there exists several meta-modeling architectures, for example "MOF: Meta-Object Facility" (OMG, 2008). However, these architectures serve to general purpose. They aim at serving to general problems coming from all engineering domains; they define their own semantic and syntactic to build meta-models. For this research project, which focuses particularly on model transformation domain, these meta-modeling architectures seem to be huge and complex (without specific focus on model transformation domain). In AMTM, model transformation mappings should be built automatically. To achieve this purpose, semantic and syntactic checking measurements are applied on meta-model layer to detect same or similar concepts between the source and target meta-models. Thus, the meta-models should be built (or deduced) with special formats; in this way, specific S&S measurements could be involved. Another reason to define such a MMM in AMTM is to help solve the granularity problems involved in model transformation process. As explained in the former chapters, one of the problems of defining automatic model transformation methodology is the granularity issues involved in. The MMM also defines a possible solution to this problem in an overall perspective. Figure III-3 shows the detail of this meta-meta-model.

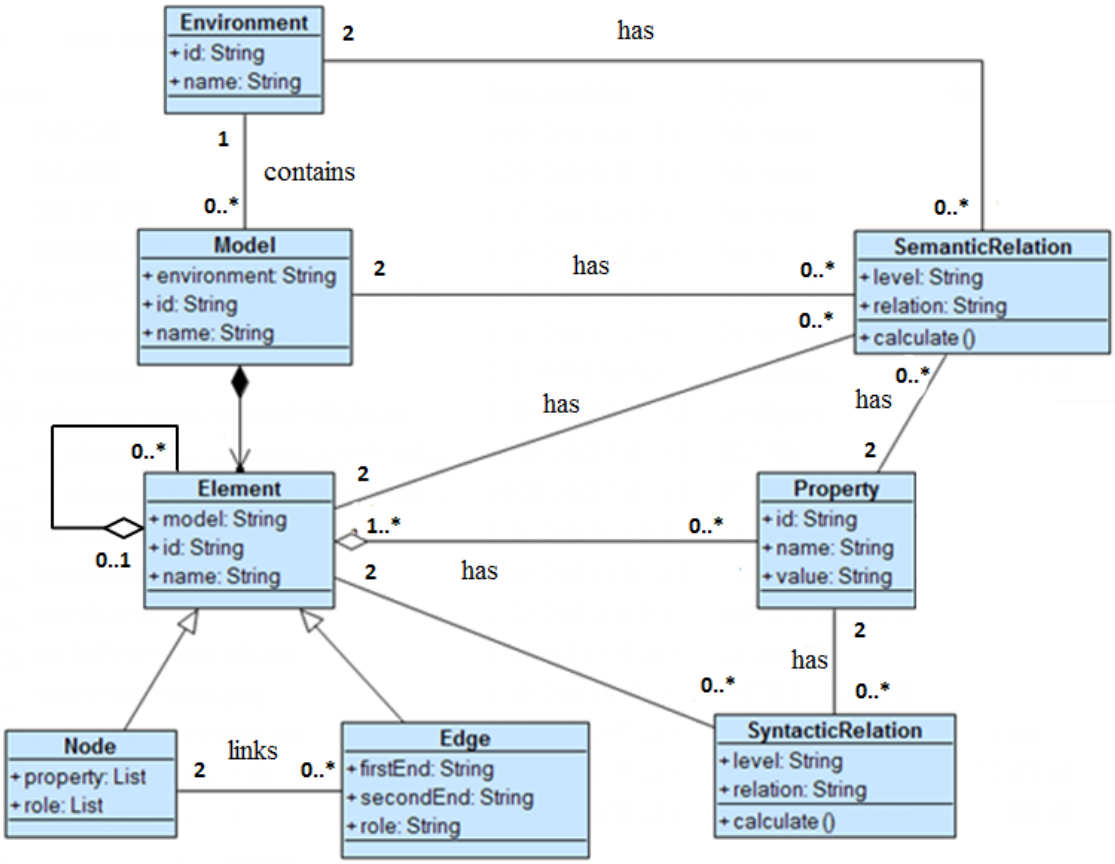


Figure III-3: The meta-meta model involved in the theoretical main framework.

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

- "**Environment**" describes the context of a system which models belong to.

- "**Model**" stands for all kinds of inputs and outputs (models at different layers: meta-model or model layer).
- "**Element**" represents all items that could be contained in models (elements are self-contained). The "Element" has two instances: "Node" and "Edge".
- "**Node**" stands for an object or a concept; it is used to describe a subject that exists in the world. A node should have at least one role to perform in model containing it.
- "**Edge**" describes the relationship between "Nodes". Every "Edge" links two roles (nodes stand for the same or different conceptual range).
- "**Property**" is used to identify and explain the "Element" (node or edge) that contains it. Each "Property" has a "Data Type". The data type could be either a "**Primitive Type**" (string, integer, double, Boolean, etc.) or an "**Enumeration type**" (defined by users); both of the two types are used to identify 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.

In a specific domain "Environment" defined in the MMM, many "model" instances could exist. Normally, an "Environment" and its "model" instances should have meaningful names to be understandable to people and differentiate to each other. So, semantic relations might exist between different "Environments" and "Models" contained. A "model" instance contains a set of "Element": a group of "Nodes" and a group of "Edges" to connect these "Nodes". The potential model transformation mappings should be built among these "Nodes" and "Edges". For a specific model instance, the "Elements" defined in it might be some specific domain concepts (terms); such specific concepts may not have exact semantic meanings or remain un-understandable to other domains. So, only semantic relation checking on "Element" is not enough; syntactic relation checking is also needed here. Furthermore, "Element" contains a group of "Property", which could be regarded as its identifiers. The same reason as on "Element", syntactic relation checking is also needed on "Property". A "Property" in a "Model" instance could also be a concept or sub-concept with specific context defined by its "Element". "Property" contains "Data Type": "Primitive" or "Enumeration". This "Data Type" could be regarded as the attribute of "Property", and it helps to differentiate "Properties" on some aspects.

As explained in former chapters, semantic and syntactic checking measurements could be used to detect same (or similar) concepts and relations contained in models; So, two of the semantic and syntactic checking methodologies that are focused on element and property defined in MMM, are introduced to detect the potential mappings (replace the manually effort) in AMTM. Also, in the MMM, the property and its dada type are highlighted; both of them are used to deduce semantic relations among elements. Furthermore, the inner attribute of element and property: their names, have also been used to define semantic and syntactic relations.

III.3 Iterative model transformation process on meta-model level

Different to many other model transformation methodologies, model transformation is regarded as an iterative process in AMTM. Several intermediate target models might be generated between original source model and final target model. Between the two models, several intermediate models could exist. Figure III-4 is an illustration of this iterative process.

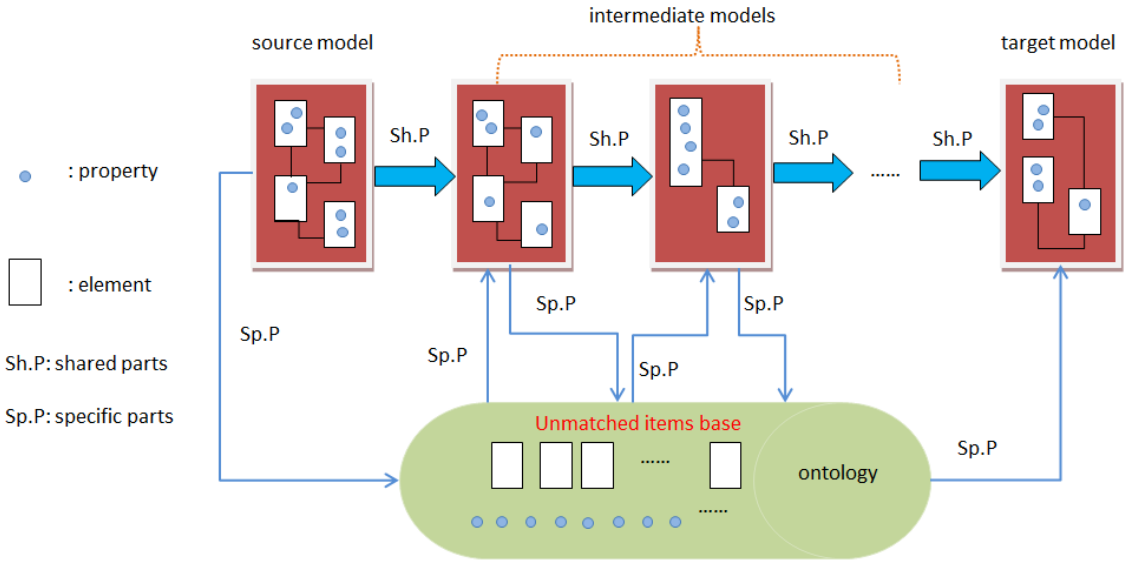


Figure III-4: Iterative model transformation process.

Within each iteration phase, the specific parts of the source model should be stored (in order not to be lost) and the ones of target models should be enriched. The iterative process allows using the specific parts from former transformation phases to enrich the specific parts of the target models that are generated in the latter transformation phases. In AMTM, a specific ontology is created to store and reuse these specific parts generated during the iterative transformation process. The reason of choosing ontology (not database schema or other storage methods) to work in AMTM is: ontology focuses on formal semantic meanings and shares understandable knowledge; it could help AMTM to enhance the semantic detecting part. This ontology is designed with the same structure as the MMM and named as "AMTM_O". To detect the shared parts within each transformation iteration phase, S&S measurements are applied. An illustration of the detecting process is shown in Figure III-5.

AMTM builds model-to-model mappings and transformation rules among elements and properties on meta-model level. The detecting process contains four mains steps.

- Applying S&S measurements on meta-model level to generate potential mappings and transformation rules.
- Using the generated mappings on model level to get transformation results.
- Validate the potential mappings and transformation rules based on the transformation results.

- Considering the validation results from the third step to define new mappings and transformation rules on meta-model level.

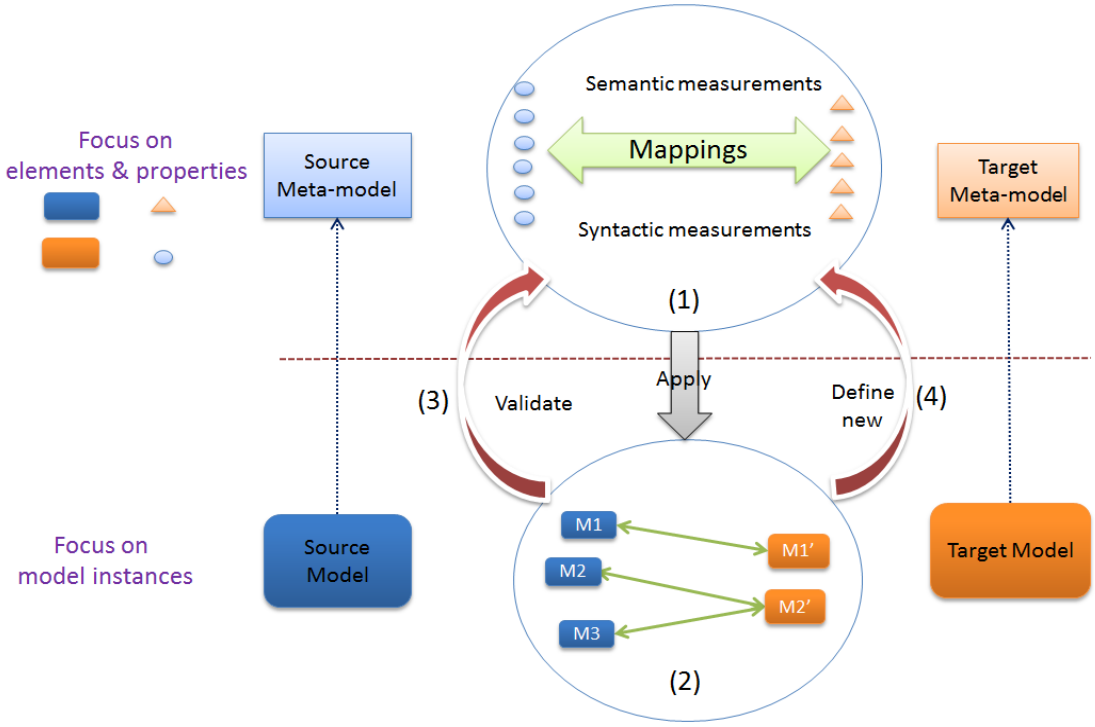


Figure III-5: Detecting process for shared parts between source and target models.

III.3.1 Principle of matching in AMTM

According to (Del Fabro et al, 2009), the main problems of existing model transformation practices are: **low reusability, repetitive tasks and huge manual work involved**, etc. In order to solve these problems, an ideal solution is defining an automatic model transformation methodology based on semantic and syntactic checking measurements. However, as stated at the end of the first chapter "**using modeling and model transformation to serve collaborations"** brings new problematic: **granularity diversity**". Considering the content conveyed by the MMM, the granularity issue reflects mainly on the mismatching between items that are belonging to "element" level and "property" level, respectively. Normally, model transformation mappings are built between the items that are belonging to the same level (considering the context of MMM: element-to-element and property-to-property). However, only such kinds of mappings are not enough to make model transformation rules capable to solve real engineering problems. So, the matching mappings between elements and properties are also necessary.

Figure III-6 shows the matching principle. According to the MMM, models are made of elements. So, the main model transformation mappings are mainly built between the elements that come from source and target meta-model, respectively. According to the theoretical main framework of AMTM, model transformation mappings should be built among the shared parts of source and target models; the specific parts of them should be stored and enriched, respectively.

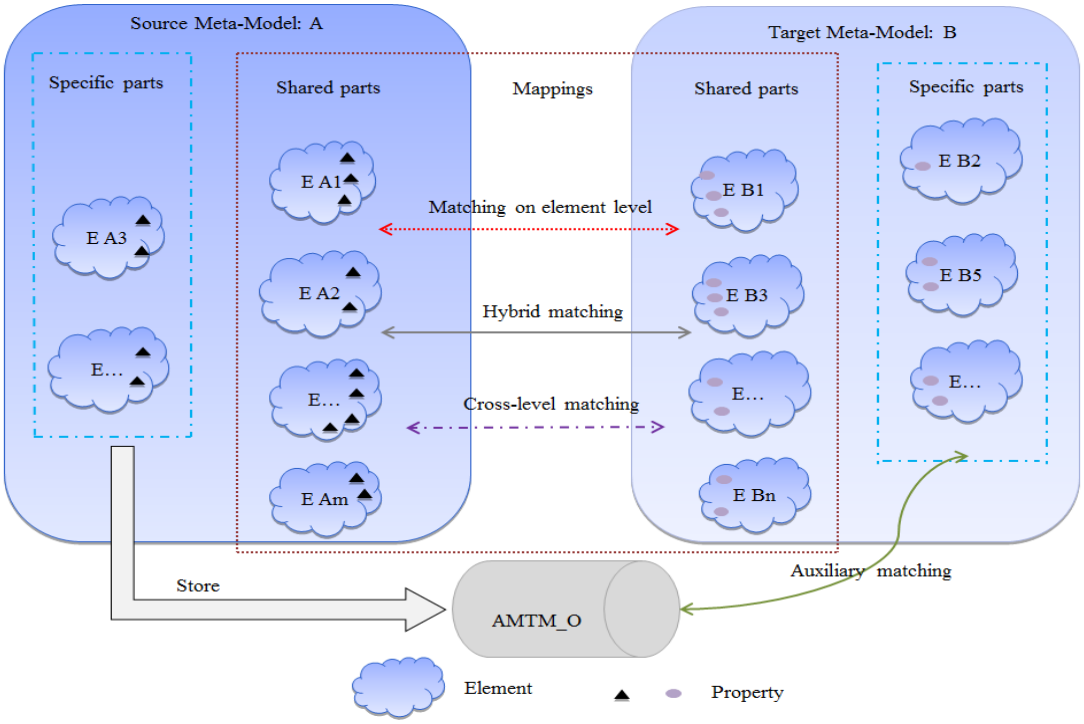


Figure III-6: Matching principle defined in AMTM.

As shown in Figure III-6, to detect the shared parts between two meta-models, three matching steps are defined (considering the granularity diversity).

- The first matching step "matching on element level" contains two phases: building mappings between element's pairs and building mappings between property's pairs that are within the matched element pair.
- The second matching step "hybrid matching" focuses on the properties (property-to-property matching), which are unmatched after the first matching step.
- The third matching step "cross-level matching" concerns making mappings between properties and elements.

For the specific parts, the ones from source meta-model are stored in AMTM_O; and the ones from target meta-model are enriched by using the fourth matching step.

- The fourth matching step "auxiliary matching" focuses on enriching the specific parts of target models by extracting additional knowledge from AMTM_O.

The details (focus and matching mechanism) of these four matching steps are illustrated in the four following subsections, respectively. Furthermore, the usage of semantic and syntactic checking measurements within these four steps is illustrated synchronously. Table III-1 shows a simple illustration of the four matching steps.

Table III-1: Simple illustration of the four matching steps

matching step	first step	second step	third step	fourth step
matching pairs	phase 1: **element – element** phase 2: **property – property** (Note: the property coming from matched elements)	**property – property** (Note: unmatched properties from SMM to all properties in TMM)	unmatched **elements** – unmatched **properties**	**mixed matching unmatched items**

III.3.2 matching on element level

According to the MMM, meta-models are made of elements and elements contain a group of properties. Therefore, model transformation mappings should be built among the elements and properties.

The first matching step focuses on detecting element's matching pair. If two elements coming from source model and target model stand for the same concept (shared concepts on meta-model layer), a mapping should be built between them. Also, if two elements are regarded as a matching pair, the properties contained in them are also matched. Figure III-7 shows the matching focus of this matching step.

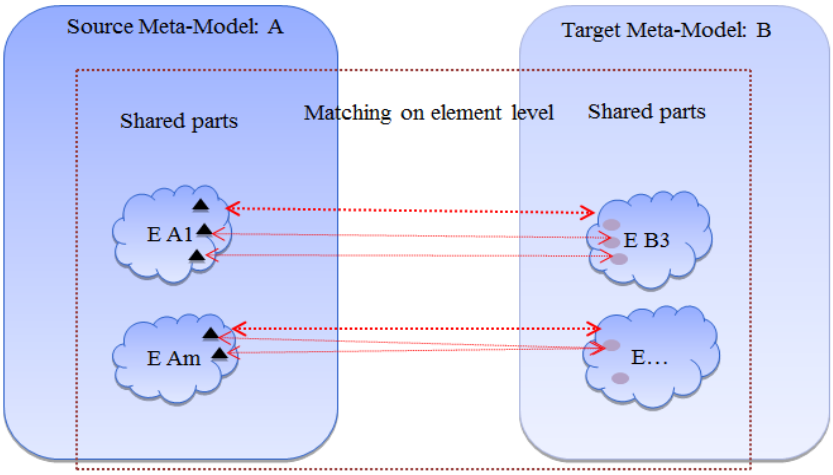


Figure III-7: Matching focus of the first matching step.

In this matching step, there is an important issue: **how to define that two elements are potential matching pair (stand for the same concepts)**? A possible solution consists in testing the semantic and syntactic relations between two elements' names and their properties' groups. Figure III-8 shows an illustration of the detecting process in the first matching step.

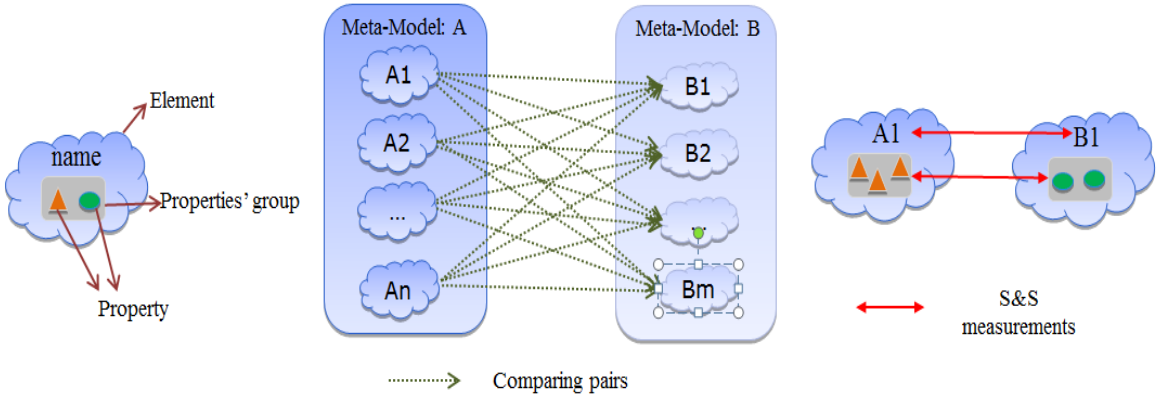


Figure III-8: Comparing mechanism in matching on element level.

Meta-model A contains 'n' elements and meta-model B contains 'm' elements. In the first matching step, the maximum number of comparisons between the two meta-models is: "m*n" (as shown by the dash lines). An element from source meta-model should be compared with all the elements from the target

meta-model until finding one or none matching element. So, in this example, every element of meta-model A will be compared at least with one element and at most 'm' elements from meta-model B.

A specific value "Ele_SSV" is calculated for every comparing pair of elements. "Ele_SSV" stands for "element's semantic and syntactic value"; it is calculated based on the semantic and syntactic relations between elements' names, and between their groups of properties. The calculation rule of "Ele_SSV" is shown in equation (1).

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

"Ele_SSV" is the sum of two independent parts: **elements' names** and **elements' properties' groups**; two **impact factors "name_weight"** and **"property_weight"** are used to determine **the weight of "elements' names" and "elements' properties"**, respectively. The range of their values is **between 0 and 1** while the **sum** of the two impact factors should always be **1**. Users could assign values to the two factors to determine the mutual importance between the two parts: elements' names and elements' properties' groups. In equation (1), "**S_SSV**" stands for "**semantic and syntactic value between two strings**"; it is calculated between two words (i.e. elements' and properties' names). This value concerns the semantic and syntactic relations between two words, the detail of calculating this value is presented in the fourth chapter. "**P_SSV**" stands for "**semantic and syntactic value between a pair of properties**". In equation (1), "**x**" stands for **the number of properties of a specific element from source meta-model**. To match a pair of elements, all the properties from the source element should be considered to make matching with the ones from the target meta-model elements. Then, an example is given to show the calculating rule for "P_SSV".

When comparing element "A1" and element "B1", their properties groups are taken into consideration. Assuming that "A1" has "x" properties and "B1" has "y" properties; the comparisons on their property level could be as maximum as "x*y". There exists a "P_SSV" in each comparing properties' pair. Equation (2) shows the calculating rule of "P_SSV".

$$P\_SSV = pn\_weight * S\_SSV + pt\_weight * Id\_type \quad (2)$$

The calculation rule of "P_SSV" is *very similar to* the one for "Ele_SSV". It is also the sum of two parts: properties' names and properties' types; and also two impact factors "pn_weight" and "pt_weight" are used to determine the weight of these two parts. *The rules for assigning values to the two impact factors are the same as for the ones in equation (1)*. In equation (2), "**S_SSV**" **stands for the semantic and syntactic value between two properties' names** while "Id_type" stands for "identify properties type (e.g. string, integer, float and double)". If two properties have the same type, "Id_type" will be assigned a value as 1; otherwise, this value could be 0.5 (e.g. double and float) or 0 (e.g. integer and string).

With the help of equation (1) and equation (2), every element from source meta-model could get **zero or several** potential matching elements from target meta-model. The "Ele_SSV" value between them is the maximum for the source element. This is only a potential mapping; to become a real transformation rule, mapping chosen mechanism, which is presented in the fourth chapter, is needed. Within potential matching pairs of elements, their properties are also mapped based on the "P_SSV" value calculated by equation (2). The simple example shown in figure III-7 could be used to explain this mechanism. Element

"E A1" and element "E B3" are regarded as a matching pair; two of the properties (out of three) in "E A1" could also be matched with two of the properties in "E B3".

III.3.3 hybrid matching

For the elements from source meta-models, some of them still **being unmatched after the first matching step**. Even for the matched elements, some of their properties might be still unmatched. The hybrid matching step focuses on these unmatched elements and properties. Hybrid matching step focuses on properties; all the matching pairs would be built among properties; this focus is shown in Figure III-9.

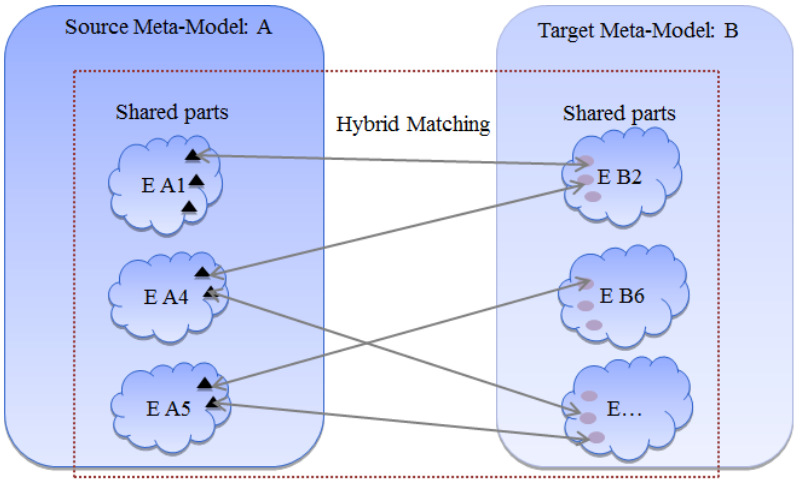


Figure III-9: Matching focus of hybrid matching step.

Within this matching step, all the matching pairs are built between source property and target property. Furthermore, these matched properties come from elements that could not be matched as a matching pair. The comparing mechanism of this matching step is simple: **comparing all the unmatched properties from source meta-model with all the properties from target model**, in order to find similar pairs. Figure III-10 is an illustration of hybrid matching step.

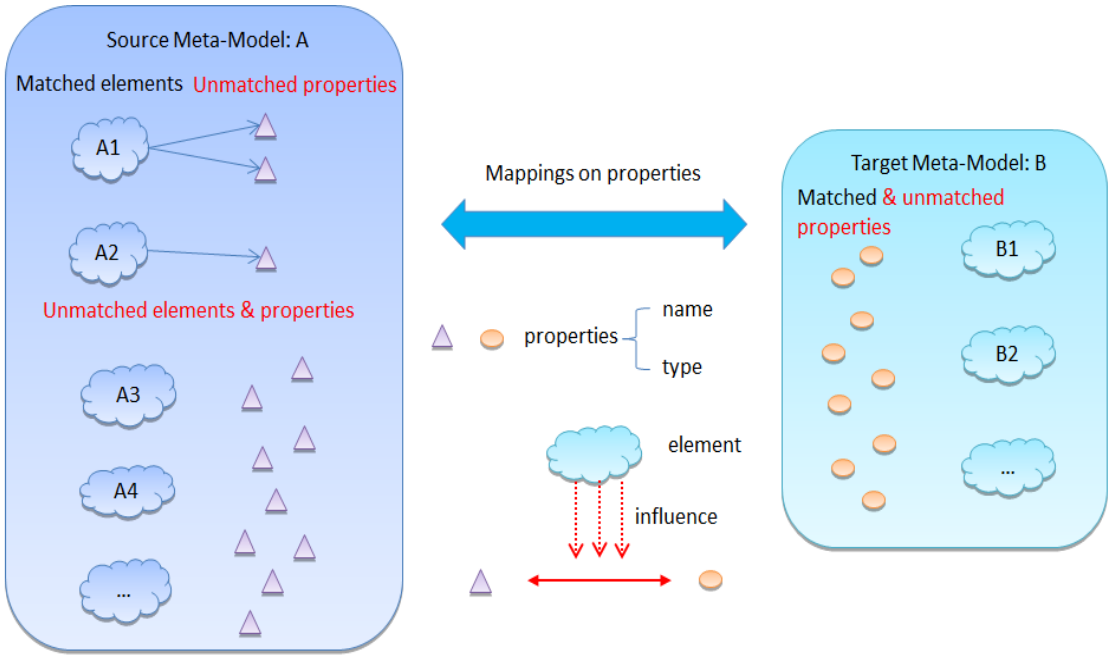


Figure III-10: Comparing mechanism of hybrid matching step.

This step aims at breaking the main granularity constraint: property matching pairs only exists within matched element's pairs. This step implements **many-to-many mappings on element level** (Properties from one element could be transformed to several target elements while one target element could be generated by combining properties that come from several source elements). Figure III-11 shows the two granularity issues.

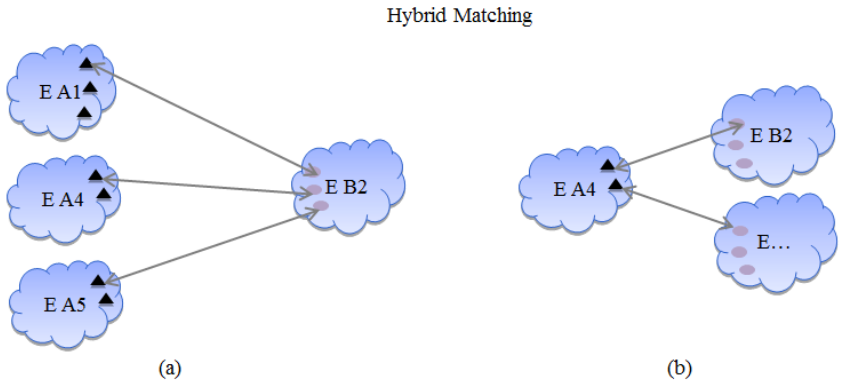


Figure III-11: Matching results in hybrid matching step.

In Figure III-11 part (a), three source elements compose the target element; in b, one source element is divideded as two parts and transformed into two elements. Hybrid matching provides the mechanism to achieve both of the two situations by overcoming the matching constraint on elements level.

When comparing two properties, this step also considers the **influence from element's level**. The matching mechanism of this step shows in equation (3).

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

"HM_SSV" stands for "hybrid matching semantic and syntactic value"; the idea of calculating it is *similar to the ones of "Ele_SSV" and "P_SSV"*. "en_weight" and "pl_weight" are two impact factors for "element level influence" and "property level influence". They perform the same role as "name_weight" and "property_weight" in equation (1). The influence from element level mainly depends on elements' names. In equation (3), "S_SSV" calculates the semantic and syntactic value between two element's names. The second calculation part in this equation concerns the property level; the comparing mechanism for this part is illustrated in equation (2).

III.3.4 Cross-level matching

This matching step focuses on the unmatched elements and properties that are left from the first and second matching steps. Cross-level means that the matchings are built among the items that come from different levels (i.e. source meta-model element – to – target meta-model property, source meta-model property – to – target meta-model element). In simple words, the matching step builds mappings between elements and properties. The matching focus of this step is shown in Figure III-12. In this matching step, source element "E A3" is transformed into a property in target element "E B5"; it is also possible to transform the property of source element "E A7" into a target element "E B8".

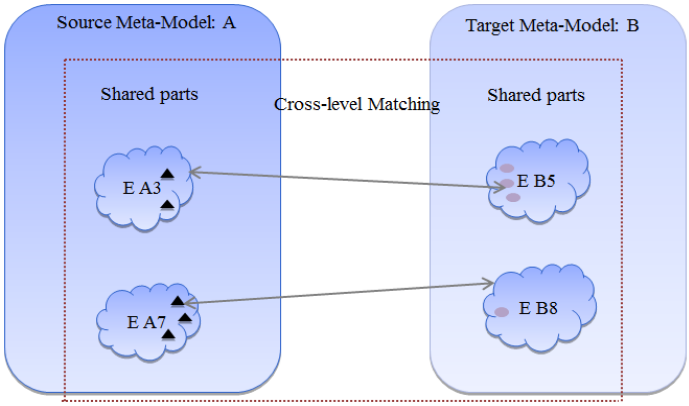


Figure III-12: Possible matching results of cross-level matching step.

When comparing an element and a property, only their names are taken into account. S&S checking are used to compare their names (both element's properties group and the type of property are ignored). Equation (4) is used to calculate the S&S relationship between a pair of element and property.

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

"CLM_SSV" stands for "cross-level matching semantic and syntactic value"; this calculation rule is only based on element's name and property's name. Semantic and syntactic values between the two names are calculated, respectively. "sem_weight" and "syn_weight" are two impact factors for "semantic relation" and "syntactic relation"; they determine the weights of the two aspects. They perform the same role as "name_weight" and "property_weight" defined in equation (1).

III.3.5 auxiliary matching

All the shared parts between source meta-model and target meta-model are regarded to be found after the three matching steps: matching on element level, hybrid matching and cross-level matching. The specific parts of source and target meta-models are still left untreated. The auxiliary matching step focuses on these specific parts. Figure III-13 illustrates the matching focus of this step.

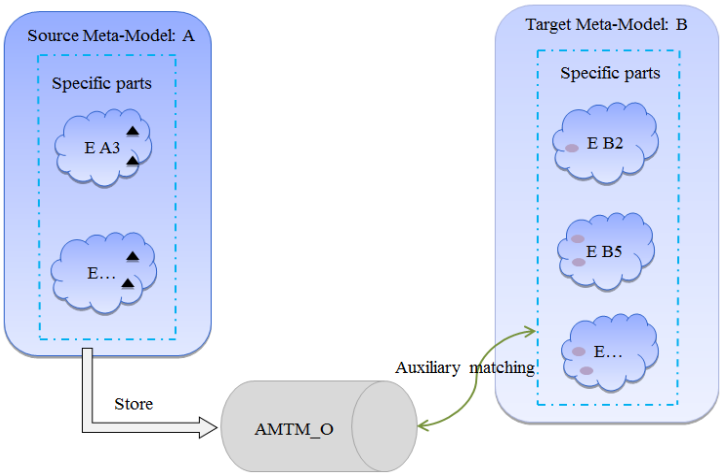


Figure III-13: Matching focus of auxiliary matching step.

Auxiliary matching step defines the mechanism of storing and reusing these specific parts (in some aspects, this step also helps to do model merging). After the first two matching steps, all the unmatched items from source meta-model are stored in AMTM_O, which is shown as Figure III-14. For a complete model transformation process, the specific parts from former iterations could be reused as the specific parts to enrich the target models generated in the latter iterations. Furthermore, the content in AMTM_O could also be enriched by the other ontologies or knowledge base from other domains (extracting concepts from other storages and storing them to AMTM_O following the specific formats). For this reason, the specific parts from source meta-models are not sufficient to generate all the specific parts needed by the target meta-models that are in the same iterative process.

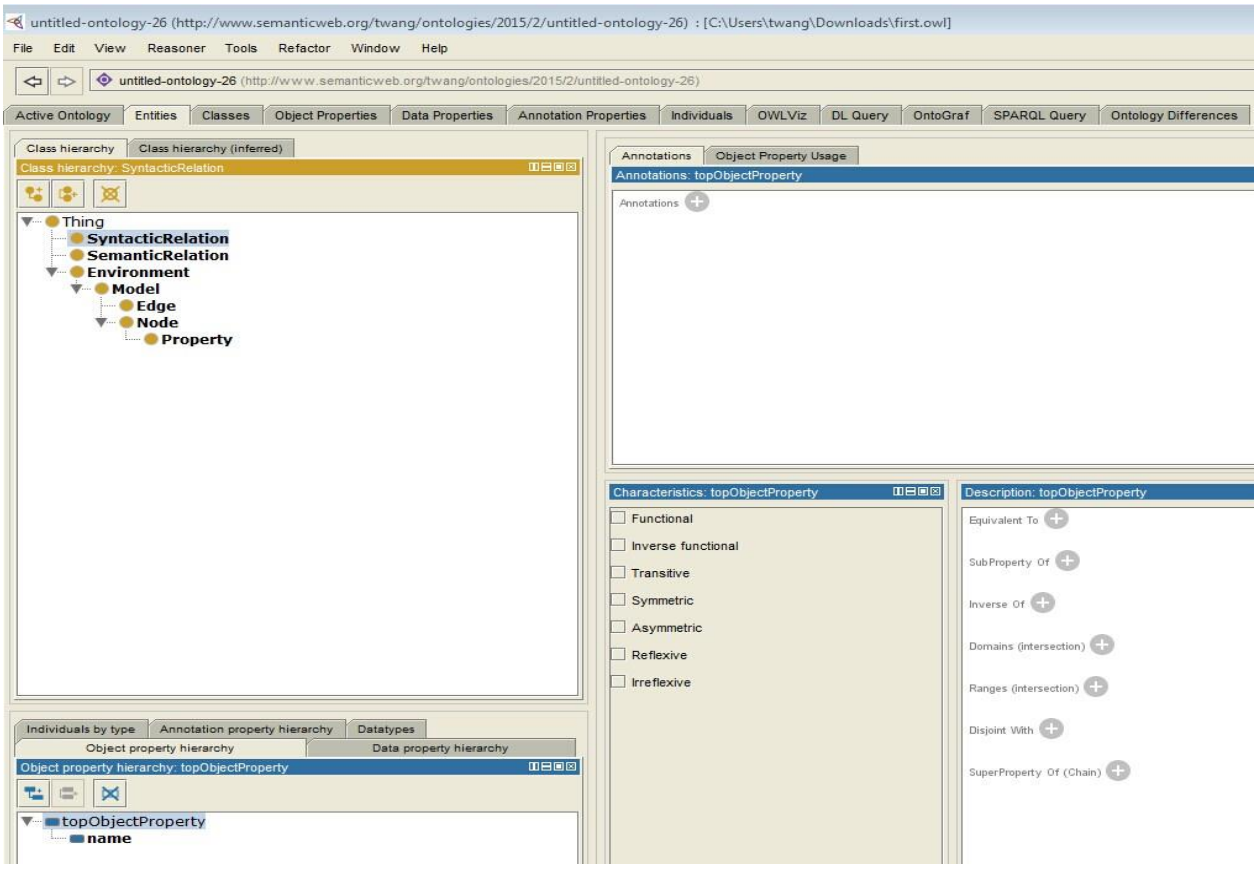


Figure III-14: Structure of AMTM_O (represented thanks to Protégé).

As stated in (Gruber, 1995), "formal ontologies are viewed as designed artifacts, formulated for specific purposes and evaluated against objective design criteria". In AMTM, AMTM_O should be designed and formulated to serve model transformation process. Since it is used as an aid to do the S&S (storing and reusing specific parts from source meta-models and items pairs have semantic relations), AMTM_O should work with the matching mechanism (relevant to S&S) that is defined in MMM. So, it is designed and formulated following the same structure of the MMM.

AMTM_O could be divided into two parts; an intermediate one and a final one. In order to save time and improve efficiency of auxiliary matching step, a temporary one (intermediate) is built. For AMTM, a final ontology is created; a specific temporary ontology is created for one entire model transformation process. In one model transformation process, all specific parts from source meta-model are stored in this temporary ontology. Furthermore, to enrich the target meta-model, this temporary ontology would also be explored first. When the whole matching process is finished, all the specific parts that are still left in this temporary ontology will be transferred to the final ontology. If there is some specific parts still left unmatched in the target meta-model, the final ontology could provide a possible solution (contents left from other model transformation processes) to enrich this part.

III.4 Validation process on model level

As shown in the theoretical main framework, the majority model transformation mappings are built on meta-model layer. Furthermore, figure III-5 illustrates the validation mechanism for these mappings.

Model transformation mappings are built based on the calculation results of S&S measurements between source meta-model and target meta-model. In order to validate if these mappings are proper, they should be used on model level (using real practices to test). These generated mappings could be applied between any source models that are conformed to the source meta-model and target models that are conformed to the target meta-models. The mapping results should be examined by domain experts or other professionals: **manual verification (MV)**. Based on the final examination results, new mappings could be built; also, the S&S measurements (impact factors used in the equations that are mentioned above) could be modified for the specific application domains, which have been examined by the domain experts.

This validation method involves manual efforts, which violates the original intention of AMTM. So, another automatic validation methodology is also involved in AMTM. This automatic validation methodology is named as "**reverse detection validation (RDV)**".

Normally, meta-model based model transformation methodologies aim at building mappings on meta-model layer: from source meta-model to target meta-model. The rules for detecting potential matching pairs have been illustrated above: from source meta-model to target meta-model. The RDV process takes the original target meta-model as source meta-model and builds mappings again from target meta-model to source meta-model. Then, generating another set of model transformation mappings rules. The overlap parts of the two model transformation rules sets generated by AMTM will be regarded as the final model transformation rules; the specific parts, which are either from the first comparing phase (source to target) or from the validation comparing phase (target to source), will be left to the system users to validate. Since model transformation validation aspect is not a core component in AMTM, just two of the validation mechanisms applying in AMTM are presented here. A simple comparison between the two validation methods is given in Table III-2.

Table III-2: Validation methods for AMTM

Methods \ Feature	Credibility	Time consuming	Costs
MV	high	yes	high
RDV	neutral	neutral	neutral

III.5 Conclusion

This chapter presents the overview of AMTM. The main content of this chapter is divided into two parts: theoretical solution basis and execution process detail.

For the first part, a theoretical main framework and the meta-meta-model involved are shown and illustrated. The basic theories of doing model transformation in AMTM are defined within this framework. Since AMTM is a meta-model based, automatic model transformation methodology, semantic and syntactic checking measurements have been combined into the model transformation process. MMM defines the mechanism of using S&S measurements on meta-model level to detect the transformation mappings automatically.

Besides this theoretical main framework, another main principle of AMTM is defined and shown in figure III-4. Model transformation is regarded as an iterative process in AMTM; each iteration phase is an independent model transformation process following the principles defined in the theoretical main framework. This iterative theory also illustrates AMTM execution process detail.

For the second part "execution process detail", it is divided into four main matching steps. According to the MMM, S&S measurements have been involved in each matching steps with different focuses and usage. One of the main weaknesses of automatic model transformation methodologies lies in the "granularity diversity"; these four main matching steps aim at solving it. AMTM implements the many-to-many and cross-level (mismatching between property level and element level) matching mechanism.

As a main aspect of doing model transformation, "model transformation validation", which is a component in AMTM, has also been illustrated in this chapter. Two main validation methods involved in AMTM: experts' validation (relies mainly on manual work) and reverse detection validation (could be implemented automatically).

Chapter IV: Combining S&S into model transformation process

IV.1 Introduction: the reason and objective of applying S&S in AMTM

This chapter focuses on the mechanism of combining semantic and syntactic checking measurements as a whole into AMTM. The content involved in this chapter contains three main parts.

- The generous purpose of using semantic and syntactic checking.
- Methods of doing semantic checking and syntactic checking, and the potential relation between the two checking measurements.
- The mechanism of using S&S in AMTM and relevant issues about using S&S in AMTM.

As stated in the third chapter, S&S measurements play a key role (involved in all the matching steps: calculation equations) in automatic model transformation process. This chapter also reveals the importance of S&S in AMTM; the position of this chapter in this thesis is shown in Figure IV-1.

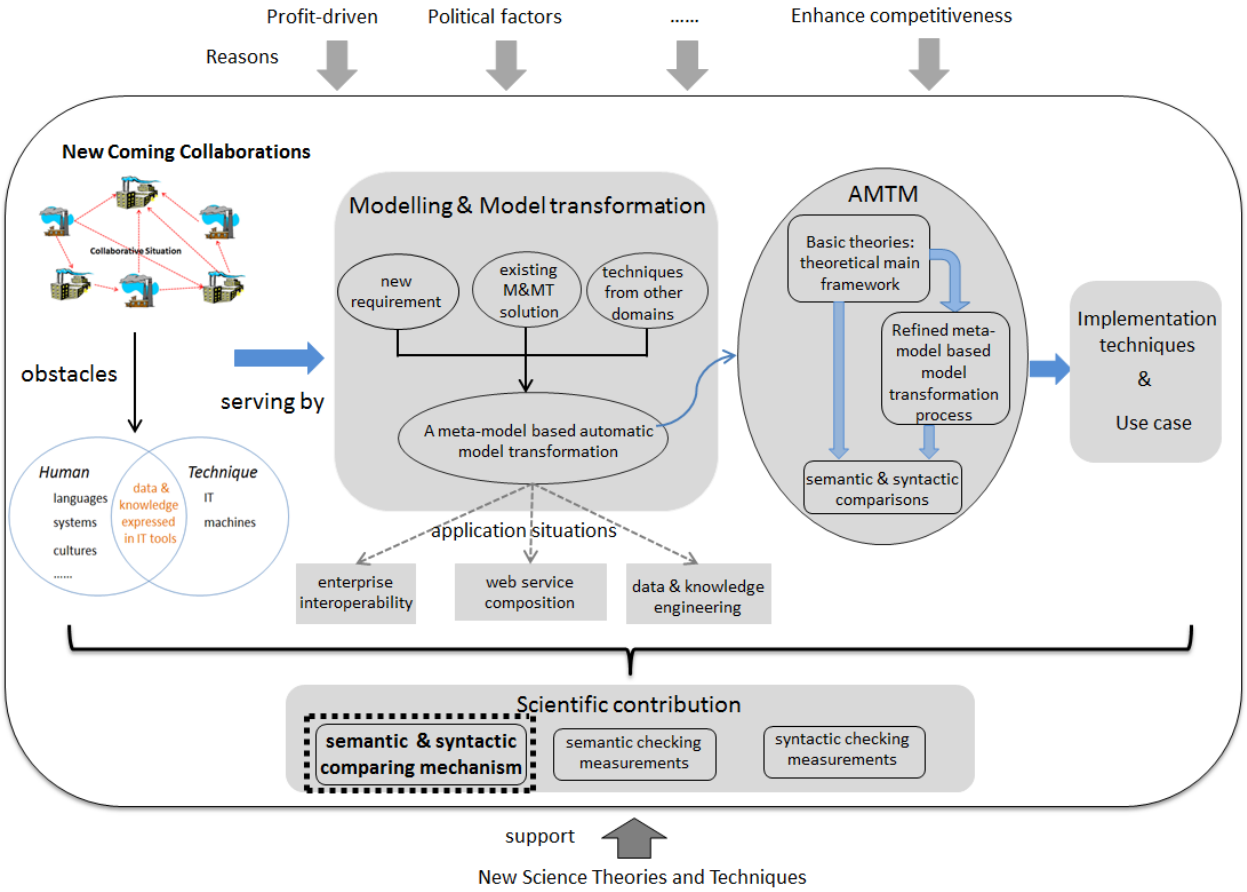


Figure IV-1: The position of chapter four in this thesis.

Semantic checking and syntactic checking are mostly used as a key point to replace manual efforts; for the reason that the checking process could be executed by using IT (computer assist) methods. These two checking methods are created on the basis of "natural words", and focus on the composition of alphabet

and semantic meanings carried by the words, respectively. Taking the word "star" as a simple example, on one hand syntactic checking focuses on the alphabet composed 's' 't' 'a' and 'r', and their positions in the word (position 1,2,3,4); on the other hand, semantic checking pays attention to the semantic meanings that are conveyed by this word: four meanings as a noun and two meanings as a verb.

The mechanism of using S&S in AMTM is regarded as one of the scientific contribution that is proposed in the research work presented in this thesis. AMTM is a general model transformation methodology; it aims at providing service to all the engineering domains adopted model driven development approaches.

The main reason for using S&S could be replacing huge amount of manual work by using high-performance computers. In some terms, semantic and syntactic checking measurements have already been used in model transformation domain; however, the usage is limited to particular model transformation methodologies (e.g. serving to particular domain, building on specific model transformation techniques). To be more precise, the scientific contribution of this chapter concerns idea of combining S&S in a general model transformation methodology.

This chapter is divided into five main sections. The second section presents the principle of applying S&S both in general situations and in AMTM: focusing on a pair of words. The third section concerns the mechanism of assigning values to the impact factors that are shown in the equations, which combines S&S into the model transformation process, in the third chapter. The fourth section illustrates the mechanism of choosing matching pairs (generated based on the semantic and syntactic relations between the items involved in potential matching pair), and the fifth section gives a simple example to explain the mechanism of applying S&S illustrated in this chapter. At the end, a conclusion of this chapter is shown.

IV.2 The mechanism of combining S&S in AMTM

As stated in the second chapter, both semantic and syntactic checking measurements are applied on a pair of words. Syntactic checking concerns the letters involved in a word and the sequence appearance of letters. Semantic checking focuses on all the semantic meanings that are conveyed by the word itself. By detecting the syntactic relation and semantic relation between a pair of words, a proximate similarity degree between the two words could be generated. Based on this similarity degree, some further processes could be executed between the two words (e.g. mapping, transform). Normally, the detecting step needs manual effort.

Considering the context of model transformation domain, the aim of applying S&S is to transform source model to target model. To achieve this aim, comparing the two models and building transformation mappings between them are necessary. Furthermore, the MMM defined within the theoretical main framework of AMTM shows the mechanism of doing S&S comparing between models. Model transformation mappings should be built between elements, properties, and among elements and properties, which are defined as the main items contained in meta-models by the MMM.

In simple words, semantic and syntactic checking rules are built on the basis of words sets. But, in the context of model transformation, the comparing items are properties and elements (not directly words themselves). In order to fulfill this gap, a bridge between words set and model items (elements and

properties) should be built. In the third chapter, a simple idea of building this bridge is illustrated. Here, Figure IV-2 shows this bridge.

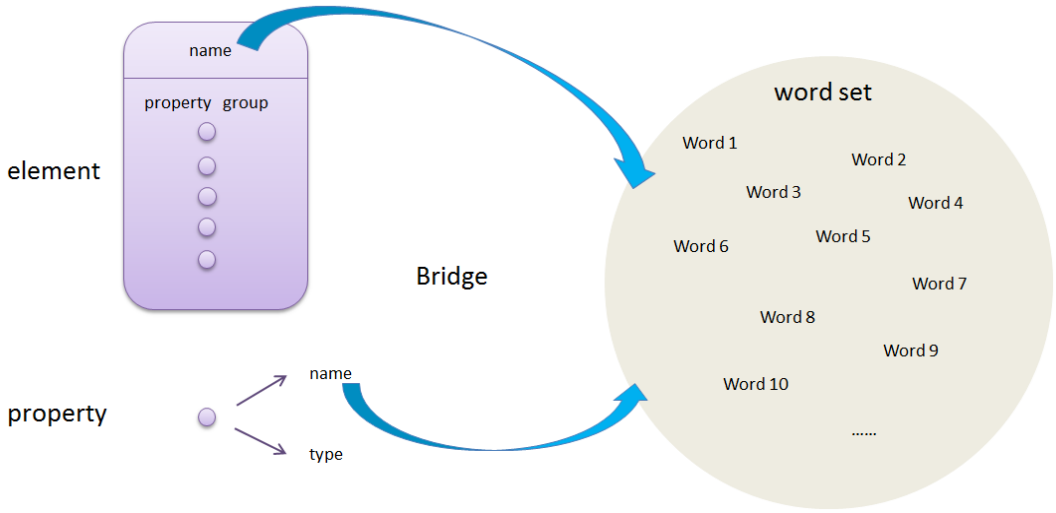


Figure IV-2: Bridge cross the gap of models' items and word set.

Element is regarded as two parts: ***element name*** and **property's group**. Element name is a word; between two elements' names, S&S could be used to get a relation between them. Property group is regarded as a whole; S&S could be used on single properties' pairs. A property could also be regarded as two parts: ***property name*** and **property type**. Property name is a word; between two properties' names, S&S could be applied. For property type, there are a range of options (e.g. string, integer and double). To compare two properties, their types could be same, different or similar (e.g. integer and double). The four equations presented in the third chapter reflect the idea of building this bridge in detail. All elements' and properties' names are units in the word set; they could be located in word set and compared by S&S as simple words.

As a short conclusion, the items contained in meta-models own names which represent by words; furthermore, S&S measurements could be applied between word pairs, and define semantic relations between specific word pairs. Therefore, S&S measurements are involved in model transformation process in this way.

IV.3 Relation between semantic checking and syntactic checking

S&S measurements are used as a whole; whereas it contains two parts: semantic checking and syntactic checking. In some terms, they are independent: focusing on different aspects of the words. Both of them could be used between a pair of words; considering the applying objects, some potential relations could be built between the two checking measurements.

Between a pair of strings, there always exists a syntactic relation between them. The syntactic relation stands for the syntactic similarity between them. If two strings are words that are with semantic meanings, there might be also semantic relation between them. So, there is one question **whether the syntactic relation between two words could influence the semantic relations between them?** The answer to this question is definitely yes. If two words have a very high syntactic similarity, it means they probably stand for one word (convey same semantic meanings) or two words with opposite semantic meanings (e.g. usual and unusual). Usually, to detect the semantic relations between two words is more complex and time-consuming (this part will be illustrated in detail in the sixth chapter) than to detect the syntactic relation between them. So, in AMTM, syntactic relation checking is always executed before semantic checking between two words. Equation (5) is defined to calculate the S&S relation between two words (if either one is just a string 'without semantic meaning', there is just syntactic relation between them).

$$S\_SSV = SeV\_weight * S\_SeV + SyV\_weight * S\_SyV \tag{5}$$

"S_SSV" has been used in the former equations (e.g. equation (1), equation (2)); it stands for the semantic and syntactic relation (in AMTM, it stands by a value between 0 and 1) between a pair of strings. "S_SeV" stands for the semantic value while "S_SyV" stands for the syntactic value between two strings. Two impact factors "SeV_weight" and "SyV_weight" are defined here; they are ranging from '0' to '1' and the sum of them is '1'. They work together to determine whether semantic or syntactic relation is more important for a specific pair of strings. A special case is between a pair of strings without semantic meanings, "SeV_weight" should be assigned as '0' and "SyV_weight" should be assigned as '1'.

Based on different application situations, the two aspects "semantic and syntactic" might have different effects (e.g. in enterprise engineering, semantic meaning may be more important but in medical field, syntactic may be more important). So, a better way of assigning values to the two impact factors should be: leaving it to domain experts. Another possible solution is applying use case to do tests and based on the test results (also judged by domain experts) to modify the ranges of the two impact factors.

IV.4 the mechanism of selecting matching pairs based on S_SSV

The idea of using semantic and syntactic checking to detect the potential relation between a pair of words has been illustrated above. Moreover, with the mechanism defined in the equations (1) to (5), S&S could be involved in AMTM to define relationship between a pair of model items defined in the MMM. However, there is still one issue to be solved: how to choose matching pairs as the final model transformation mappings based on the S&S relations (stands by values) between the items involved in the pairs.

To analysis this issue, two levels are defined: S&S relation between a pair of words and S&S relation between a pair of items that are defined in MMM. The mechanism of choosing potential matching pairs is needed to build the final model transformation rules in model transformation process. The mechanism of

using S&S to detect potential mappings has been presented above; this part focuses on the matching pairs choosing mechanism.

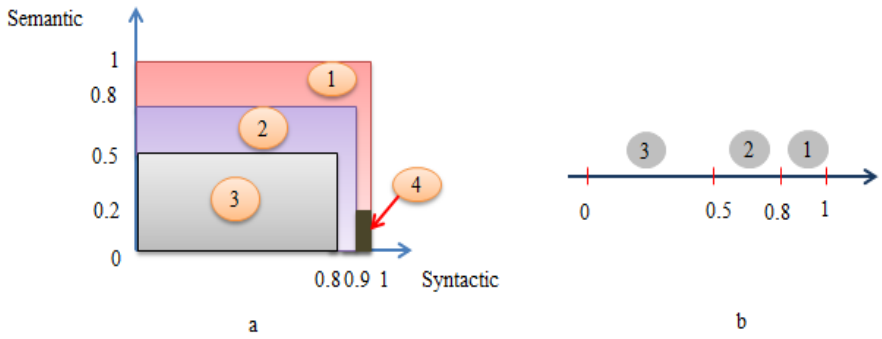


Figure IV-3: Matching pair choosing mechanism.

Figure IV-3 shows two levels of this issue: situation (a) and situation (b). Normally, for most engineering domains, strong semantic relation means high possibility of making mappings between two words. In AMTM, S&S relations stand by values that range from '0' to '1'; the higher this value is, the stronger the relation is. Figure IV-3 situation (a) shows the mechanism of choosing potential matching word pairs.

- **Region 1** stands for two words that have close relation (**strong syntactic relation and strong semantic relation**); two words have such a relation could transform to each other (matching pairs).
- **Region 2** stands for two words that have **high semantic and high syntactic relations (or normal semantic relation and strong syntactic relation**); the words pairs belong to this region are regarded as potential transformable pairs.
- **Region 3** stands for two words having weak relations (**weak semantic relation and not strong syntactic relation**); such a word pair has low possibility to be a matching pair or a potential matching pair.
- The special word pairs are in **Region 4**, which stands for word pairs that have **strong syntactic relation but very weak semantic relation**. For example, word pair: common and uncommon. Such a pair of words is regarded to have an antonym semantic relation between each other, and thus no transformation mappings would be built between them. In model transformation context, only same or similar concepts pairs are considered valuable (not concepts with opposite meanings).

The second level of choosing matching pairs concerns the relation between two potential matching items defined in the MMM (coming from source and target meta-models, respectively). For example, the relation between two "elements" is represented by a value "Ele_SSV", which ranges from 0 to 1. "Ele_SSV" is calculated based on semantic and syntactic comparisons among two elements' names and properties' groups. The mechanism of selecting matching elements pairs depends particularly on the range of this value. As explained in the former sections "there is a gap between words pairs and model items pairs", so generating an S&S value between a pair of words is not enough to define model items mappings pairs. Considering about this gap, assigning threshold values for choosing potential matching model items is necessary.

As shown in Figure IV-3, two threshold values: 0.5 and 0.8 are defined to choose model items' (i.e. element and property) matching pairs. Taking "Element" as example, if two elements have "Ele_SSV" 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 (this situation will be left to users to decide if make mappings or not); else, if this value is **in region 3** (Less than 0.5), no mappings will be built between the two elements. When comparing property pairs or element-property pairs, this choosing mechanism stays the same.

Since the mechanism of choosing potential model items matching pairs is the same for all the three matching levels: **element to element**, **property to property** and **element to property**, an element (or a property) may have from "0" to several potential matching items. So, in this way, a "many-to-many" transformation rules are built on both element and property levels from source meta-model to target meta-model.

IV.5 Simple use case of using S&S

This section presents a simple use case, which aims at showing the mechanism of applying S&S that is defined within AMTM. The use case shown in this section is just carried out within words' pairs; a more complicated and model transformation functional concerned use case will be given in the seventh chapter. The use case in this section contains three groups of word pairs; Table VI-1 shows the detail of them.

Table IV-1: Word pairs defined in this use case

Group	Word pair	Note
No.1	acridorex & acrisorcin	only syntactic relation
No.2	students & student	syntactic checking is enough
No.3	teacher & person	Both semantic and syntactic checking

The word pairs in the first group are some specific strings, which do not have semantic meanings leaving their special context. The example "acridorex" and "acrisorcin" are two names of some medicines. Without the context of medical science, they are meaningless. However, for majority semantic thesaurus, such kinds of words would not be included in. So, dealing with such a pair of words, the impact factors "SeV_weight" and "SyV_weight" defined in equation (5) should be assigned with special issues: '0' and '1'.

The word pairs in the second group have a strong syntactic relation (high syntactic similarity) between the two involved items. With powerful syntactic checking measurements, the semantic relation between a pair of words could be detected automatically. Just as shown in the example "students" and "student", it is obviously that they convey the same semantic meaning (within the context of model transformation domain). The purpose of differentiating this group is to avoid unnecessary semantic checking steps. Considering the usage of equation (5), the "S_SSV" for such a pair of words should always be '1', no matter what values are assigned to the impact factors "SeV_weight" and "SyV_weight". Actually, in AMTM, syntactic checking is divided into two steps; the first step is trying to use syntactic checking to define potential semantic relations between two words (This part is detailed in chapter six).

The word pairs in the third group have both semantic relation and syntactic relation between the two words involved. Both of the syntactic checking measurements and semantic checking measurements should be applied on such pairs, and work together to generate a "S_SSV" for them. As shown in the example "teacher" and "person", it is obviously that they have a weak syntactic relation but a neutral semantic relation (a teacher must be a person, but a person may not be a teacher). Normally, to determine the relation between a pair of words, semantic relation influences more than syntactic relation. Depending on different application domains, the influence degree might be a little different. Considering the equation (5), two impact factors "SeV_weight" and "SyV_weight" could be assigned with the values of '0.8' and '0.2' or '0.9' and '0.1', respectively. The basic assigning principle is: the value of "SeV_weight" is greater than the value of "SyV_weight" and the sum of the two values is '1'.

IV.6 Conclusion

This chapter presents the S&S checking mechanism involved in AMTM. In order to develop an automatic model transformation methodology, semantic and syntactic checking measurements are two key issues to be involved.

In some terms, semantic and syntactic checking could be executed automatically with the help of IT domains; in this way, huge amount of manual efforts could be replaced. In AMTM, semantic and syntactic checking measurements work as a whole, but they are independent to each other in some aspects. Normally, the two checking measurements are applied between a pair of words. To fulfill the gap between word sets and model items, AMTM builds a bridge by defining a MMM and S&S comparing mechanism (four equations defined in the third chapter). In AMTM, a value is generated to stand for the S&S relation between two model items. In order to differentiate matching pairs, potential matching pairs and un-matching pairs, matching pair choosing mechanism is also presented in this chapter. Furthermore, the matching pair choosing mechanism also guarantees the many-to-many matching relation between the items coming from source meta-model and target meta-model, respectively. Finally, a simple use case of doing S&S in AMTM between word pairs is given. This use case illustrates the idea that S&S checking should be executed differently based on the specific word pairs. Consequently, the impact factors defined in equation (5) might be assigned with different pairs of values with small range changes. Normally, the values assigning part could be done intuitive.

This chapter presents the basic theoretical idea for doing S&S in AMTM and the relevant issues about S&S. The detail of doing semantic checking and syntactic checking will be illustrated in details in the following two chapters, respectively.

Chapter V: Semantic checking measurements involved in AMTM

V.1 Introduction

In AMTM, semantic checking can be regarded as a process of detecting the semantic meanings of words (e.g. elements' and properties' names); one specific word may have several semantic meanings that are defined by people or defined by its context. Also within the context of AMTM, semantic checking measurements stand for the methods of comparing semantic meanings and building semantic relations between specific items, which contain semantic issues. Such an item could be directly a word (in any languages), something derived from words and some special marks (e.g. math symbols, modeling elements).

In AMTM, as the main objective concerns detecting of potential model transformation mappings, semantic checking measurements are used to detect semantic meanings carried by model items (elements and properties) and to build semantic relations between model items. Semantic checking measurements involved in AMTM aim at comparing semantic meanings conveyed by items that come from source meta-model and target meta-model, respectively. The potential matching items between the two meta-models should have same (or similar) semantic meanings or semantic representations. This chapter focuses on illustrating the mechanism of semantic checking measurements involved in AMTM. The position of this chapter in the whole thesis is shown in Figure V-1.

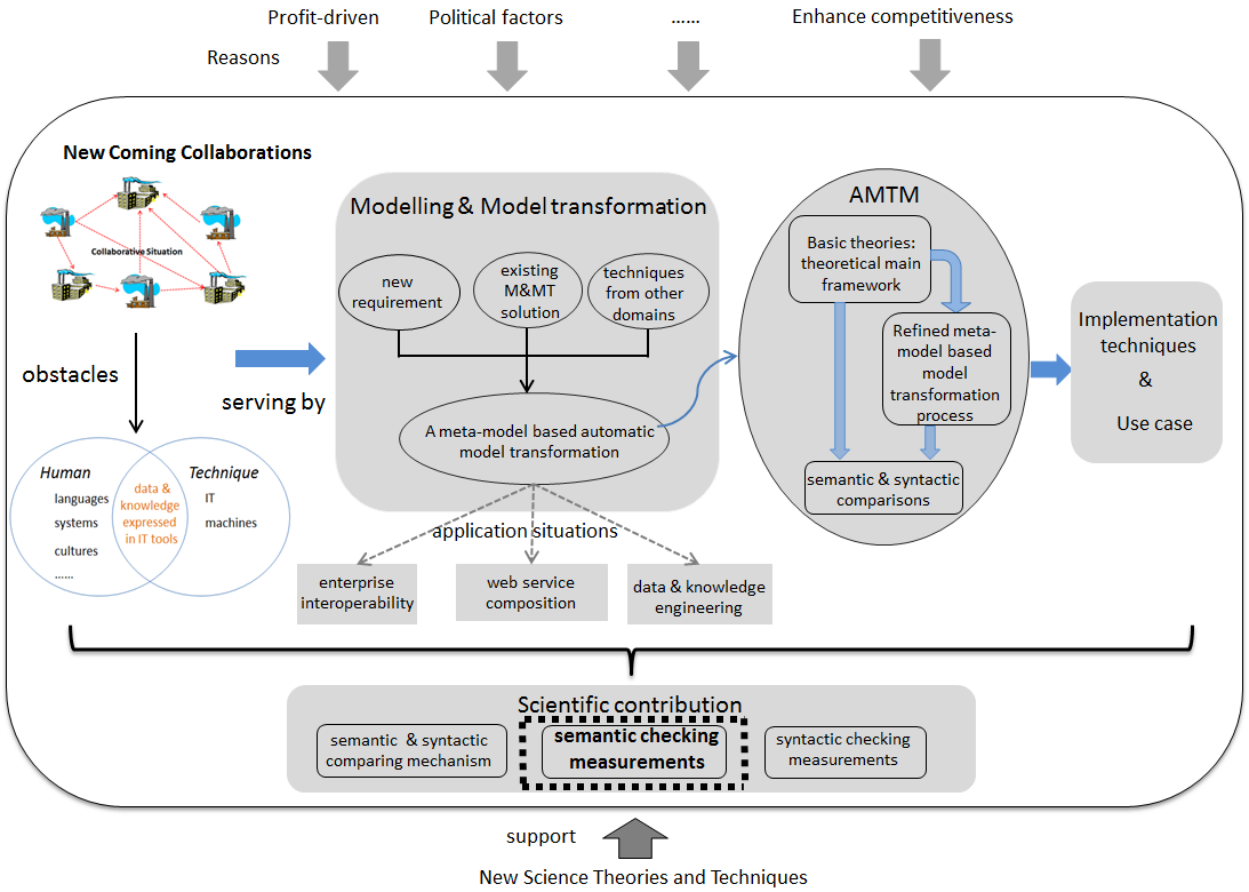


Figure V-1: The position of chapter five to the thesis.

Semantic checking mechanism involved in AMTM is regarded as one of the scientific contribution proposed by this thesis. AMTM adjusts semantic checking measurements to be suitable to model transformation domain; furthermore, specific software tool is developed for AMTM.

This chapter is divided into six parts. The second section presents the basic requirement of doing semantic checking measurements: a significant semantic thesaurus. This section divides into two subsections: (i) a general semantic thesaurus "WordNet" (Fellbaum, 1998), which provides the original data for AMTM semantic checking, and (ii) the semantic thesaurus that is built particularly for AMTM (AMTM_ST). On the basis of AMTM_ST, the third section illustrates the mechanism of the detecting process for semantic relations existing between potential mappings items. The semantic relations to be detected are divided into two kinds: simple semantic relation and iterative semantic relation. The fourth section shows simple use case of the detecting process for the two kinds of semantic relations. A possible improvement point for enhancing AMTM_ST is presented in the fifth section. At last, a conclusion of this chapter is given.

V.2 Basic requirement of doing semantic checking

A word may have more than one semantic meaning; to determine the exact semantic meaning of one word, the context of the word should be taken into consideration. There exist different relations among different semantic meanings, such as: **similar**, **hypernym** and **antonym** (such semantic relations are explained in the second subsection of this section). To do semantic checking, a powerful semantic thesaurus is necessary. The semantic thesaurus should at least contains semantic meanings of majority words (here, we only consider English words). A better semantic thesaurus should also provide semantic relations building among different semantic meanings. AMTM_ST, which is adopted from "WordNet", is built for AMTM.

V.2.1 Semantic thesaurus "WordNet"

"WordNet" is a lexical database for English, which could be regarded as a large electronic dictionary. It contains huge amounts of **simple words**, **phrasal verbs** and **idioms**. Comparing to the traditional paper dictionaries, which entries have to be arranged according to their spelling (and thus, to some extent, their pronunciation), entries in WordNet are organized in terms of their semantics. Specifically, words in WordNet that are similar in meaning are interlinked by means of pointers that stand for a semantic relation. Formally, WordNet is a semantic network, an acyclic graph (Fellbaum, 1998). WordNet has been employed as a resource for many applications in natural language processing and information retrieval.

WordNet is a widely used machine-readable lexicon in natural language processing. Word semantic senses are organized as a synonym set, which is called "**synset**"; every synset consists of a list of synonymous word forms. Here, a word form can be *a single word*, a *phrasal verb* and *an idiom.* Furthermore, semantic relations that describe relationships are built among these synsets. WordNet divides words in four groups: *nouns*, *verbs*, *adjectives* and *adverbs*. Totally, there are **155 327 words** that have **207 016 different word senses**; these word senses belong to **117 597 different synsets**. Table V-1

shows the number of words, word senses and synsets that are divided in four groups: noun, verb, adverb and adjective.

Table V-1: Number of words, word senses, and synsets stored in WordNet 2.1 (Huang, 2007)

\	words	word senses	synsets
noun	117 097	145 104	81 426
verb	11 488	24 890	13 650
adjective	22 141	31 320	18 877
adverb	4 601	5 720	3 644
total	155 327	207 016	117 597

To connect different synsets, many semantic relations are built among them. The details of these semantic relations are shown in Table V-2.

Table V-2: Semantic relations maintained in WordNet 2.1 (Huang, 2007)

Semantic relation	noun	verb	adjective	adverb
hypernym	75 134	13 124	-	-
hyponym	75 134	13 124	-	-
instance hypernym	8515	-	-	-
instance hyponym	8515	-	-	-
part holonym	8874	-	-	-
part meronym	8874	-	-	-
member holonym	12 262	-	-	-
member meronym	12 262	-	-	-
substance holonym	793	-	-	-
substance meronym	793	-	-	-
attribute	643	-	643	-
domain category	4 147	1 237	1 113	37
domain member category	6534	-	-	-
domain region	1 247	2	76	2
domain member region	1 327	-	-	-
domain usage	942	16	227	73
domain member usage	1258	-	-	-
entail	-	409	-	-
cause	-	219	-	-
also	-	589	2 683	-
verb group	-	1 748	-	-
similar-to	-	-	22 622	-
antonym	2 142	1 089	4 080	718
derivation	35 901	23 095	12 911	1
participle	-	-	124	-
pertainym	-	-	4 852	3 213
total	265 297	54 652	49 331	4 044

The semantic relations that are maintained in WordNet are shown in Table V-2. For different words groups (i.e. noun, verb adjective and adverb), different semantic relations have been defined. For example, **hypernym and hyponym** are two corresponding semantic relations that are defined on nouns group and verbs group. The majority semantic relations maintained in WordNet are on three word groups: noun, verb and adjective. Furthermore, among the three groups, noun group has the most semantic relations.

Since not all of these semantic relations are adopted in AMTM, we do not explain the details of each semantic relation (e.g. meanings, examples and possible application domains) in this thesis. The details of WordNet (including these semantic relations) are stated in (Fellbaum, 1998).

As a short conclusion, WordNet is a generous semantic thesaurus. It contains large amounts of words, their semantic senses and the semantic relations among all these senses. It could provide solutions to semantic checking problems for all the domains (not focusing on specific engineering domains).

Model transformation is a specific engineering domain. Since semantic checking measurements play a key role in defining automatic model transformation methodology, a specific semantic thesaurus should be created for model transformation domain. In AMTM, a specific semantic thesaurus AMTM_ST is built; WordNet has been chosen to use as the basis to AMTM_ST. There are two main reasons for building AMTM_ST.

- **Improving efficiency of semantic checking process**: WordNet contains huge content; model transformation domain needs only parts of this content. Less content stores in semantic thesaurus could save the time of accessing and searching the exact information involved in.
- **Suitable to serve model transformation domain**: model transformation domain mainly focuses on noun, verb and adjective word groups; furthermore, model transformation focuses only on several main semantic relations existing among these word groups. WordNet defines and maintains too many semantic relations to be used by model transformation domain.

For the two main reasons, AMTM_ST is created in AMTM; the detail of this thesaurus is illustrated in the next subsection.

V.2.2 AMTM Semantic thesaurus: AMTM_ST

The mechanism of doing semantic checking in AMTM has been illustrated in the former chapters and sections. This section focuses on the basic requirement of doing semantic checking in AMTM: creating a particular semantic thesaurus "AMTM_ST".

"AMTM_ST" is also a huge semantic thesaurus, which contains large amounts of words, their semantic meanings and semantic relations among these words. It is created on the base of WordNet, and it adopts the majority words and a small part of the semantic relations that are defined in WordNet. Figure V-2 shows the structure of AMTM _ST.

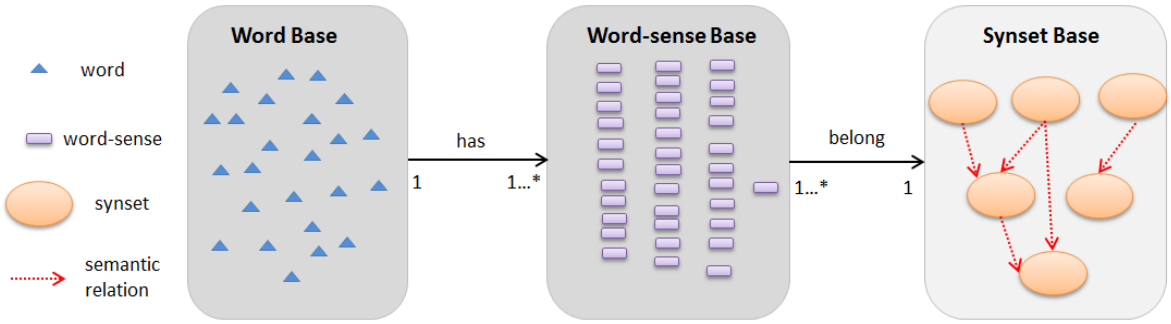


Figure V-2: Structure of AMTM _ST.

Similar to WordNet, the items stored in AMTM _ST could also be divided into three categories.

- **Word Base** contains majority normal English words (coming mainly from three word groups: noun, verb and adjective); these words have high possibility to be used in model transformation domain: building models and defining meta-models.
- **Word-sense Base** contains all the word senses owning by words stored in "Word Base"; a word could have "one to several" semantic senses. Taking the word "star" as an example; it belongs to word group: noun and verb. Totally, it has six semantic senses (meanings); as noun, it has four senses; as verb, it has another two senses.
- **Synset Base** contains many groups of word senses. A group of word senses is called "Synset" here. The word senses that are contained in one synset (a group of word senses) own synonym semantic meanings. As in WordNet, semantic relations are also built among these different synsets.

Considering the context of defining automatically model transformation mappings and rules, seven kinds of semantic relations are defined and maintained among synsets in AMTM _ST. They are **"synonym", "hypernym", "hyponym", "similar-to", "antonym", "iterative hypernym" and "iterative hyponym"**. If two words have any kinds of these seven semantic relations, they could be located in AMTM_ST.

- **Synonym:** a word or phrase that means exactly or nearly the same as another word or phrase in the same language. In AMTM_ST, all the words belong to the same synset have this semantic relation between each other.
- **Hypernym:** a word with a broad meaning that more specific words fall under; a superordinate. In AMTM_ST, this semantic relation exists between the synsets that contain nouns and verbs, respectively.
- **Hyponym:** a word of more specific meaning than a general or superordinate term applicable to it. Similar to the hypernym semantic relation, this semantic relation exists between the synsets that contain nouns and verbs.
- **Similar-to:** a word or phrase has similar meanings as another word or phrase. In AMTM_ST, this semantic relation only exists between the synsets of adjectives.
- **Antonym:** a word opposite in meaning to another. In AMTM_ST, this semantic relation exists on all the synsets of the three groups of words.
- **Iterative hypernym:** a word has an iterative hypernym relation to another. Following the rules of hypernym semantic relation, this kind of semantic relation only exists on synsets of noun and verbs.
- **Iterative hyponym:** a word has an iterative hyponym relation to another. Similar to iterative hypernym semantic relation, this kind of semantic relation also exists on synsets of noun and verbs.

Considering the context of model transformation domain "model transformation mappings should be built between same or similar concepts", six semantic relations: **"synonym", "hypernym", "hyponym", "similar-to", "iterative hypernym" and "iterative hyponym",** are defined in AMTM_ST. Since the **"antonym"** semantic relation could be detected by syntactic checking (using less resource and more efficient than semantic checking) and be used as an assistant to make model transformation mappings in AMTM, it is also defined and maintained in AMTM_ST. Four of the seven semantic relations: **"hypernym", "hyponym", "similar-to" and "antonym"** are adopted from WordNet, the other three semantic relations: **"synonym", "iterative hypernym" and "iterative hyponym"** are particularly

defined in AMTM_ST (on the basis of existing structure of WordNet) to help detect potential model transformation mappings.

In order to use these semantic relations to define model transformation mappings (to be used in the equations presented in the third and fourth chapters), a specific value (between 0 and 1) is assigned to each of the semantic relations. Table V-3 shows these semantic relations, their values pairs, and for each of the semantic relations, a concrete example of words pairs is presented.

Table V-3: Semantic relations built in AMTM _ST and their value pairs

Semantic relation	S_SeV	Example
synonym	0.9	shut & close
hypernym	0.6	person-creator
hyponym	0.8	creator-person
similar-to	0.85	perfect & ideal
antonym	-1	good & bad
iterative hypernym	0.6^n	person-creator-maker-author
iterative hyponym	0.8^n	author-maker-creator-person

For each of the semantic relations, an example of word pairs, which owns it, is shown in table V-3. The corresponding "S_SeV" (semantic value between a pair of strings: first introduced in equation (1)) value for a particular semantic relation stands for the similarity of the words pair on semantic aspect. The higher of this value means the closer of the two words in semantic dimension.

At this moment, all these "S_SeV" values are assigned directly (based on experience that is got from self-made tests); a better solution to assign these values should also consider the different domain factors and leave this task to domain expects. In the general conclusion of this thesis, the assigning values issue will be discussed in detail; furthermore, another possible method of assigning these values is presented.

Thanks to "WordNet", huge amount of words, word-senses and synsets could be collected and defined in AMTM _ST. AMTM _ST adopts the majority of the word sets, word sense sets and a small part of the semantic relations that are defined and maintained in WordNet. Table V-4 shows the quantity of three category items: words, word senses and synsets that are stored in AMTM _ST.

Table V-4: Content stored in AMTM _ST

Items	Number
words	147306
word senses	206941
synsets	114038

With the huge semantic-related content stored in AMTM_ST, AMTM provides the possibility of doing strong semantic checking measurements for model transformation domain to detect automatically model transformation mappings and rules.

V.3 Semantic relation detecting mechanism

As illustrated in the fourth chapter, semantic checking measurements take place among elements' and properties' names; the semantic relations between a potential matching pair of names are detected. In AMTM, both elements' and properties' names are regarded as normal words. Defining the semantic relations between two names is the same to define semantic relations between two words.

On the basis of AMTM_ST, AMTM defines a process to detect semantic relations between two words. The first step of doing semantic relations detection is to locate the two comparing words. Figure V-3 shows the locating part of this process.

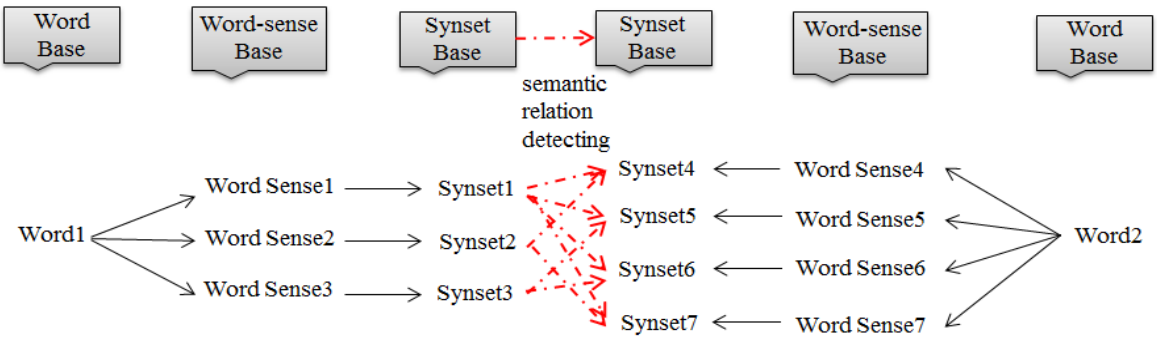


Figure V-3: Locating step of semantic relations detecting process.

Normally, a common word could be stored and located in AMTM_ST. The first step of locating process is *to find the two comparing words in AMTM_ST*. A word may have several semantic meanings (word senses stored in AMTM _ST); so the second step in this process is *to locate all the word senses that are owned by the two comparing words*. For the reason "**one word sense belongs to one synsets**", a word might belong to one or several synsets. Thus, the final step of locating words in AMTM_ST is to *form two groups of synsets that two words belong to*.

After getting two synsets groups, the next step is to detect the semantic relations that are existed among all the possible synset pairs (one from word1 side, the other from word2 side). In Figure V-3, the dash lines show those possible matching synset pairs (not all of them). One point to be stated clearly here, **the semantic relations between two words are not limited to zero or one**, there may exist several semantic relations between a specific pair of words (as one word may have several different semantic meanings).

As illustrated in table V-3, there are seven semantic relations defined and maintained in AMTM_ST. According to the detecting mechanism, these seven semantic relations could be divided into two groups: **simple semantic relations** and **iterative semantic relations**. *In the simple semantic relations group, there are five semantic relations: synonym, hypernym, hyponym, antonym, and similar-to*. The detecting mechanism of these five semantic relations is the same, and it is easy to be implemented in AMTM. *For the iterative semantic relations group, there are two semantic relations: iterative hypernym and iterative hyponym*. The detecting mechanism of the two semantic relations is also the same, but more complex than the one defined for the simple semantic relations group. The details of the two kinds of detecting mechanism are presented in the following two subsections, respectively.

V.3.1 Simple semantic relations detection

The principle idea of detecting semantic relations for the simple semantic relation group is: using *one loop* tries to find two same synsets for all the five semantic relations: **synonym, similar-to, hypernym, hyponym and antonym**, respectively. Figure V-4 is the simplification of this mechanism.

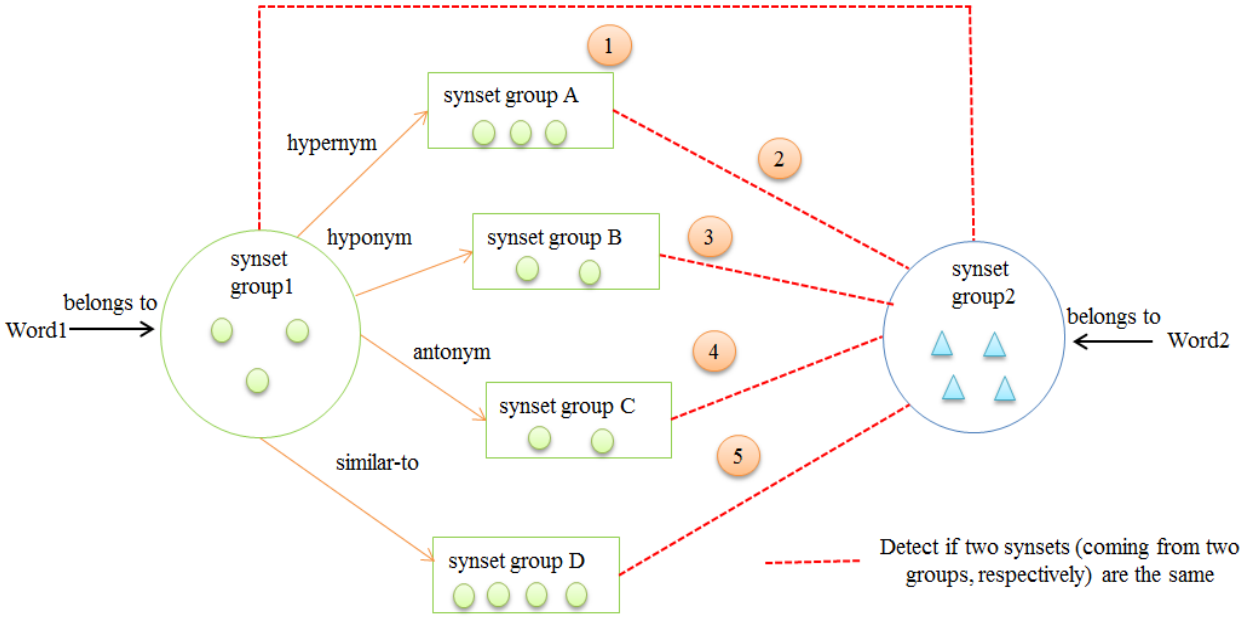


Figure V-4: Detecting mechanism of simple semantic relations.

The locating phase of detecting process has been illustrated in the beginning of this section; it assumes that for both word1 and word2, all of their synsets have been located and form two groups as: synset group1 and synset group2. All the potential semantic relations are to be found within synset pairs (one comes from synset group1 and the other comes from synset group2).

In simple word, the principle of detecting simple semantic relations is: for one word (in this case, either word1 or word2 is ok), search all the synsets that have the five kinds of semantic relations with its synsets (formed as synset group A, synset group B, etc.), then comparing if there exists one same synset in the other word's synset group (synset group 2). Figure V-4 shows five comparing loops; each loop tries to detect a specific semantic relation between the two words.

V.3.2 Iterative semantic relations detection

For the other group of semantic relations, the detecting process of "iterative hypernym" and "iterative hyponym" semantic relations is complex. The detecting principle of the two: "iterative hypernym" and "iterative hyponym" semantic relations is same. The locating phase should be executed at first for both simple semantic relations detecting and iterative semantic relations detecting. The iterative semantic relations detecting process will be carried out only after the failure of the simple semantic relation detecting process. The principle of detecting semantic relations for this semantic relation group is: locating iteratively the synsets that have hypernym or hyponym

relations with word1's synsets and comparing with the synsets that word 2 belongs to, in order to find two same synsets. This process is shown clearly in Figure V-5.

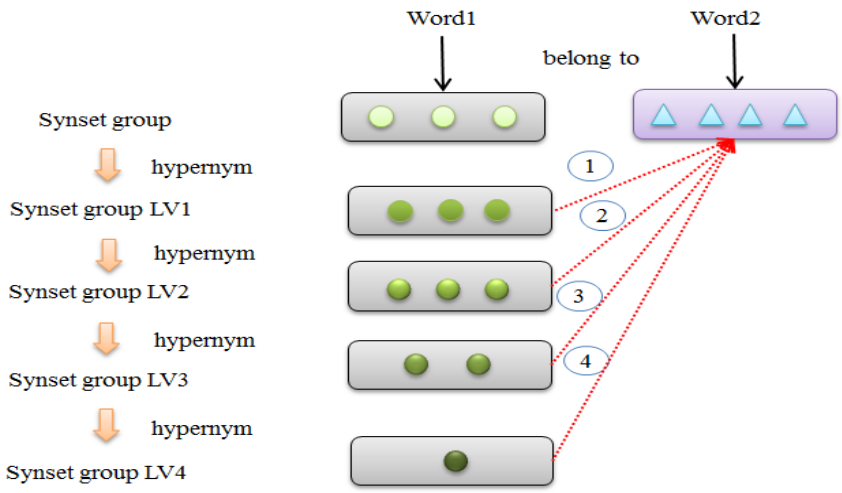


Figure V-5: Iterative semantic relations detecting process illustration.

Considering the efficiency issue and the strength of semantic relations between two words, the iterative loop will be executed at most four times. If there is no semantic relation being found after four iterative comparisons, we regard that the two words have no iterative semantic relations between each other. Then the "S_SeV" between them might be "0" in the context of AMTM.

The main difference between iterative semantic relations detecting and simple semantic relations detecting is the iterative synsets searching process. Moreover, the calculation rules for "S_SeV" values of the two groups' semantic relations are also different to each other (assigned directly or calculated by applying predefined formulas).

V.4 Simple use case

In this section, the semantic relation "iterative hypernym" is taken as an example to show the structure and working mechanism of AMTM_ST. As illustrated above, semantic relations are not built directly between different words in AMTM_ST; they are built among synsets that words belong to. All the items in the use case shown in this section are synsets, which are represented by the words that belong to them. The iterative semantic relations among several synsets are defined and maintained in AMTM_ST as shown in Figure V-6.

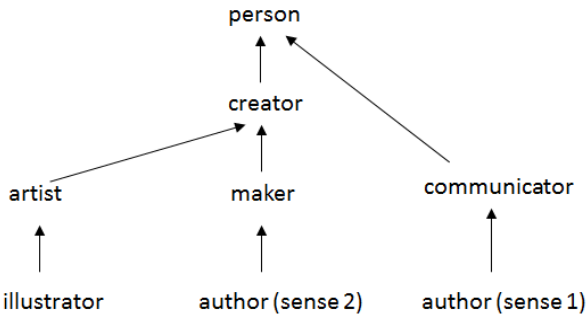


Figure V-6: Iterative hypernym semantic relations illustration.

In Figure V-6, there are eight synsets and fourteen hypernym (seven of the fifteen are iterative hypernym semantic relations) semantic relations are defined among them. Based on the semantic relations and their values pairs that are defined in Table V-3, Table V-5 is generated for this example.

Table V-5: Semantic relations and values detected in this use case

Word 1	Word 2	Semantic Relation	Semantic Value
person	creator	hypernym	0.6
person	communicator	hypernym	0.6
person	artist	1 iterative hypernym	0.36
person	maker	1 iterative hypernym	0.36
person	author (sense 1)	1 iterative hypernym	0.36
person	illustrator	2 iterative hypernym	0.216
person	author (sense 2)	2 iterative hypernym	0.216
creator	artist	hypernym	0.6
creator	maker	hypernym	0.6
creator	illustrator	1 iterative hypernym	0.36
creator	author (sense 2)	1 iterative hypernym	0.36
artist	illustrator	hypernym	0.6
maker	author (sense 2)	hypernym	0.6
communicator	author (sense 1)	hypernym	0.6

All the semantic relations mentioned in Table V-3 have been defined and maintained in AMTM_ST; the implemented algorithms just to search for them (the specific comparing words pairs are used as the input of the implemented algorithms). Another point to be mentioned in this example is: hypernym and hyponym are two opposite semantic relations. Applying this example again, "person" to "creator" is hypernym relation while "creator" to "person" is hyponym relation. Considering the context of transformation, a creator must be a person but a person may not be a creator; so, the semantic value for hyponym is higher (easier to build transformation mapping) than the value for hypernym relation.

V.5 Specific content is needed to enrich AMTM_ST

Model transformation could serve many engineering domains to solve different kinds of collaborative problems (exchanging and sharing data and information). In order to improve efficiency, an automatic model transformation methodology is required. Semantic checking, which could be used as an artificial intelligent issue, can replace the manual work from the process of defining model transformation

mappings and rules. The relationship of model transformation domain and semantic checking measurements could be seen in Figure V-7.

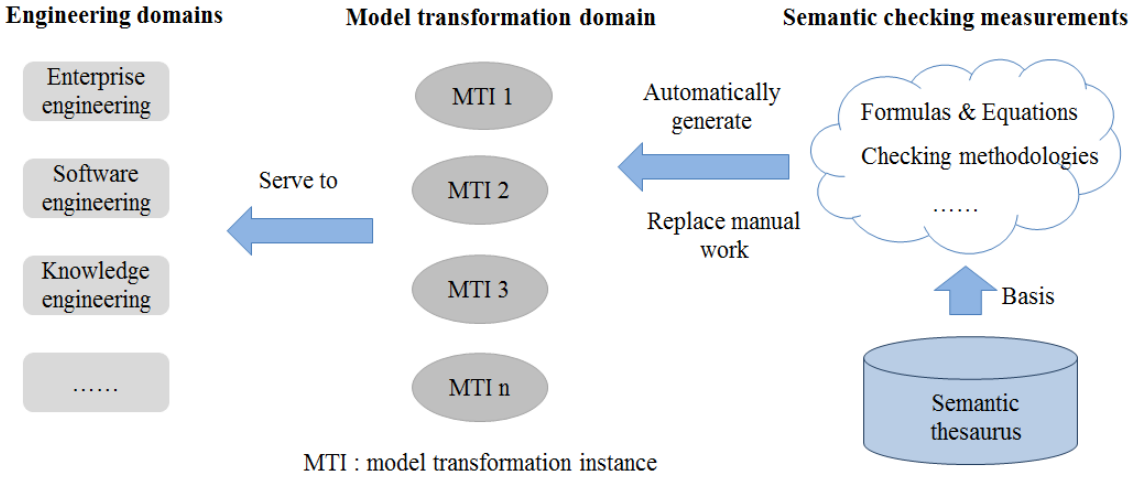


Figure V-7: Relationship between model transformation domain and semantic checking.

Semantic thesaurus is used as the fundamental basis for semantic checking; it plays a key role in the performance of the checking results. At this moment, the semantic thesaurus in AMTM "AMTM_ST" is created on the basis of a huge general semantic thesaurus WordNet. Normally, the content (e.g. words and semantic relations) stored in AMTM_ST is sufficient to deal with problems coming from different engineering domains. However, in some particular situations, such a general semantic thesaurus does not support specific domain problems. Specific engineering domains adopt special terms (words with particular semantic meanings & semantic relations) that are not stored in AMTM_ST. In order to deal with such situations, some additional content from engineering domains could be added into AMTM_ST. Figure V-8 shows this issue. It is necessary to enrich the general semantic thesaurus "AMTM_ST". There are two data sources that could be used to enrich AMTM_ST: special terms (coming from practical business of engineering domains) and special knowledge that is stored in ontologies (also categorized by domains).

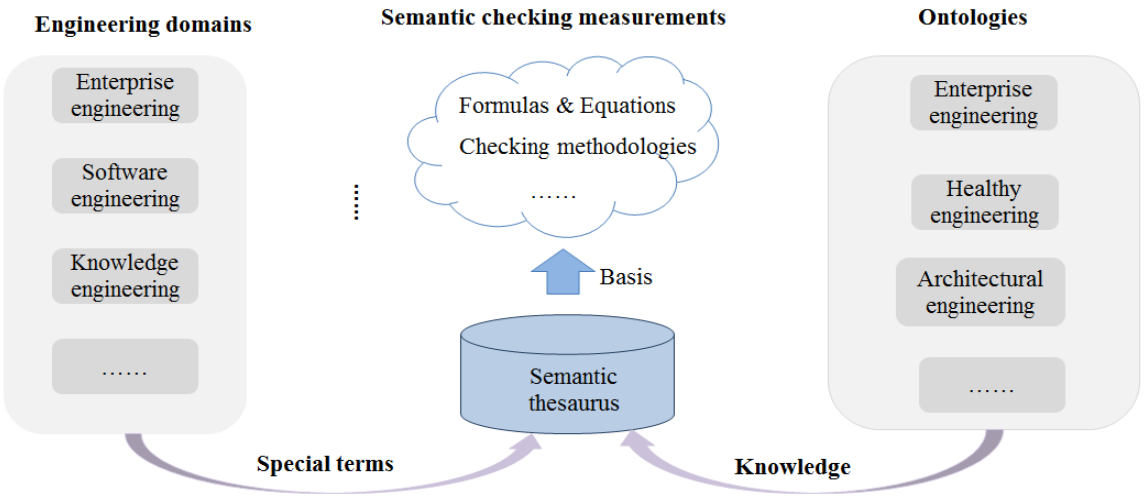


Figure V-8: Methods of enriching the general semantic thesaurus.

The idea of enriching AMTM_ST is dedicated to improve the performance of doing semantic checking. However, this idea is difficult to be carried out due to two main reasons.

- **The special structure defined in AMTM_ST**: the content stored in AMTM_ST follows the three-level structure: word, word sense and synset. The new added content must obey to this structure. However, the "special terms" and "knowledge" from specific engineering domains and their ontologies always have special structures; it is complex to transform the structures (remove unnecessary parts & add necessary parts).
- **The connection between new enriched content and old content stored in AMTM_ST**: one of the most important issues to enrich AMTM_ST is to build connections between the existing content and the new added content. But, the experts, who provide the special terms or knowledge, could not build these connections for AMTM_ST. So, the step of building connections relies on the managers of AMTM. It is a huge manual work to complete.

Since solving domain specific problems is not the purpose of AMTM the two problems mentioned above could not be solved at this moment, we just propose the idea of enriching AMTM_ST here. Another issue that could be improved for doing semantic checking in AMTM lies in the equations, formulas and checking mechanism defined to do semantic checking.

In other research or practical works, similar research works about doing semantic checking to replace manual work has been carried out. The semantic checking measurements involved in AMTM could be regarded as a summary of several of these works (considering the characteristics of model transformation domain). These similar research works are listed in references as: (Kappel et al, 2006), (Kappel et al, 2006), (Dolques, 2011) and (Shvaiko et al, 2005).

V.6 Conclusion

In this chapter, the semantic checking measurements involved in AMTM are presented in detail.

In AMTM, semantic checking measurements take place between a pair of comparing words. The comparing pair of words coming from source meta-model and target-meta model, respectively; and they stand for model items' (i.e. elements and properties defined in the MMM) names. The semantic checking measurements are created on the basis of a huge semantic thesaurus "AMTM_ST", which is built particularly for AMTM (on the basis of another general semantic thesaurus "WordNet").

Semantic checking measurements are regarded as a scientific contribution provided by this thesis. The semantic checking measurements involved in AMTM contain two main parts: (i) create the semantic thesaurus, and (ii) defining the checking mechanism according to the semantic thesaurus and model transformation context. Both the two parts have been presented in detail in this chapter. Furthermore, a simple use case is given to show how the checking mechanism works. At the fifth section of this chapter, possible improvements on AMTM semantic checking measurements are illustrated.

Chapter VI: Syntactic checking measurements involved in AMTM

VI.1 Introduction

Syntactic checking is a process of detecting the syntactic similarity between a pair of comparing words. For a specific word, it is made of a set of letters. The sequence and positions of these letters involved in this word determines the syntactic characteristic of it.

In AMTM, syntactic checking measurements are used as an assistant to semantic checking measurements. For the reason, semantic checking is a time-consuming task and syntactic checking could reveal several of the semantic relations between two words in some aspects. So, this chapter is presented just following the semantic checking measurements chapter.

Same as semantic checking measurement involved in AMTM, syntactic checking measurements defined in AMTM are also regarded as scientific contribution provided by this thesis. Several existing syntactic checking methods have been learned and adopted to generate the syntactic checking measurements for AMTM. The position of this chapter in the thesis is shown in Figure VI-1.

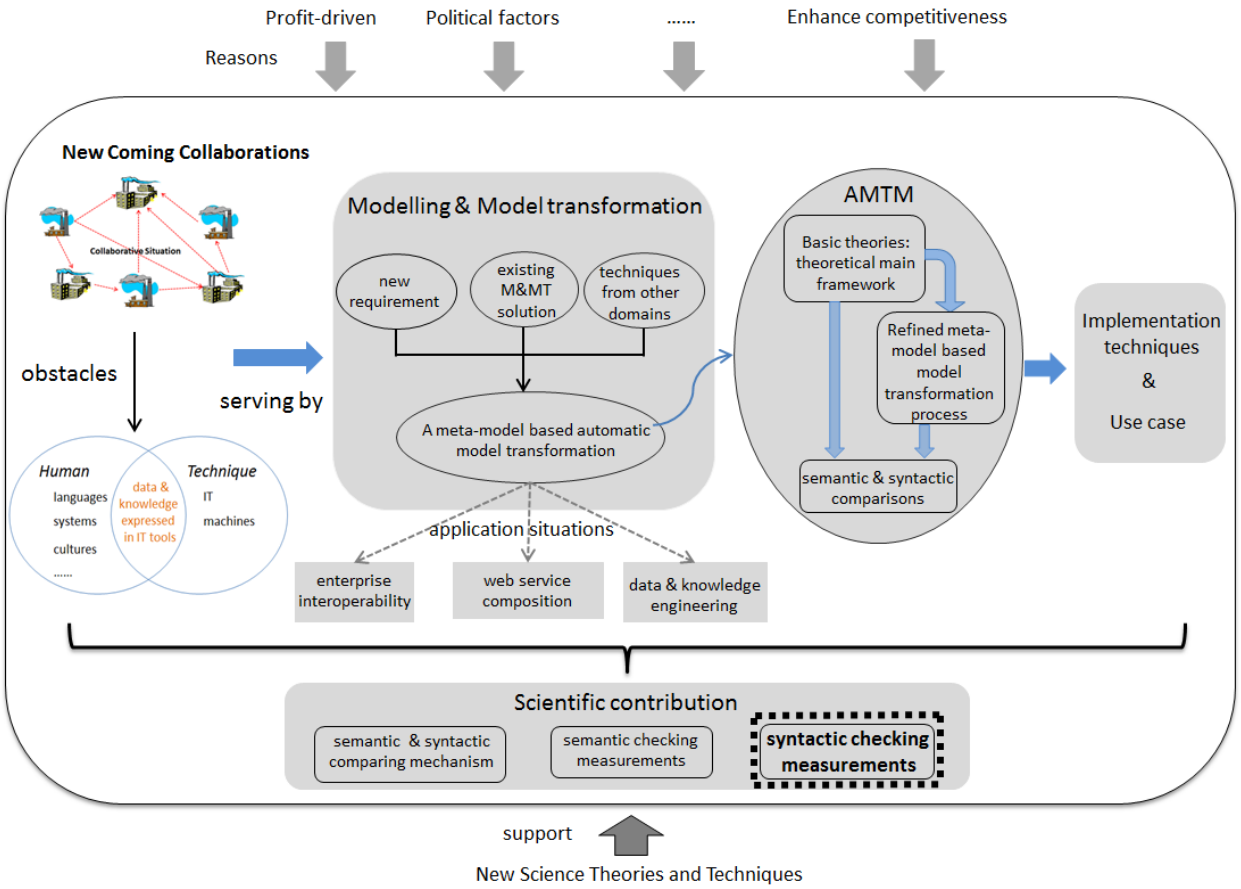


Figure VI-1: The position of chapter six in this thesis.

In AMTM, as one of the means to detect potential model transformation mappings, syntactic checking measurements are used to detect syntactic similarity carried by the names of model items (elements and properties). This chapter focuses on the mechanism of doing syntactic checking measurements involved in AMTM.

VI.2 Syntactic checking measurements in AMTM

The syntactic checking measurements involved in AMTM contain two steps: (i) test if two words that are in different forms stand for the same word (with same semantic meanings), (ii) calculate the syntactic similarity between two different words (both in forms and meanings). The two following subsections present the two syntactic checking steps in detail, respectively.

VI.2.1 Predefined syntactic checking measurements

Comparing to semantic relations, syntactic relations might be more important to build model transformation mappings in a few specific domains. Also, syntactic checking could reveal some hints of semantic relations also (with less time and high efficiency). Moreover, syntactic checking could solve some problem for semantic checking, for example: detecting the words that are in special forms, which have not been stored in the semantic thesaurus (AMTM_ST).

VI.2.1.1 Special situations focused in predefined measurements

In AMTM, syntactic checking aims several kinds of words that are in special forms. Table VI-1 shows several situations and examples of words that are in special forms.

Table VI-1: Several situations and examples of words in special forms

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	……	……	……

Table VI-1 shows clearly that AMTM syntactic checking pretreatment focuses mainly on two situations: words in plural forms and words in gerund forms. For these situations, a special profile could be built to record them; there are two ways that syntactic checking could deal with these situations, 1) comparing these words with their formal forms, and 2) modify these words to their formal forms that are stored in the semantic thesaurus (further to be detected of semantic meanings). The purpose of pretreatment is to detect the same word that shows in different forms or modify a word in special form to its original form.

In some terms, the pretreatment phase of syntactic checking step enhances the performance of semantic checking by expanding the words sets in AMTM_ST. Another possible solution to do pretreatment of syntactic checking is “applying Porter stemming algorithm” (Willett, 2006), which is illustrated in next chapter.

VI.2.1.2 applying stemming algorithm to do predefined measurements of syntactic checking

English words have several special composition methods. A majority part of these methods are concerning stemming. Usually, the stemming parts could determine the semantic meanings of the words (words with the same stemming convey similar semantic meanings). In AMTM; it seems necessary to analysis the stemming part of the words to enhance the syntactic checking part.

Lots of research works about learning and analyzing stemming have been carried out. The original stemming algorithm paper was written in 1979 in the Computer Laboratory, Cambridge (England), and appeared as Chapter 6 of the final project report (van Rijsbergen et al, 1980) and (Porter, 1980). And since then it has been reprinted in (Jones, 1997).

A fully automated alternative to truncation is provided by a "stemming algorithm" explained in (Hooper et al, 2009) and (Porter, 2001). This reduces all words with the same root to a single form, the 'stem' by stripping the root of its derivational and inflectional affixes; in most cases, only suffixes that have been added to the right-hand end of the root are removed. The removals of prefixes (i.e. strings that have been added at the left-hand end of a root) have been much less studied in the case of English-language retrieval.

In AMTM, we try to adopt "Porter stemming algorithm" (Willett, 2006) to do the syntactic checking in the pretreatment step. "Porter stemming algorithm" has two main major features.

- A significant reduction in the complexity of the rules associated with suffix removal.
- The use of a single, unified approach to the handling of context.

Rather than rules based on the number of characters remaining after removal, Porter uses a minimal length based on the number of consonant-vowel-consonant strings (the "measure") remaining after removal of a suffix. Furthermore, the Porter algorithm had become the standard for stemming English, and it hence provided a natural model for the processing of other languages.

As majority of the common stemming algorithms, "Porter stemming algorithm" also has the problem: considering comprehensive situations as a whole. To execute such an algorithm is also a time-consuming task and the potential getting result may be not good. In some terms, running "Porter stemming algorithm" may take more time than running semantic checking algorithms that are defined in AMTM. Due to this reason, involving "Porter stemming algorithm" into AMTM is just a proposal at this moment. Maybe, a better solution is "adopting part of this algorithm or using a starter edition of it".

VI.2.2 "Levenshtein Distances" algorithm

As the second step of doing syntactic checking, "Levenshtein Distances" (Gilleland, 2009) algorithm has been chosen to detect syntactic similarity by AMTM. "Levenshtein Distances" algorithm is one of the similarity metrics, which is used as one of the prominent methodology in syntactic domain, based words' syntactic checking methodology.

In information theory and computer science, the Levenshtein distance 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. It is named after Vladimir Levenshtein, who considered this distance in 1965 (Levenshtein, 1966). Levenshtein distance may also be referred to as edit distance, although that may also denote a larger family of distance metrics (Navarro, 2001). It is closely related to pairwise string alignments. Normally, "Levenshtein Distances" algorithm could be used in approximate string matching; the objective is to find matches for short strings in many longer texts, in situations where a small number of differences are to be expected. The short strings could come from a dictionary, for instance. Here, one of the strings is typically short, while the other is arbitrarily long. This has a wide range of applications;

for instance, spell checkers, correction systems for optical character recognition, and software to assist natural language translation based on translation memory. The Levenshtein distance can also be computed between two longer strings, but the cost to compute it, which is roughly proportional to the product of the two string lengths, makes this impractical. Thus, when used to aid in fuzzy string searching in applications such as record linkage, the compared strings are usually short to help improve speed of comparisons.

Mathematically, the Levenshtein distance between two strings: string a and string b with the length $|a|$ and $|b|$, respectively) is given by "$Lev_{a,b}(i,j)$"

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \mathrm{lev}_{a,b}(i-1,j) + 1 \\ \mathrm{lev}_{a,b}(i,j-1) + 1 \\ \mathrm{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

Where "1 (ai ≠ bj)" is the "indicator function"; it is equal to 0 when "ai = bj" and equal to 1 otherwise. Note that the first element in the minimum corresponds to deletion (from string a to string b), the second to insertion and the third to match or mismatch, depending on whether the respective symbols are the same.

A simple example is given below to illustrate the mechanism of using "Levenshtein Distances" algorithm to calculate the syntactic similarity between two words. This example is to calculate the "Levenshtein Distances" between two different words: "sun" and "son". The calculation process contains two steps: initiate a two-dimension table and calculate each missed values in this table based on the rules defined in equation (5). The first step "creating a two-dimension table for the two comparing words" is shown in Table VI-2.

Table VI-2: Initiate calculation table of "Levenshtein Distances"

\	son	s	o	n
sun	0	1	2	3
s	1	ABS?		
u	2			
n	3			

The two words are listed in column and line of Table VI-2, respectively; the letters (ignore cases) involved in the words are listed and marked with their positions in the words. "Levenshtein Distances" algorithm defines the rules to calculate the values to fill in the blank. The value in position "ABS?" should be calculated based on three values, the one from upper to it: "1", the one from left to it: "1" and the value in the upper left corner: "0". The concrete rule is: the value from upper and left should add "1", thus get two "2"; the two corresponding letter (from upper and left) of "ABS?" are the same "s", so the value in the upper left corner should add "0" (otherwise, add "1"). Then, the minimum value from the three (2, 2 and 0) is: "0"; the minimum value is chosen to fill in "ABS?".

Based on the calculating rules, all the blanks "ABS?" in Table VI-2 could be filled. The final result is shown in Table VI-3. The final value, which is regarded as "Levenshtein Distances" between the two

words, comes from the lower right corner of the table: "1"; it means only one operation is needed to transform the word "sun" to the word "son".

Table VI-3: "Levenshtein Distances" calculation results of this case

\	son	s	o	n
sun	0	1	2	3
s	1	0	1	2
u	2	1	0	1
n	3	2	1	1

By applying "Levenshtein Distances" algorithm, the syntactic difference between two strings (words) could be calculated, which is represented by an integer value. This integer value stands for the number of steps needed to change one string to another. In AMTM, syntactic checking methods are needed to calculate syntactic similarity between two elements' (or properties') names. AMTM adopts "Levenshtein Distances" algorithm as its main syntactic checking methodology. "Levenshtein Distances" algorithm serves as the second syntactic checking step in AMTM: to calculate the syntactic similarity between two different words. "Levenshtein Distances" algorithm only focuses on the two comparing words; the algorithm is easy to implement and takes little resource to execute. Comparing to the other syntactic checking measurements, "Levenshtein Distances" algorithm is simple, with high efficiency and also powerful. It has been used by many other research works, as a usage example, the basic theory and concrete executing process of this algorithm is stated in (Heeringa, 2004). However, "Levenshtein Distances" algorithm also has its weakness in detecting syntactic similarity (comparing with the high complexity algorithms, such as "Porter stemming algorithm"). So, in AMTM, syntactic checking part is divided into two steps. The first checking step aims at offsetting the weaknesses of "Levenshtein Distances" algorithm. Also, in order to use "Levenshtein Distances" in AMTM, equation (6) is defined. As illustrated above, "Levenshtein Distances" stands by an integer value. However, a float whose range is '0' to '1' is used to describe the similarity between two words in AMTM. Equation (6) aims at transferring this integer value ("Levenshtein Distances") to such a float.

$$S\_SyV = 1 - LD / max(word1.length, word2.length) \quad (6)$$

In equation (6), "S_SyV" stands for the syntactic similarity value between two words; "LD" means "Levenshtein distances" between the two words. The value of "S_SyV" should always be in the range of 0 to 1; the higher of this value, the higher syntactic similarity between the two comparing words. With the help of this equation, "Levenshtein Distances" algorithm is involved in AMTM. Together with the predefined treatment phase (or part of rules defined in Porter stemming algorithm), "Levenshtein Distances" and equation (6) build up the syntactic checking part of AMTM.

VI.3 Conclusion

This chapter describes the syntactic checking methodology involved in AMTM. As the semantic checking methodology involved in AMTM, syntactic checking methodology is also regarded as scientific contribution proposed by this thesis to model transformation domain.

Syntactic checking could be used in many applications that coming from different domains and several syntactic checking methodologies have been proposed and implemented. The most common usage of it is to matching entities names. In model transformation domain, in order to build potential model transformation mappings, entities' (model items') names matching is also necessary. In AMTM, syntactic checking and semantic checking work together to replace the manual work from the process of defining model transformation mappings and rules. Figure VI-2 is a simple illustration of syntactic checking parts and its relation with the semantic checking part involved in AMTM.

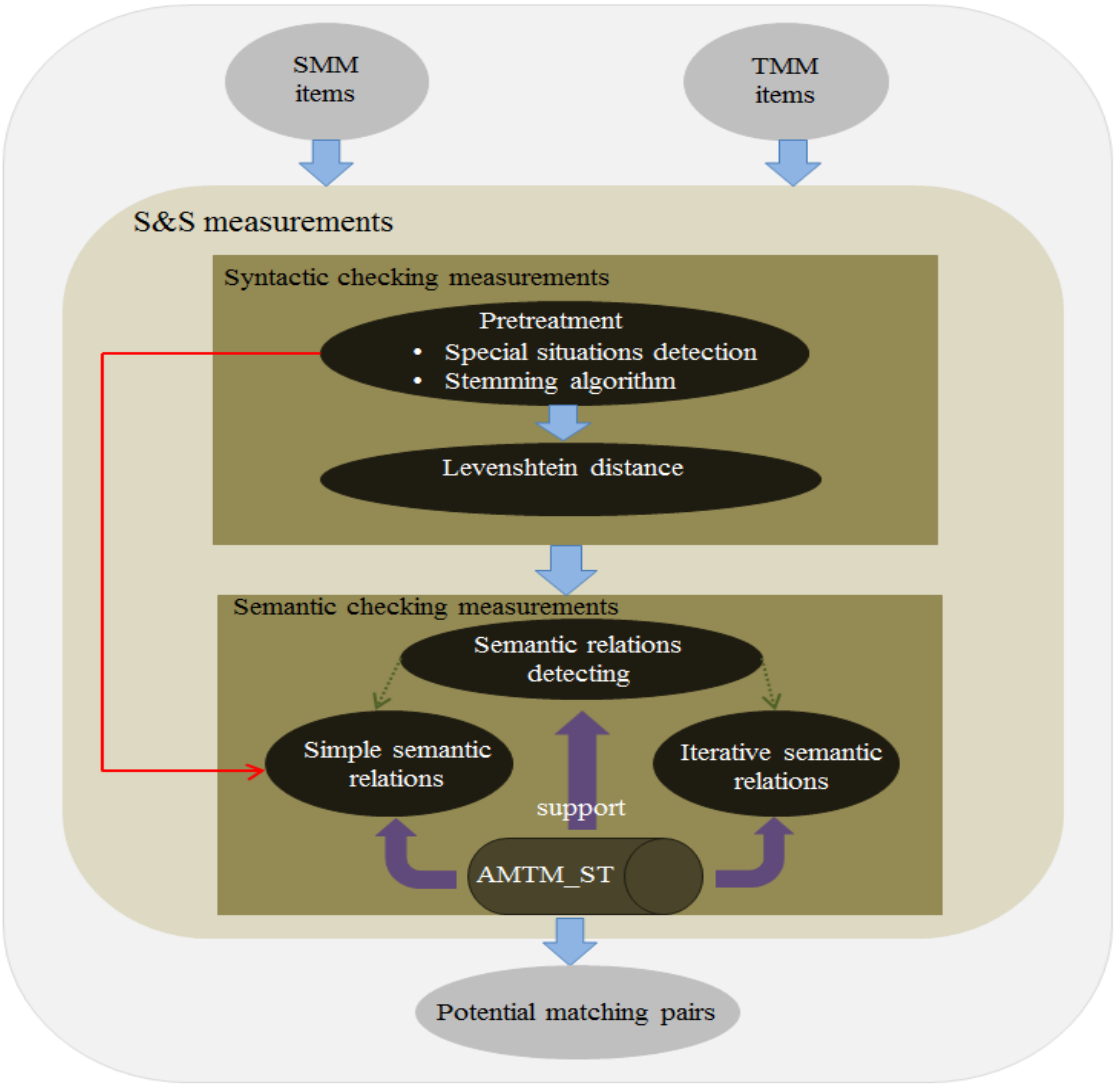


Figure VI-2: A simple illustration of syntactic checking measurements involved in S&S.

As shown in Figure VI-2, the S&S measurements involved in AMTM contains two parts: **syntactic checking measurements and semantic checking measurements**. The S&S takes source meta-model (SMM) and target meta-model (TMM) items as inputs and the outputs are potential matching pairs of these items. For the syntactic checking part, it contains two steps: **predefined treatment (pretreatment) and "Levenshtein distance" algorithm**. The first step "predefined treatment" also contains two phases: **special situations detection** and **stemming algorithms.** In AMTM, both of the two phases could discover special semantic relations (e.g. synonym and antonym) between a pair of words by applying

syntactic checking measurements. If the pretreatment step fails in discovering such kinds of semantic relations, then "Levenshtein distance" algorithm will be used to calculate the syntactic similarity between a pair of words. This syntactic similarity stands by a value and works together with the semantic relation value between the same pair of words to determine potential matching pairs.

Chapter VII: Software tool implementation & use case

VII.1 Introduction

The theoretical solution, which has been illustrated in the former chapters, requires information techniques to support; one of the purposes of AMTM "remove manual effort from model transformation process" also reveals the requirement on artificial intelligent domain. This chapter aims at illustrating briefly the process of developing such a software tool, and explaining the working mechanism and testing the working performance of AMTM by carrying out a complete use case. The position of this chapter in the thesis is shown in Figure VII-1 (the square marked by dash lines).

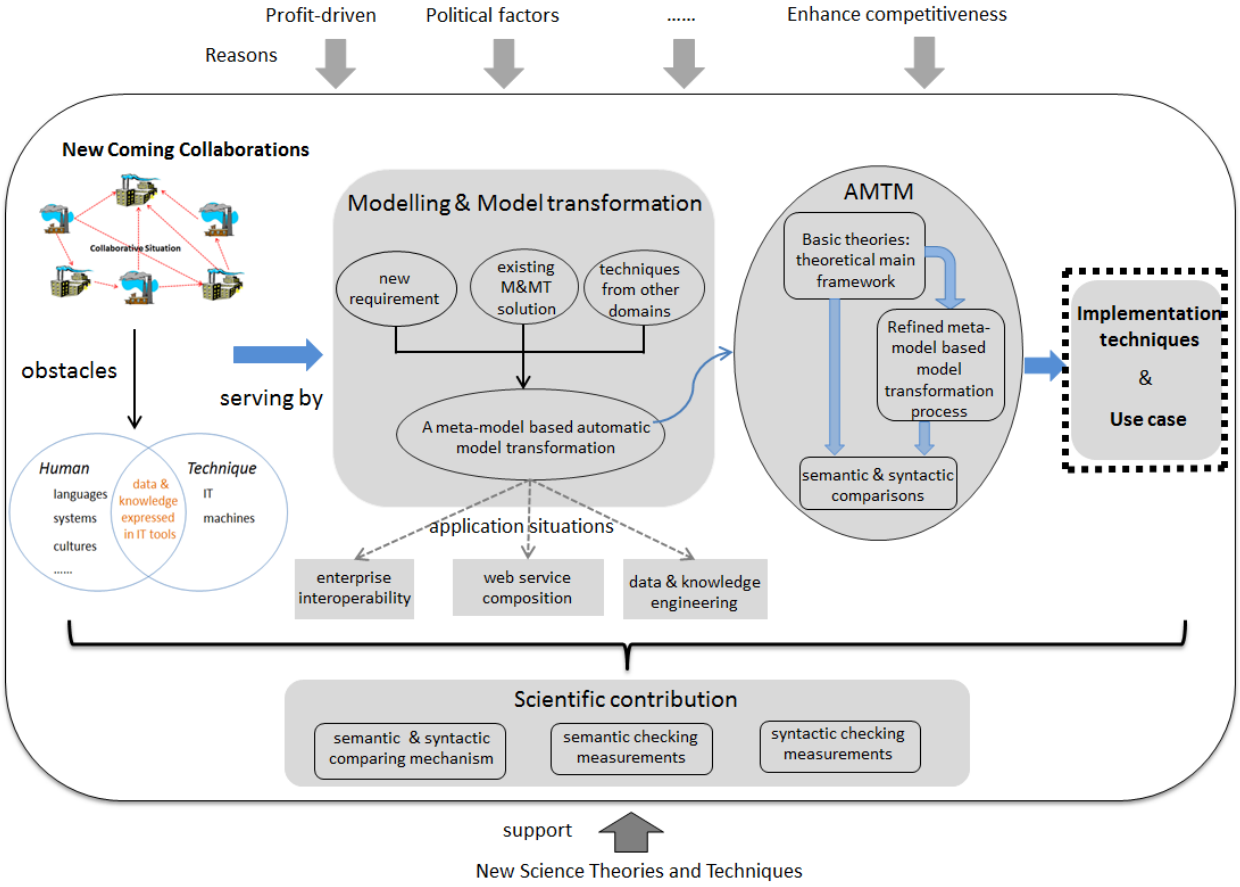


Figure VII-1: Positon of chapter seven in this thesis.

This chapter presents briefly part of the implementation phase of the theoretical solution defined in AMTM (the detail implementing process "AMTM-SS developing process" is shown in the Annex). With the help of the software tool "AMTM-SS", a complete use case is carried out. This use case shows clearly the working mechanism of AMTM; based on the testing results of the use case, the advantage and disadvantage of AMTM is summarized.

VII.2 Software tool implementation

Software engineering (Pressman, 2005) provides solutions to develop complex and huge software systems. There are several traditional software developing life cycles, such as: waterfall model, rapid prototype model, spiral model and agile model.

Normally, software developing contains four main steps (organized according to the previously mentioned developing cycles): requirement analysis, system design, coding and testing. In this chapter, we only present briefly the requirement analysis step and the design step involved in the developing process of AMTM-SS; the details of the two steps and the details of the other two implementing steps could be consulted in the annex of this thesis.

VII.2.1 Requirement analysis

There are six main functional modules that should be implemented in AMTM-SS. Furthermore, each of them could be divided into several small functional modules concerning the detail of storing intermediate results, connecting semantic thesaurus and reusing the results from other functional modules. The main function of each main functional module is explained here.

- **Models analysis module**: this module aims at dealing with users' inputs. The inputs of AMTM-SS are different kinds of model sets; AMTM builds potential model transformation mappings between two specific meta-models, which are conformed to the MMM (illustrated in the third chapter). So, there is a gap between the real inputs and the inputs required by AMTM-SS. The main task of this module is to analyze the input model sets, and based on the analyzing results to deduce the two specific meta-models required by the following functional modules. Since the input models might be built in any modeling techniques and contain different semantic and syntactic representations, the task of this module could be really huge and complex. The design and implementation parts of this module will be detailed in the following sections and in the annex.
- **Semantic relations detecting module**: this module focuses on the semantic checking part, which is created on the basis of two specific meta-models passed from the "models analysis module". Specific meta-models contain semantic representations, which are needed to be detected. Detecting and comparing the semantic representations between two models help to define the potential model transformation mappings. This module is the most time-consuming functional part, and it needs to be completed with high efficient algorithms. Furthermore, this module needs to be supported by a huge semantic thesaurus "AMTM_ST", which has been illustrated in the fifth chapter.
- **Syntactic relations detecting module**: this module performs a similar role as the semantic relations detecting module; but it focuses on the syntactic representations carried by the specific meta-models to build the potential model transformation mappings. This module implements the "Levensthein distances" (a part of Porter stemming algorithms is also involved as pretreatment) syntactic similarity checking algorithm. The efficiency of this module also determines the efficiency of AMTM-SS; so, it is necessary to consider the efficiency aspect while considering

about the powerful of syntactic checking methods used here. Different to the semantic relations detecting module, this module does not need the support from other resources.

- **Relations selecting module**: this module aims at selecting potential model transformation mappings based on the syntactic and semantic relations checking results. The syntactic relations detecting module and semantic relations detecting module generate a value for each potential matching pairs (one item from the source meta-model, the other from the target meta-model). The mechanism of choosing potential matching pairs is defined and implemented in this module. This module regards the syntactic and semantic checking results as an integrate value and set threshold values to category the relational values. This module works as a bridge that crosses the gap between semantic and syntactic checking results and model transformation mappings. Furthermore, it also combines the two checking methodologies into model transformation process.
- **Transformation mappings generating module**: this module builds potential model transformation mappings and rules based on the selecting results generated in the functional module "relations selecting". The selecting results generated in the relations selecting module might be conflict (e.g. overlap, Mutex) to each other. This module tries to simplify (solve the overlap and Mutex problems) the potential matching pairs. For instances, in the potential matching pairs, **one item from source meta-model could be matched with several items in the target mate-model**; however, according to the priority level, maybe only one matching pair is considerable. This module could reduce the interactions between AMTM-SS and its users. It is necessary to have such a functional module to replace manual efforts from the software system.
- **User interaction module**: all the functional modules presented above work on generating automatically the potential model transformation mappings. This module provides a solution to validate these automatically generated mappings. This module provides interfaces to systems users to validate and define new model transformation mappings and rules. The final validated mappings and redefined mappings will be regarded as model transformation rules; and these rules will be applied on the source model to generate the target model.

All of the six main functional modules provide the basic functions implemented in AMTM-SS. These functional modules focus on their own core functions, but they are not independent to each other. Among them, data and information are passed as parameters of functions. They work together as a whole to complete the process of detecting automatically model transformation mappings.

Besides these functional requirements, non-functional requirements also defined in AMTM-SS. The non-functional requirements concern two main aspects: user effort involved and system executing efficiency.

AMTM aims at defining model transformation mappings automatically; one of its purposes is to totally remove users' effort from the process of detecting mappings. So, AMTM-SS should reduce the interactions with system users as few as possible.

As one of the common evaluation standards, the execution efficiency for all the software systems is important. The final purpose of AMTM-SS is to serve the data and information sharing (exchanging) problems existing in collaborations. So, high efficiency is an inner requirement on AMTM-SS. The efficiency of executing AMTM-SS is mainly determined by the execution of syntactic and semantic checking measurements, which are used to replace the manual efforts. So, developing two efficient

algorithms (we are still working on this point) to do the two checking measurements could improve the efficiency of AMTM-SS.

In detail, the non-functional requirements on AMTM-SS are: (i) shielding the detecting process of model transformation mappings to the users (however, users have to be involved in the mappings rules validation part), and (ii) developing high efficient algorithms of doing syntactic checking and semantic checking.

VII.2.2 System design

This section presents briefly the system design part. The design part follows the data flow in AMTM-SS, a simple design idea is shown in Figure VII-2.

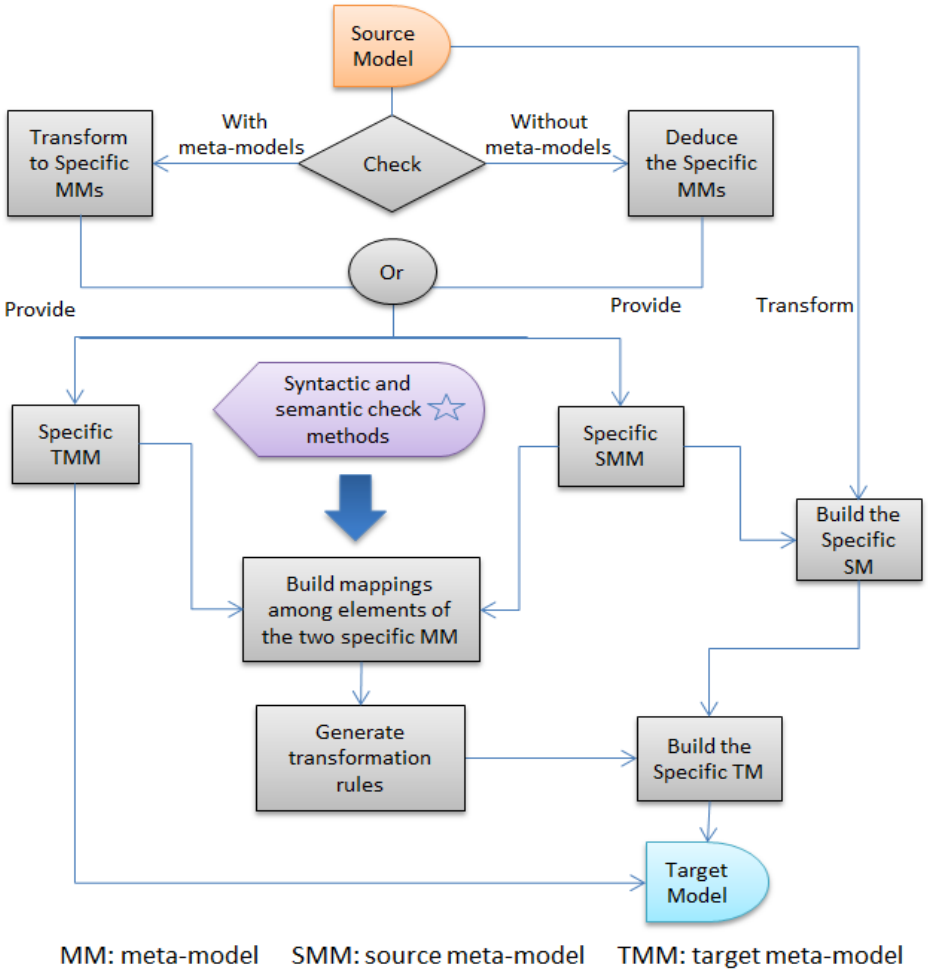


Figure VII-2: A simple illustration of designing AMTM-SS.

The software system provides three main interfaces to the system users. The first one is for uploading input model sets, the second one is used to connect with users to validate the automatically generated model transformation mappings, and the third one is used to show the final transformation results (target model) to users. The information flow in this software system has three states: information contained in

the input models, information conveyed by the specific meta-models and information carried by the target model. Figure VII-2 is created based on the information states to show the design step of AMTM-SS.

In AMTM, models are categorized by different layers (model and meta-model); the majority mapping rules should be built on the abstract model (meta-model) layer. In order to get the specific meta-models, which are conformed to the MMM, for both the source and target models, two approaches could be applied.

- Deducing from models (meta-models are not included in the input model sets).
- Transforming from meta-models (meta-models are provided by users).

Both of the two approaches can generate the specific meta-models. The specific part, which is marked with a star in Figure VII-2, concerns the core functions in AMTM-SS. It points out that the syntactic and semantic checking methods are applied on the specific meta-models; the mechanism of involving the two checking methodologies into model transformation process has been illustrated in former chapters. Based on the requirement analysis, the design of packages composition is created and shown in Figure VII-3.

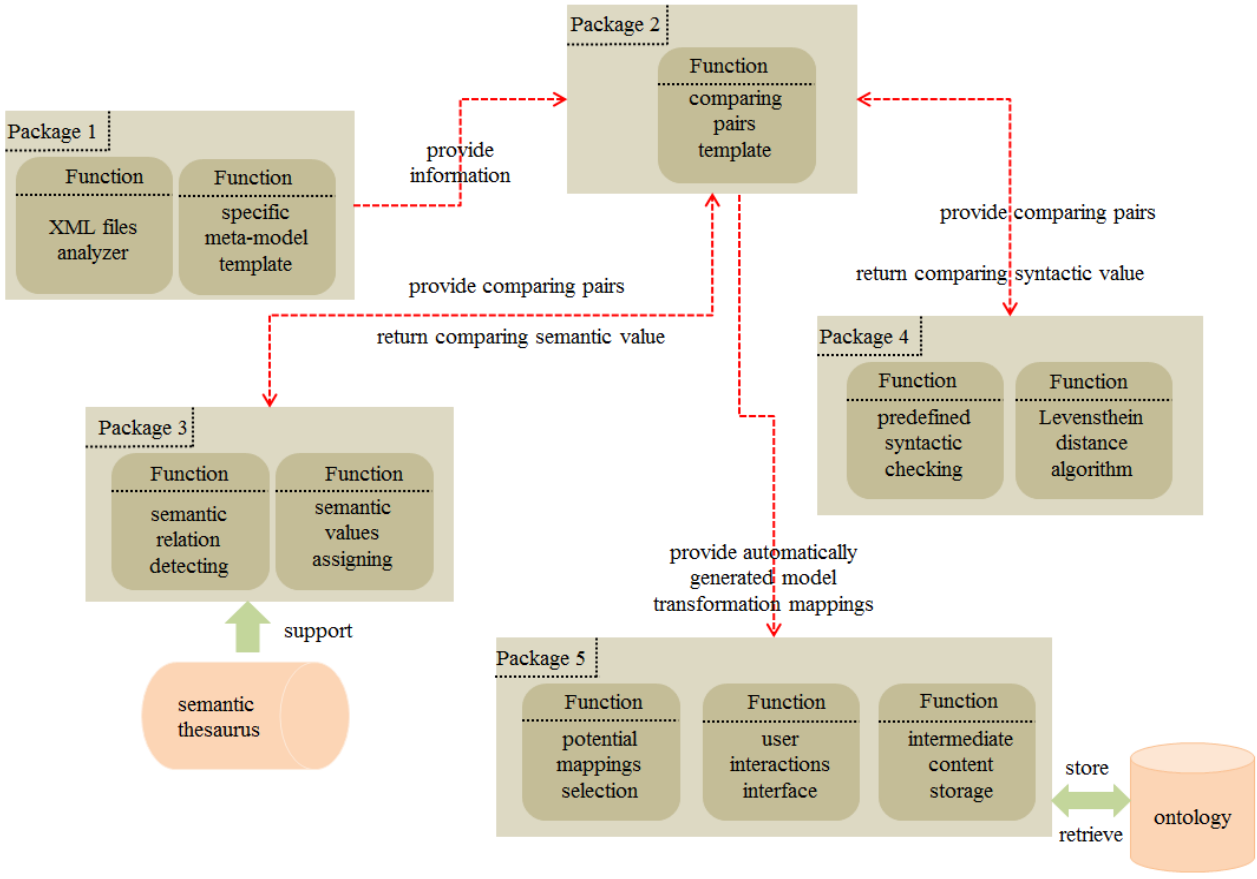


Figure VII-3: Package design illustration.

As shown in Figure VII-3, five main packages are defined in the design step. Moreover, within each main package, the important classes are clarified. The main task of each class is marked as the name of it. Here, a simple illustration of each package and its main classes is given.

- Package 1: this package contains several functional classes to **analyze the input model sets, and store the information contained in these input models to the template that are conformed to the MMM**. The classes in this package also provide an interface or method to users to upload their inputs into the software system. In simple words, this package has main tasks: (i) receive inputs from system users, (ii) analyze the inputs and extract the useful information, and (iii) store and adjust the extract information into a special format (specific meta-models conformed to MMM). All these tasks could be regarded as the preparation steps for doing automatic model transformation in AMTM. The outputs of this package are two specific models. The prepared specific meta-models would be used by the following detecting mappings steps.
- Package 2: this package **works as a bridge which crosses the gap between model transformation domain and syntactic & semantic checking measurements**. The inputs of this package are two specific meta-models generated by the first package. It provides several templates to store the potential comparing pairs (one item comes from the source meta-model, the other comes from the target meta-model). These template classes also define a comparing value and comparing functions. These comparing functions define the mechanism of calling syntactic and semantic checking measurements, and also define different mechanism of using the two checking measurements based on different comparing item pairs (i.e. element to element, property to property, and element to property). The return value of these functions is used to assign the comparing value defined in these functions. Normally, for each potential matching pair, there exists such a comparing value, which could be used to select the final matching pairs and make model transformation rules.
- Package 3: this package contains the **semantic checking measurements**. The classes (functions) to get access to semantic thesaurus are also contained in this package. The inputs of this package are two names (coming from elements and properties). After being called and receiving two names, the first step is to detect the possible semantic relations between the two names, then according to the "semantic relation and value" pairs defined in the semantic checking chapter, a final comparing value is assigned to this comparing pair of names. The output (return value) of this package is this comparing value. This value just determines a part of the final comparing value between of pair of comparing model's items (only concerns the names of the items).
- Package 4: this package performs the similar function as package 3. It focuses on the **syntactic checking measurements**. The syntactic checking methodologies: predefined treatment (part of Porter stemming algorithm) and "Levensthein distances" algorithm, are defined in this package. The inputs of this package are also a comparing pair of two names; the return value (ranges from 0 to 1) is the comparing result: syntactic similarity. As illustrated in former chapters, syntactic checking results could influence the semantic checking part by detecting semantic relations through syntactic checking; the calling mechanism of these two packages is determined by package 2.
- Package 5: this package provides the functions of **generating the final model transformation mappings and rules**. The inputs of this package come from package 2; all the potential mappings pairs and their comparing relation values. The main tasks of this package are: (i) select the useful potential mapping pairs (remove the overlap, Mutex ones), (ii) provide connecting interfaces to system users, and allow them to validate the automatically generated mappings and define new mapping rules, (iii) store the final mapping rules into ontology and apply these rules on source model to generate the target model. In order to complete these tasks, this package needs get

access to the ontology (ontology is also needed to provide knowledge to fulfill the specific parts of the target meta-model). Interactions between system users and AMTM-SS are important in this package; friendly and efficient user interface is needed here.

All the five packages compose AMTM-SS. In Figure VII-3, only the main functional classes are presented. Two outside resources to AMTM-SS are the semantic thesaurus "AMTM_ST" and the ontology "AMTM_O". AMTM-ST provides three main interfaces: connecting with system users, getting access to AMTM_ST, and getting access to AMTM_O.

VII.3 Complete use case

This section presents a complete use case to test the functions and performance of AMTM-SS. This use case aims at covering all the matching theories defined in AMTM, and using all the main algorithms implemented in AMTM-SS. This use case is shown in Figure VII-4. It simulates the model transformation process defined in AMTM among three models.

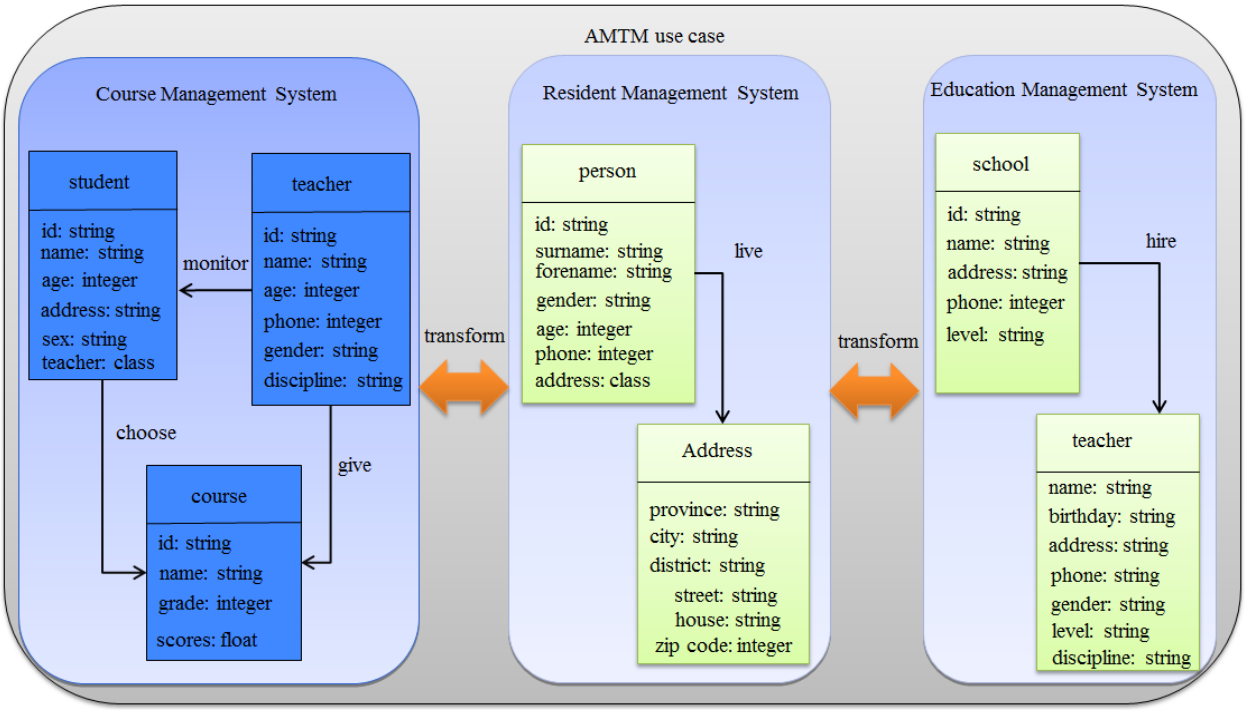


Figure VII-4: The complete use case of AMTM.

This use case contains three meta-models: course management system, resident management system and education management system. This use case tries to make model-to-model mappings and transformations between the meta-models "course management system" & "resident management systems", and between "resident management system" & "education management". The model transformation process involved in this use case is an iterative one. The "resident management system" could be regarded as an intermediate target model to test the "specific parts store and reuse" mechanism.

This use case focuses on how to use AMTM-SS to detect automatically mappings among the elements (Especially "Node" responds to the MMM) that are coming from source meta-model and target meta-model.

VII.3.1 The first model transformation in this use case

The details of each meta-models contained in this use case are shown in figure VII-4. In the first model transformation process iteration, the source meta-model contains three main concepts: student (with six properties), teacher (with six properties) and course (with four properties). All of the three concepts will be compared with the two concepts: person (with seven properties) and address (with six properties) in the target meta-model.

There is several uncertain impact parameters in the model transformation mechanism defined in AMTM; the first step of executing this use case should be assigning concrete values to these uncertain parameters. Table VII-1 shows the impact parameters and their values pairs defined for this use case. Actually, these are the default values assigned to these impact parameters; AMTM-SS also provides the mechanism of allowing users to modify them.

Table VII-1: Assigning values to uncertain impact parameters for this use case

No. Group	Parameter 1	Assign value 1	Parameter 2	Assign value 2	In Equation
1	name_weight	0.5	property_weight	0.5	1
2	pn_weight	0.8	pt_weight	0.2	2
3	en_weight	0.5	pl_weight	0.5	3
4	sem_weight	1.0	syn_weight	0	4
5	SeV_weight	0.9	SyV_weight	0.1	5

With all these assigned values, the comparing mechanism defined in AMTM could be implemented (executable) in AMTM-SS. The reason and significance of assigning these values pairs will be illustrated with the usage of each equation (defined in the third chapter) in this use case.

In the first model transformation iteration "from source meta-model 'course management system' to target meta-model 'resident management system'", in order to show the AMTM-SS working mechanism in detail, the detecting process of comparing two concepts "student" and "person" is shown step by step.

The first comparing step is to calculate the "S_SSV" between two elements' names (student and person); equation (5), which is shown below again, is used to do this step.

$$S\_SSV = SeV\_weight * S\_SeV + SyV\_weight * S\_SyV$$

Figure VII-5 is the screenshot of the equation (5) executing in AMTM-SS with "student" and "person" as two comparing words. According to figure VII-5, in AMTM_ST, the word "student" has two semantic meanings (belongs to two synsets) and the word "person" has three semantic meanings (belongs to three synsets). 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 impact factors: "SeV_weight" and "SyV_weight" in equation (5) are assigned with values as 0.9 and 0.1, respectively. The final S&S value between the two words is: 0.5903.

Actually, such a detail intermediate testing result would not be shown to the users in real use of AMTM-SS; in this use case, in order to show clearly and to make the testing result understandable to readers, we divide the whole use case into many small test cases. Within each of the particular small test case, we show the testing results and give explanations of the results.

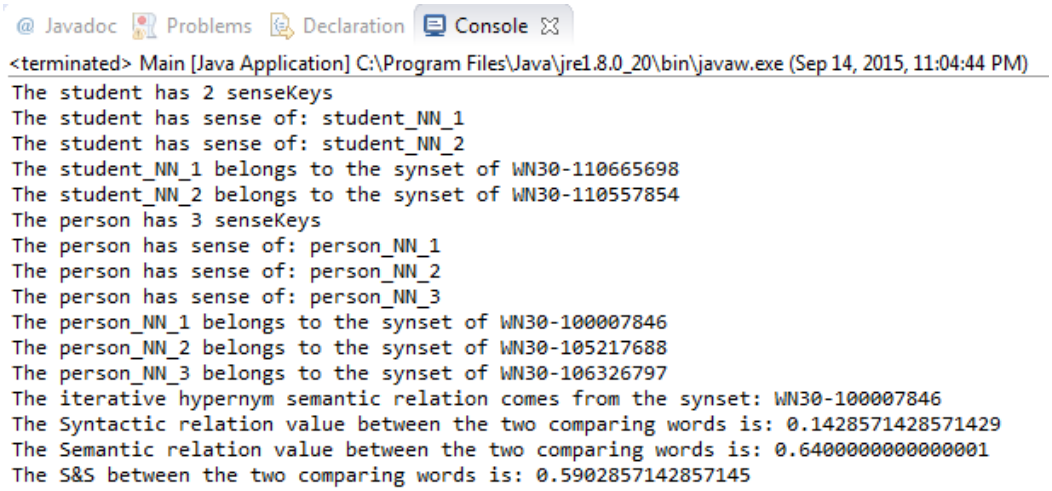


Figure VII-5: The S&S comparison executed between "student" and "person".

The possible semantic relations (and corresponding values) between the two names are shown in table V-3, which is defined in the fifth chapter. The algorithms implemented in AMTM-SS search in AMTM_ST to detect the potential semantic relations between the two names. To complete the syntactic checking part involved in equation (5) and equation (6), "Levensthein distances" algorithm is applied between the two names; for the reason, the two names do not belong to the situation that two words in different forms but stand for the same meaning.

According to equation (1), to compare two elements, both their names and properties' groups should be taken into consideration. So, the second calculating step is to calculate the S&S value on their property level. For doing this step, the comparing mechanism is defined both in equation (1) and equation (2).

When calculating S&S values on property level, both properties' names and their types are taken into account. In this use case, property's name is regarded more important than its type for the process of making transformation mappings. So in equation (2), which is shown below again, "pn_weight" and "pt_weight" are assigned with values 0.8 and 0.2, respectively. The executing results are shown is Figure VII-6.

$$P\_SSV = pn\_weight * S\_SSV + pt\_weight * Id\_type$$

As shown in Figure VII-6, to compare a pair of elements, an S&S relation value "P_SSV" is generated for each potential matching pairs of properties. In this test case, there are totally "6 (source properties) * 7 (target properties): 42" potential properties' matching pairs.

To improve the efficiency, some strategies are applied during the comparing process. When the "P_SSV" value between two properties is "1", the source property will not be used to compare with other properties from target element. To be more readable, table VII-2 is created to store all these "P_SSV" values.

With all these "P_SSV" values and the "S_SSV" value "0.5903" calculated in the first step (between two elements' names: student and person), the "Ele_SSV" between elements "student" and "person" could be calculated by using equation (1). Equation (1) is shown below again.

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

In this use case, two impact factors: "name_weight" and "property_weight" are assigned with values 0.5 and 0.5, respectively. The "element's name" and "property group" are regarded as same important to make mappings between elements. In this use case, the final "Ele_SSV" between "student" and "person" is: 0.673.

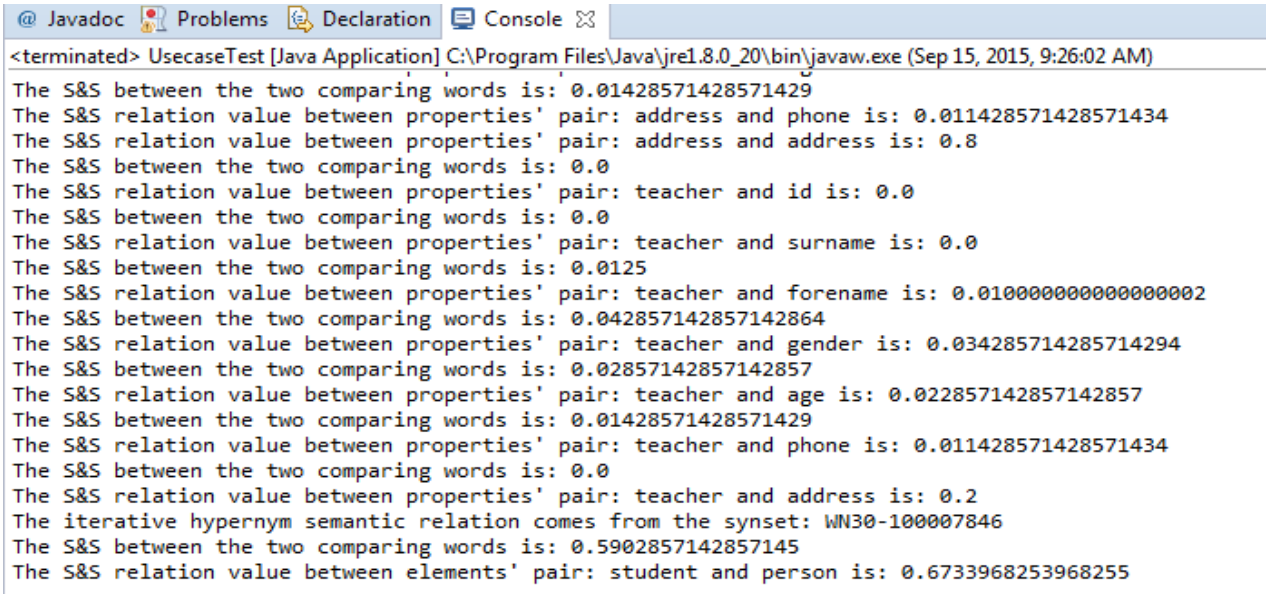


Figure VII-6: The testing results of matching on element level with two elements: "student" and "person".

The testing results shown in Figure VII-6 are stored and presented in Table VII-2. For this model transformation iteration, there are six tables that are similar to this one are created (on element level, there are six potential matching pairs, e.g. student to person, teacher to person and course to address).

Table VII-2: Comparisons on property level for this use case

student \ person	id	surname	forename	gender	age	phone	address
id	1	-	-	-	-	-	-
name	0.2	0.6777	0.672	0.2133	0.04	0.016	0.0114
age	0	0.0229	0.02	0.0133	1	-	-
address	0.2114	0.2	0.21	0.2114	0.0228	0.0114	0.8
sex	0.2	0.2114	0.21	0.8613	0	0	0.0114
teacher	0	0	0.01	0.0343	0.0228	0.0114	0.2

Following the same calculating rules and steps, all the "Ele_SSV" values for potential matching pairs of elements (coming from source model and target model, respectively) could be generated. Table VII-3 shows the final result.

Table VII-3: Potential matching pairs on element level in this use case

Course MS \ Resident MS	person	address
student	0.695	0.1439
teacher	0.7502	0.10
course	0.2585	0.243

Based on the result shown in table VII-3 and the mechanism of choosing potential matching pairs, which is illustrated in the fourth chapter, the potential matching element pairs could be chosen. Furthermore, the properties matching pairs (exist within the matching element pairs) could also be defined with the help of contents stored in Table VII-2 (a set of this kind of tables). Figure VII-7 shows the comparing results, which is got from the former matching steps.

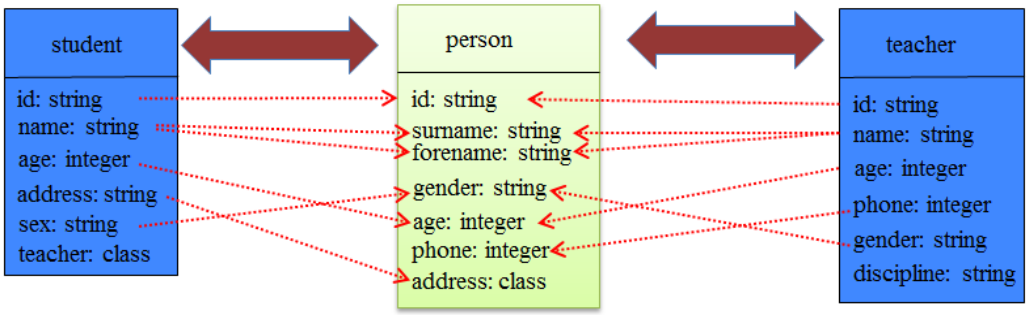


Figure VII-7: Matching results on element level in the first transformation iteration of this use case.

Each model transformation iteration phase defined in AMTM contains four matching steps: matching on element level, hybrid matching, cross-level matching and auxiliary matching. Only the first matching step "matching on element level" has been used until now. For the second matching step "Hybrid matching", the matching ideas and techniques used are the same as the ones used in first matching step. "Hybrid matching" focuses on the unmatched properties left from the first matching step. To complete it, Table VII-4 is created and fulfilled automatically by AMTM-SS. This matching mechanism of this step is defined in equation (3), which is shown below again.

$$HM\_SSV = en\_weight * S\_SSV + pl\_weight * P\_SSV$$

The two impact factors: "en_weight" and "pl_weight" are assigned with values 0.5 and 0.5, respectively. In this use case, when making hybrid matchings, the influence from element level and the property level are regarded as equal important in building mappings.

Table VII-4: Hybrid matching results in this use case

Element: Property	person: id	person: phone	……	address: house	address: zip code
student: teacher	0.2951	0.3715	…	0	0
teacher: discipline	0.3918	…	…	…	…
course: id	0.5	…	…	…	…
course: name	0.05	…	…	…	…
course: grade	0.05	…	…	…	…
course: scores	0.05	…	…	…	0.01

All the unmatched properties from source meta-model are used to compare with all the properties of target meta-model. Parts of the final result of this step have been shown in Table VII-4. Based on the comparing results and the potential matching pairs choosing mechanism, no matching pairs should be

built. Since the comparing process of this step is complex, we just take one comparing pair "student: teacher – person: id" (element: property) as example and show the executing result in Figure VII-8.

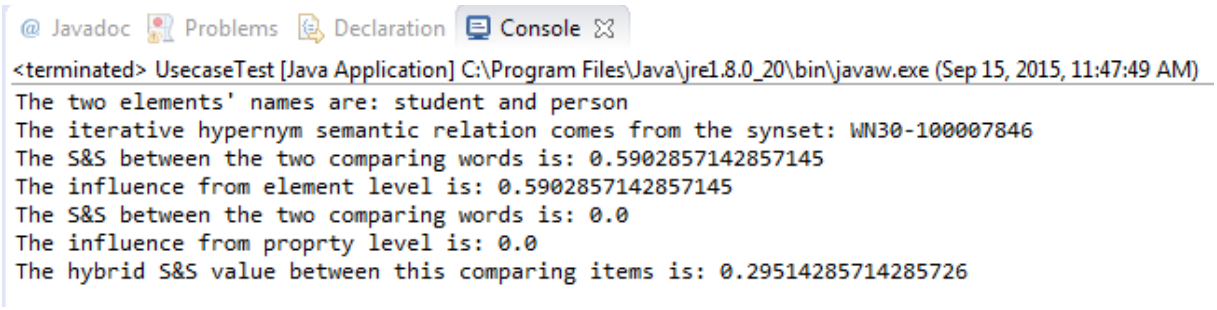


Figure VII-8: Illustration of Hybrid matching testing results.

As shown in Figure VII-8, in this small test case, the influence on element level "between two names: student and teacher" is 0.59; the influence of property level (between property "teacher" with type "class" and property "id" with type "string") is: 0. The final hybrid S&S relation value between this pair of properties is: 0.2951. According to the potential matching pairs choosing mechanism, this pair of items would not be regarded as model transformation mappings.

Within the first model transformation iteration of this use case, there are totally "5 (unmatched source properties) * 13 (all target properties): 65" comparing pairs needed to be tested in the hybrid matching step. The mechanism of comparing them is the same.

The third matching step included in AMTM is "cross-level matching". In order to make cross-level matching, unmatched elements and unmatched properties are comparing with each other. In this use case, Table VII-5 is created and fulfilled by AMTM-SS (the comparing mechanism is defined in equation (4)). In this use case, while doing the third matching step, syntactic relation between two names is ignored. The impact factor of semantic relation is assigned with value "1", and syntactic relation is assigned with value "0".

Table VII-5: Cross-level matching results in this use case

Element or Property	element: address	property: province	……	property: house	property: zip code
element: course	-	0	…	0	0
property: discipline	0	-	-	-	-
property: id	0	-	…	…	…
property: name	0	-	…	…	…
property: grade	0	-	…	…	…
property: scores	0	-	…	…	-

In this matching step, the items from same level (i.e. element to element, property to property) will not be compared with each other; in Table VII-5, the "-" is used to mark such pairs. The testing results show that there is no potential matching pairs could be built in this matching step for this use case.

The fourth matching step included in AMTM is "Auxiliary matching". This step stores specific parts (unmatched items) of source model in AMTM_O and tries to enrich the specific parts (unmatched items) in target model by extracting content from AMTM_O. In this use case, only the specific parts from source model are stored in AMTM_O; since this is the first model transformation iteration phase, there is no content stored in AMTM_O could be used to enrich the target model.

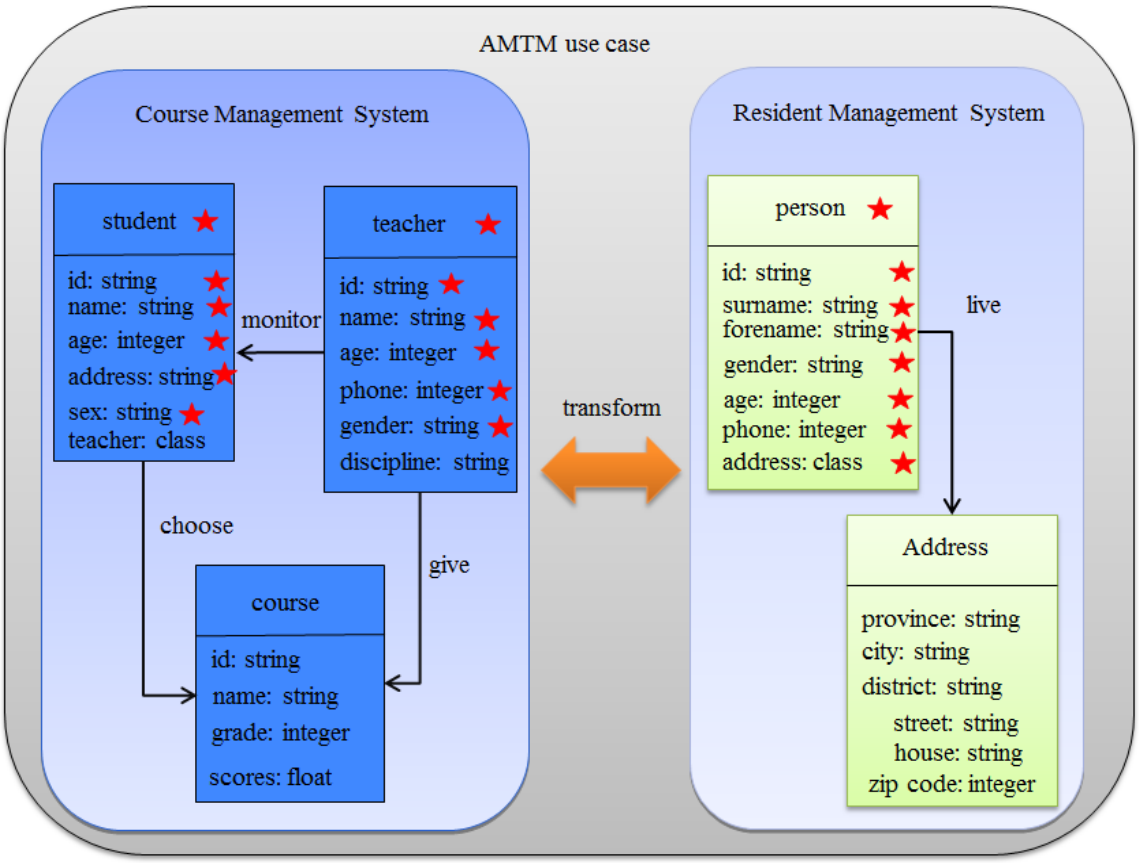


Figure VII-9: Matching result of the first model transformation iteration in this use case.

The final matching result of the first model transformation iteration in this use case is shown in Figure VII-9. In Figure VII-9, the items marked with a "star" are matched; all the other items are stored in AMTM_O and prepared to be used by the second model transformation iteration phase. It is normal that both source and target meta-model have some specific parts, which could not be matched. The significance of doing this automatic model transformation is to detect the shared (same or similar) concepts between source and target meta-models. AMTM replaces manual effort from the process of building model transformation mappings among shared concepts.

VII.3.2 The second model transformation iteration in this use case

In the second model transformation iteration, the source meta-model is the target meta-model "resident management system" in the former model transformation iteration, and the target meta-model is

"education management system", which has two concepts: school (with five properties) and teacher (with six properties).

Now, the process of the second model transformation iteration phase is presented. Comparing to the first model transformation iteration phase, the different part (also the significance) of this iteration is the possibility to test the mechanism of enriching target meta-model. The illustration of this model transformation iteration is shown in Figure VII-10.

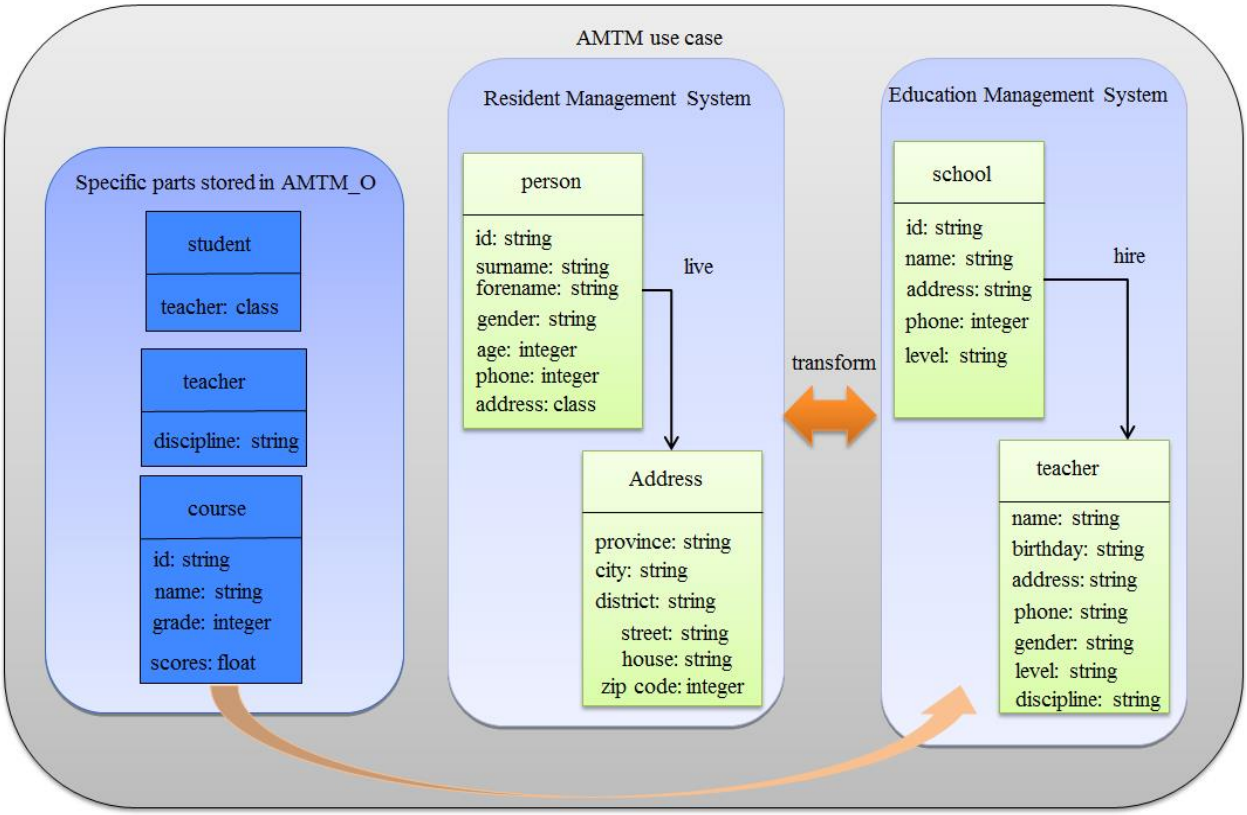


Figure VII-10: Illustration of second model transformation matching iteration in this use case.

The process of detecting potential model transformation matching pairs in this iteration phase is the same with the one of the first iteration; it follows the four matching steps: **matching on element level, hybrid matching, cross-level matching and auxiliary matching**. In this iteration, the testing results of the first matching step: **matching on element level**, are shown in Table VII-6.

Table VII-6: Potential matching pairs on element level of second iteration in this use case

Resident MS \ Education MS	school	teacher
person	0.2882	0.3284
address	0.0954	0.0989

The testing results show that element "person" and element "teacher" is a potential matching pair, since they have the maximum S&S relation value of the four potential matching pairs on element level. Figure VII-11 shows the testing results in detail (the properties matching pairs within the two elements are also generated).

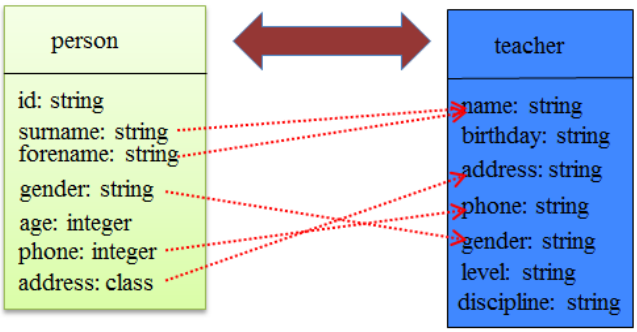


Figure VII-11: Matching result on element level of the second iteration phase.

All the unmatched properties from source meta-model after the first matching step will be used to compare with the properties in the target meta-mode in the second matching step. The comparing process of the second step "**Hybrid matching**" is similar to the one in the first model transformation iteration phase. The testing results of this step show that no potential matching pairs within this model transformation iteration.

The third matching step is **cross-level matching**; in this step, the element "address" in the source meta-model is matched with the property "address" from element teacher, which is in the target meta-model. Equation (4), which is presented and explained in the third chapter, defines the mechanism of doing this comparing process.

$$CLM\_SSV = sem\_weight * S\_SeV + syn\_weight * S\_SyV$$

In this test case, the two impact factors "sem_weight" and "syn_weight", which are shown in the equation (4) above, are assigned with values: 1 and 0. This means that in **cross-level matching**, we only consider the influence of the semantic aspect of elements' names and property names. Table VII-7 records all the comparing pairs involved in this matching step.

Table VII-7: Comparing pairs in cross-level matching step

Resident MS \ Education MS	element: school	property: id	……	property: address
element: address	-	0		**0.9**
property: id	0	-	-	-
property: age	0	-	-	-
	0	-	-	-
property: province	0	-	-	-
property: city	0	-	-	-

In this matching step, only "element to property" and "property to element" pairs are considered. Figure VII-12, shows the testing results of this matching step. The "CLM_SSV" value between element "address" and property "address" is 0.9, because they are regarded as coming from the same synset; the semantic relation between them is synonym.

The fourth matching step auxiliary matching step tries to enrich the specific parts (unmatched items) in target meta-model; for this test case, the property "discipline" in target meta-model could be enriched with the specific part (property "discipline" in meta-model course management system) stored in AMTM_O.

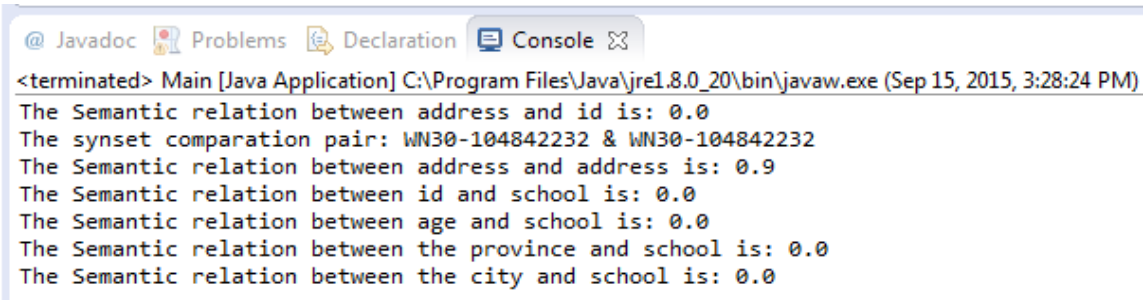


Figure VII-12: Matching result on element level of the second iteration phase.

So, after all these matching steps, the final model transformation mappings and rules are shown in Figure VII-13. The items marked with red stars are all matched.

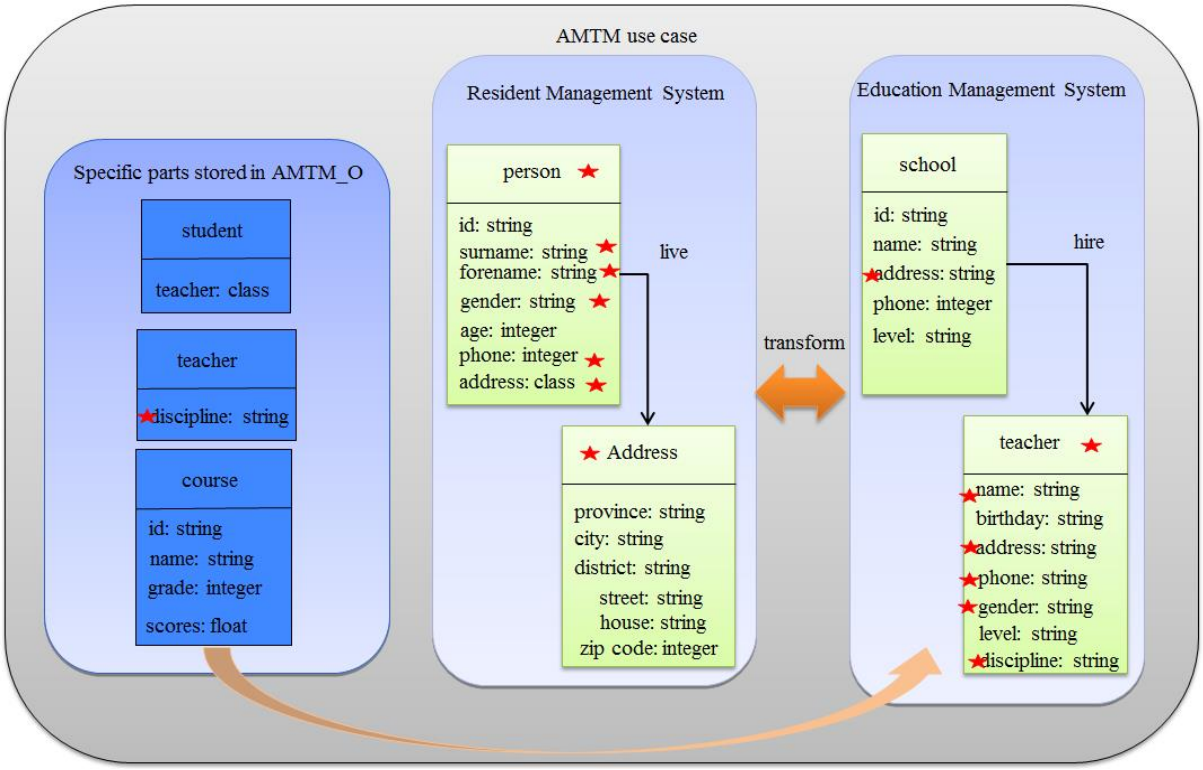


Figure VII-13: Final matching result of the second iteration phase in this use case.

As a short conclusion, this use case shows a complete example that tests all the main matching theories defined in AMTM and all the main functions (algorithms) implemented in AMTM-SS to detect automatically model-to-model transformation mappings and rules. All the equations presented in the third and fourth sections have been used in this use case. The main model transformation mappings and rules are built within the first matching step "matching on element level"; while the other three matching steps: hybrid matching, cross-level matching and auxiliary matching phases solve the granularity issue involved in model transformation process. All the three latter matching steps perform important roles in detecting potential matching pairs even the transformation rules detected in these three steps are not as many as the ones generated in the first matching step. On both element level and property level, the mappings implement "many-to-many" transformation.

This use case simulates the model transformation mappings detecting process that is executed on meta-model layer. All the model transformation mappings and rules are built between source meta-models and target meta-models. Finally, these automatically generated mappings and rules should be used on specific source models to generate the target models. So, we present a source model which is conformed to the course management system meta-model. Also, the targets models, which are generated automatically based on the model transformation rules built above, are presented. Figure VII-14 shows the source model and two target models.

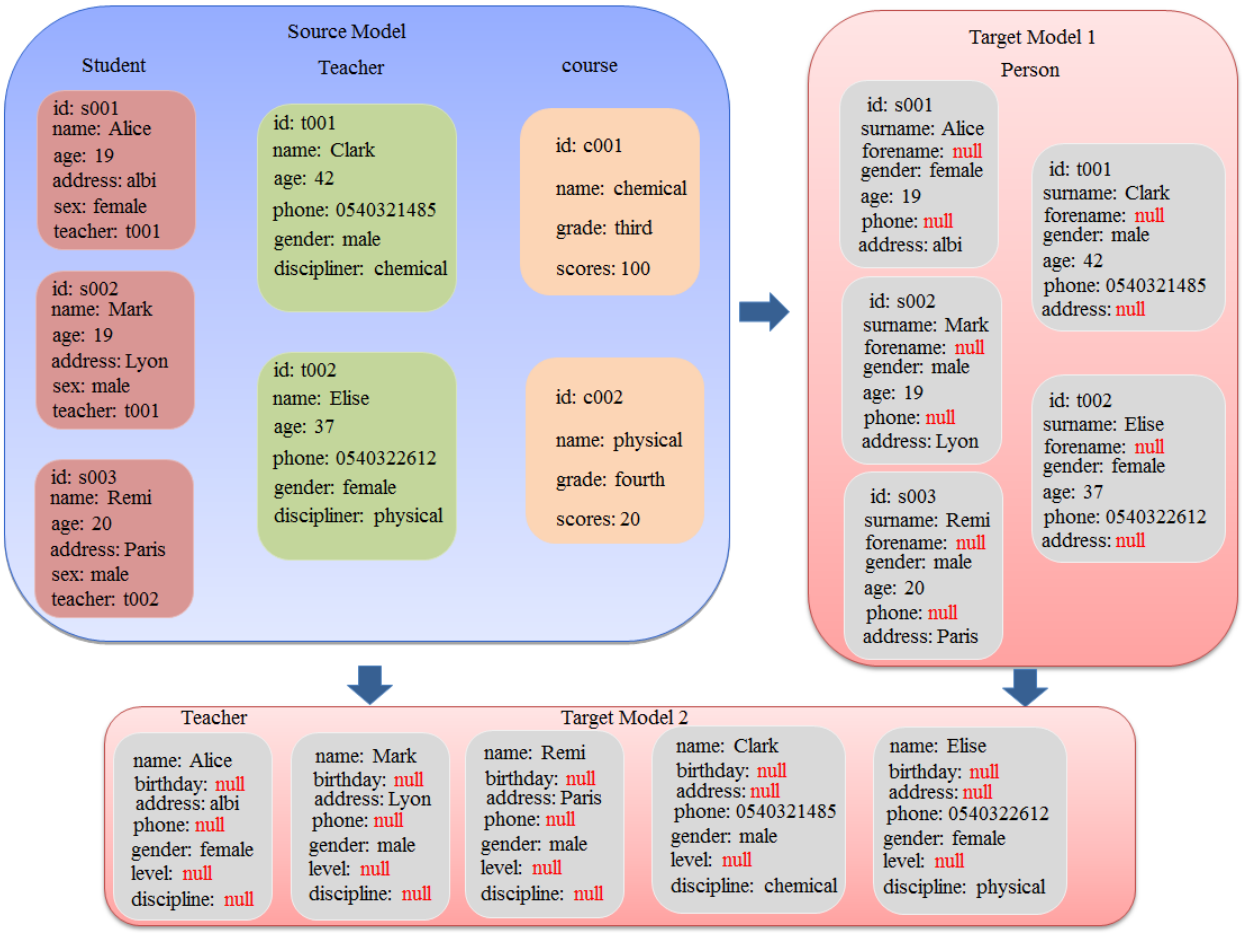


Figure VII-14: Target models generated by using the automatically detected mappings.

As shown in Figure VII-14, the "Source Model" contains seven "nodes": three "student" instances, two "teacher" instances and two "course" instances, and the relations among the seven "nodes" are maintained by their properties. All of the properties of the seven nodes: six properties of each "student", six properties of each "teacher" and four properties of each "course", have been assigned with values.

In the context of AMTM, the first model transformation iteration phase is taking in this "Source Model" and generating automatically the "Target Model 1". The automatically generated model transformation rules (the generating process, the selecting mechanism and the final selecting results) on meta-model have been illustrated above. The transformation results are: the "Target Model 1" contains five instances of "person" that are transformed from instances of "student" and "teacher" in the "Source Model" and the properties of these new instances of "person" are also partly fulfilled ("student" instances have no property of "phone" and "teacher" instances have no property of "address"). The second model

transformation iteration phase is taking in the "Target Model 1" (as source model) and generating the "Target Model 2". Based on model transformation rules presented above, the transformation results are: "Target Model 2" contains five "teacher" instances transformed from the "person" instances in "Target Model 1". Some of the properties of the new generated "teacher" instances could not be fulfilled: all the properties of "birthday" and properties of "level", parts of the properties of "phone" (originate from the "Source Model" node "student") and parts of the properties of "address" (originate from "Source Model" node "teacher"). One point to be made clearly here: in "Target Model 2", only the last two nodes "Clark" and "Elise" have the properties of "discipline. The two properties are enriched by extracting the content from AMTM_O; the extracted content is stored in AMTM_O, as specific parts of the "Source Model", in the first model transformation iteration phase.

As a short conclusion of this testing results, some elements and properties in the target model could not be transformed from the source model (or enriched by AMTM_O); the left unmatched parts require manual effort to deal with. In some aspects, AMTM could remove manual effort from defining the same (or similar) concepts between source and target meta-models. The testing result carried out for this use case is acceptable; there is one problem "the efficiency of AMTM-SS": it takes more than thirty minutes to comparing two elements (each of them with five properties). The efficiency part will be one of the focuses in the future work: the second (or the third) iteration of AMTM.

General Conclusion

This part aims at giving a general conclusion of this thesis. The content in this part could be divided into four subparts: (i) a conclusion of the content presented in this thesis, (ii) the prospect usage of AMTM, (iii) the points involved in AMTM that require to be improved; and (iv) a social perspective of applying AMTM.

Conclusion of the research work

As illustrated in the general introduction part, a large number of collaborations among heterogeneous partners are appearing and disappearing within specific periods at this moment. A typical problem brought by these collaborations is "**how to exchange and share data (information and knowledge) among the heterogeneous partners**". Concerning this problematic, this thesis presents a model-based methodology to serve it. In simple words, each of the partners involved in collaborations is a system, which could be represented by a set of models (built based on different functional point of views); furthermore, model transformations could be used to simulate the processes of connecting heterogeneous partners (especially on data, information and knowledge sharing and exchanging aspects). As the inner requirement of collaborations, the model transformation methodology adopted to serve them should be a high efficient one.

The main difficulty of defining a high efficient model transformation methodology to serve collaborations comes from three aspects: **semantic checking**, **syntactic checking** and **granularity issue involved**. Table GC-1 shows the three aspects and the coming sources of them. The detail of this table is presented in chapter one.

Table GC-1: three difficulties of defining high efficient model transformation methodology

Origin \ Difficulty	semantic detecting	syntactic detecting	granularity issue
application domains	√	√	
modeling techniques	√	√	√
model transformation domain			√

After reviewing existing solutions, this thesis presents an automatic model transformation methodology (AMTM), which answers the requirements of collaboration. This methodology combines syntactic and semantic checking measurements into model transformation process to detect the potential mappings automatically. Comparing with the existing model transformation methodologies, the aim of AMTM is "supporting to effectively build the collaborations". The involved syntactic and semantic checking measurements could **replace the manual efforts from model transformation process** (defining model transformation mappings). In order to apply the theories defined in AMTM, a software system "AMTM-SS" has been developed. AMTM-SS is used to test the working mechanism of AMTM and its performance; a complete use case is carried out and the testing results of this use case are explained.

The contribution of this thesis could be divided into two categories: **scientific contribution** and **technical contribution**.

- Scientific contribution: providing a model transformation methodology "AMTM" to support "**data (and information) exchanging and sharing issues among heterogeneous partners**". AMTM could help to **build efficiently collaborations** (e.g. enterprise collaboration, crisis management and traffic accidents management).

- Technical contribution: defining **semantic checking measurements** and **syntactic checking measurements**, and **taking them as a whole to combine into model transformation process**. To define the semantic checking measurements, a specific semantic thesaurus, which aims at serving particularly to model transformation domain, has been created. To define the syntactic checking measurements, several existing syntactic checking methods (e.g. "Porter stemming" algorithm and "Levenshtein distance" algorithm) have been adopted in AMTM considering the context of model transformation domain. Finally, to combine the two checking measurements into model transformation process, a theoretical solution has been defined in the theoretical main framework of AMTM (the MMM and five equations presented in the third chapter).

The whole structure of this thesis is shown in Figure GC-1.

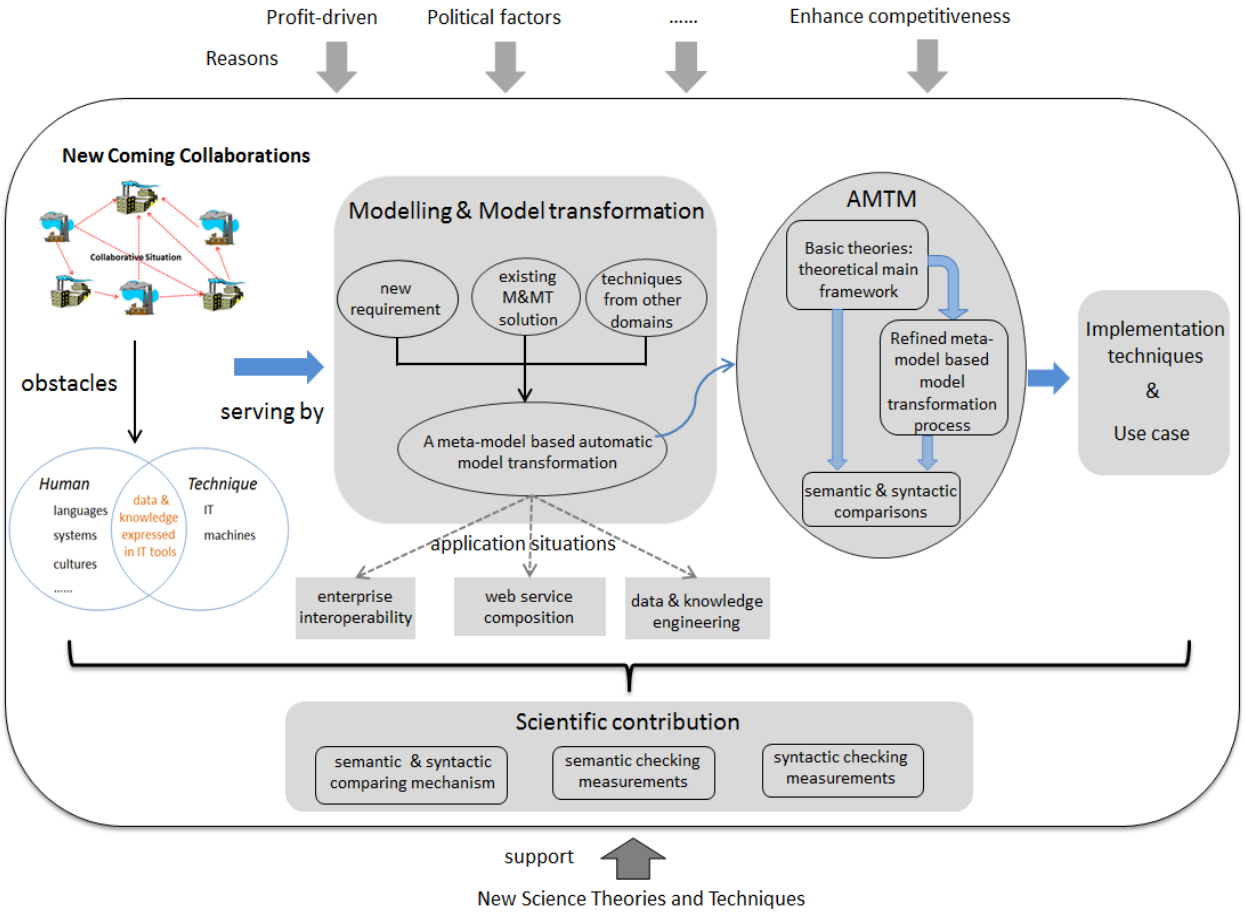


Figure GC-1: Content structure of this thesis.

Prospect usage of AMTM: serves to MISE 3.0

The original purpose of creating this PhD project is to support the research project "MISE". Actually, "automatic model transformation methodology" is an important support for MISE project. The "MISE" project was in its second iteration (MISE 2.0) when this PhD project has been launched. At this moment, the third version of "MISE" is on the developing process.

MISE 3.0 platform implements the same model-driven engineering approach than MISE 1.0 and MISE 2.0. This approach is structured according to four layers.

- Gathering of individual and collaborative knowledge.
- Designing of potential collaborative behavior.
- Deploying of accurate collaborative behavior.
- Managing and adapting of collaborative behavior.

The principle of MISE structure has been illustrated briefly in the first chapter of the thesis based on the second iteration of this approach: MISE 2.0. MISE 1.0 and MISE 2.0 provided improved solutions to support collaborative situations by deploying a MIS among heterogeneous organizations. However, there are still some concrete research avenues to explore. MISE 3.0 brings new characteristics to this huge research project in six main aspects; as stated in (Benaben et al, 2012), the details of these six improving aspects are also presented briefly below.

- **Knowledge gathering: collaboration model.** In MISE 1.0 and MISE 2.0, knowledge gathering step depended on the users: to fill the instantaneous information available concerning the collaborative situation. In MISE 3.0, the ambition is to use Event Driven Architecture (EDA) to continuously gather the knowledge (about organizations and situation) and continuously update the models (describing organizations and situation).
- **Behavior design: model of collaborative behavior.** Comparing with the "collaborative process(es) deduction" step defined in MISE 1.0 and MISE 2.0, MISE 3.0 includes a more soft principle of doing this step. This main idea of the new principle is "integrating decision support system to assist the user in selecting potential behaviors.
- **Implementation: deployment of Mediation IS.** MISE 3.0 uses **non-functional requirements** extracted from previously deduced "design-time" indicators to select the most fitting technical elements to implement the deduced business collaborative behavior. Otherwise, **non-functional requirements** have not been used in MISE 1.0 and MISE 2.0 to complete this step.
- **Agility: detection and adaptation.** MISE 3.0 uses detection through EDA system and adaptation through a new run of one of the design-time steps (function of the nature of the problem detected).
- **MISE 3.0 synthesis.** This part concerns the mechanism of integrating the four new improving aspects mentioned above as a working process in MISE 3.0. The four improving aspects focus on different steps in MISE, and there are interactions between them.
- **Application domains.** MISE 1.0 and MISE 2.0 aimed at serving three application domains about collaborative situations (but there might be really more): support of logistics systems, support of health care systems and support of crisis management systems. The usage of MISE 3.0 to the last domain mentioned (crisis management) is detailed in (Benaben et al, 2012): a geographical area may be watched through an EDA platform, in order to gather all events (from sensors, services, people, devices, etc.) in order to build and maintain a global picture of that area. According to some unexpected (or expected) negative changes (such as a lot of tweets mentioning the same problem, a lot of GPS data showing that a lot of vehicles are stopped, some abnormal values of temperature sensors, etc.), the MISE 3.0 platform could start the behavior deduction based on (i) information concerning the situation (risk, facts, etc.) and (ii) information concerning rescue means (resource, potential actors, etc.) both extracted from the global picture.

Figure GC-2 illustrates the global MISE approach (three steps in an agile framework) and underlines schematically the specificities of first, second and third iterations.

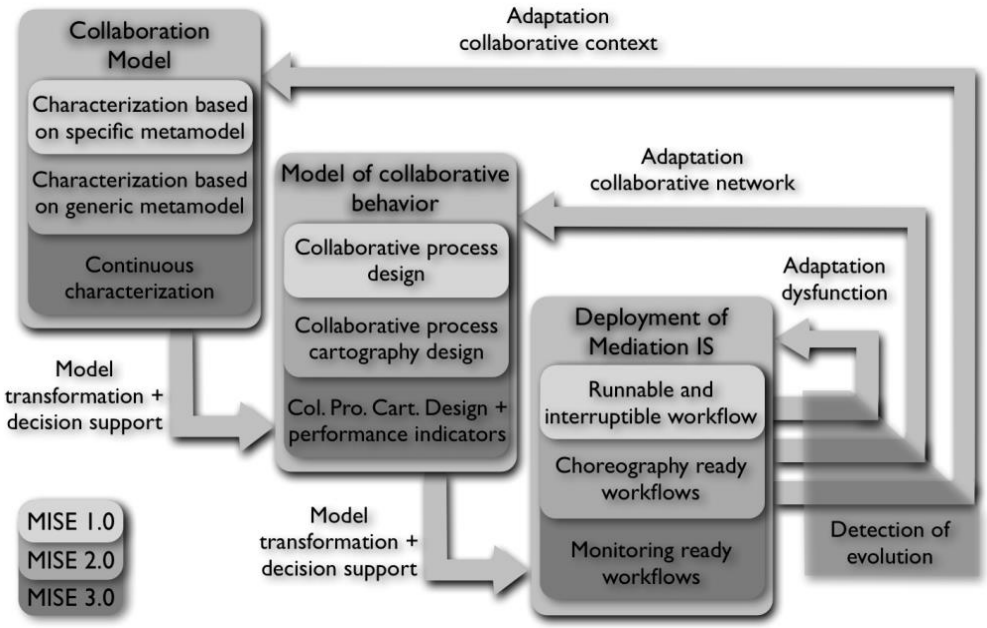


Figure GC-2: MISE 1.0, MISE 2.0 and MISE 3.0 iterations (Benaben et al, 2012).

AMTM performs a key role in MISE 3.0; it helps MISE 3.0 to achieve the agility characteristic by replacing manual efforts from the whole iterative process. Model transformation practices are used in all the four layers mentioned above. Actually, model transformations involved in MISE could be divided into two groups: horizontal model transformations and vertical model transformations. The model transformations take place within one MISE layer (connecting models from same layer) are belong to the first group: horizontal model transformations; the model transformations take place between two MISE layers (connecting models from two adjacent layers) are belong to the second group: vertical model transformations.

Future works and improvements

Based on the testing results of AMTM-SS, several points in both AMTM and AMTM-SS have to be improved in future works. The details of these improving points are listed below.

- **Automatic validation of the automatically generated model transformation mapping rules.** At this moment, there is only one way to validate the automatically generated model transformation mappings in AMTM: demanding system users or domain experts to do the validation. However, this validation method involves manual efforts, which contraries to the original intention of AMTM. So, an automatic validation method would be a better solution to AMTM. Here, just a proposal is shown in Figure GC-3.

  In AMTM, model transformation mappings rules are built on meta-model level (from source meta-model to target meta-model); the potential mapping rules concern how to transform model items in source meta-model to model items in target meta-model. The idea of doing validation on these mappings is to reverse this building process: changing the roles between source meta-model and target meta-model and building mappings to transform target meta-model to source meta-model. The mapping building process also depends on AMTM. Then, two sets of mapping rules will be generated (one set from source meta-model to target meta-model and the other set from target meta-model to source meta-model). Figure GC-3 shows the relation of the two sets of

mappings rules. The overlapping part between the two sets could be regarded as the final model transformation mappings; the mapping rules only appear in one set will be left to system users or domain experts to assess. In this way, the manual efforts involved in AMTM could be reduced (not necessary to validate all the automatically generated model transformation mappings).

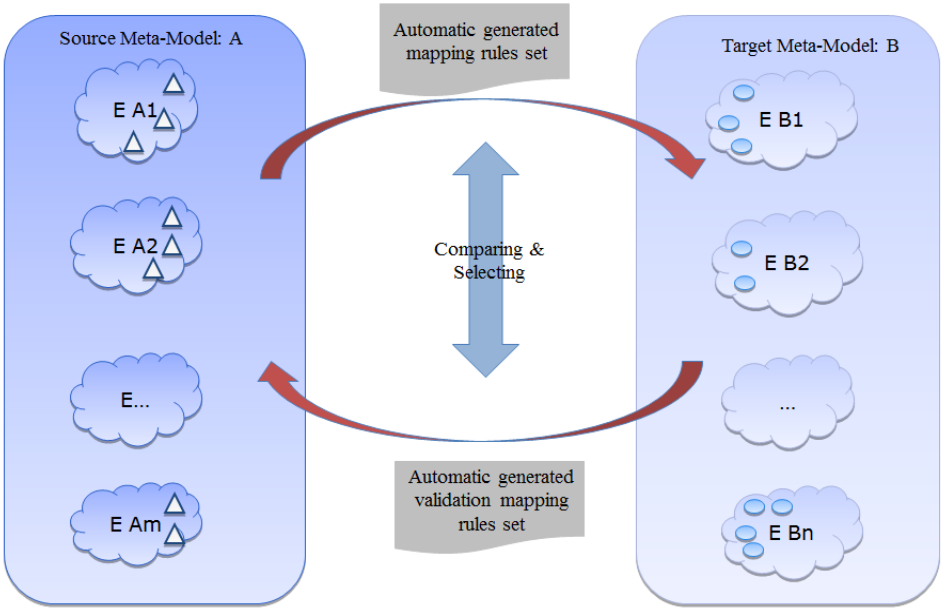


Figure GC-3: Illustration of possible automatic validation solution.

- **Mechanism of assigning values to the impact factors in equations** (illustrated in the third chapter) and **thread values** (illustrated in the fourth chapter about choosing mechanism of potential matching pairs). Actually, assigning values to these uncertain impact factors should concern about the context of different applying domains and situations. However, such consideration would lead this process to be done by manual work. There are two possible solutions to this problem: **assign manually and assign automatically**. The two meta-models may come from different domains, and usually the target meta-model plays a more important role in determination of values for these uncertain factors. So, the manual assign method depends on the expert from the domain of target meta-model. The automatic assign method depends on **mathematic strategies**, such as "choquet integral" (Abril et al, 2012); these mathematic strategies could detect the importance of each parameter that involved in a formula and assign the weights to them automatically. The second solution seems to be a better solution, but it may bring in new kinds of uncertain problems and reduce the efficiency of AMTM-SS.
- **Adding new semantic relations and modifying the respond values (or calculating formulas) to these semantic relations**. As stated in the fifth chapter, seven semantic relations have been defined and maintained in AMTM_ST and for each of them a respond value is assigned directly. One of the main problems is **"if the seven semantic relations are enough to detect potential model transformation mappings?"** Some new semantic relations (defined and maintained in WordNet) could be adopted into AMTM_ST, such as: "part holonym (part and whole)", "domain category", etc. More real test cases are needed to make the final decision about enriching AMTM_ST or not. The huger AMTM_ST is, the lower efficiency of semantic checking algorithms will be. Another main problem concerns the values assigned to different semantic

relations. These assigned values (also with the help of matching pairs' selection mechanism) determine directly the creation of final potential mapping rules. Some particular test cases or real use cases should be executed in relevant software tools (e.g. AMTM-SS) to modify correspondingly these semantic values and threshold values for choosing potential matching pairs.

- **The efficiency of the algorithms that focus on doing semantic checking measurements**. Since semantic checking measurements rely on a huge semantic thesaurus, the relevant algorithms are really time-consuming processes of detecting semantic relations between two words. The most time-consuming part in semantic relation detecting algorithms is the comparing loops for potential matching pairs. So, there are two solutions to improve the efficiency of this algorithm. The first solution is to **reduce the number of potential matching pairs**, and the second solution is to **reduce the number of comparing loops** for a specific matching pair. Both the two solutions are possible and will work. In the second version of AMTM (also AMTM-SS), the improving efficiency part of it will become a focus.
- **The usage of AMTM_ST**. At this moment, only formal English words are stored in AMTM_ST; it means semantic checking measurements could only take place between formal English words. However, in model transformation domain, the words involved might be in specific forms (e.g. plural form, gerund form and past tense forms), which are not stored in AMTM_ST. So, to enhance the usage of AMTM_ST, some preparation work should be done: either modify the words' forms before locating them in AMTM_ST or add words with special forms into AMTM_ST (it is necessary to build connections between the new added words and the existing content in AMTM_ST). Comparing the two methods, the first method "modify words' forms" is easier to implement. Some syntactic relevant algorithms should be introduced in to modify words' forms. Another problem in using AMTM_ST is the vocabulary stored in it. In some specific domain, some vocabularies are used; these vocabularies are not stored in AMTM_ST or possess different meanings as they should do. For this situation, either creating (or using) a specific semantic thesaurus or ignoring those specific vocabularies.

The five points mentioned above are needed to be improved. However, without these optimizations, both AMTM and AMTM-SS could work at this moment.

Prospect usage of AMTM: managing data sets

The usage of AMTM is not limited to serve to MISE project; the syntactic and semantic checking measurements and the ideas of doing this could serve to many domains (not only in model transformation domain). The basic function provided by AMTM, which is implemented as AMTM-SS, is to share and exchange information (data) among heterogeneous partners. A broader vison of the usage of AMTM is shown in Figure GC-4.

Converting rough data to information might be the further usage of AMTM. This work could help to solve cross-domain problems. Many data collectors (e.g. sensors, smart equipment and computers) could gather rough data from a particular region or domain. The collected data focus on various purposes and reflect different views of a system. Moreover, different collectors store data in their own structures which might be heterogeneous to each other. So, it is difficult to make use of this kind of data as a whole.

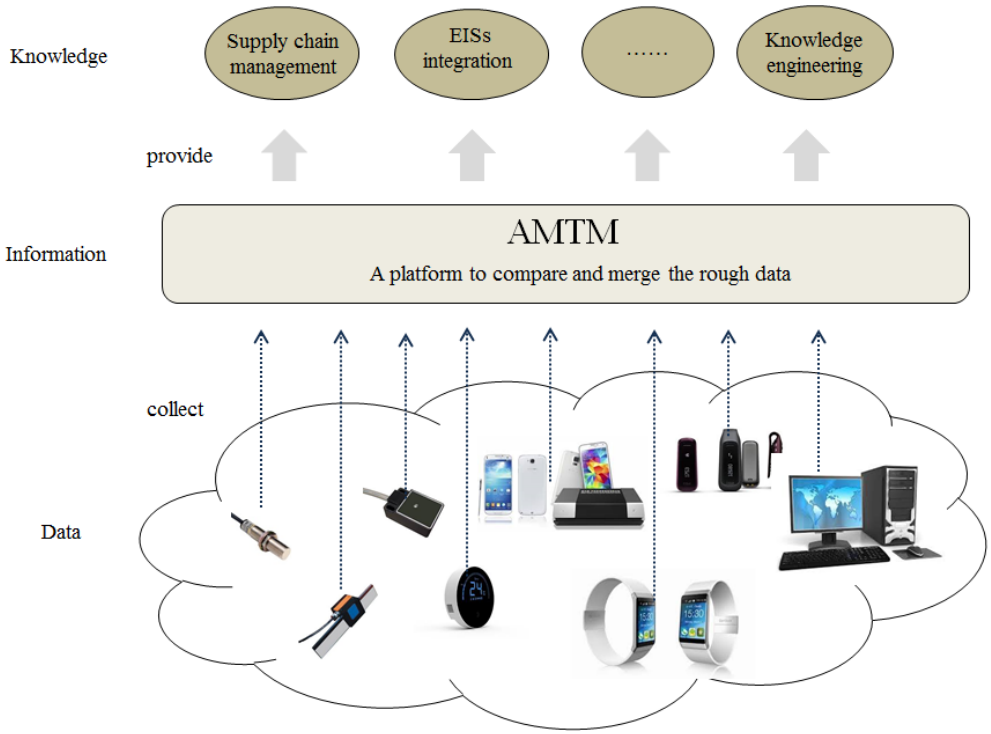


Figure GC-4: A broader vision of the usage of AMTM.

In the context of AMTM, each collected data could be regarded as single models. Thus, AMTM could use semantic and syntactic checking measurements to detect the intrinsic links between data and domain meta-models. Finally, after comparing and transforming these data, a final target model (overview of a specific system) could be generated. This final target model could be used by different domains; with domain specific rules, the information contained in this model could be used as knowledge.

A possible further usage of AMTM is to serve the building process of "smart city". A smart city uses digital technologies or information and communication technologies (ICT) to enhance quality and performance of urban services, to reduce costs and resource consumption, and to engage more effectively and actively with its citizens. To build a smart city, large amounts of sensors should be used to collect specific data or information of the city. In a specific region of smart city, different kinds of sensors should be used to collect different kinds of information depending on different views; in different regions of smart cities, same sensors are used to collect same kind of data or information to compare to each other. Finally, the data and information collected by these sensors (or a part of all sensors) should be used by professional people (focusing on different point of views) to analyze the exact situations of the city and thus provide suggestions of improving urban services. Before to be used by the professional people, the data and information should be compared horizontally (data collected by the same sensors) and vertically (data and information collected by different sensors).

AMTM could be used as a middleware between the collected data and professional people (e.g. city managers). Based on the real viewpoints (meta-models) made by professional people, AMTM could generate the specific views (models) by extracting and transforming the collected data (regarding each data as a single model and applying semantic and syntactic checking measurements between them).

References

(Abril et al, 2012) Abril, D., Navarro-Arribas, G., & Torra, V. (2012). Choquet integral for record linkage. Annals of Operations Research, 195(1), 97-110.

(Ackoff, 1971) Ackoff, R. L. (1971). Towards a system of systems concepts. Management science, 17(11), 661-671.

(Ackoff, 2010) Ackoff, R. L. (2010). From data to wisdom. Journal of applied systems analysis, 16, 3-9.

(Akehurst et al, 2002) Akehurst, D., & Kent, S. (2002). A relational approach to defining transformations in a metamodel. In ≪ UML≫ 2002—The Unified Modeling Language (pp. 243-258). Springer Berlin Heidelberg.

(Alessandro et al, 2008) D'Atri, A., De Marco, M., & Casalino, N. (Eds.). (2008). Interdisciplinary Aspects of Information Systems Studies: The Italian Association for Information Systems. Springer Science & Business Media.

(Andries et al, 1999) Andries, M., Engels, G., Habel, A., Hoffmann, B., Kreowski, H. J., Kuske, S., ... & Taentzer, G. (1999). Graph transformation for specification and programming. Science of Computer programming, 34(1), 1-54.

(Atzori et al, 2010) Atzori, L., Iera, A., & Morabito, G. (2010). The internet of things: A survey.Computer networks, 54(15), 2787-2805.

(Balogh et al, 2006) Balogh, A., Varró, G., Varró, D., & Pataricza, A. (2006, April). Compiling model transformations to EJB3-specific transformer plugins. In Proceedings of the 2006 ACM symposium on Applied computing (pp. 1288-1295). ACM.

(Bataille et al, 2001) Bataille, V., & Castellani, X. (2001). Métamodélisation et ingénierie des systèmes d'information. Ingénierie des systğmes dinformation, 149-174.

(Bénaben et al, 2010) Bénaben, F., Mu, W., Truptil, S., Pingaud, H., & Lorré, J. P. (2010, April). Information Systems design for emerging ecosystems. In Digital Ecosystems and Technologies (DEST), 2010 4th IEEE International Conference on (pp. 310-315). IEEE.

(Benaben et al, 2012) Benaben, F., Lauras, M., Truptil, S., & Lamothe, J. (2012). Mise 3.0: an agile support for collaborative situation. In Collaborative networks in the internet of services (pp. 645-654). Springer Berlin Heidelberg.

(Benaben et al, 2015) Benaben, F., Mu, W., Boissel-Dallier, N., Barthe-Delanoe, A. M., Zribi, S., & Pingaud, H. (2015). Supporting interoperability of collaborative networks through engineering of a service-based Mediation Information System (MISE 2.0). Enterprise Information Systems, 9(5-6), 556-582.

(Bernstein et al, 2011) Bernstein, P. A., Madhavan, J., & Rahm, E. (2011). Generic schema matching, ten years later. Proceedings of the VLDB Endowment, 4(11), 695-701.

(Bézivin, 2001) Bézivin, J. (2001, July). From object composition to model transformation with the MDA. In tools (p. 0350). IEEE.

(Bézivin et al, 2001) Bézivin, J., & Gerbé, O. (2001, November). Towards a precise definition of the OMG/MDA framework. In Automated Software Engineering, 2001.(ASE 2001). Proceedings. 16th Annual International Conference on (pp. 273-280). IEEE.

(Bézivin, 2006) Bézivin, J. (2006). Model driven engineering: An emerging technical space. InGenerative and transformational techniques in software engineering (pp. 36-64). Springer Berlin Heidelberg.

(Biehl, 2010) Biehl, M. (2010). Literature study on model transformations. Royal Institute of Technology, Tech. Rep. ISRN/KTH/MMK.

(Bilenko et al, 2002) Bilenko, M., & Mooney, R. J. (2002). Learning to combine trained distance metrics for duplicate detection in databases. Submitted to CIKM-2002, 1-19.

(Bollati et al, 2011) Bollati V A. MeTAGeM: a framework for model-driven development of model transformations[D]. Ph. D. Thesis. University Rey Juan Carlos. http://www. kybele. etsii. urjc. es/members/vbollati/Thesis, 2011.

(Bollati et al, 2013) Bollati, V. A., Vara, J. M., Jiménez, Á., & Marcos, E. (2013). Applying MDE to the (semi-) automatic development of model transformations. Information and Software Technology, 55(4), 699-718.

(Börger et al, 2012) Börger, E., & Stärk, R. (2012). Abstract state machines: a method for high-level system design and analysis. Springer Science & Business Media.

(Box et al, 2000) Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H. F., ... & Winer, D. (2000). Simple object access protocol (SOAP) 1.1.

(Brambilla et al, 2012) Brambilla, M., Cabot, J., & Wimmer, M. (2012). Model-driven software engineering in practice. Synthesis Lectures on Software Engineering, 1(1), 1-182.

(Brickley et al, 2000) Brickley, D., & Guha, R. V. (2000). Resource Description Framework (RDF) Schema Specification 1.0: W3C Candidate Recommendation 27 March 2000.

(Brinkman et al, 2013) Brinkman, S., Abu-Hanna, A., de Jonge, E., & de Keizer, N. F. (2013). Prediction of long-term mortality in ICU patients: model validation and assessing the effect of using in-hospital versus long-term mortality on benchmarking. Intensive care medicine, 39(11), 1925-1931.

(Bruel et al, 1998) Bruel, J. M., & France, R. B. (1998). Transforming UML models to formal specifications. The Unified Modeling Language, UML.

(Bruel et al, 2000) Bruel, J. M., Lilius, J., Moreira, A., & France, R. B. (2000). Defining precise semantics for UML. In Object-Oriented Technology (pp. 113-122). Springer Berlin Heidelberg.

(Cabot et al, 2010) Cabot, J., Clarisó, R., Guerra, E., & De Lara, J. (2010). Verification and validation of declarative model-to-model transformations through invariants.Journal of Systems and Software, 83(2), 283-302.

(Cabot et al, 2012) Cabot, J., & Gogolla, M. (2012). Object constraint language (OCL): a definitive guide. In Formal Methods for Model-Driven Engineering (pp. 58-90). Springer Berlin Heidelberg.

(Carey et al, 1994) Carey, S., & Spelke, E. (1994). Domain-specific knowledge and conceptual change. Mapping the mind: Domain specificity in cognition and culture, 169-200.

(Chapron, 2006) Chapron, J. (2006). L'urbanisme organisationnel: méthode et aides à la décision pour piloter l'évolution du système d'information de l'entreprise(Doctoral dissertation, Ecole Nationale Supérieure des Mines de Saint-Etienne).

(Chen et al, 2007) Chen, D., Dassisti, M., & Elves 忙 ter, B. (2007). Enterprise Interoperability Framework and Knowledge Corpus-Final report Annex: Knowledge Pieces.Contract no.: IST508, 11.

(Chen et al, 2008) Chen, D., Doumeingts, G., & Vernadat, F. (2008). Architectures for enterprise integration and interoperability: Past, present and future. Computers in industry, 59(7), 647-659.

(Cheng-Leng et al, 1999) Cheng-Leong, A., Li Pheng, K., & Keng Leng, G. R. (1999). IDEF*: a comprehensive modelling methodology for the development of manufacturing enterprise systems. International Journal of Production Research, 37(17), 3839-3858.

(Christensen et al, 2001) Christensen, E., Curbera, F., Meredith, G., & Weerawarana, S. (2001). Web services description language (WSDL) 1.1.

(Clark, 1999) Clark, J. (1999). Xsl transformations (xslt). World Wide Web Consortium (W3C). URL http://www. w3. org/TR/xslt.

(Cleaveland et al, 2001) Cleaveland, C. C., & Cleaveland, J. C. (2001). Program Generators with XML and Java with CD-ROM. Prentice Hall PTR.

(Cohen et al, 2002) Cohen, W. W., & Richman, J. (2002, July). Learning to match and cluster large high-dimensional data sets for data integration. In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 475-480). ACM.

(Cohen et al, 2003) Cohen, W., Ravikumar, P., & Fienberg, S. (2003, August). A comparison of string metrics for matching names and records. In Kdd workshop on data cleaning and object consolidation (Vol. 3, pp. 73-78).

(Crowston, 1994) Crowston, K. (1994). A taxonomy of organizational dependencies and coordination mechanisms. Center for Coordination Science, Alfred P. Sloan School of Management, Massachusetts Institute of Technology.

(Curbera et al, 2003) Curbera, F., Khalaf, R., Mukhi, N., Tai, S., & Weerawarana, S. (2003). The next step in web services. Communications of the ACM, 46(10), 29-34.

(Czarnecki et al, 2003) Czarnecki, K., & Helsen, S. (2003, October). Classification of model transformation approaches. In Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture(Vol. 45, No. 3, pp. 1-17).

(Davis et al, 1988) Davis, A. M., Bersoff, E. H., & Comer, E. R. (1988). A strategy for comparing alternative software development life cycle models. Software Engineering, IEEE Transactions on, 14(10), 1453-1461.

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

(Del Fabro et al, 2009) 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.

(Del Fabro et al, 2005) Del Fabro, M. D., Bézivin, J., Jouault, F., Breton, E., & Gueltas, G. (2005). AMW: a generic model weaver. In 1 ere Journées sur l'Ingénierie Dirigée par les Modèles (IDM05) (pp. 105-114).

(Dolques, 2011) Dolques, X., Dogui, A., Falleri, J. R., Huchard, M., Nebut, C., & Pfister, F. (2011). Easing model transformation learning with automatically aligned examples. In Modelling Foundations and Applications (pp. 189-204). Springer Berlin Heidelberg.

(Feldmann et al, 2014) Feldmann, S., Kernschmidt, K., & Vogel-Heuser, B. (2014). Combining a SysML-based modeling approach and semantic technologies for analyzing change influences in manufacturing plant models. Procedia CIRP, 17, 451-456.

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

(Fowler, 2010) Fowler, M. (2010). Domain-specific languages. Pearson Education.

(Fox et al, 1998) Fox, M. S., & Gruninger, M. (1998). Enterprise modeling. AI magazine,19(3), 109.

(France et al, 2007) France, R., & Rumpe, B. (2007, May). Model-driven development of complex software: A research roadmap. In 2007 Future of Software Engineering (pp. 37-54). IEEE Computer Society.

(Friedenthal, 2014) Friedenthal, S., Moore, A., & Steiner, R. (2014). A practical guide to SysML: the systems modeling language. Morgan Kaufmann.

(Gábor et al, 2013) Gábor, A., Kő, A., Szabó, I., Ternai, K., & Varga, K. (2013, January). Compliance Check in Semantic Business Process Management. In On the Move

to Meaningful Internet Systems: OTM 2013 Workshops (pp. 353-362). Springer Berlin Heidelberg.

(Galhardas et al, 1999) Galhardas, H., Florescu, D., Shasha, D., & Simon, E. (1999). An extensible framework for data cleaning.

(Gerber et al, 2002) Gerber, A., Lawley, M., Raymond, K., Steel, J., & Wood, A. (2002). Transformation: The missing link of MDA. In Graph Transformation (pp. 90-105). Springer Berlin Heidelberg.

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

(Grangel et al, 2010) Grangel, R., Bigand, M., & Bourey, J. P. (2010). Transformation of decisional models into UML: application to GRAI grids. International Journal of Computer Integrated Manufacturing, 23(7), 655-672.

(Gruber, 1995) Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing?. International journal of human-computer studies, 43(5), 907-928.

(Gruninger, 2000) Gruninger, M., Tissot, F., Valois, J., Lubell, J., & Lee, J. (2000). The process specification language (PSL) overview and version 1.0 specification. US Department of Commerce, Technology Administration, National Institute of Standards and Technology.

(Heeringa, 2004) Heeringa, W. J. (2004). Measuring dialect pronunciation differences using Levenshtein distance (Doctoral dissertation, [University Library Groningen][Host]).

(Hernández et al, 1995) Hernández, M. A., & Stolfo, S. J. (1995, June). The merge/purge problem for large databases. In ACM SIGMOD Record (Vol. 24, No. 2, pp. 127-138). ACM.

(Herrmannsdoerfer et al, 2009) Herrmannsdoerfer, M., Benz, S., & Juergens, E. (2009). COPE-automating coupled evolution of metamodels and models. In ECOOP 2009–Object-Oriented Programming (pp. 52-76). Springer Berlin Heidelberg.

(Hooper et al, 2009) Hooper, R., & Paice, C. (2005). The Lancaster stemming algorithm.Retrieved June, 20, 2009.

(Huang, 2007) Huang, X. X. (2007). An OWL-based WordNet lexical ontology. Journal of Zhejiang University SCIENCE A, 8(6), 864-870.

(Hwang et al, 2008) Hwang, S. Y., Lim, E. P., Lee, C. H., & Chen, C. H. (2008). Dynamic web service selection for reliable web service composition. Services Computing, IEEE Transactions on, 1(2), 104-116.

(Ide, 2010) Ide, N., & Pustejovsky, J. (2010, January). What does interoperability mean, anyway? Toward an operational definition of interoperability for language technology. In Proceedings of the Second International Conference on Global Interoperability for Language Resources. Hong Kong, China.

(Jackson et Keys, 1984) Jackson, M. C., & Keys, P. (1984). Towards a system of systems methodologies. Journal of the operational research society, 473-486.

(Jaro, 1989) Jaro, M. A. (1989). Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. Journal of the American Statistical Association, 84(406), 414-420.

(Jaro, 1995) Jaro, M. A. (1995). Probabilistic linkage of large public health data files.Statistics in medicine, 14(5- 7), 491-498.

(Jenz, 2003) Jenz, D. E. (2003). BPMO Tutorial: Defining a Private Business Process in a Knowledge Base. Tutorial, Jenz & Partner GmbH.

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

(Jones, 1997) Jones, K. S. (1997). Readings in information retrieval. Morgan Kaufmann.

(Kappel et al, 2006) Kappel, G., Kapsammer, E., Kargl, H., Kramler, G., Reiter, T., Retschitzegger, W., ... & Wimmer, M. (2006). Lifting metamodels to ontologies: A step to the semantic integration of modeling languages (pp. 528-542). Springer Berlin Heidelberg.

(Kappel et al, 2007) Kappel, G., Kargl, H., Kramler, G., Schauerhuber, A., Seidl, M., Strommer, M., & Wimmer, M. (2007, March). Matching Metamodels with Semantic Systems-An Experience Report. In BTW Workshops (pp. 38-52).

(Karsai et al, 2003) 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.

(Kim et al, 2012) Kim, J., Kang, S., Lee, J., & Choi, B. W. (2012). A semantic translation method for data communication protocols. Journal of Systems and Software,85(12), 2876-2898.

(Konstantas et al, 2005) Konstantas, D., Bourrières, J. P., Léonard, M., & Boudjlida, N. (Eds.). (2006). Interoperability of enterprise software and applications. Springer Science & Business Media.

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

(Kosanke et al, 1999) Kosanke, K., Vernadat, F., & Zelm, M. (1999). CIMOSA: enterprise engineering and integration. Computers in industry, 40(2), 83-97.

(Lee et al, 2011) Lee, J. (Ed.). (2011). Service Life Cycle Tools and Technologies: Methods, Trends and Advances: Methods, Trends and Advances. IGI Global.

(Levenshtein, 1966) Levenshtein, V. I. (1966, February). Binary codes capable of correcting deletions, insertions, and reversals. In Soviet physics doklady (Vol. 10, No. 8, pp. 707-710).

(Malone et al, 2003) Malone, T. W., Crowston, K., & Herman, G. A. (2003). Organizing business knowledge: the MIT process handbook. MIT press.

(Maier, 1998) Maier, M. W. (1996, July). Architecting principles for systems-of-systems. InINCOSE International Symposium (Vol. 6, No. 1, pp. 565-573).

(Matula, 2003) Matula, M. (2003). NetBeans metadata repository. NetBeans Community.

(McDonald et al, 2013) McDonald, N., & Goggins, S. (2013, April). Performance and participation in open source software on github. In CHI'13 Extended Abstracts on Human Factors in Computing Systems (pp. 139-144). ACM.

(Merrill, 1916) Merrill, S. (1916). The moose book. EP Dutton.

(Miller et al, 2003) Miller, J., & Mukerji, J. (2003). MDA Guide Version 1.0. 1.

(Min et al, 2010) Min, F. Y., Yang, M., & Wang, Z. C. (2010). Knowledge-based method for the validation of complex simulation models. Simulation Modelling Practice and Theory, 18(5), 500-515.

(Monge et al, 1996) Monge, A. E., & Elkan, C. (1996, August). The Field Matching Problem: Algorithms and Applications. In KDD (pp. 267-270).

(Monge et al, 1997) Monge, A., & Elkan, C. (1997). An efficient domain-independent algorithm for detecting approximately duplicate database records.

(Morley, 2002) Morley, C. (2002). La modélisation des processus: typologie et proposition utilisant UML. Processus & Systèmes d'information-Journée ADELI, 13.

(Navarro, 2001) Navarro, G. (2001). A guided tour to approximate string matching. ACM computing surveys (CSUR), 33(1), 31-88.

(Neches et al, 1991) Neches, R., Fikes, R. E., Finin, T., Gruber, T., Patil, R., Senator, T., & Swartout, W. R. (1991). Enabling technology for knowledge sharing. AI magazine, 12(3), 36.

(Nikolova et al, 2012) Nikolova, S., Boyd-Graber, J., & Fellbaum, C. (2012). Collecting semantic similarity ratings to connect concepts in assistive communication tools. InModeling, Learning, and Processing of Text Technological Data Structures(pp. 81-93). Springer Berlin Heidelberg.

(OMG, 2000) OMG. The XMI Specification 1.0. http://www.omg.org. 2000

(OMG, 2008) Omg, Q. (2008). Meta object facility (mof) 2.0 query/view/transformation specification. Final Adopted Specification (November 2005).

(OMG, 2014) Object Management Group Model Driven Architecture (MDA) MDA Guide rev. 2.0 OMG Document ormsc/2014-06-01

(Papazoglou et al, 2003) Papazoglou, M. P. (2003, December). Service-oriented computing: Concepts, characteristics and directions. In Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on(pp. 3-12). IEEE.

(Pasula et al, 2002) Pasula, H., Marthi, B., Milch, B., Russell, S., & Shpitser, I. (2002). Identity uncertainty and citation matching. In Advances in neural information processing systems (pp. 1401-1408).

(Poole et al, 2002) Poole, J., Chang, D., Tolbert, D., & Mellor, D. (2002). Common warehouse metamodel (Vol. 20). John Wiley & Sons.

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

(Porter, 2001) Porter, M. F. (2001). Snowball: A language for stemming algorithms.

(Pressman, 2005) Pressman, R. S. (2005). Software engineering: a practitioner's approach. Palgrave Macmillan.

(Radatz et al, 1990) Radatz, J., Geraci, A., & Katki, F. (1990). IEEE standard glossary of software engineering terminology. IEEE Std, 610121990(121990), 3.

(Rahm et Bernstein, 2001) Rahm, E., & Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. the VLDB Journal, 10(4), 334-350.

(Rajsiri et al, 2010) Rajsiri, V., Lorré, J. P., Benaben, F., & Pingaud, H. (2010). Knowledge-based system for collaborative process specification. Computers in Industry,61(2), 161-175.

(Rao et al, 2005) Rao, J., & Su, X. (2005). A survey of automated web service composition methods. In Semantic Web Services and Web Process Composition (pp. 43-54). Springer Berlin Heidelberg.

(Ristad et al, 1998) Ristad, E. S., & Yianilos, P. N. (1998). Learning string-edit distance. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 20(5), 522-532.

(Romero et al, 2011) Romero, V., Rutherford, B., & Newcomer, J. (2011, April). Some Statistical Procedures to Refine Estimates of Uncertainty when Sparse Data are Available for Model Validation and Calibration. In 52nd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference 19th AIAA/ASME/AHS Adaptive Structures Conference 13t (p. 1709).

(Salem et al, 2008) Salem, R. B., Grangel, R., & Bourey, J. P. (2008). A comparison of model transformation tools: Application for Transforming GRAI Extended Actigrams into UML Activity Diagrams. Computers in Industry, 59(7), 682-693.

(Santiago et al, 2012) Santiago, I., Jiménez, Á., Vara, J. M., De Castro, V., Bollati, V. A., & Marcos, E. (2012). Model-Driven Engineering as a new landscape for traceability management: A systematic literature review. Information and Software Technology, 54(12), 1340-1356.

(Schmidt, 2006) Schmidt, D. C. (2006). Guest editor's introduction: Model-driven engineering.Computer, 39(2), 0025-31.

(Shvaiko et al, 2005) Shvaiko, P., & Euzenat, J. (2005). A survey of schema-based matching approaches. In Journal on Data Semantics IV (pp. 146-171). Springer Berlin Heidelberg.

(Soley, 2000) Soley, R. (2000). Model driven architecture. OMG white paper, 308(308), 5.

(Steinberg et al, 2008) Steinberg, D., Budinsky, F., Merks, E., & Paternostro, M. (2008). EMF: eclipse modeling framework. Pearson Education.

(Tejada et al, 2001) Tejada, S., Knoblock, C. A., & Minton, S. (2001). Learning object identification rules for information integration. Information Systems, 26(8), 607-633.

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

(Truptil, 2011) Truptil, S. (2011). Etude de l'approche de l'interopérabilité par médiation dans le cadre d'une dynamique de collaboration appliquée à la gestion de crise.

(Terrasse et al, 2005) Terrasse, M. N., Savonnet, M., Leclercq, E., Grison, T., & Becker, G. (2005). Points de vue croisés sur les notions de modèle et métamodèle. 1ères journées sur l'Ingénierie Dirigée par les Modèles, 17-28.

(Touzi et al, 2007) Touzi, J., Lorré, J. P., Bénaben, F., & Pingaud, H. (2007). Interoperability through model-based generation: The case of the collaborative information system (CIS). In Enterprise Interoperability (pp. 407-416). Springer London.

(Touzi et al, 2009) Touzi, J., Benaben, F., Pingaud, H., & Lorré, J. P. (2009). A model-driven approach for collaborative service-oriented architecture design. International Journal of Production Economics, 121(1), 5-20.

(Richards, 2006) Richards, R. (2006). Universal Description, Discovery, and Integration (UDDI). In Pro PHP XML and Web Services (pp. 751-780). Apress.

(Uschold et al, 1998) Uschold, M., King, M., Moralee, S., & Zorgios, Y. (1998). The enterprise ontology. The knowledge engineering review, 13(01), 31-89.

(Van Amstel et al, 2012) van Amstel, M. F., van den Brand, M. G., & Serebrenik, A. (2012). Traceability visualization in model transformations with tracevis. In Theory and Practice of Model Transformations (pp. 152-159). Springer Berlin Heidelberg.

(Van der Aalst, 2014a) van der Aalst, W. M. (2014). How People Really (Like To) Work. In Human-Centered Software Engineering (pp. 317-321). Springer Berlin Heidelberg.

(Van der Aalst, 2014b) van der Aalst, W. M. (2014). Data scientist: The engineer of the future. InEnterprise Interoperability VI (pp. 13-26). Springer International Publishing.

(van Rijsbergen et al, 1980) Van Rijsbergen, C. J., Robertson, S. E., & Porter, M. F. (1980). New models in probabilistic information retrieval. Computer Laboratory, University of Cambridge.

(Varró et al, 2003) Varró, D., & Pataricza, A. (2003). VPM: A visual, precise and multilevel metamodeling framework for describing mathematical domains and UML (The Mathematics of Metamodeling is Metamodeling Mathematics). Software and Systems Modeling, 2(3), 187-210.

(Varró et al, 2004) Varró, D., & Pataricza, A. (2004). Generic and meta-transformations for model transformation engineering. In «UML» 2004—The Unified Modeling Language. Modeling Languages and Applications (pp. 290-304). Springer Berlin Heidelberg.

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

(Vernadat, 1999) Vernadat, F. (1999). Techniques de modélisation en entreprise: applications aux processus opérationnels. Economica.

(Wang et al, 2014a) Wang, T., Truptil, S., & Benaben, F. (2014, January). Semantic approach to automatically defined model transformation. In Model-Driven Engineering and Software Development (MODELSWARD), 2014 2nd International Conference on (pp. 340-347). IEEE.

(Wang et al, 2014b) Wang, T., Truptil, S., & Benaben, F. (2014). Semantic Approach to Automatically Defined Model Transformation. In Enterprise Interoperability VI(pp. 127-138). Springer International Publishing.

(Wang et al, 2015a) Wang, T., Truptil, S., & Benaben, F. (2015). A General Model Transformation Methodology to Serve Enterprise Interoperability Data Sharing Problem. In Enterprise Interoperability (pp. 16-29). Springer Berlin Heidelberg.

(Wang et al, 2015b) Wang, T., Truptil, S., & Benaben, F. (2015, June). An Automatic Model Transformation Methodology to Serve Web Service Composition Data

Transforming Problem. In Services (SERVICES), 2015 IEEE World Congress on (pp. 135-142). IEEE.

(Wang et al, 2015c) Wang, T., Truptil, S., & Benaben, F. (2015, January). Applying a Semantic & Syntactic Comparisons Based Automatic Model Transformation Methodology to Serve Information Sharing. In Proceedings of the International Conference on Information and Knowledge Engineering (IKE) (p. 3). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

(Wiederhold, 1992) Wiederhold, G. (1992). Mediators in the architecture of future information systems. Computer, 25(3), 38-49.

(Willett, 2006) Willett, P. (2006). The Porter stemming algorithm: then and now. Program,40(3), 219-223.

(Winkler, 1999) Winkler, W. E. (1999). The state of record linkage and current research problems. In Statistical Research Division, US Census Bureau.

(W3C, 2000) XML Schema Definition 2000. http://www.w3.org/TR/xmlschema-0/

(Yu et al, 2012) Yu, Y., Lin, Y., Hu, Z., Hidaka, S., Kato, H., & Montrieux, L. (2012, June). Maintaining invariant traceability through bidirectional transformations. InProceedings of the 34th International Conference on Software Engineering(pp. 540-550). IEEE Press.

Annex: AMTM-SS implementation

The annex part of this thesis concerns about the developing process of AMTM-SS. This part contains three main sections: (i) requirement analysis, (ii) software implementation, and (iii) software testing. All the three sections focus on AMTM-SS.

I. Requirement analysis

The requirement analysis activity focuses on the main functional and non-functional points that should be involved in AMTM-SS. In other words, this step should make clear that the work needed to be done by this software tool and how the work should be done (depending on the efficiency point of view).

The basic functional requirement of AMTM is building potential model transformation mappings based on the input models sets. Figure Annex-1 shows this requirement.

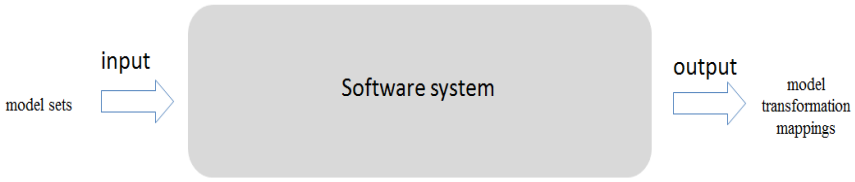


Figure Annex-1: Main functional requirement on AMTM-SS.

The input of AMTM-SS is "model sets". Normally, this "model sets" could contain several combinations, for instances: a set of "source model, source meta-model and target "meta-model", a set of "source model and target model instances" and a set of "source meta-model and target model instances". The output is a set of potential model transformation mappings.

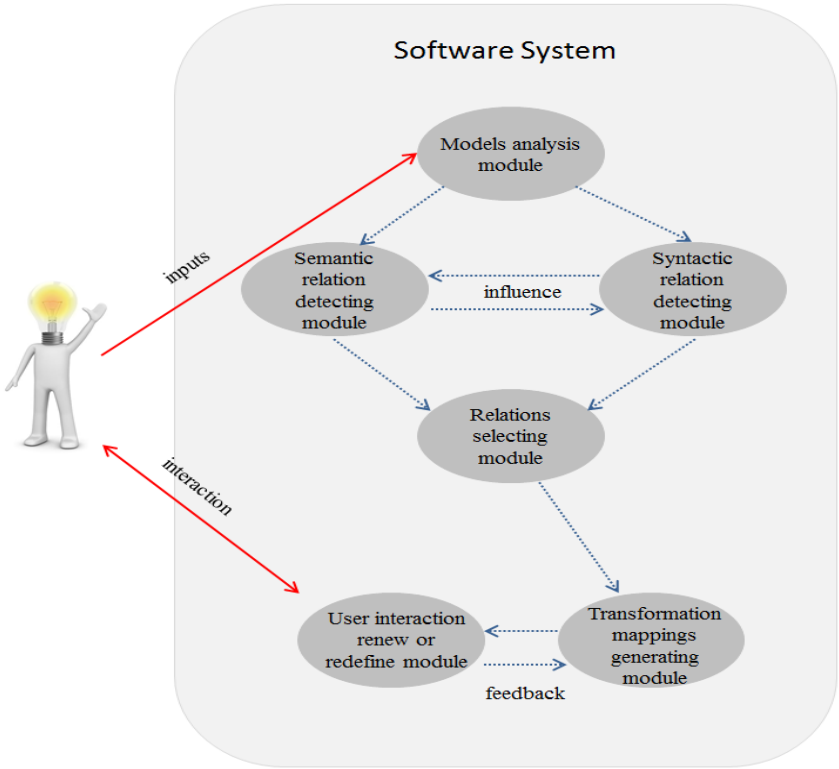


Figure Annex-2: Main functional modules in AMTM-SS.

According to the principle functional requirement, the software system should contain several main functional modules. These functional modules are shown in Figure Annex-2. The details (functional and non-functional requirement and design) of each functional module have been presented in chapter seven. Here, we mainly focus on the system implementation and testing parts.

II. System implementation

This section focuses on the implementation phase of AMTM-SS. The developing environment "relevant techniques and main algorithms involved" are presented in the following subsections, respectively.

II.1 AMTM-SS developing environment

AMTM-SS is developed as a desktop application in its first version. It is only used and tested in one local computer, which is also used to develop it. The basic configuration of this computer is shown in Figure Annex-3.

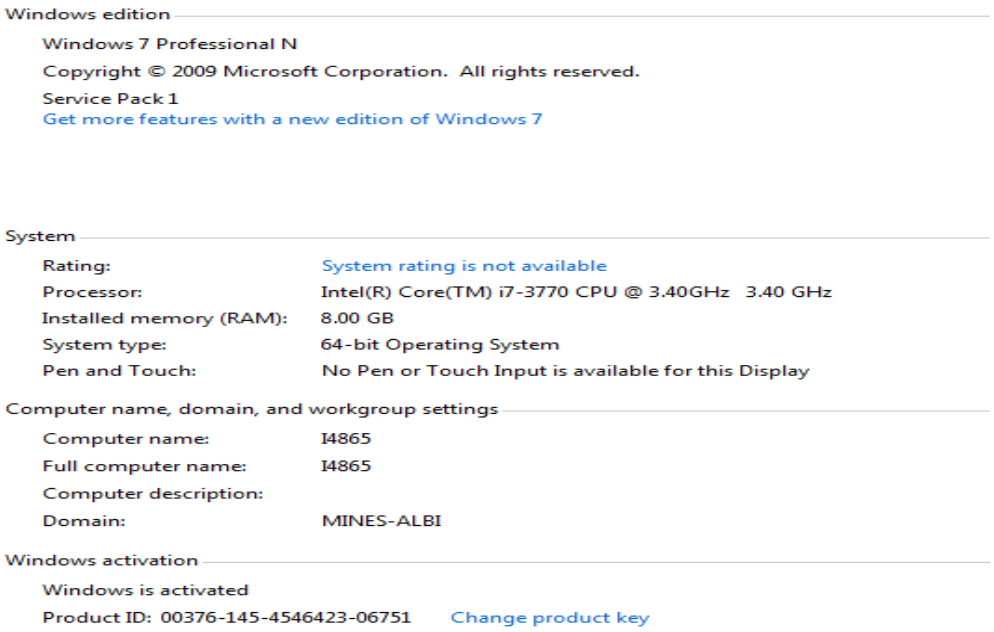


Figure Annex-3: Basic configuration of the developing computer.

As a personal computer, this one has a high configuration. The basic technique parameters are: (i) the operating system: Windows 7 Professional N, (ii) CPU: Inter Core i7-3770, 3.4GHz, (iii) main memory (RAM): 8.0 GB, etc. Comparing to the other personal computers, this one has high efficiency of executing programs. It provides a good environment of developing and testing software system.

The main programming language is java, three books about using this programming language are listed in (Arnold et al, 1996), (Wellings, 2004) and (Dietel, 2009); the version being used is "1.8.0_20". The relevant java developing environment contains: the version of runtime environment is "1.8.0_20-b26", and the java virtual machine is 64-Bit server, 25.20-b23 mixed mode. The integrated developing environment (IDE) of java programs is "Eclipse", the detail of this IDE could be consulted in (Geer, 2005), (Murphy et al, 2006) and (Vogel, 2014); the chosen version is "eclipse-java-luna-SR1-win32-x86_64". This is the new and stable version when the project of AMTM-SS was launched. The other developing environment techniques are: Maven 2.0, which is presented in (Smart, 2005) and (Casey et al,

2006), and GitHub stated in (McDonald et al, 2013) and (Vasilescu et al, 2013). The semantic thesaurus "AMTM_ST" is stored in MongoDB; the introduction and use tips of this database could be consulted in (Chodorow, 2010), (Membrey et al, 2010) and (Chodorow, 2013).

For developing AMTM-SS, several existing java jar packages are also involved in. These jar packages could be regarded as functional black boxes; they provide functions to achieve specific tasks and hide the implementation details to their users. Applying these packages could save developing time and improve the efficiency of developing software systems. Two of the main jar packages are: (i) the jar packages provided by CGI laboratory, and (ii) the jar package "JDOM" (Hunter, 2002). The usage of these two jar packages is shown in the following phrases, respectively.

- Jar packages provided by CGI laboratory: these jar packages provide the functions of connecting XSD files and java classes (transforming automatically XSD files to java classes). In AMTM-SS, such functions help to generate automatically the specific meta-model templates conformed to the MMM. Furthermore, these jar packages could also help to extract the content stored in the user input model sets into specific meta-model template.
- JDOM jar package: the functions provided by this jar package aims at building connections between XML files and java classes. It provides many functions dealing with XML files: extracting information from these files and generating new XML files with special formats that are defined in the java programs. AMTM-SS uses functions provided by this jar package to analyze the input model sets (models are uploaded to the software system in XML format). Also, the output of AMTM-SS "final target model" is a XML file that is generated by JDOM jar package and based on the transformation results (content stored in java classes).

All the content presented above is about the developing relevant environment and techniques. As AMTM-SS is a huge and complex project, the developing phrase is a long term process. In the original plan, AMTM-SS would have several versions; each version will be depending on different developing techniques (even application situations). Also, there are several points in the original version of AMTM-SS needed to be improved, such as: enrich the semantic thesaurus, improve the efficiency of algorithms involved and optimize the user interface. In the further version of AMTM-SS, these improvements would be added one by one.

II.2 Main algorithms implemented in AMTM-SS

In this subsection, three main functional algorithms involved in AMTM-SS will be presented and explained. The three algorithms are: syntactic checking methodology ("Levensthein distance" algorithm), semantic checking methodology (the algorithms of detecting semantic relations between two words and assigning values to detected relations) and the algorithm of combining syntactic checking and semantic checking as a whole. The implementation (executable programming codes) of the three algorithms is shown in the following subsections.

II.2.1 Implementation of "Levensthein distance" algorithm

As a part of the syntactic checking method involved in AMTM, "Levensthein distance" algorithm plays a very important role. In the majority comparing (to detect potential matching pairs) cases, the two names (words) involved are different, which could not be defined by the pretreatment phase in syntactic checking. The syntactic similarity between two different words (in syntactic aspect) needed to be

measured by "Levensthein distance" algorithm. The implementation of this algorithm in AMTM-SS is shown in Figure Annex-4.

```
// The concrete execution process
public double getSyntacticRelation() {
  // get the length of the two words
  int len1 = word1.length();
  int len2 = word2.length();

  //initiate an array to store the intermediate results
  int[][] matrix = new int[len1 + 1][len2 + 1];

  // initialization the "LD" comparison matrix
  for (int i = 0; i <= len1; i++) {
    matrix[i][0] = i;
  }
  for (int j = 0; j <= len2; j++) {
    matrix[0][j] = j;
  }
  // a temporary variable
  int temp;
  // mathematical mechanism defined in LD
  for (int i = 1; i <= len1; i++) {
    for (int j = 1; j <= len2; j++) {
      if (word1.charAt(i - 1) == word2.charAt(j - 1)) {
        temp = 0;
      } else {
        temp = 1;
      }
      int minimumValue = 0;
      // selecting the minimum value from three comparing values
      if (matrix[i - 1][j - 1] + temp <= matrix[i][j - 1] + 1) {
        minimumValue = (matrix[i - 1][j - 1] + temp) <= (matrix[i - 1][j] + 1) ? (matrix[i - 1][j - 1] + temp)
            : (matrix[i - 1][j] + 1);
      } else {
        minimumValue = (matrix[i][j - 1] + 1) <= (matrix[i - 1][j] + 1) ? (matrix[i][j - 1] + 1)
            : (matrix[i - 1][j] + 1);
      }
      matrix[i][j] = minimumValue;
    }
  }
  // formula to transform LD to a syntactic similar value ranges from 0 to 1
  syntacticRelation = 1 - (double) matrix[len1][len2] / (getBiggerValue(word1.length(), word2.length()));
```

Figure Annex-4: Implementation of "Levensthein distance" algorithm.

The main executable codes of this algorithm are only around fifty lines. The calculation idea behind these lines of code is more valuable. The three main variables defined in this algorithm are used to store the two comparing words and comparing syntactic similarity value. The lengths and letters involved in the two comparing are two kinds of information that are important to this algorithm; a comparison matrix for the two comparing words is built and the initialization values of this matrix are coming from the two kinds of information. The calculating, mathematical mechanism has been illustrated in the former chapter and this mechanism is easy to be implemented with codes (around fifteen lines of code). Finally, the comparing value calculated is the number of steps needed to transform one word to another. A formula is shown at the bottom of the code fragment; this formula transforms the "Levensthein distance" to the syntactic similarity value (between 0 and 1) between two comparing words.

II.2.2 Implementation of semantic relation detecting algorithm

Semantic relation detecting plays a key role in making potential model transformation mappings for AMTM. Model items convey semantic representations as their main identifiers; these identifiers are used to differentiate model items and provide clues to make matching pairs. Semantic relation checking methods aim at revealing these semantic representations conveyed by model items; furthermore, assign different values for different potential matching pairs that own different semantic relations. As illustrated in former chapters, a specific semantic thesaurus "AMTM_ST" has been created in AMTM to do the semantic relation detecting process. AMTM-SS needs to define the programs to get access to AMTM_ST; different functions should be implemented to extract information and make use of AMTM_ST. To make this point clearly, the main functions provided by AMTM-SS to operate AMTM_ST contains: (i) locating

one word in AMTM-SS, (ii) tracing and grouping all the word-senses belonged to the located word, (iii) tracing and grouping all the synsets that contains all the located word-senses, and (iv) detecting semantic relations that are built (in AMTM_ST) between synsets.

In AMTM-SS, AMTM_ST is implemented in MongoDB database. All the content of AMTM_ST is stored in MongoDB following the standard format of XML file. The first function provided by AMTM-SS is to get access to the MongoDB database; so, before running the programs in AMTM-SS, the MongoDB program should be started independently. Figure Annex-5 shows the start command and running interface of MongoDB tool.

Figure Annex-5: MongoDB start command and running interface.

It is shown clearly in Figure Annex-5 that the "Windows PowerShell" program opens "MongoDB" program, and it is waiting other programs to connect it at the "27017" port (this port number is defined in MongoDB configuration file, which could be modified). The content stored in MongoDB is shown in Figure Annex-6.

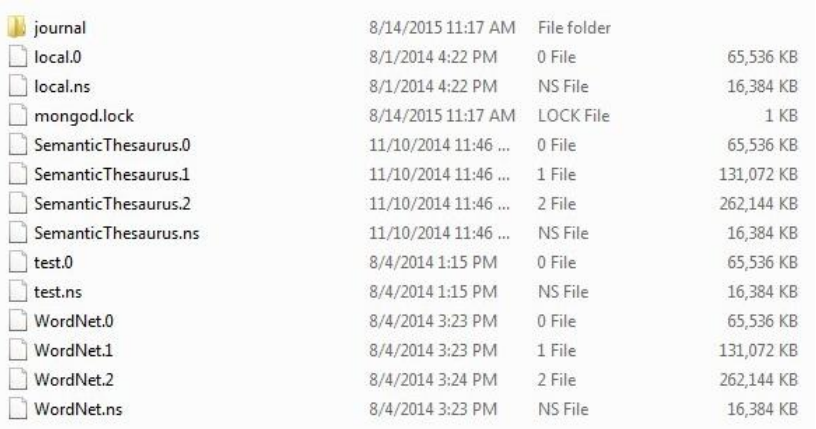


Figure Annex-6: Contents store in MongoDB.

The file named as "SemanticThesaurus" is the target to get connected by AMTM-SS. The content store in "SemanticThesaurus" is coming from an "OWL" file named as "WordNet.owl", which is regarded as the semantic thesaurus "AMTM_ST". The code fragment to implement the function of extracting content from "WordNet.owl" file and storing to MongoDB is shown in Figure Annex-7.

```
private File                ontologyFile;
private SAXBuilder          ontoBuilder;
private Document            document;
private Element             rootElement;
private Namespace           rootNameSpace;
private List<Element>       elementsList;
private MongoDBDao          mdbd;

List<GJaxbNounSynset>       gJaxbNounSynset       = new ArrayList<GJaxbNounSynset>();
List<GJaxbVerbSynset>       gJaxbVerbSynsets      = new ArrayList<GJaxbVerbSynset>();
List<GJaxbAdjectiveSynset>  gJaxbAdjectiveSynsets = new ArrayList<GJaxbAdjectiveSynset>();
List<GJaxbLetter>           gJaxbLetters          = new ArrayList<GJaxbLetter>();
List<GJaxbWordSense>        gJaxbwordSenseList    = new ArrayList<GJaxbWordSense>();

public void initiate() throws Exception {

  ontologyFile = new File("WordNet.owl");

  ontoBuilder = new SAXBuilder(false);

  try {
    document = ontoBuilder.build("WordNet.owl");
  } catch (JDOMException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
  } catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
  }

  rootElement = document.getRootElement();
  loop = rootElement.getChildren().size();
  elementsList = new ArrayList<Element>();
  elementsList = rootElement.getChildren();
  rootNameSpace = rootElement.getNamespace();
  System.out.println("The name of rootElement: " + rootElement.getName());
  System.out.println("The number of rootElement's children: " + loop);

  mdbd = new MongoDBDao();
  mdbd.initiate();
  categoryElements();

}
public void storeToMongoDB() throws Exception{
  mdbd.storeAdjectiveSynsetToMongoDB(gJaxbAdjectiveSynsets);
  mdbd.storeNounSynsetToMongoDB(gJaxbNounSynset);
  mdbd.storeVerbSynsetToMongoDB(gJaxbVerbSynsets);
  mdbd.storeWordsToMongoDB(gJaxbLetters);
  mdbd.storeWordSenseToMongoDB(gJaxbwordSenseList);
}
```

Figure Annex-7: Extracting useful information from “WordNet.owl” file.

The analysis part of the “WordNet.owl” file uses the functions provided by “JDOM” jar. A variable (type is File) is defined to read the “WordNet.owl” file into the software system. Then a variable is defined as a “pointer”; this “pointer” is used to traverse the whole “WordNet.owl” file. Then, all the useful information stored in this file could be extracted from it and stored in the structures that stand by java variables. Finally, these variables, which store the useful information extracted the “WordNet.owl”, could be used to store into the MongoDB. The code fragment for storing java variables into MongoDB is shown in Figure Annex-8.

```
private Mongo             mg = null;
private DB                db;
private DBCollection      nounsynset;
private DBCollection      verbsynset;
private DBCollection      adjectivesynset;
private DBCollection      words;
private DBCollection      wordsense;

private byte[]            javaObjectByte;
private InputStream       inputStream;
private ObjectInputStream in;

@SuppressWarnings("deprecation")
public void initiate() throws UnknownHostException {
  mg = new Mongo();
  db = mg.getDB("SemanticThesaurus");
  nounsynset = db.getCollection("nounsynset");
  verbsynset = db.getCollection("verbsynset");
  adjectivesynset = db.getCollection("adjectivesynset");
  words = db.getCollection("words");
  wordsense = db.getCollection("wordsense");
}

public void storeNounSynsetToMongoDB(List<GJaxbNounSynset> gns)
    throws Exception {
  for (int i = 0; i < gns.size(); i++) {
    DBObject dbo = new BasicDBObject();
    ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
    ObjectOutputStream out = new ObjectOutputStream(outputStream);
    out.writeObject(gns.get(i));
    dbo.put("JavaObject", outputStream.toByteArray());
    out.close();
    outputStream.close();
    nounsynset.insert(dbo);
  }
}
```

Figure Annex-8: Storing content into MongoDB.

The functions of operating MongoDB database are provided by external jars, which are involved into AMTM_ST by Maven 2.0. The useful information stored in the java variables are written into MongoDB as data flow. MongoDB is a database management software tool; many databases could be defined and created in it (with a specific name), and each database could contain several tables. The data (information) is written as data flow into the specific database and table based on their names. In Figure Annex-9, only the connecting part (one insert method) between java program and MongoDB tool is shown.

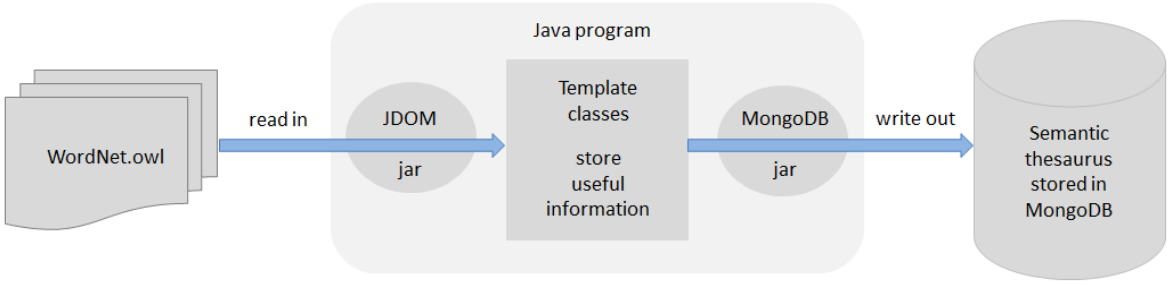


Figure Annex-9: Pre-treatment for doing semantic relation detecting.

All the content presented above is the preparation work of doing semantic relation detecting in AMTM-SS. It reveals the process of building semantic thesaurus for AMTM-SS to use. Figure Annex-9 shows simply the work done by the two code fragments shown in Figure Annex-7 and Figure Annex-8.

The original content of semantic thesaurus is stored in "WordNet.owl" file. Java program adopts functions provided by "JDOM" jar to read in the useful information from "WordNet.owl" to its template classes; then these template classes (with useful content in them) are written into MongoDB as data flow by the functions provided by the specific external jars. The semantic thesaurus stored in MongoDB is used by AMTM-SS to do the semantic relation detecting for building automatically model transformation mappings.

The java program of transforming content from "WordNet.owl" file to the database "semantic thesaurus" in MongoDB is complex and reduces the efficiency of AMTM-SS. Now, here comes the question "why do not use the 'WordNet.owl' file directly as the semantic thesaurus"? The answer to this question concerns two main aspects: (i) structure and validity of the content in semantic thesaurus, and (ii) efficiency of doing semantic relation detecting.

- Structure and validity of the content in semantic thesaurus. The content stored in WordNet has been illustrated in the former chapter; the "WordNet.owl" file (in ontology format) contains all the content presented in the WordNet. So, not all the content in "WordNet.owl" file is useful to do model transformation. Furthermore, the content stored in "WordNet.owl" file has its specific structure, which is not designed to be used specially for model transformation domain. So, a new semantic thesaurus with more relevant content that are stored with specially designed structure is needed.
- Efficiency of doing semantic relation detecting. As explained in the former chapters, semantic relation detecting relies on the basis of a huge semantic thesaurus. However, one of the problems of java programming language is the inefficiency of processing huge files. On the other hand, with the special index and search mechanism, database provides an efficient solution to process huge data. Comparing to the traditional relational database, MongoDB has its own advantage in processing the data. In MongoDB, The mechanism of storing and retrieving data has high efficiency. So, choosing MongoDB as the carrier of semantic thesaurus is a better solution than just use the "WordNet.owl" file.

It is true that it takes time and resource to transform "WordNet.owl" file into database in MongoDB, but this process is only needed to be executed one time. After creating the "semantic thesaurus" database in MongoDB, it could be used in anywhere and anytime. Furthermore, the actions of enriching and modifying the semantic thesaurus becomes operations (i.e. add, delete, update and query) on database. Comparing to processing huge files, java programs take less resource (e.g. main memory 'RAM' and CPU time) in doing database operations. So, the step of transformation the content (necessary to AMTM) from "WordNet.owl" file to MongoDB is meaningful in AMTM-SS.

The semantic relation detecting mechanism in AMTM-SS depends on the querying operations of the semantic thesaurus "AMTM_ST" stored in MongoDB. The code fragment, which focuses on these querying operations, is shown in Figure Annex-10.

```
private MongoDBDao mdbd;

private String sourceWord;
private String targetWord;

private List<String> sourceWordsenses;
private List<String> sourceSynsets;

public HypernymCalculation(String sourceWord, String targetWord){
  this.sourceWord = sourceWord;
  this.targetWord = targetWord;
}

public void initiate() throws UnknownHostException{
  mdbd = new MongoDBDao();
  mdbd.initiate();
  sourceWordsenses = new ArrayList<String>();
  sourceSynsets = new ArrayList<String>();
  targetWordsenses = new ArrayList<String>();
  targetSynsets = new ArrayList<String>();
}

public double getHypernymRelation() throws ClassNotFoundException, IOException{

  sourceWordsenses = mdbd.querySpecificWord(sourceWord);
  for(int i=0;i<sourceWordsenses.size();i++){
    sourceSynsets.add(mdbd.querySpecificWordSense(sourceWordsenses.get(i)));
  }

  targetWordsenses = mdbd.querySpecificWord(targetWord);
  for(int i=0;i<targetWordsenses.size();i++){
    targetSynsets.add(mdbd.querySpecificWordSense(targetWordsenses.get(i)));
  }

  for(int i=0;i<sourceSynsets.size();i++){
    for(int j=0;j<targetSynsets.size();j++){
      System.out.println("The synset comparation pair: " + sourceSynsets.get(i) + " & " + targetSynsets.get(j));
      if(sourceSynsets.get(i).equals(targetSynsets.get(j))){
        return synonymValue;
      }
    }
  }
```

Figure Annex-10: Pre-treatment for doing semantic relation detecting.

The querying operations could be divided into four groups: querying specific word, query word senses of specific word, query synsets of specific word sense, and query specific semantic relations between two specific synsets. All these query functions use loop to implement and concerns database (built in MongoDB) operations. The exact functions that operate on database are shown in Figure Annex-11 as a fragment of code. This code fragment only shows two main database operations: searching all the word senses for a specific words and search the specific synset for a word sense.

```
public List<String> querySpecificWord(String word) throws IOException,
    ClassNotFoundException {
  DBCursor wordsCur = words.find();
  int number = wordsCur.size();
  wordsenses = new ArrayList<String>();

  for (int i = 0; i < number; i++) {

    DBObject object = wordsCur.next();
    javaObjectByte = (byte[]) object.get("JavaObject");
    inputStream = new ByteArrayInputStream(javaObjectByte);
    in = new ObjectInputStream(inputStream);
    GJaxbLetter readWord = (GJaxbLetter) in.readObject();
    in.close();
    inputStream.close();
    if (readWord.getLabel().equalsIgnoreCase(word)) {
      System.out.println("The " + word + " has "
          + readWord.getSenseKey().size() + " senseKeys");
      for (int j = 0; j < readWord.getSenseKey().size(); j++) {

        String temp = readWord.getSenseKey().get(j);
        temp = temp.substring(15);
        System.out.println("The " + word + " has sense of: " + temp);
        wordsenses.add(temp);
      }
      break;
    }
  }
  return wordsenses;
}

public String querySpecificWordSense(String senseKey) throws IOException,
    ClassNotFoundException {
  DBCursor wordsenseCur = wordsense.find();
  int number = wordsenseCur.size();

  for (int i = 0; i < number; i++) {

    DBObject object = wordsenseCur.next();
    javaObjectByte = (byte[]) object.get("JavaObject");
    inputStream = new ByteArrayInputStream(javaObjectByte);
    in = new ObjectInputStream(inputStream);
    GJaxbWordSense readWordSense = (GJaxbWordSense) in.readObject();
    in.close();
    inputStream.close();
    if (readWordSense.getLabel().equalsIgnoreCase(senseKey)) {

      String temp = readWordSense.getSynset();
      temp = temp.replace("#", "");
      System.out.println("The " + senseKey + " belongs to the synset of "
          + temp);
      return temp;
    }
  }
  return null;
}
```

Figure Annex-11: Functions operate on database in MongoDB.

In Figure Annex-11, two main variables: “DBCursor” and “DBObject” are defined to get access to the database created in MongoDB. “DBCursor” concerns a specific table defined in database (associate by the table name), and “DBObject” reads a record in table. The two variables work together to get any records defined in any tables in the databases of MongoDB. Then the two functions define mechanism to select the real records that are required according to the parameters passed to them. Finally, the selected records are stored into variables and returned to the calling functions. The final purpose of semantic relation detecting is to define the relation between two words; so, after locating two groups of synsets (belong to the source word and target word, respectively), semantic relations detecting between synsets

pairs are needed. The code fragment, which concerns the synsets comparing, is shown in Figure Annex-12.

```
int numberNs = nounSynsetCur.size();
int numberVs = verbSynsetCur.size();
int numberAs = adjectiveSynsetCur.size();

antonymSynset = new ArrayList<String>();

for (int i = 0; i < numberNs; i++) {
  DBObject object = nounSynsetCur.next();
  javaObjectByte = (byte[]) object.get("JavaObject");
  inputStream = new ByteArrayInputStream(javaObjectByte);
  in = new ObjectInputStream(inputStream);
  GJaxbNounSynset readNounSynset = (GJaxbNounSynset) in.readObject();
  in.close();
  inputStream.close();

  if (readNounSynset.getId().equalsIgnoreCase(synset)) {
    for (int j = 0; j < readNounSynset.getAntonym().size(); j++) {
      antonymSynset.add(readNounSynset.getAntonym().get(j));
    }
    break;
  }
}

for (int i = 0; i < numberVs; i++) {
  DBObject object = verbSynsetCur.next();
  javaObjectByte = (byte[]) object.get("JavaObject");
  inputStream = new ByteArrayInputStream(javaObjectByte);
  in = new ObjectInputStream(inputStream);
  GJaxbVerbSynset readVerbSynset = (GJaxbVerbSynset) in.readObject();
  in.close();
  inputStream.close();
  if (readVerbSynset.getId().equalsIgnoreCase(synset)) {
    for (int j = 0; j < readVerbSynset.getAntonym().size(); j++) {
      antonymSynset.add(readVerbSynset.getAntonym().get(j));
    }
    break;
  }
}
for (int i = 0; i < numberAs; i++) {
  DBObject object = adjectiveSynsetCur.next();
  javaObjectByte = (byte[]) object.get("JavaObject");
  inputStream = new ByteArrayInputStream(javaObjectByte);
  in = new ObjectInputStream(inputStream);
  GJaxbAdjectiveSynset readAdjectiveSynset = (GJaxbAdjectiveSynset) in
      .readObject();
  in.close();
  inputStream.close();
  if (readAdjectiveSynset.getId().equalsIgnoreCase(synset)) {

    for (int j = 0; j < readAdjectiveSynset.getAntonym().size(); j++) {
      antonymSynset.add(readAdjectiveSynset.getAntonym().get(j));
    }
    break;
  }
}
return antonymSynset;
```

Figure Annex-12: Semantic relation detecting for specific synset.

Figure Annex-12 shows the fragment of codes that implement the function of detecting antonym semantic relation for a specific synset. All the synsets that have antonym relations with the synset passed in as parameter will be return back a group. Since the number of synset records is huge, synsets are divided and stored in three groups: noun synset, verb synset and adjective synset. So, detecting semantic relations for a specific synset, the three groups of synsets are needed to be search one by one. One word could have

several semantic meanings (e.g. as noun, as verb and as adjective), and thus it could belong to different synset groups.

The syntactic and semantic checking measurements have been presented and explained, respectively. However, for the reason "syntactic checking could replace part of semantic checking with less time and resource occupied", the two checking methods should work together in AMTM-SS as a whole process. The code fragment, which combines them together, is shown in Figure Annex-13.

```
private static SyntaxRelationBTW synRV;
private static SemanticRelationCalculation semRV;

private static HypernymCalculationAmongWords semRCAV;

private double syntacticValue = 0;
private double semanticValue = 0;

private double factor = 0.1;
private double weight = 1;

private String sourceWord;
private List<String> wordlist = new ArrayList<String>();

private List<Double> synValues = new ArrayList<Double>();
private List<Double> semValues = new ArrayList<Double>();

public void initiate(String sourceWord, String targetWord){
  synRV = new SyntaxRelationBTW(sourceWord,targetWord);
  semRV = new SemanticRelationCalculation(sourceWord,targetWord);
  try {
    semRV.initiate();
  } catch (UnknownHostException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
  }
}

public double getSSValue(){
  syntacticValue = synRV.getSyntacticRelation();
  if(syntacticValue>=weight){
     return 1.0;
  }
  else{
    try {
      semanticValue = semRV.getHypernymRelation();
      System.out.println(semanticValue);
    } catch (ClassNotFoundException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    } catch (IOException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    }
    return factor*syntacticValue + (1-factor)*semanticValue;
  }
}
```

Figure Annex-13: Semantic relation detecting for specific synset.

A function "getSSValue()" has been defined to determine the executing sequence and execution mechanism between syntactic and semantic checking measurements. Furthermore, in this code fragment a special variable "factor" has been defined; this variable concerns the weights of semantic and syntactic within one model transformation process. Until now, all the functional code fragments shown above are only concerned with comparing within a pair of words. The final purpose of AMTM-SS is: "building model transformation mappings and generating target model"; so, combining the two checking measurements into model transformation process is another important functional part for AMTM-SS. Figure Annex-14 shows the functional code fragment concerns about this part.

```
private Model                                sourceModel     = new Model();
private Model                                targetModel     = new Model();
private double[][]                           nodesMatrix;
private double[][]                           edgesMatrix;
private List<Match_PairOfNodes>              elementsMatchPairs;
private List<Match_PairOfProperties>         propertiesMatchPairs;

private static Semantic_Syntactic_Relation ssr;
double                                       weight_name     = 0.3;
double                                       weight_property = 0.7;
double weight_EdgeName = 0.6;
double weight_EdgeFromTo = 0.4;

public MappingRulesGenerated() {

}

public MappingRulesGenerated(Model sm, Model tm) {
  this.sourceModel = sm;
  this.targetModel = tm;
  ssr = new Semantic_Syntactic_Relation();
  elementsMatchPairs = new ArrayList<Match_PairOfNodes>();
  propertiesMatchPairs = new ArrayList<Match_PairOfProperties>();
}

public void execute(){
  getSSValueOnNodes();
  hybridMatching();
}
```

Figure Annex-14: Combining two checking methods into model transformation process.

The inputs "model sets" have been analyzed and useful information contained has been extracted into specific template, which is implemented as java class. Syntactic and semantic checking measurements are used on the two instances "sourceModel" and "targetModel". As illustrated in the former chapters, model transformation mappings are built among the models' items: element and properties. For each potential matching pair (element-to-element, property-to-property and element-to-property), a relational value, which is calculated based on items' names and maybe other relevant attributes, is assigned. For different potential matching pairs, different comparing mechanisms (calculation formulas and parameters defined in the formulas) are used. The basis for all these comparing mechanisms is syntactic and semantic checking methods, which are shown and explained in this section.

As a short conclusion of this section, the implementations of all the main functional parts have been mentioned here. These functional parts contain: (i) building semantic thesaurus "AMTM_ST", (ii) analyzing inputs model sets, (iii) connecting "AMTM_ST", (iv) syntactic checking algorithm, (v) semantic checking algorithm, and (vi) combining the two checking methods into model transformation process. The core code fragments of these functions have been shown. At this moment, the efficiency of the algorithms mentioned above have been paid attention; however, more work is needed to be done to improve the algorithm and also for the whole AMTM-SS system.

III. System test

The testing part is very important in the developing process of software systems. There are two main methods of doing this step are: validation and verification. Similar to the system implementation section, this section focuses on several main functional test parts. These functional parts are: "inputs model sets analysis", "syntactic checking algorithm", "semantic checking algorithm", and an instance of comparing

two elements. For each of the functional test part, a simple test case is shown; furthermore, the executed results of java programs are also shown by screenshot of the console.

III.1 Inputs model sets analysis module test

The original inputs to AMTM-SS are model sets. The aim of AMTM-SS is to build potential model transformation mappings (on meta-model layer) automatically based on semantic and syntactic checking measurements; so, the perfect inputs to AMTM-SS are two meta-models: source meta-model and target meta-model. For this test module, the test case inputs are two meta-models that are built with XSD techniques. Part of the contents of the two meta-models is shown in Figure Annex-15.

Source meta-model

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<metaModel xmlns="http://www.gind.emac.fr/modeler/metaModel">
    <backgroundColor>#fdf2cf</backgroundColor>
    <title>TreatmentSystem</title>
    <version>1.0-SNAPSHOT</version>
    <backgroundColor>#fdf2cf</backgroundColor>
    <edgeType>FIXED</edgeType>
    <node>
        <type>Demand</type>
        <view>/webjars/ioda/treatment_system/images/model/nodes/Demand.svg
        </view>
        <role>Demand</role>
        <role>ResourceEvent</role>
        <magnets>
            <managedPosition>RECT</managedPosition>
        </magnets>
        <property name="name" defaultvalue="Demand_${cpt}" type="string"
            optional="false" visible="true" posXFromParentCenter="0"
            posYFromParentCenter="0" fontSize="20">
            <description>description</description>
        </property>
        <property name="near" visible="false" optional="false" type="string"
            readonly="true">
            <values>
                <dynamicValues>
                    <buttonName>find</buttonName>
                    <action>ontologyLink.getNearConcepts()</action>
                </dynamicValues>
            </values>
        </property>
        <property name="same as" visible="false" optional="false"
            type="string" readonly="true">
            <values>
                <dynamicValues>
                    <buttonName>find</buttonName>
                    <action>ontologyLink.getSameAsConcepts()</action>
                </dynamicValues>
            </values>
        </property>
```

Target meta-model

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<metaModel xmlns="http://www.gind.emac.fr/modeler/metaModel">
    <title>CollaborativeProcess</title>
    <version>1.0-SNAPSHOT</version>
    <icon></icon>
    <fileExtension>poo</fileExtension>
    <backgroundColor>#fdf2cf</backgroundColor>
    <edgeType>FIXED</edgeType>
    <node>
        <type>PartnerPool</type>
        <view>/webjars/ioda/collaborative_process/images/model/nodes/PartnerPool.svg
        </view>
        <role>PartnerPool</role>
        <role>Pool</role>
        <magnets>
            <managedPosition>RECT</managedPosition>
        </magnets>
        <property name="PartnerName" defaultvalue="20" type="string"
            optional="false" visible="false" posXFromParentCenter="0"
            posYFromParentCenter="0" fontSize="20">
            <description>20</description>
        </property>
        <property name="name" defaultvalue="Partner Pool ${cpt}"
            type="string" optional="false" visible="true" posXFromParentCenter="-${parentWidth}/2 + 30"
            posYFromParentCenter="0" rotate="-90" fontSize="32">
            <description>description</description>
        </property>
    </node>
    <node>
        <type>SIMPool</type>
        <view>/webjars/ioda/collaborative_process/images/model/nodes/SIMPool.svg
        </view>
        <role>SIMPool</role>
        <role>Pool</role>
        <magnets>
            <managedPosition>RECT</managedPosition>
        </magnets>
        <property name="name" defaultvalue="SIM Pool ${cpt}" type="string"
            optional="false" visible="true" posXFromParentCenter="-${parentWidth}/2 + 30"
            posYFromParentCenter="0" rotate="-90" fontSize="32">
            <description>description</description>
        </property>
    </node>
    <node>
        <type>EndEvent</type>
        <view>/webjars/ioda/collaborative_process/images/model/nodes/EndEvent.svg
        </view>
        <role>EndEvent</role>
        <role>SIMComponents</role>
        <magnets>
            <managedPosition>CROSS</managedPosition>
```

Figure Annex-15: Test case inputs: two meta-models in XSD format.

XSD could be regarded as a special file format, which is always used as the technique to build meta-model. The target of this test case is to read these two files into AMTM-SS, then analyze and extract useful information from the two files to build specific meta-model that are conformed to the MMM. The test result of this case is shown in Figure Annex-16.

```
1  package util;
2
3  public class MetamodelAnalysisTest {
4
5    public static void main(String[] args){
6
7      FileReader ff;
8      ff = new FileReader("TreatmentSystemMetaModel.xml","CollaborativeProcessMetaModel.xml");
9
10     if(ff.getSourceFile().exists()){
11       System.out.println("Read file successful!");
12     }else{
13       System.out.println("File does not exist!");
14     }
15
16    ff.excute();
17   }
18 }
19
```

Problems @ Javadoc Declaration Console

```
<terminated> MetamodelAnalysisTest [Java Application] C:\Program Files\Java\jre1.8.0_20\bin\javaw.exe (Aug 17, 2015, 2:40:25 PM)
Read file successful!
This specific source meta-model has: 35 elements
The 1 node: Demand has 4 properties;it has: 2 roles
The 2 node: ResourceStatus has 4 properties;it has: 2 roles
The 3 node: Offer has 4 properties;it has: 2 roles
The 4 node: CrisisLocation has 4 properties;it has: 2 roles
The 5 node: SupplierLocation has 4 properties;it has: 2 roles
The 6 node: ServiceOfResourceRequisition has 4 properties;it has: 2 roles
The 7 node: ServiceOfResourceTransport has 4 properties;it has: 2 roles
The 8 node: ServiceOfActor has 4 properties;it has: 2 roles
The 9 node: MediationService has 4 properties;it has: 2 roles
The 10 node: ConsumableResource has 7 properties;it has: 2 roles
The 11 node: Condition has 4 properties;it has: 1 roles
The 12 node: ResourceAllocation has 5 properties;it has: 1 roles
The 13 node: Actor has 4 properties;it has: 1 roles
The 14 node: DurableResource has 6 properties;it has: 2 roles
The 15 node: Risk has 4 properties;it has: 2 roles
The 16 node: Consequence has 4 properties;it has: 2 roles
The 17 node: Role has 4 properties;it has: 1 roles
This specific target meta-model has: 19 elements
The 1 node: PartnerPool has 2 properties;it has: 2 roles
The 2 node: SIMPool has 1 properties;it has: 2 roles
The 3 node: EndEvent has 1 properties;it has: 2 roles
The 4 node: IntermediaryEvent has 1 properties;it has: 2 roles
The 5 node: StartEvent has 1 properties;it has: 2 roles
The 6 node: SIMTask has 1 properties;it has: 3 roles
The 7 node: GatewayO has 1 properties;it has: 2 roles
The 8 node: Gateway+ has 1 properties;it has: 2 roles
The 9 node: Gatewayx has 1 properties;it has: 2 roles
The 10 node: PartnerTask has 3 properties;it has: 2 roles
```

Figure Annex-16: Executing results of the test case.

The executing result shows that the source meta-model contains 17 nodes and the target meta-model contains 10 nodes. The name and property group of each node are also read out. So, the templates for both the source and target meta-models could be generated based on the analysis. Two points needed to be made clearly in this test case: (i) meta-models could follow different formats (XSD file is only one of the common accepted standards); "JDOM" is used to analyze these inputs model sets, and (ii) not all the information contained in the original meta-models is useful to AMTM-SS, only the information needed to build specific meta-models is extracted and used.

III.2 Syntactic checking measurement module test

This module focuses on testing the "Levensthein distance" algorithm. The pretreatment step in syntactic checking phase is also implemented, but it only concerns several special situations (there is no need to do the test). The inputs of this test module are two words, and the output of it should be a value between o and 1, which stands for the syntactic similarity between the two words. The syntactic similarity value is calculated based on the "Levensthein distance" between two words and the length of them. Here, two word pairs: person & perfect and artist & maker are taken in as test cases. The executing results are shown in Figure Annex-17.

```
5  public class LevenshteinTest {
6
7    public static void main(String[] args){
8
9      SyntaxRelationBTW srb1 = new SyntaxRelationBTW("person","perfect");
10
11     SyntaxRelationBTW srb2 = new SyntaxRelationBTW("artist","maker");
12
13     System.out.println("The syntactic similarity between word person and perfect is: " + srb1.getSyntacticRelation());
14     System.out.println("The syntactic similarity between word artist and maker is: " + srb2.getSyntacticRelation());
15
16
17   }
18 }
19
```

Javadoc Problems Declaration Console

```
<terminated> LevenshteinTest [Java Application] C:\Program Files\Java\jre1.8.0_20\bin\javaw.exe (Aug 17, 2015, 4:12:58 PM)
The syntactic similarity between word person and perfect is: 0.4285714285714286
The syntactic similarity between word artist and maker is: 0.0
```

Figure Annex-17: Test case for syntactic checking measurement.

The executing result shows that the syntactic similarity between “person” and “perfect” is 0.4286 and between “artist” and “maker” is 0. Intuitively speaking, this result is acceptable. This test also reveals the truth that “Levenshtein distance” algorithm could not detect the semantic relation at all. The syntactic checking algorithm is easy to be implemented and the mechanism is easy to understand. So, the test case is easy to design and carry out.

III.3 Semantic checking measurement module test

The most important and complex functional part in AMTM-SS is the semantic relation detecting. This part concerns operations on database stored in MongoDB; the efficiency and accuracy are two main aims to be tested in this part. The inputs of this test module are two words, and the outputs should be a value that stands for the semantic relation between the two words. As illustrated above, the hypernym (hyponym) semantic relations among several words maintained in “AMTM_ST” are known. The inputs of this test case use three pairs of words: “person & author”, “artist & creator” and “good & bad”. The executing result of the first test case “person & author” is shown in Figure Annex-18.

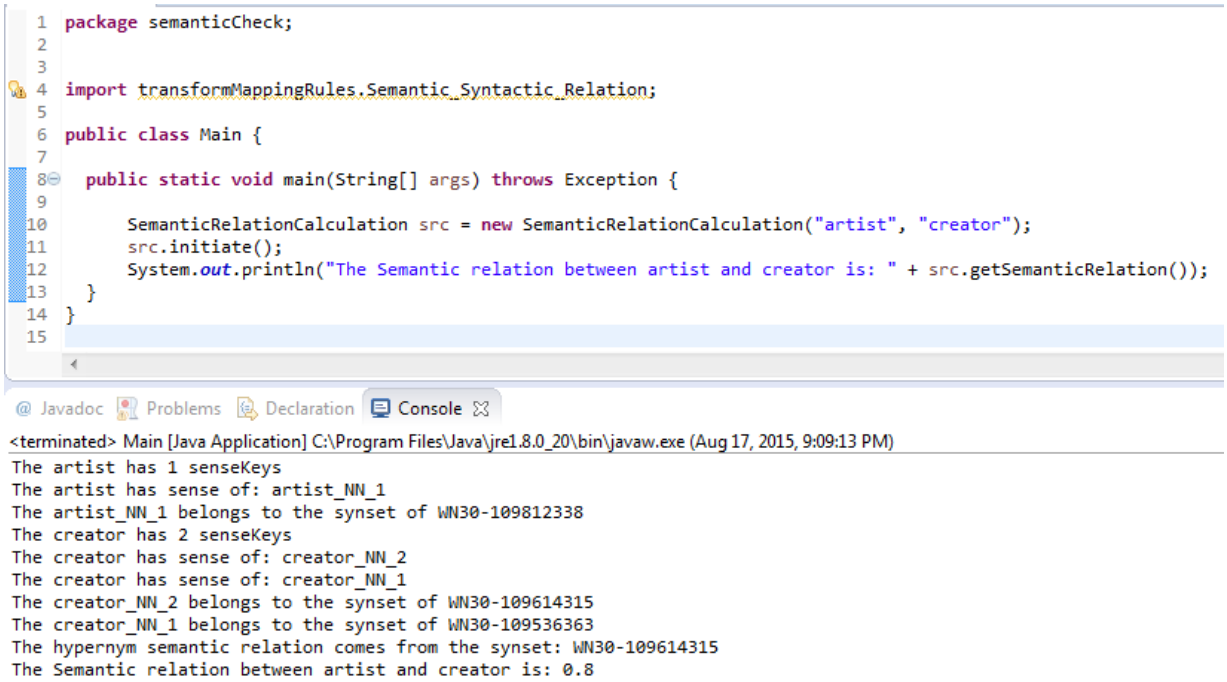
```
8    public static void main(String[] args) throws Exception {
9
10       SemanticRelationCalculation src = new SemanticRelationCalculation("person", "author");
11       src.initiate();
12       System.out.println("The Semantic relation between person and author is: " + src.getSemanticRelation());
13   }
14 }
```

Javadoc Problems Declaration Console

```
<terminated> Main [Java Application] C:\Program Files\Java\jre1.8.0_20\bin\javaw.exe (Aug 17, 2015, 6:27:43 PM)
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 author has 3 senseKeys
The author has sense of: author_NN_1
The author has sense of: author_NN_2
The author has sense of: author_VB_1
The author_NN_1 belongs to the synset of WN30-110794014
The author_NN_2 belongs to the synset of WN30-110126177
The author_VB_1 belongs to the synset of WN30-201704452
The iterative hyponym semantic relation comes from the synset: WN30-100007846
The Semantic relation between person and author is: 0.36
```

Figure Annex-18: Semantic relation detecting test case between “person & author”.

The result shown in the console presents that “person” has three semantic meanings (word senses) and belongs to three synset and “author” also has three semantic meanings and belongs to three synsets. The semantic relation between them is “iterative hypernym” (person-to-author), and the semantic relation value between the two words is 0.36 (0.6*0.6). It means that the possibility of transforming “person” to “author” is 0.36 on semantic aspect: because “person” has a broad meaning than the more specific meanings conveyed by “author”. As an opposite test case, the second comparing group is between “artist” and “creator”. The executing result of this test group is shown in Figure Annex-19.



```
1  package semanticCheck;
2
3
4  import transformMappingRules.Semantic_Syntactic_Relation;
5
6  public class Main {
7
8    public static void main(String[] args) throws Exception {
9
10       SemanticRelationCalculation src = new SemanticRelationCalculation("artist", "creator");
11       src.initiate();
12       System.out.println("The Semantic relation between artist and creator is: " + src.getSemanticRelation());
13   }
14 }
15
```

```
@ Javadoc  Problems  Declaration  Console
<terminated> Main [Java Application] C:\Program Files\Java\jre1.8.0_20\bin\javaw.exe (Aug 17, 2015, 9:09:13 PM)
The artist has 1 senseKeys
The artist has sense of: artist_NN_1
The artist_NN_1 belongs to the synset of WN30-109812338
The creator has 2 senseKeys
The creator has sense of: creator_NN_2
The creator has sense of: creator_NN_1
The creator_NN_2 belongs to the synset of WN30-109614315
The creator_NN_1 belongs to the synset of WN30-109536363
The hypernym semantic relation comes from the synset: WN30-109614315
The Semantic relation between artist and creator is: 0.8
```

Figure Annex-19: Semantic relation detecting test case between “artist & creator”.

The testing result shows that the word “artist” only has one word sense and belongs to one synset, and the word “creator” has two word senses and belongs to two synsets. The semantic relation between the two words is “hypernym” (artist-to-creator), and the semantic relation value is: 0.8. It means that the possibility of transforming “artist” to “creator” is high on semantic aspect. For the reason “artist is one kind of creator”; in other words, the concept of “creator” contains the concept of “artist”. Here, it is necessary to make clearly that the semantic relation between “artist” and “creator” is not the same as the semantic relation between “creator” and “artist”. In model transformation domain, the transformation concerns the range of concepts. For instance, a student must be a person but a person might be student; so student could be transformed to person while person might not be transformed to student. So, in AMTM, semantic relations “hypernym” and “hyponym” are distinguished. The final test case concern a pair of adjective words: good and bad. The executing result of this pair is shown in Figure Annex-20.

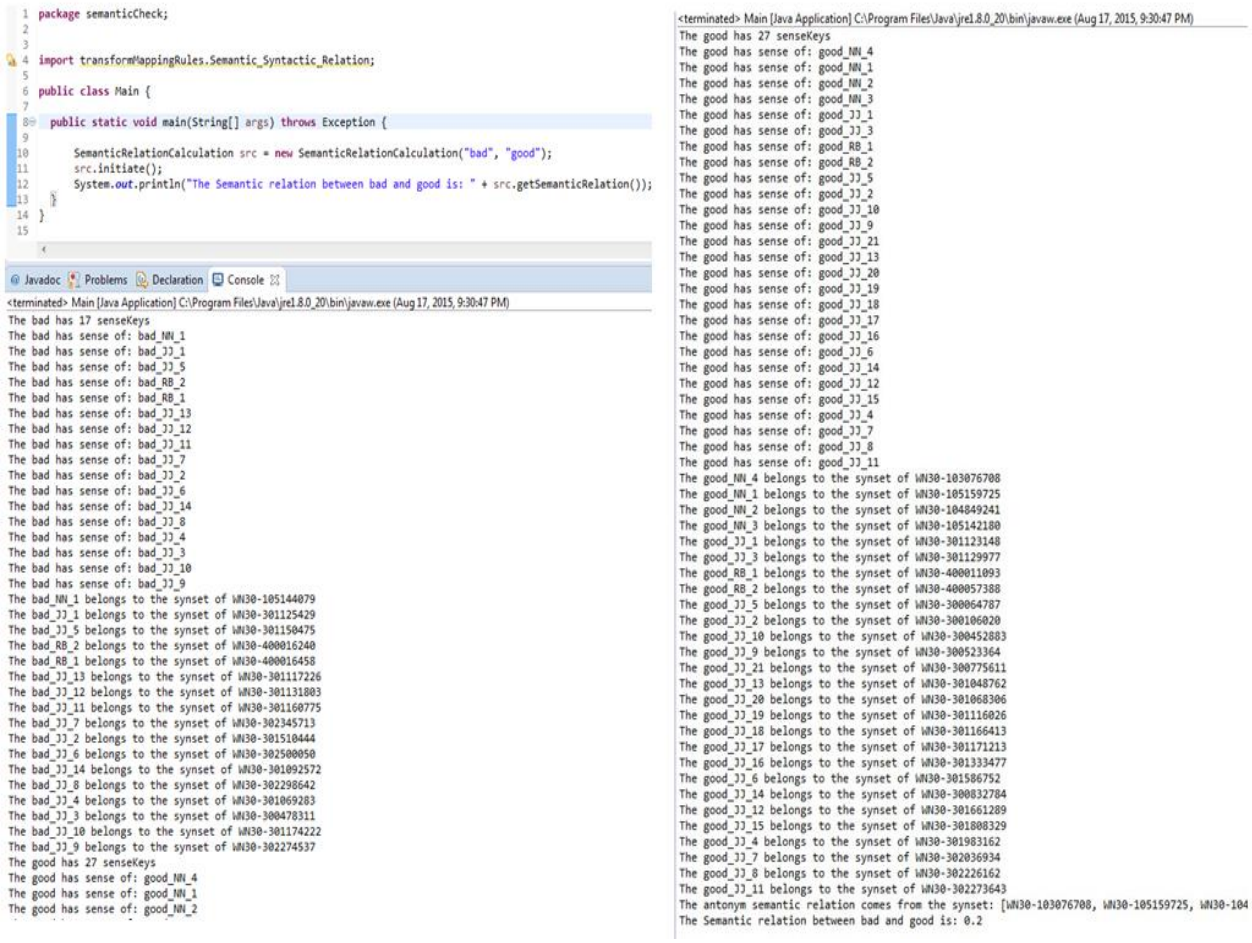


Figure Annex-20: Semantic relation detecting test case between "bad & good".

The testing result shows that the word "bad" has seventeen word sense and thus belongs to seventeen synsets, and the word "good" has twenty-seven word senses and belongs to twenty-seven synsets. The semantic relation between the two words is "antonym", and the semantic relation value is: 0.2. This value is assigned manually before executing this program; when two words have antonym semantic relation between each other, the better assigned value may be negative.

Three test cases are presented in this subsection; they show the process of detecting three different semantic relations: iterative hypernym, iterative hyponym and antonym between pairs of comparing words. The mechanism of detecting other semantic relations is similar to the ones shown in these use cases. The only different aspect is the assigned semantic relation values.

IV. References

(Arnold et al, 1996) Arnold K, Gosling J, Holmes D, et al. The Java programming language[M]. Reading: Addison-wesley, 1996.

(Casey et al, 2006) Casey, J., Massol, V., Porter, B., & Sanchez, C. (2006). Better Builds with Maven.

(Chodorow, 2010) Chodorow, C. (2010). Introduction to mongodb. In Free and Open Source Software Developers' European Meeting (FOSDEM).

(Chodorow, 2013) Chodorow, K. (2013). MongoDB: the definitive guide. " O'Reilly Media, Inc.".

(Dietel, 2009) Dietel, P. (2009). Java how to program. PHI.

(Geer, 2005) Geer, D. (2005). Eclipse becomes the dominant Java IDE. Computer, 38(7), 16-18.

(Hunter, 2002) Hunter, J. (2002). JDOM makes XML easy. In Sun's 2002 Worlwide Java Developer Conference.

(McGuinness, 2004) McGuinness, D. L., & Van Harmelen, F. (2004). OWL web ontology language overview. W3C recommendation, 10(10), 2004.

(Membrey et al, 2010) Membrey, P., Plugge, E., & Hawkins, D. (2010). The definitive guide to MongoDB: the noSQL database for cloud and desktop computing. Apress.

(Murphy et al, 2006) Murphy, G. C., Kersten, M., & Findlater, L. (2006). How are Java software developers using the Elipse IDE?. Software, IEEE, 23(4), 76-83.

(Smart, 2005) Smart, J. F. (2005). An introduction to Maven 2. JavaWorld Magazine. Available at: http://www. javaworld. com/javaworld/jw-12-2005/jw-1205-maven. html.

(Vasilescu et al, 2013) Vasilescu, B., Filkov, V., & Serebrenik, A. (2013, September). StackOverflow and GitHub: associations between software development and crowdsourced knowledge. In Social Computing (SocialCom), 2013 International Conference on (pp. 188-195). IEEE.

(Vogel, 2014) Vogel, L., & IDE, E. J. (2014). Eclipse IDE tutorial.

(Wellings, 2004) Wellings, A. (2004). Concurrent and real-time programming in Java. John Wiley & Sons.

Résumé long en français de la thèse

Introduction

Dans le but de décrire ou mesurer un objet ou un système, il est possible d'utiliser différent termes avec chacun un référentiel propre. Par exemple, il est possible d'utiliser l'une des unités suivantes pour décrire une température : *Celsius, Fahrenheit* ou encore *Kelvin.* Ainsi 15 °C, 59 °F et 288.15°K sont des températures équivalentes. Un autre exemple de notre quotidien porte sur les dates. En effet, une même date peut être écrite de manières différentes tel que le 15 Septembre 2015 s'écrira *15/09/2015* en France, *09/15/2015* aux Etats-Unis d'Amérique et *2015-09-15* en Chine. Ce dernier exemple illustre une différence **syntaxique**, ou comment représenter le même sujet avec différents formats d'écriture. En complément à la différence syntaxique, il existe aussi une différence **sémantique** basée sur la possibilité d'utiliser des mots similaires afin de décrire un même sujet. Cette possibilité à plusieurs origines : les synonymes (*content, heureux)* l'inclusion (*véhicule – camionnette)*, la composition (*université – faculté*) et l'héritage (*enseignant – enseignant de chimie)*, etc.

De plus, le développement rapide de la Science et des technologies dans certains domaines rend le monde de « plus en plus petit » en facilitant les moyens de communication. D'une certaine manière, ces avancées permettent l'accroissement du nombre de collaboration à travers les pays, les organisations et les personnes en recherchent d'une solution optimale. Ainsi, les collaborations traditionnelles laisse petit à petit la place à un nouveau type de collaboration « plus volatile » ayant comme caractéristiques : d'avoir un cycle de vie très rapide (de sa constitution à sa dissolution), un partenariat opportuniste basé sur les compétences des partenaires (Touzi *et al.,* 2007). En conséquence, les partenaires évoluant dans ce type de collaboration sont hétérogènes par nature. En effet, ils sont des cultures différentes, ils parlent des langages différents, ils ont des méthodes de travail différentes et enfin des outils différents (notamment au niveau de la science de l'information et de la communication). Il est par ailleurs possible de caractériser ces collaborations de « Systèmes de Systèmes » d'après les critères définis par (Maier, 1998). En effet, ces collaborations ont les cinq critères suivants : (i) une indépendance managériale, (ii) une indépendance opérationnelle, (iii) les partenaires sont répartis géographiquement, (iv) ces collaborations sont évolutives dans le temps et (v) la collaboration a un comportement émergent basé sur la composition des comportements des partenaires. Parmi ces critères, l'indépendance managériale et opérationnelle impliquent que chaque partenaire travail en utilisant ses propres ressources et son propre vocabulaire. Ce nouveau type de collaboration fait par conséquent émerger de nouvelles problématiques et particulièrement : **Comment construire de façon efficace et efficiente une collaboration entre des partenaires hétérogènes de façon dynamique ?** Généralement, un des premiers obstacles à franchir est le problème de **partage et d'échange d'information entre les partenaires**. Une origine de cet obstacle repose sur les différences **syntaxique et sémantique** de représentation des données et des informations utilisées par les différents acteurs. Par conséquent, **faciliter la réconciliation syntaxique et sémantique entre différents formats est un premier pas vers la création efficiente et efficace des collaborations.**

L'ingénierie système basée sur les modèles, MBSE, est une approche permettant de faciliter les collaborations. Cette approche se base sur deux concepts clés : **les modèles** et **les transformation de modèles**. En effet, il est possible de représenter un système (ou sujet) en soulignant ces caractéristiques au travers de modèles puisque qu'un « **modèle** représentent les caractéristiques d'un système en fonction d'un point de vue spécifique » (Bézivin, 2006). Cette définition, similaire à celles proposées par (Venadat, 1999) et (Terrasse *et al.*, 2005), impliquent que chaque partenaire de la collaboration, pouvant être vu comme un système d'après la définition précédente du système de systèmes, peut être représenté par un ou plusieurs modèles en fonction des points de vue souhaités. Ainsi, les interactions ou les comparaisons entre partenaires peuvent être représentées par une transformation de modèle. Il existe plusieurs définitions d'une transformation de modèles. (Tratt, 2005) défini une transformation de modèles comme un programme qui mute un modèle source en un modèle cible. *L'Object Management Group* (OMG) défini une transformation de modèles dans le contexte d'une architecture dirigée par les modèles (MDA) comme un processus de conversion d'un modèle en un autre modèle portant sur le même système (Miller et al., 2003). (Kleppe et al, 2003) défini une transformation de modèles comme une génération automatique d'un modèle cible sur la base d'un modèle source selon une description de la transformation. Sur la base de ces définitions, il est possible de synthétiser une **transformation de modèle** comme étant **un processus de génération d'un modèle cible sur la base de modèle source**. Ainsi, dans le domaine de la collaboration, il est possible de représenter par des modèles les caractéristiques des partenaires, notamment leurs données et informations. Il est alors possible de transférer des

données ou informations entre les partenaires grâce à des transformations de modèles tels que représentés sur la figure 1.

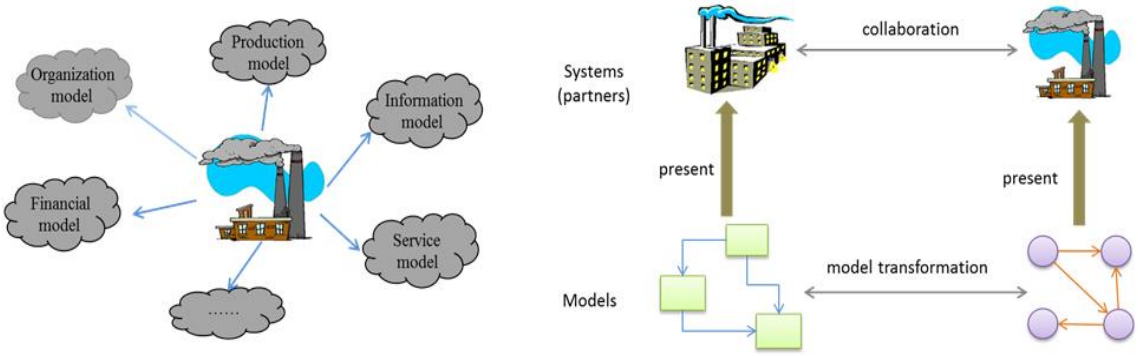


Fig. 1 Illustration de l'utilisation de modèles et de transformation de modèles en support de la collaboration.

Sur la base de l'idée d'utilisé une MBSE et dans le but de faciliter la réconciliation syntaxique et sémantique entre différents formats, cette thèse propose une méthodologie automatique de transformation de modèles (AMTM) basée sur une comparaison sémantique et syntaxique des éléments des modèles. Contrairement aux méthodologies existantes de transformation de modèles, AMTM ambitionne aussi de résoudre les problèmes liés aux collaborations inter-domaines. Pour cela, AMTM repose sur une mesure combinée de comparaison sémantique et syntaxique sur la base d'un méta-méta-modèle et d'un processus de transformation de modèles. L'articulation de cette thèse est présentée par la Fig. 2.

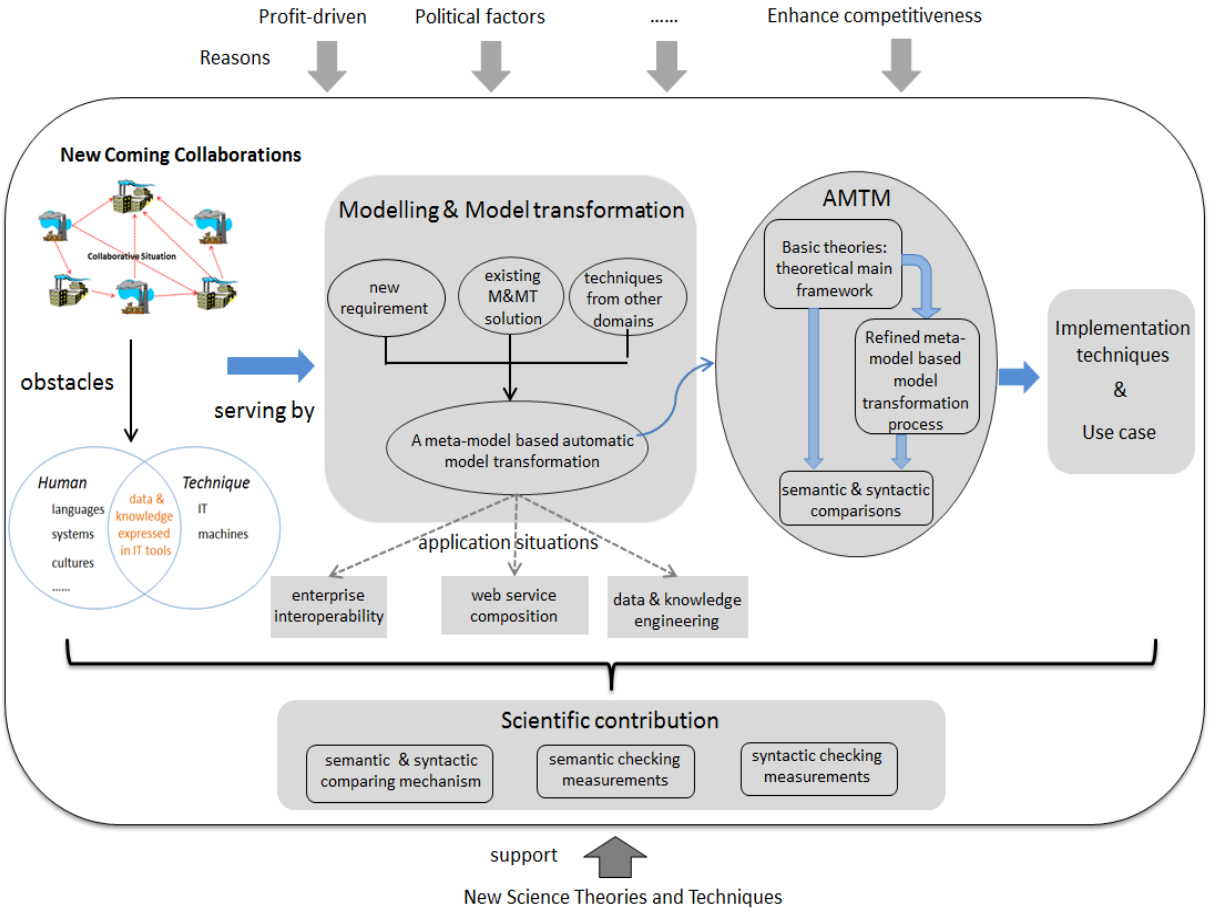


Fig. 2 Vue d'ensemble de cette thèse

Cette thèse est décomposée en huit chapitres. Le premier chapitre positionne la problématique de transfert d'information dans différent domaine tel que la collaboration inter-organisation, la composition de web-service ou

encore l'interopérabilité. Le deuxième chapitre présentera les notions de bases du domaine de la modélisation et de la transformation de modèles ainsi que les méthodologies existantes. Le troisième chapitre donnera une vue d'ensemble de l'approche AMTM ainsi que ses bénéfices par rapport aux méthodologies existantes. La méthodologie AMTM étant basée sur une comparaison sémantique et syntaxique des éléments des modèles : le chapitre quatre explique les différents mécanismes de comparaisons utilisés alors que le chapitre cinq est focalisé sur la comparaison sémantique et le chapitre six sur la comparaison syntaxique déployée dans AMTM. Avant de conclure, le chapitre sept explique l'implémentation technique de la preuve de concept ainsi que des résultats de tests effectués.

Conclusion

Cette thèse propose une méthodologie efficiente de transformation de modèle, nommée AMTM, pouvant être utilisée pour échanger des informations entre organisations hétérogènes et ainsi faciliter la mise en place de leur collaboration occasionnelle. La contribution de cette méthodologie porte sur l'automatisation totale ou partielle de la définition de transformation de modèles permettant ainsi de limiter les efforts humains, problème récurrent dans la définition des transformations de modèles. AMTM repose sur la comparaison de méta-modèle afin de définir les transformations de modèles, ainsi un méta-méta-modèle (MMM) a été défini afin de définir des correspondances syntaxique et sémantique entre les différents éléments des méta-modèles. De plus, AMTM est basée sur une approche itérative se basant sur l'utilisation d'une ontologie (AMTM_O). Cette ontologie sauvegarde les éléments des différents méta-modèles ainsi que les résultats des correspondances entre les éléments afin de pouvoir déduire des relations de transitivité entre les éléments des modèles ou encore de compléter la transformation de modèles par l'ajout de connaissance additionnelle. A chaque itération, quatre étapes de correspondances sont exécutées afin de résoudre notamment le problème de granularité : **au niveau des éléments, la comparaison hybride, la comparaison inter-niveau et enfin la comparaison auxiliaire**. L'ensemble de ces étapes est basé sur une comparaison syntaxique et sémantique. La comparaison sémantique repose sur une base de référence conséquence, nommé AMTM_ST, construit sur la base de WordNet alors que le la comparaison syntaxique est basée sur l'utilisation de plusieurs algorithmes combinés, notamment l'algorithme de « *Porter Streaming* » et celui de « *Leveinstein Distance* ». Ces résultats sont ensuite combinés grâce à six équations permettant de proposer les liens de correspondance entre les différents méta-modèles. Ces équations reposent sur des seuils permettant de paramétrer la méthodologie et ainsi essayer de l'optimiser.

Suite à cette thèse, certaines pistes améliorations peuvent encore être explorées :

- Réaliser une validation automatique des correspondances déduites
- Déduire automatiquement les règles de transformations entre les éléments des modèles (*i.e.* conversion, etc.)
- Compléter la base AMTM_ST en ajoutant de nouveaux types de relation sémantique, notamment les relations propres à un domaine spécifique.
- Améliorer la performance du programme développé comme preuve de concept.
- Enrichir la base AMTM_ST sur les bases de données spécifiques à un ou plusieurs domaines.

Avec ces améliorations, l'usage de la méthodologie AMTM pourra être élargi à la mutualisation des sources de données telles que les objets connectés ou encore l'open data. La Fig. 3 représente un possible usage de la méthodologie AMTM.

En effet la méthodologie AMTM pourrait être utilisée pour la conversion de données brutes en information (grâce à de la connaissance). Cette vision est cohérente avec les problématiques des collaborations inter-domaine puisque différents émetteurs de données (*i.e.* les capteurs, les ordinateurs ou encore les objets connectés) peuvent émettre de la donnée brute sur une région particulière ou encore un domaine particulier. Les données collectées doivent ensuite être comparées et ce malgré leurs hétérogénéités de format (structure de données) et de sémantique (termes utilisées).

Afin de pouvoir appliquer la méthodologie AMTM à ce contexte, chaque donnée collectée doit être vue comme un modèle. Ainsi, les liens de correspondances entre les données pourraient être déduits ou proposés et ainsi faciliter la mise en relation des données et ce indépendamment du domaine d'étude.

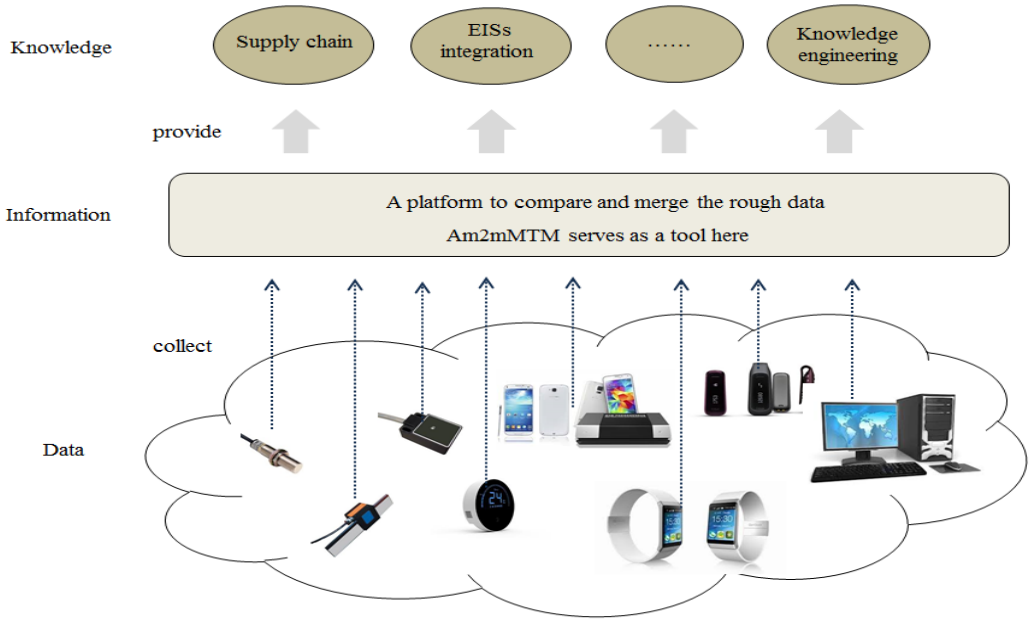


Fig. 3 Usage future de AMTM

Du modèle à la problématique de cette thèse :

Une transformation de modèles est un processus qui agit sur les modèles. Le modèle et la modélisation jouent donc un rôle important dans le domaine de recherche portant sur la transformation de modèle. Ainsi cette partie débutera sur une présentation de la notion de modèle avant de s'attarder plus en détail sur les transformations de modèles.

Généralement, afin de résoudre un problème complexe, les personnes décomposent ce problème et en construise une représentation leur permettant d'appliquer un raisonnement dans le but de trouver une solution. Cette représentation peut être écrite sur la forme d'un modèle (*cf. partie Introduction page 1*). (Bézivin, 2016) explique que pour exploiter un modèle, il est nécessaire de connaître exactement les concepts qui le composent. Cette description est formalisée dans un **méta-modèle**.

La figure 4 est une représentation des relations existantes entre un système, un modèle et un méta-modèle. Un méta-modèle est un modèle décrivant le contenu d'un modèle. Par conséquent, un méta-modèle doit lui-même être conforme à son propre méta-modèle. Ainsi, un méta-modèle peut exister à différent niveau d'abstraction (Miller *et al.*, 2003).

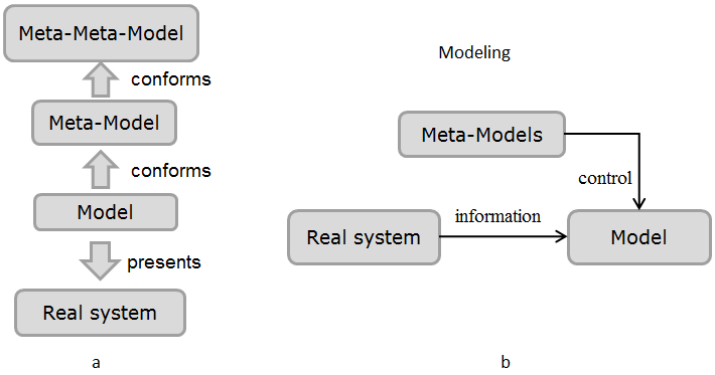


Fig. 4 Relations entre système, modèle et méta-modèle

Les transformations de modèles ont une place de choix dans le domaine de l'ingénierie dirigée par les modèles. En effet, l'automatisation intégrale ou même partielle d'approche, tel que le MDA, MDE, etc., représente un gain en efficience non négligeable dans la mise en œuvre de solution technique. Bien qu'il existe aujourd'hui de nombreux

développement de transformation de modèles et ce dans différents domaines d'application, il existe encore des difficultés ou piste d'amélioration des transformations de modèles (Del Fabro et al, 2009) : **une faible réutilisation, des tâches répétitives, ceci impliquant un important effort des réalisateurs**. Ces difficultés prennent leurs essences depuis :

- **La complexité des modèles**. Les modèles sont une représentation d'un système et par conséquent plus ou moins complexe en fonction du choix des informations à représenter ainsi que de la dimension et de la complexité du système.
- **La diversité des informations contenues par le modèle**. La diversité des systèmes provenant de leur domaine d'application ou encore de leur fonctionnement, est reflétée au niveau des modèles puisque un modèle est une représentation d'un système.
- **L'hétérogénéité des modélisations possible**. Un modèle est construit sur la base d'un méta-modèle exprimant les règles de représentation des informations du système. La diversité de méta-modèle possible, notamment dû à la gestion de version (tel que les normes par exemple) ou encore la multitude de méta-modèles existant pouvant être utilisés afin de représenter la même information.
- **La diversité des niveaux d'abstraction des méta-modèles**. La comparaison entre éléments de différents méta-modèles est soumise à la différence de niveau d'abstraction des méta-modèles (comme illustré par la Fig 4.)

Ainsi, il est possible d'en déduire qu'une lacune actuelle des transformations de modèles et le manque d'une méthodologie efficiente s'applicable à différent domaines. La méthodologie AMTM ambitionne à combler cette lacune.

Afin de souligner le côté inter-domaine de la méthodologie AMTM, sa mise en œuvre afin de répondre à ces difficultés a été réalisée dans trois domaines différents : l'interopérabilité des organisations (Wang et al, 2015a), la composition de web-services (Wang et al, 2015b), et l'ingénierie de la connaissance (Wang et al, 2015c).

Etat de l'art :

Cette partie consiste en une exploration des trois dimensions du domaine de la transformation de modèle : les catégories, les techniques et enfin des exemples de transformation de modèles.

Les transformations de modèles peuvent s'appliquer à des modèles ou à des textes. Il est ainsi possible de diviser les types de transformation de modèles selon trois catégories : *modèles à texte, modèle à modèle et texte à modèle*.

La méthodologie AMTM ambitionnant d'automatiser les transformations de modèle à modèle, la suite de cette partie se focalisera sur les transformations de modèle à modèles. Il existe cinq principales approches de transformation de modèles à modèles (Czarnecki *et al.*, 2003) : *approche de manipulation directe, approche relationnel, approche basée sur la transformation de graphe, approche dirigée par les structures et les approches hybrides*. Ces approches sont basées sur les méthodes de transformation suivante : *balise et schéma prédéfinis, transformation automatique, transformation basée sur les méta-modèles et fusion de modèles*.

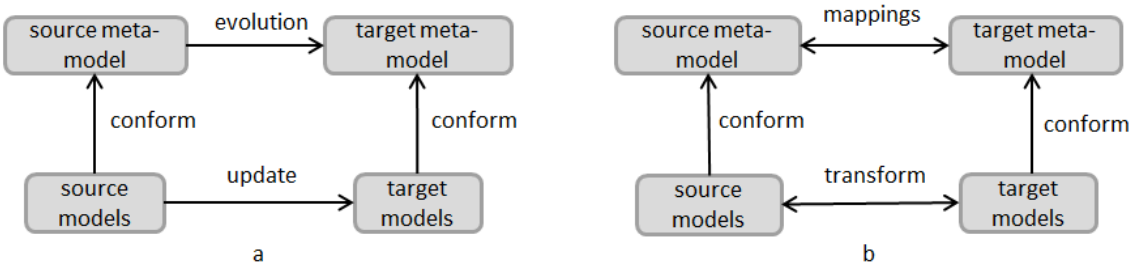


Fig. 5 illustration de transformation basée sur les méta-modèles

Par la suite, l'approche basée sur la transformation de méta-modèles est étudiée en détail puisqu'elle correspond à l'approche privilégiée par la méthodologie AMTM. Il est possible de distinguer deux types de cette approche : *évolution* et *transformation*. Ces deux situations sont illustrées par la Fig.5 : la partie (a) représente l'évolution. Ainsi le méta-modèle cible est généré par une évolution du méta-modèle source (exemple : ajouter de nouvelles caractéristiques). Les modèles sources, conforment aux méta-modèle source, sont mis à jour et deviennent des modèles cibles conforment au méta-modèle cible. Les activités de recherche, les théories proposées ainsi que les réalisations effectuées ont notamment permis de définir un cadre mature de réalisation de ce type de transformation

à travers la méthodologie « COPE » (Herrmannsdoerfer *et al*, 2009). La partie (b) de la Fig.5 représente un exemple de transformation. Dans cette situation, les méta-modèles source et cible étant différents, au sens qu'ils ne sont pas une évolution de l'un vers l'autre, il est nécessaire de définir des correspondances aux niveaux des méta-modèles et de les appliqués au niveau modèle afin de transformer le modèle source en modèle cible.

Le tableau 1 est une synthèse de la comparaison de différentes solutions techniques existantes permettant de réaliser une transformation de modèles selon les différentes catégories. Il est a noté que ce tableau n'est pas exhaustif et ne représente que les techniques les plus réputées actuellement.

Table 1 Comparaison de technique de transformation de modèles

name	hybrid	rule scheduling	M-to-N	note	Référence
ATL	yes	implicit internal explicit	yes	self-executed	(Jouault et al., 2008)
QVT	yes	implicit internal explicit	yes	based on MOF 2.0	(OMG, 2008)
VIATRA2	yes	external explicit	yes	based on VPM	(Varro et al., 2007)
GReAT	yes	external explicit	yes	on UML models	(Börger et al., 2012)

Les techniques de transformation de modèles peuvent être classifiées en deux groupes : *générique* ou *spécifique*. Les techniques spécifiques ont été réalisées pour un domaine spécifique et se focalise sur un problème particulier. Par conséquent, ces techniques sont peu utilisables dans un cas général de par leur manque de flexibilité. Cependant, ces techniques peuvent être exécutées de façon automatique ou semi-automatique. Les techniques génériques sont quant à elles complexes et proposent un large éventail de fonction afin de répondre à des problématiques inter-domaine. Ainsi la maîtrise de ces techniques peut être chronophage pour les utilisateurs et provoque d'importants efforts de la part des utilisateurs ainsi que de nombreuses tâches répétitifs (Del Fabro et al, 2009). Ces problèmes impliquent une utilisation limitée de ces techniques par les utilisateurs.

Le tableau 2 synthétise la comparaison de différentes expérimentations de transformation de modèles. En plus de ces expérimentations, certaines ont adoptés la comparaison sémantique afin de détecter les éventuelles correspondances tel que les travaux présentés dans (Kappel et al, 2006), (Kappel et al, 2007), (Bruel et al, 2000) and (Dolques, 2011).

Table 2 comparaisons de différentes expérimentations de transformation de modèles

name	technique	domain specific	note
Applying CIM-to-PIM model transformations for the service-oriented development of information systems (De Castro et al, 2011)	MDA-based	Yes	Combining MDA with service-oriented development of information system
Transformation of decisional models into UML: application to GRAI grids (Grangel et al, 2010)	ATL	yes	GRAI Grids to UML model
Applying MDE to the (semi-) automatic development of model transformations (Bollati et al, 2013)	MeTAGeM (Bollati et al, 2011)	no	applying MDE principles to define model transformation

Sur la base de l'étude de ces expérimentations, certaines faiblesses peuvent être identifiées : *elles sont spécifiques à un domaine, elle ne résout pas les problématiques liées à la diversité des granularités, elles ne produisent pas de thesaurus sémantique permettant de détecter automatiquement des relations entre les instances*.

Par conséquent, une méthodologie de transformation de modèles automatique pouvant être utilisée indépendamment du domaine d'application est encore à définir. Cette méthodologie devra reposer sur une réconciliation sémantique afin de déduire les correspondances entre les instances. Cette réconciliation nécessitera l'utilisation d'un thesaurus couvrant à la fois la généricité ainsi que les besoins spécifiques à un domaine.

Vue d'ensemble de la démarche AMTM

L'approche AMTM a été construite sur la base d'un cadre de référence reprenant les principaux concepts de la transformation de modèles basée sur les méta-modèles. Afin de pouvoir réaliser une comparaison syntaxique et sémantique (S&S) des éléments d'un modèle au cours du processus de transformation de modèles, un méta-méta-modèle a été proposé. La Fig. 6 représente le cadre théorique de la démarche (a) ainsi qu'une représentation UML du méta-méta-modèle (b).

L'approche AMTM consiste à déduire des règles de transformation au niveau des méta-modèles afin de pouvoir les appliquées par la suite sur la partie de concepts partagés des modèles. En effet, un méta-modèle décrit la façon de représenter un point de vue d'un système impliquant que deux méta-modèles peuvent dans un cas représenter le

même point de vue, ou alors deux point de vues partiellement différents. Il en découle que seule une partie du modèle source peut générer une partie du modèle cible. Ces parties, nommées concepts partagés, seront la base de la transformation alors que les parties spécifiques devront être sauvegardées, pour le modèle source, ou enrichies, pour le modèle cible.

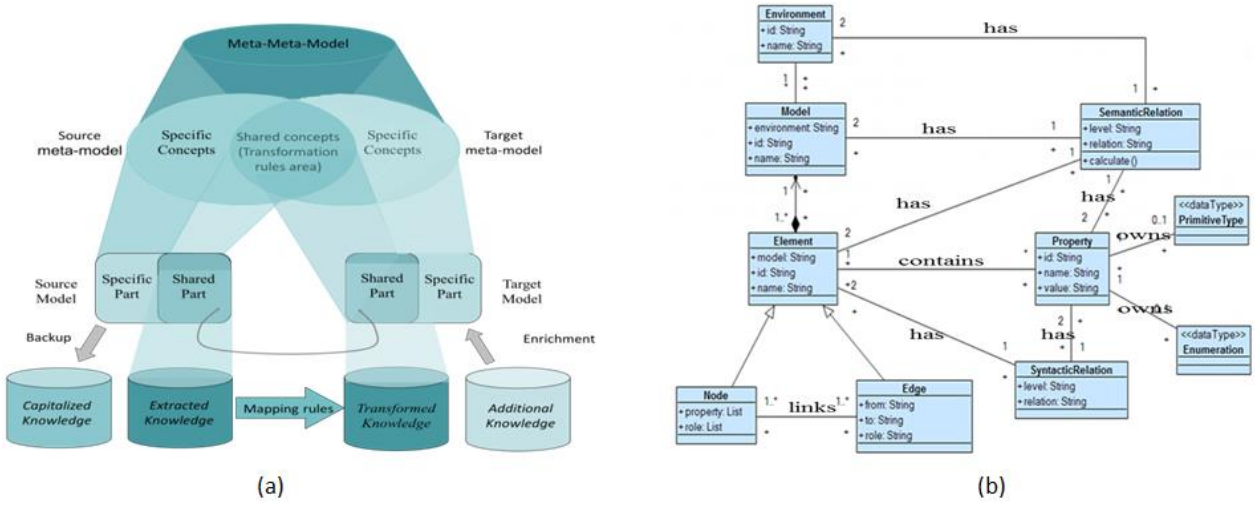


Fig. 6 cadre théorique de la démarche AMTM et représentation du méta-méta-modèle

L'originalité de la démarche AMTM réside dans son approche itérative. En effet, la transformation se réalise en différentes transformations intermédiaires, produisant chacune un modèle intermédiaire. Cette approche, représentée par la Fig. 7, permet d'une part d'améliorer la réconciliation S&S entre les modèles et d'autre part de rechercher la connaissance additionnelle, nécessaire à la complétude du modèle cible, au sien d'autres modèles.

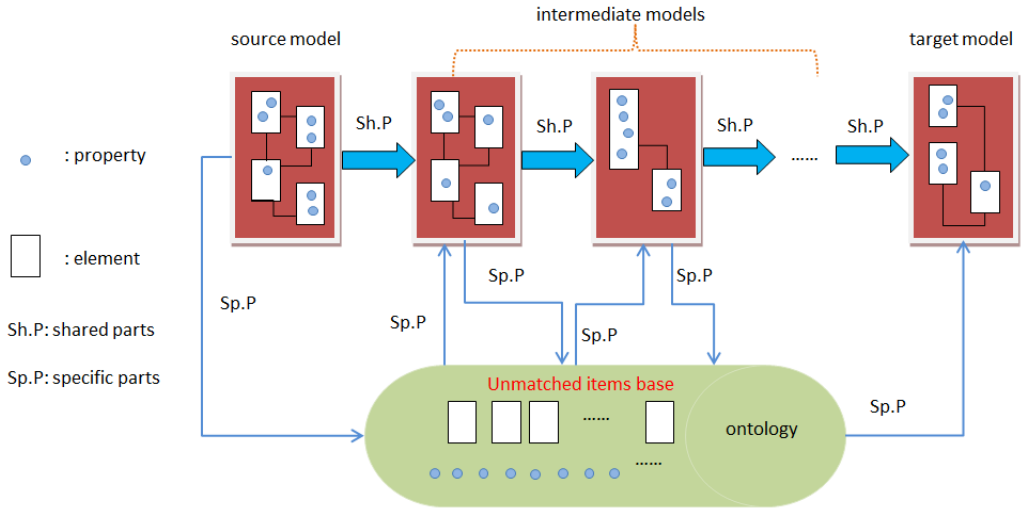


Fig. 7 représentation de l'approche itérative de AMTM

Pour cela et en respectant le cadre théorique de référence (Fig. 6 (a)), suite à chaque transformation, la partie spécifique du modèle est sauvegardée et l'ensemble des éléments du modèle cible encore orphelin de lien de correspondance (la partie spécifique) devient la source de la transformation suivante. Ces éléments sont alors comparés à des modèles précédemment utilisés afin de compléter la transformation précédente et ainsi de suite. Cette approche itérative repose sur une ontologie, nommée AMTM_O, en charge de capitaliser les éléments des modèles afin de permettre leur réutilisation pour l'approche itérative.

Par conséquent, l'approche AMTM est une répétition de recherche de correspondance syntaxique et sémantique des éléments de modèles. Chaque étape de l'approche, illustrée par la Fig. 8, se décompose selon quatre étapes principales :

- Calculer la réconciliation S&S au niveau des éléments des méta-modèles dans le but de générer de potentielles correspondances et des règles de transformation.
- Appliquer les résultats de la première phase au niveau des modèles dans le but de voir le résultat de la transformation.
- Demander une validation, utilisateur pour le moment, du résultat de l'étape précédente dans le but d'affiner les règles de transformation.
- Sur la base des résultats de l'étape précédente, les règles de transformation sont affinées ou de nouvelles sont créées au niveau des méta-modèles.

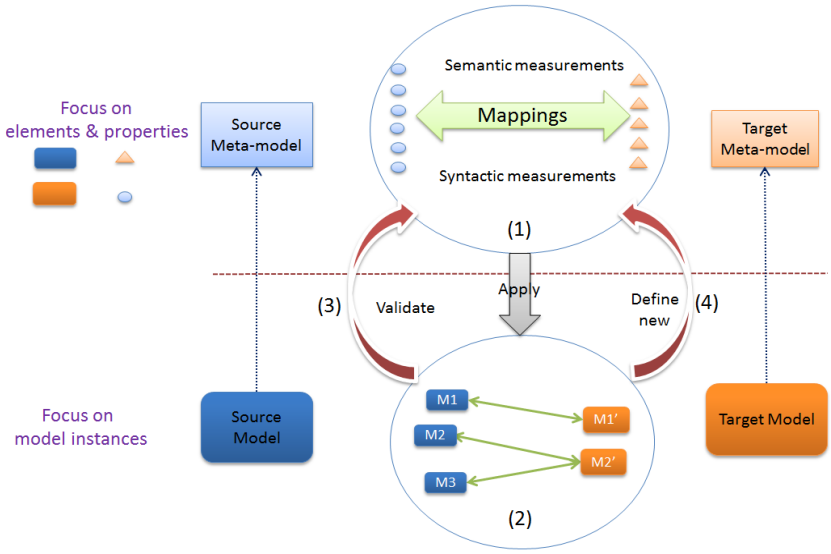


Fig. 8 Vue d'ensemble d'une étape de transformation de l'approche AMTM

Comme expliquer précédent, l'élément central de l'approche AMTM consiste en sa précision de mesure de correspondance syntaxique et sémantique d'éléments. Cette correspondance est déduite en quatre étapes, représentées par la Fig. 9 : *correspondance au niveau des éléments*, *correspondance hybride*, *correspondance inter-niveau* et *correspondance auxiliaire*.

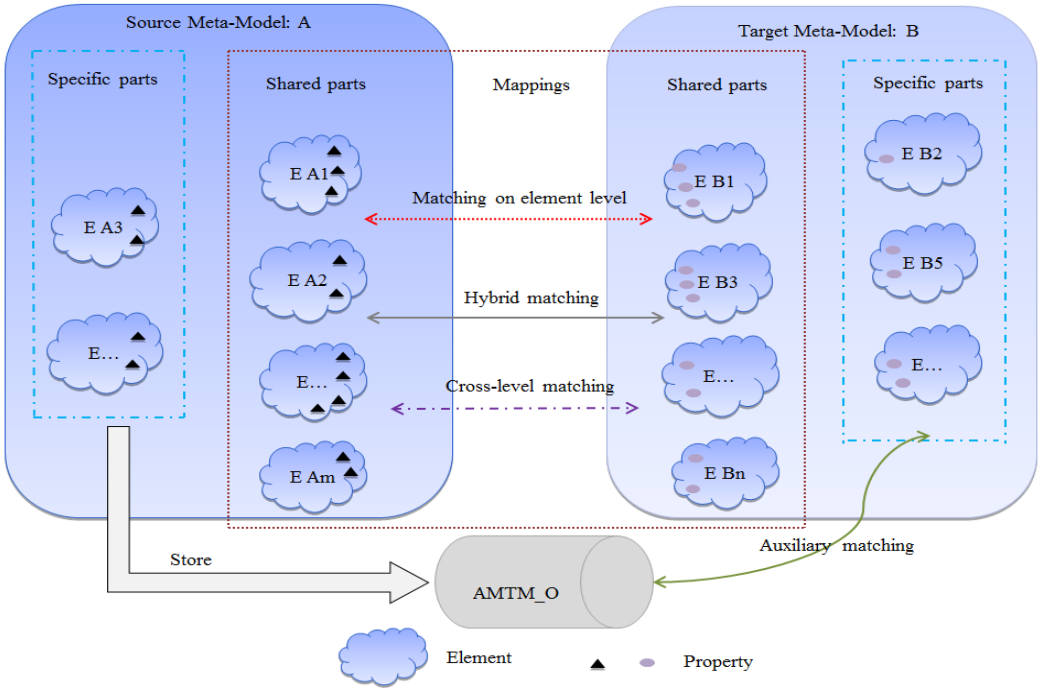


Fig. 9 les quatre correspondances S&S définies dans l'approche AMTM

- *correspondance au niveau des éléments* : cette étape consiste à calculer la correspondance entre éléments en incluant le calcul de correspondance entre les propriétés et le nom de ces éléments. Ce calcul est basé sur deux équations (1) et (2). L'équation (2) calcul la similarité de deux propriétés tant dis que l'équation (1) calcul la similarité des éléments.

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

$$P\_SSV = pn\_weight * S\_SSV + pt\_weight * Id\_type \quad (2)$$

- *correspondance hybride* : les propriétés, non réconciliées lors de l'étape précédente, sont comparées aux autres propriétés grâce à l'équation (3).

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

- *correspondance inter-niveau* : cette étape consiste à comparer des propriétés à des éléments dans le but de résoudre des problématiques de granularité grâce à l'équation (4).

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

- *correspondance auxiliaire* : cette étape consiste à rechercher au sein de l'ontologie des correspondances pour les éléments non réconcilié par les étapes précédentes grâce à l'équation (5) présentée dans la partie suivante.

Méthode de calcul des correspondances S&S

La méthodologie AMTM consiste à comparer un couple de mots selon deux dimensions : la syntaxe et la sémantique. La correspondance syntaxique consiste à comparer les lettres des mots ainsi que leur séquence contrairement à la correspondance sémantique s'intéressant aux sens portés par les mots.

Dans le contexte de la transformation de modèles, cette comparaison est effectuée entre éléments, entre propriétés et entre éléments et propriétés en fonction de l'étape de la démarche AMTM en cours. Cependant, les règles de correspondance syntaxique et sémantique sont basées sur les mots, alors que les transformations de modèles portent sur des éléments ou propriétés et non sur de mots. Par conséquent, il est nécessaire de ramener la problématique au niveau des mots tel que représenté par la Fig.10.

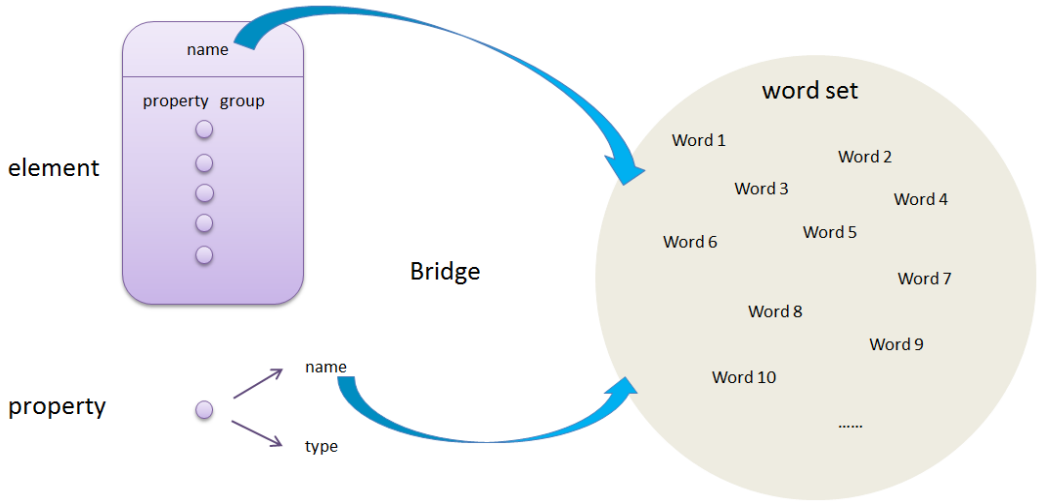


Fig. 10 lien entre les mots et les éléments et les propriétés de modèles.

Un *élément* d'un modèle est composé de : un *nom* et d'un *groupe de propriétés*. Le nom d'un élément est un mot, il est donc possible d'appliquer une correspondance S&S entre les noms de différents éléments. Le groupe de

propriétés quant à lui doit être comparé de façon intégrale. Par conséquent, le résultat de la comparaison de deux groupes de propriétés se base sur les comparaisons du produit cartésien des propriétés. Une *propriété* est comparée avec une autre propriété sur la base de deux facteurs les caractérisant : leur *nom* et leur *type*. Le nom des propriétés étant un mot, la comparaison S&S peut être directement appliquée à l'instar de la comparaison du nom des éléments. Il existe une multitude de type des propriétés possible (*i.e.* chaîne de caractères, entier, réel, etc.) ainsi deux propriétés peuvent avoir des types identiques, différents ou similaire (*i.e.* entier et réel) impliquant une valeur de correspondance différente.

Une relation syntaxique et sémantique entre deux mots peut être évaluée. De plus, si les deux mots ont une valeur importante de similarité syntaxique, il est probable qu'il s'agisse du même mot (*i.e.* le même mot au singulier et au pluriel) ou d'un antonyme (*i.e.* normale et anormale). Ainsi il n'est pas possible de se limiter à une comparaison syntaxique des mots, il est nécessaire de comparer à chaque fois les valeurs de correspondance syntaxique et sémantique et d'en déduire une valeur de correspondance calculé grâce à l'équation suivante :

$$S\_SSV = SeV\_weight * S\_SeV + SyV\_weight * S\_SyV \quad (5)$$

Dans cette équation, S_SSV correspond à la valeur de la correspondance syntaxique et sémantique des mots. Cette valeur, comprise entre 0 et 1, est calculée sur la base de la valeur de la correspondance syntaxique, notée S_SyV, et de la valeur de la correspondance sémantique, notée S_SeV. Ces deux valeurs sont pondérées par leur poids respectif. La somme des poids doit être égale 1 et dans le cas particulier d'un calcul de correspondant sans relation sémantique, le poids de la relation syntaxique, notée SyV_weight, sera égale à 1.

En se Basant sur les valeurs de l'équation précédente, il est possible de construire un repère permettant de qualifier les correspondances entre les mots et par conséquence les éléments des modèles. La Fig.11 est une représentation de ce repère divisé en quatre régions. La **région 1** correspond à une correspondance automatique entre les mots ou les éléments. La **région 2** correspond à une correspondance éventuelle laisser au choix de l'utilisateur. La **région 3** correspond à un défaut de correspondance entre les éléments et enfin la **région 4** correspond aux relations particulières comme les antonymes.

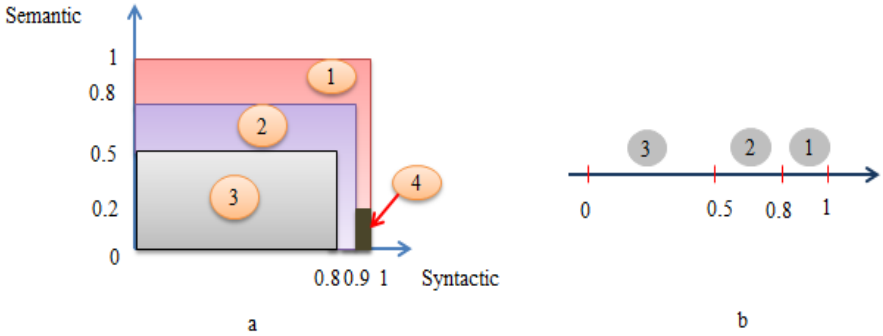


Fig. 11 Repère permettant de qualifier les relations entre éléments.

Calcul de la correspondance sémantique.

Le calcul de la correspondance sémantique de l'approche AMTM repose sur un thesaurus sémantique, nommée AMTM_ST, basé sur l'ontologie WordNet (Fellbaum, 1998). La Fig.12 représente la structure du thesaurus sémantique.

Ce thesaurus est composé de trois niveaux :

- **Word Base:** les mots de base, essentiellement provenant de la langue Anglaise, font référence à des noms, verbes et adjectifs pouvant être utilisées au sein d'un modèle ou d'un méta-modèle.
- **Word-sense Base:** lexique de sens des mots. Les mots peuvent être porteur de plusieurs sens tel que le mot star peut faire référence à de l'astrologie, ou à une célébrité ou tout autre de ces quatre autre sens. Ainsi un mot de base peut être lié à plusieurs sens (*i.e.* Word-sense).
- **Synset Base:** est composé de groupes de sens de mots ayant une relation (*i.e.* antonyme, similaire, etc.) entre eux.

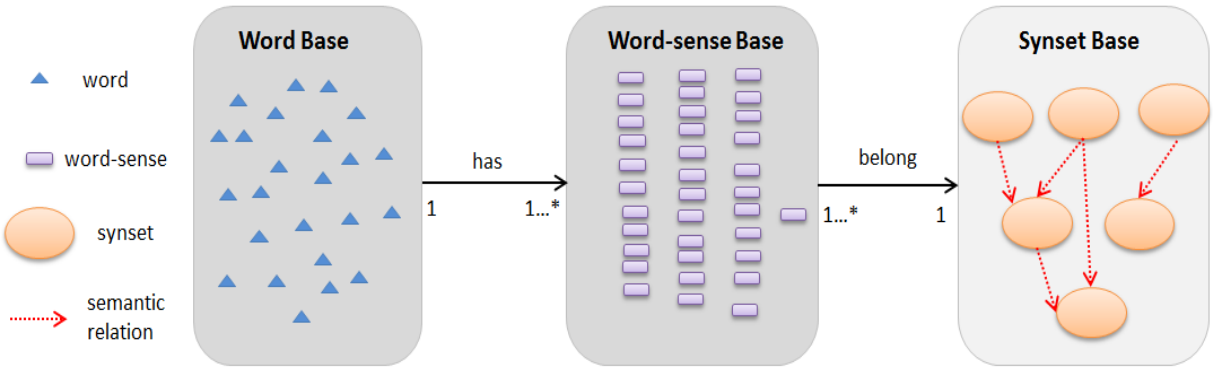


Fig. 12 Structure du thesaurus sémantique AMTM_ST

Dans le cadre de l'approche AMTM, sept relations entre Synsets (groupe de sens de mots) ont été définies et maintenues. Chaque relation s'est vue attribuée une valeur de pondération en fonction de son degré de similarité. Le tableau 3 liste ces relations ainsi que leurs pondérations.

Table 3 Liste des relations sémantiques utilisées par l'approche AMTM

Semantic relation	S_SeV	Example
synonym	0.9	shut & close
hypernym	0.6	person-creator
hyponym	0.8	creator-person
similar-to	0.85	perfect & ideal
antonym	-1	good & bad
iterative hypernym	0.6^n	person-creator-maker-author
iterative hyponym	0.8^n	author-maker-creator-person

Il est nécessaire d'avoir un thesaurus sémantique conséquent afin d'obtenir des résultats exploitables de réconciliation sémantique. Le tableau 4 énumère le nombre de mots de base, de sens de mot et de synsets contenus dans AMTM_ST.

Table 4 Items stored in AMTM_ST

Items	Number
words	147 306
word senses	206 941
synsets	114 038

La Fig. 13 est une illustration du processus de calcul de correspondance sémantique entre deux mots. Ce calcul se base sur deux méthodes : *directe* et *itérative*. La partie (a) de la Fig. 13 représente la méthode directe. La méthode directe consiste à rechercher, entre les synsets, les relations de types : hypernyme, hyponyme, antonyme, synonyme et similaire à. Si la méthode directe ne donne pas de résultat, la méthode itérative est alors utilisée afin de parcourir les synsets liés par une relation de type hypernyme jusqu'à l'obtention d'un résultat de la méthode directe. Le principe de la méthode itérative est représenté par la partie (b) de la Fig.13.

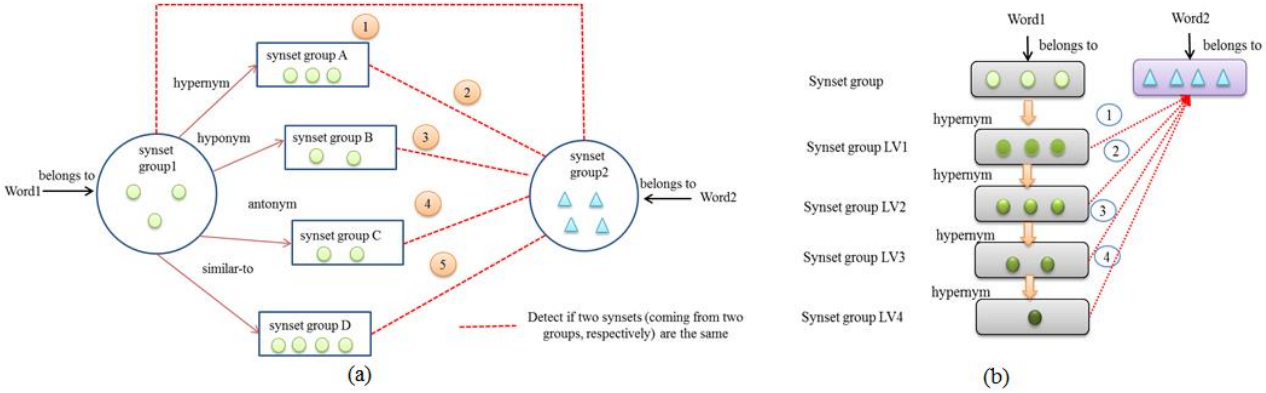


Fig. 13 méthode de calcul de la correspondance sémantique entre deux mots.

La Fig.13 illustre la méthode de calcul de correspondance sémantique entre deux mots. La partie (a) illustre le mécanisme de détection directe entre deux mots alors que la partie (b) illustre l'approche itérative. Le calcul direct entre deux mots repose sur une correspondance, listée dans le tableau 3, entre un type de relation et une valeur prédéfinie. Dans le cas d'une relation d'hypernyme, l'ensemble des relations sémantique de ce sysnset est exploré et ainsi de suite ce qui permet de proposer une approche itérative.

Calcul de la correspondance Syntaxique

Le calcul de la correspondance syntaxique se déroule en deux étapes : un *prétraitement de mots* puis l'application de l'*algorithme de calcul de distance de Levensthein*. Le prétraitement de mots ambitionne à trouver un mot et ceux malgré ces diverses écritures possibles (*i.e.* singulier et pluriel par exemple). Le tableau 5 liste un ensemble, non-exhaustif, de prétraitement possible. Le prétraitement de mots repose sur l'algorithme de *Poter stemming algorithm* (Willett, 2006).

Table 5 Exemple de prétraitement

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	……	……	……

L'objectif de ce prétraitement est retrouvé au sein d'un mot, un mot de base du thesaurus AMTM_ST afin de pouvoir y appliquer un calcul de correspondance sémantique.

La seconde partie du calcul de la correspondance syntaxique consiste à appliquer l'algorithme de calcul de distance de *Levensthein*. Cet algorithme est un des algorithmes le plus utilisés dans le domaine de la réconciliation syntaxique (Gilleland, 2009). Le calcul de distance entre deux mots correspond au nombre minimum d'opérations à effectuer sur les caractères afin d'obtenir le mot cible. Ces opérations peuvent être de l'insertion, de la suppression ou du remplacement.

La formule ci-dessous représente les formules de l'algorithme de calcul de distance de Levensthein (Gilleland, 2009) où a et b sont deux chaînes de caractères de tailles respectivement notées |a| et |b|.

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \mathrm{lev}_{a,b}(i-1,j) + 1 \\ \mathrm{lev}_{a,b}(i,j-1) + 1 \\ \mathrm{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

Dans cette formule « 1 (ai ≠ bj) » est la fonction d'indicateur égale à 0 si « ai = bj » et égale à 1 autrement. Il est à noter que le premier élément du minimum correspond à la suppression de caractère de a vers b, le deuxième élément correspond à l'insertion de caractère et le troisième élément correspond à la cohérence ou l'incohérence des caractères en fonction de l'équivalence des caractères.

Cas d'étude et présentation du preuve de concept

Une preuve de concept, nommée AMTM_SS, a été réalisée afin d'illustrer la démarche et évaluer ces performances. Cette preuve de concept, consistant en un programme informatique, a été réalisée suivant un processus classique de développement informatique divisée en quatre étapes : recueil des besoins, définition de l'architecture logique, développement et réalisation de tests. Ce chapitre se focalise essentiellement sur la définition de l'architecture logique de la preuve de concept AMTM_SS et évoquera les étapes de recueil des besoins, de développement et de réalisation de tests.

AMTM_SS est divisé en six modules fonctionnels : **analyse des modèles, détection de relations sémantique, détection de relations syntaxique, sélection de relations, génération de correspondance de transformation, interaction avec l'utilisateur**. De plus, ces six modules sont eux même décomposés en sous-modules permettant la réutilisation de « brique élémentaire » dans plusieurs modules.

La Fig. 14 est une représentation de la décomposition de AMTM_SS en cinq packages représentant ici uniquement les Classes principales et deux ressources externes à savoir le thesaurus sémantique AMTM_ST et l'ontologie AMTM_O.

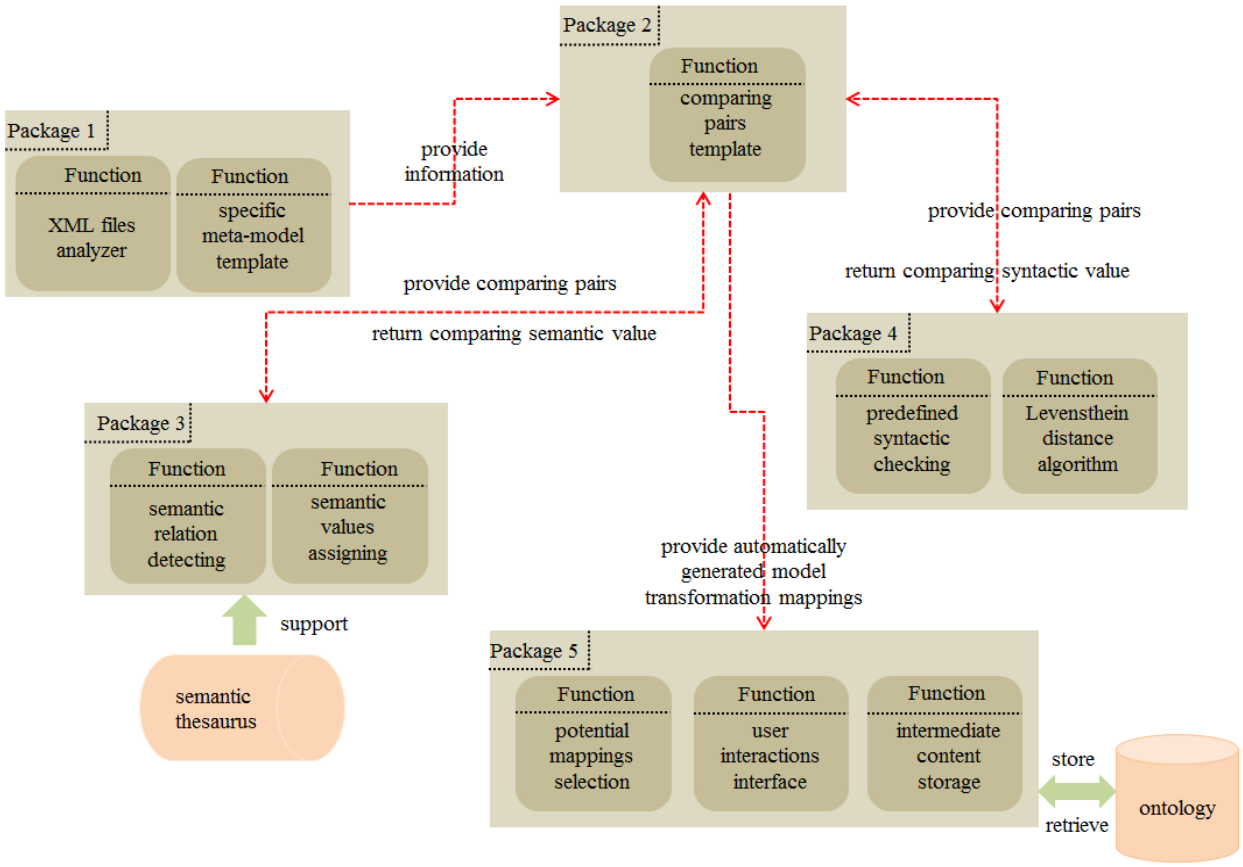


Fig. 14 décomposition de AMTM_SS en package

La démarche AMTM reposant sur la réconciliation sémantique et syntaxique des éléments, la suite de ce chapitre illustre un exemple simple de cette réconciliation entre deux éléments : *étudiant* et *personne*. La Fig.15 illustre les deux méta-modèles basiques utilisés. L'élément *Student* a cinq propriétés : *name, age, address, sex* et *teacher.* Alors que l'élément *person* a sept propriétés : *id, surname, forename, gender, age, phone* et *address.*

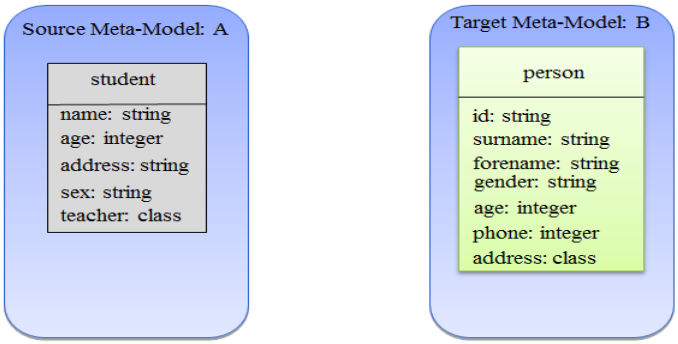


Fig.15 exemple basique utilisé pour illustrer AMTM_SS

La Fig. 16 est une impression écran de la comparaison des éléments. Il est possible d'y noter que le mot *student* est lié à deux synsets alors que le mot *person* est lié à trois synsets. La relation sémantique entre les deux mots est une relation d'hypernyme obtenue par itération et la valeur de S&S entre les deux mots est de 0.5903

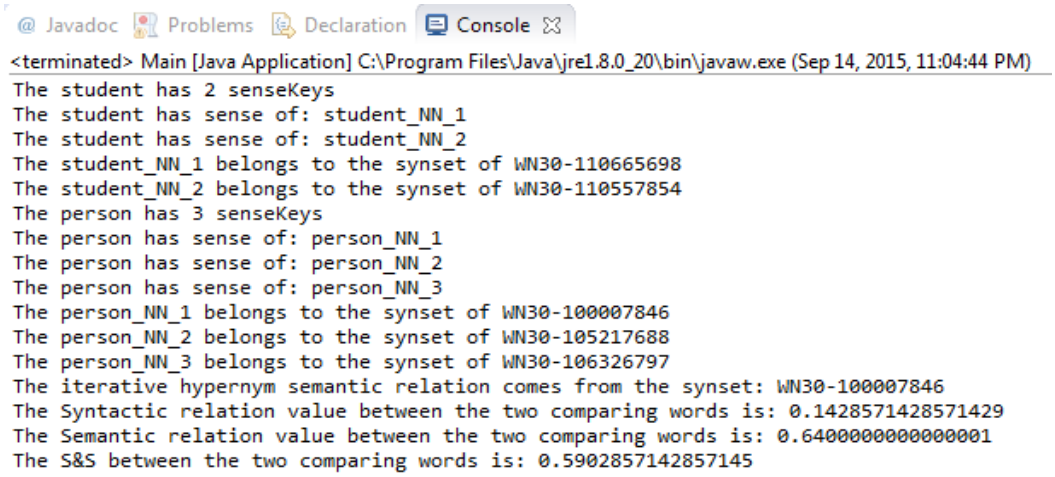


Fig. 16 résultat de la comparaison S&S entre "student" et "person"

Enfin, la Fig. 17 est une capture d'écran du résultat de la comparaison des deux groupes de propriétés.

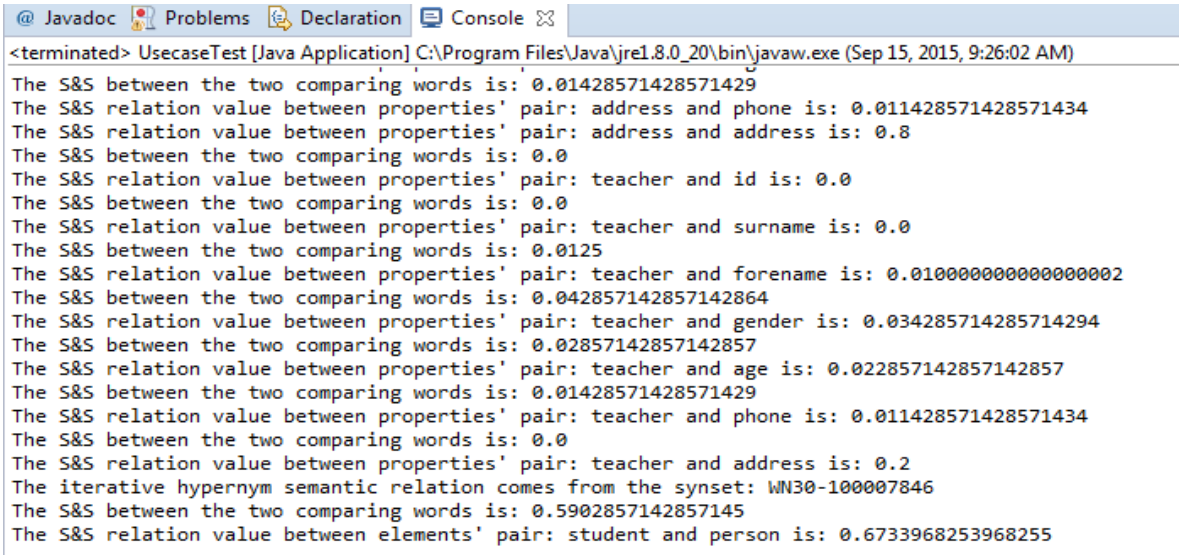


Fig. 17 résultat de la comparaison des deux groupes de propriétés

Finalement, la réconciliation finale entre les éléments *student* et *person* est de : 0.673. Cette correspondance est par conséquent qualifiée comme correspondance potentielle d'après le repère expliquer dans la partie précédente. Cet exemple simple, illustre la faisabilité de détection de correspondance entre modèles hétérogènes.

References

(Bézivin, 2006) Bézivin, J. (2006). Model driven engineering: An emerging technical space. InGenerative and transformational techniques in software engineering (pp. 36-64). Springer Berlin Heidelberg.

(Bollati et al, 2011) Bollati V A. MeTAGeM: a framework for model-driven development of model transformations[D]. Ph. D. Thesis. University Rey Juan Carlos. http://www. kybele. etsii. urjc. es/members/vbollati/Thesis, 2011.

(Bollati et al, 2013) Bollati, V. A., Vara, J. M., Jiménez, Á., & Marcos, E. (2013). Applying MDE to the (semi-) automatic development of model transformations. Information and Software Technology, 55(4), 699-718.

(Bruel et al, 2000) Bruel, J. M., Lilius, J., Moreira, A., & France, R. B. (2000). Defining precise semantics for UML. In Object-Oriented Technology (pp. 113-122). Springer Berlin Heidelberg.

(Czarnecki et al., 2003) Czarnecki, K., & Helsen, S. (2003, October). Classification of model transformation approaches. In Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture(Vol. 45, No. 3, pp. 1-17).

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

(Del Fabro et al, 2009) 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.

(Dolques, 2011) Dolques, X., Dogui, A., Falleri, J. R., Huchard, M., Nebut, C., & Pfister, F. (2011). Easing model transformation learning with automatically aligned examples. In Modelling Foundations and Applications (pp. 189-204). Springer Berlin Heidelberg.

(Grangel et al, 2010) Grangel, R., Bigand, M., & Bourey, J. P. (2010). Transformation of decisional models into UML: application to GRAI grids. International Journal of Computer Integrated Manufacturing, 23(7), 655-672.

(Herrmannsdoerfer et al, 2009) Herrmannsdoerfer, M., Benz, S., & Juergens, E. (2009). COPE-automating coupled evolution of metamodels and models. In ECOOP 2009–Object-Oriented Programming (pp. 52-76). Springer Berlin Heidelberg.

(Kappel et al, 2006) Kappel, G., Kapsammer, E., Kargl, H., Kramler, G., Reiter, T., Retschitzegger, W., & Wimmer, M. (2006). Lifting metamodels to ontologies: A step to the semantic integration of modeling languages (pp. 528-542). Springer Berlin Heidelberg.

(Kappel et al, 2007) Kappel, G., Kargl, H., Kramler, G., Schauerhuber, A., Seidl, M., Strommer, M., & Wimmer, M. (2007, March). Matching Metamodels with Semantic Systems-An Experience Report. In BTW Workshops (pp. 38-52).

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

(Maier, 1998) Maier, M. W. (1996, July). Architecting principles for systems-of-systems. InINCOSE International Symposium (Vol. 6, No. 1, pp. 565-573).

(Miller et al., 2003) Miller, J., & Mukerji, J. (2003). MDA Guide Version 1.0. 1.

(Touzi et al, 2007) Touzi, J., Lorré, J. P., Bénaben, F., & Pingaud, H. (2007). Interoperability through model-based generation: The case of the collaborative information system (CIS). In Enterprise Interoperability (pp. 407-416). Springer London.

(Terrasse et al., 2005) Terrasse, M. N., Savonnet, M., Leclercq, E., Grison, T., & Becker, G. (2005). Points de vue croisés sur les notions de modèle et métamodèle. 1ères journées sur l'Ingénierie Dirigée par les Modèles, 17-28.

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

(Vernadat, 1999) Vernadat, F. (1999). Techniques de modélisation en entreprise: applications aux processus opérationnels. Economica.

(Wang et al, 2015a) Wang, T., Truptil, S., & Benaben, F. (2015). A General Model Transformation Methodology to Serve Enterprise Interoperability Data Sharing Problem. InEnterprise Interoperability (pp. 16-29). Springer Berlin Heidelberg.

(Wang et al, 2015b) Wang, T., Truptil, S., & Benaben, F. (2015, June). An Automatic Model Transformation Methodology to Serve Web Service Composition Data Transforming Problem. In Services (SERVICES), 2015 IEEE World Congress on (pp. 135-142). IEEE.

(Wang et al, 2015c) Wang, T., Truptil, S., & Benaben, F. (2015, January). Applying a Semantic & Syntactic Comparisons Based Automatic Model Transformation Methodology to Serve Information Sharing. In Proceedings of the International Conference on Information and Knowledge Engineering (IKE) (p. 3). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

A study to define an automatic model transformation approach based on semantic and syntactic comparisons

Abstract. The models are increasingly used both for the description of a view of a complex system or for information exchange. However, to share the information, transferring information from one model to another is an issue related to the interoperability of systems now. This problem can be approached in three ways: integrated (all identical models), unified (all models refer to a pivot model), federated (no specific rules on the models). Although standards exist, they are rarely respected rigorously. The federated approach therefore seems to be the most realistic approach. However, because of the different models, this approach is complicated. Models can have a very heterogeneous structure and different vocabulary to describe the same concept. Therefore, we must identify the common concepts of different models before defining the transformation rules for transforming from one format to another. This thesis proposes a methodology to achieve these goals. It is partly based on the proposal of a meta-meta-model (to unify the description of the model structure), i.e. the meta-model, and secondly calculating the distance between each element of models to deduce the transformation rules. This distance reflecting both syntactic distance (words occurrence) and semantic relation that related to the synonymous. Researching synonym relation is based on the use of knowledge base, represented as ontology, such as WordNet.

Keywords: Model transformation, Model-driven engineering, Ontology, Semantic & Syntactic checking

---

Etude d'une approche de transformation de modèles automatisée basée sur des comparaisons sémantique et syntaxique

Résumé. Les modèles sont de plus en plus utilisés que ce soit pour la description d'un point de vue d'un système complexe ou pour l'échange d'information. Cependant, le partage d'information, le transfert d'information d'un modèle à un autre est aujourd'hui une problématique liée à l'interopérabilité des systèmes. Cette problématique peut être abordée selon trois approches : intégrée (tous les modèles identiques), unifiée (tous les modèles font référence à un modèle pivot), fédérée (pas de règles précises sur les modèles). Bien que des standards existent, ils sont rarement respectés avec rigueur. L'approche fédérée semble par conséquent l'approche la plus réaliste. Cependant, cette approche est complexe car les différents modèles, bien que comportant des concepts communs, peuvent avoir une structure et un vocabulaire très hétérogène pour décrire le même concept. Par conséquent, il faut identifier les concepts communs des différents modèles avant de définir les règles de transformation permettant de passer d'un format à un autre. Cette thèse propose une méthodologie permettant d'atteindre ces objectifs, elle se base d'une part sur la proposition d'un méta-méta-modèle permettant d'unifier la description de la structure des modèles, *i.e.* le méta-modèle, et d'autre part sur le calcul de distance entre chaque élément des modèles qui permettront de déduire les règles de transformation. Cette mesure de distance reflète la distance à la fois syntaxique, écritures différentes d'un même terme, ainsi que sémantique liée à l'utilisation de synonyme. La recherche de synonyme est basée sur l'utilisation de base de connaissance, représentée sous forme d'ontologie, tel que WordNet.

Mots-clés : Transformation de modèles, Ingénierie dirigée par les modèles, Ontologie, Sémantique et Syntaxique réconciliation