An automatic model-to-model mapping and transformation methodology to serve model-based systems engineering — Source link

An automatic model-to-model mapping and transformation methodology to serve model-based systems engineering

Tiexin Wang, Sébastien Truptil, Frederick Benaben

An automatic model-to-model mapping and transformation methodology to serve model-based systems engineering

Tiexin Wang[1] · Sebastien Truptil[1] · Frederick Benaben[1]

[1] Centre Genie Industriel, University de Toulouse - Mines Albi, Campus Jarlard, 81000 Albi, France

Abstract With enterprise collaboration becoming increasingly frequent, the ability of an enterprise to cooperate with others has become one of the core factors in gaining competitive advantage. This trend has led to an urgent requirement to improve cooperation ability. To this end, model-based systems engineering is being adapted so that it can be used to represent and simulate the working processes of enterprises. Model-to-model mappings and transformations, as important aspects in model-based systems engineering, have become two of the key factors in improving the cooperation capabilities of enterprises. However, the foundations for achieving automatic model-to-model transformation have not yet been built. Normally, model transformation rules are built on the basis of model mappings, and model mappings concern semantic or syntactic representations. One of the difficulties in achieving model-to-model mappings and transformations lies in detecting the semantics and semantic relations that are conveyed in different models. This paper presents an automatic model-to-model mapping and transformation methodology, which applies semantic and syntactic checking measurements to detect the meanings and relations between different models automatically. Both of the semantic and syntactic checking measurements are combined into a refined meta-model based model transformation process. To evaluate the performance of this methodology, we demonstrate its applicability with a realistic example.

Keywords Enterprise collaboration · Model-driven engineering · Model-to-model mappings · Automatic model-to-model transformation · Semantic and syntactic checking

✉ Tiexin Wang
tiexin.wang@mines-albi.fr

1 Introduction

Because of advances in science and technology, enterprise collaboration projects are becoming increasingly common. In order to improve their competitiveness, enterprises prefer to focus on their core business and to cooperate with other qualified partners where they need to provide a more complex function (or service). To achieve a common goal (e.g. maximize profits), enterprises also need to find ways to collaborate with each other (Li 2012). They are thus paying increasing-attention to improving their ability to work efficiently in collaboration with other enterprises. This involves understanding the concept of collaboration.

Collaboration may be structured according to four levels (Benaben et al. 2006):

- *Communication (data exchange)* organizations exchange or share data (such as invoices, purchase orders, etc.) to improve their individual performance. This is the lowest level of collaboration.
- *Coordination (sharing and synchronization-of tasks)* organizations manage their sequence of tasks to optimize the whole set of individual behaviours. This level implicitly suggests an embryo of collaborative behaviour (through task sharing and synchronization). However, this is mainly focused on individual performance without collective goal.
- *Cooperation (pursuing a common goal)* organizations have (at least) one shared common goal and set up collaborative processes to reach this (these) goal(s). These common goals legitimize the existence of a network of organizations. These organizations have to work collectively, harmoniously and with respect for each other in order to achieve the common purpose. This might require a lot of effort for each of the organizations involved.
- *Integration (seamless involvement in one virtual entity)* for this level, there is no real evolution in the quality of the collaboration (compared to the cooperation level). However, the way it is achieved is different. It is far more facilitated and fluid.

Based on this vision, the capability criteria of organizations to collaborate may also be defined accordingly. For instance, organizations must be able to join communication networks to perform communication, they must also be able to manage their resources and manage schedules to perform coordination, etc. One element that might be seen as a criterion in reaching a high level of collaboration (in terms of performance and efficiency), is interoperability.

In Ide and Pustejovsky (2010), interoperability is defined as “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”. In Zdravkovic et al. (2015) interoperability is considered as a property of a single system: “interoperability determines the capacity of a system (in a general sense) to adapt, respond, act internally or externally, etc. to specific circumstances. This capability depends on the understanding of the interfaces”. This second vision is essentially concerns the ability to take part in deep collaboration. Finally, and historically, the concept of interoperability has been defined, on the one hand in IEEE (1991) as “the ability of two or more systems or components to exchange information and to use the information that has been exchanged”, and on the other hand in Konstantas et al. (2005) by the InterOp Network of Excellence (NoE) as “the ability of a system or a product to work with other systems or products without special effort from the customer or user”.

Consequently, the interoperability of organizations appears to be a major requirement to succeed in deep and seamless high-level collaborations. Therefore, enterprises need to adopt the required interoperability functions: exchange of information, coordination of functions and orchestration of processes (Benaben et al. 2006). Furthermore, inside these organizations, Information Systems (ISs) and computerized systems are assuming the role of interfaces (external and internal exchanges), functional engines [driving processes and business activities (Ramirez et al. 2010)] and data providers (creating a drastically increasing amount of measurements, data and reports from devices, software and reporting tools). Thus, ISs must be able to support the above-mentioned interoperability functions. The issue is to ensure that the partners’ ISs will be able to work together (thanks to these interoperability functions) to constitute a coherent and homogeneous IS set (the IS of the collaborative network). Providing organizations with methods, tools and platforms that are able to ensure these interoperability functions makes good sense.

The Enterprise Interoperability Framework: EIF (Chen et al. 2008), which is shown in Fig. 1, proposes an idea to analysis enterprise interoperability.

In the EIF, the concerns of interoperability are divided into four layers: data, service, process and business. All four of these concerns need to overcome three kinds of barriers: conceptual, technological and organizational. To conquer the three kinds of barriers on four concern layers, three possible approaches are proposed.

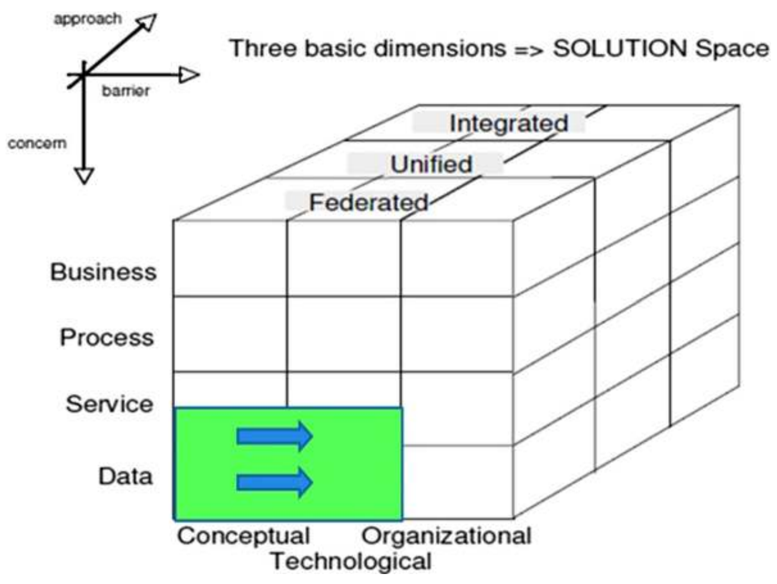


Fig. 1 EIF illustration and concerns studied this paper

They are an “Integrated” approach, a “Unified” approach and a “Federated” approach.

Each of these three approaches has its range of uses (best situations to be applied). The “Integrated” approach suits the situation where enterprises use: one common structure, one set of management rules, the same formats of data, etc. The enterprises can cooperate with each other directly. The “Unified” approach suits the situation where enterprises adapt their structures, process methods, etc. to a unified standard. Enterprises here can cooperate indirectly with each other (one step transformation to the unified standard). The “Federated” approach suits the situation where the enterprises use their own rules and formats to manage their daily business, and more manual effort is required to build cooperating connections amongst them.

At the same time, EIF also illustrates the way model-based systems engineering can be use to solve enterprise interoperability problems. An enterprise can be simulated by a set of models: different models (or a set of models) are created to describe a specific department or a production process in the enterprise, and the connections between these models can represent the interactions between different departments or processes.

In Fig. 2, an enterprise is represented by a set of models that are created based on different viewpoints. In the 1990s, as stated in Scheer (1992), several standard viewpoints were proposed to describe an enterprise. More recently, as an important

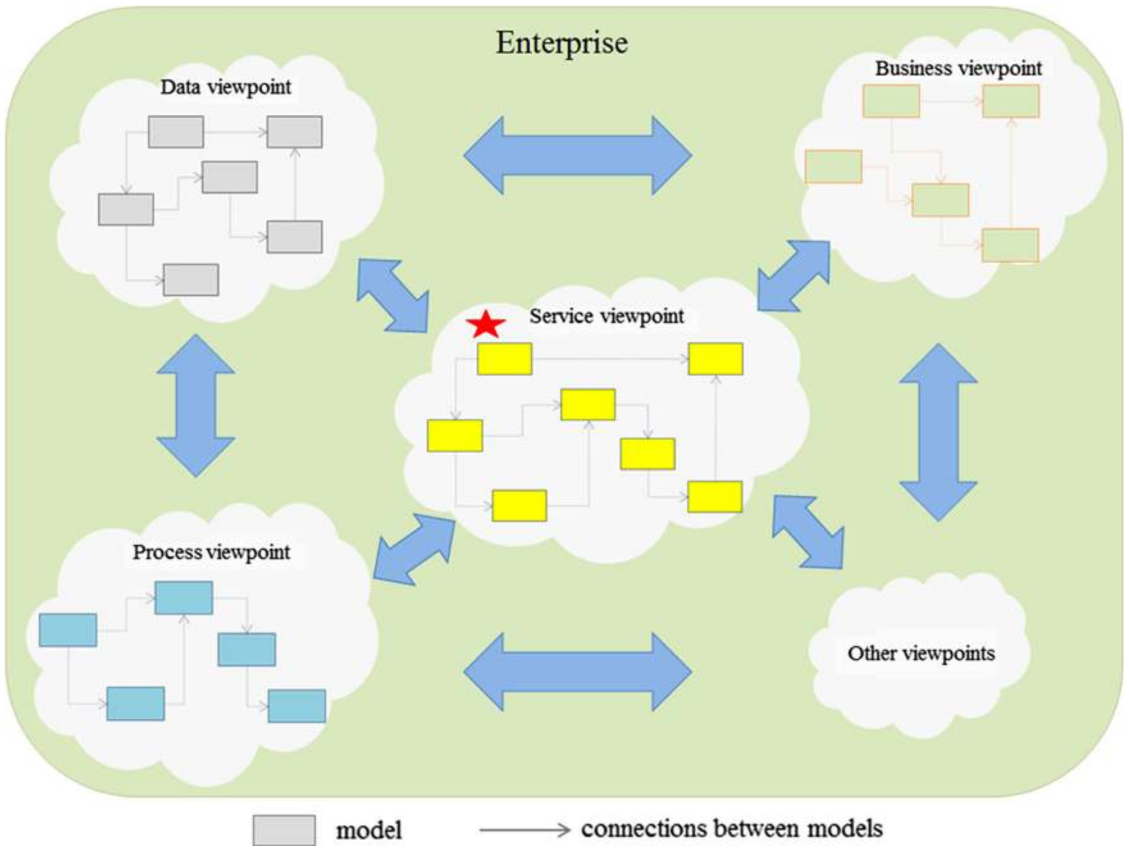


Fig. 2 An illustration of combining model-based engineering with enterprise engineering

aspect of modern enterprises, “service” viewed as crucial, so it is listed as a specific viewpoint in Fig. 2. All these different model sets describe a specific viewpoint of this enterprise. Within a model set, the models involved are connected. Between different viewpoints, the inevitable and potential relationships can also be represented by model connections. Normally, these model connections can be built using model transformations. In this way, model-based engineering and enterprise engineering are combined.

By considering enterprise collaboration, enterprise interoperability and EIF, this paper proposes an automatic model transformation methodology that focuses on breaking down the barriers of the conceptual and technological dimensions for the concerns of the data layer and part of the service layer. These barriers involve the communication and coordination levels of enterprise collaboration. In plain terms, model transformation methodology can be used to share and exchange data (& information), and given the efficiency issue in enterprise collaboration, automatic solutions would obviously be the best choice.

This paper is divided into six sections. The second section presents the problem statement in using model transformation to solve data sharing and exchange issues in the field of enterprise interoperability. The state-of-the-art of the research presented in this paper is outlined in the third section. The fourth section gives an overview of the automatic model-to-model mapping and transformation methodology (AMTM), and details the semantic & syntactic checking measurements involved in AMTM. Before the final conclusion, the design part of the software tool and a use case (with evaluation) of AMTM are described in the fifth section.

2 Problem statement

Since modern enterprises use information systems to manage their daily business, enterprise interoperability depends partly on the interoperability of their information systems. The interoperability of enterprise information systems (EISs) needs to be improved and the integration process of these kinds of information systems should be simplified.

When seeking to improve interoperability and simplify the integration process of EISs, one of the typical problems to be solved is how to share and exchange data (& information) between EISs. Given the nature of the new kinds of enterprise collaboration, as summarized in Touzi et al. (2007) and Camarinha-Matos and Afsarmanesh (2008), e.g. global and dynamic combinations of partners, short periods of duration, etc., the sharing and exchanging of data (as one of the fundamental factors in integrating EISs) needs to be done efficiently and effectively.

As illustrated in the introduction section, model-based systems engineering can be used to improve enterprise interoperability: the data sharing and exchanging issue could be solved by model transformation methodology. However, there are several weaknesses in traditional model transformation practices, as defined by Del Fabro and Valduriez (2009): they have low reusability, contain repetitive tasks, involve huge manual effort, etc. These weaknesses limit the use of model transformation to serve various engineering domains (especially in collaborative situations). Thus, the problem is how to develop an efficient and effective model transformation methodology to serve EIS interoperability and integration? To solve this problem, our proposal is to use AMTM.

Here, we give a general analysis of the difficulties in defining AMTM. In AMTM, the model transformation process contains two steps: building model transformation mappings and defining model transformation rules. The relationship between building mappings and defining transformation rules is shown in Fig. 3.

Normally, models are made of elements. Model transformation mappings are built (between those valid elements) based on some pre-defined detecting measurements or by domain experts (manual work). After generating these potential mappings, a mappings selection mechanism must be applied to define the final transformation rules.

By detecting transformation mappings and defining transformation rules, connections can be built between elements from different models. The connected model elements should have some intrinsic links that can be detected. However, these intrinsic links are difficult to find automatically, for several reasons. Normally, the main reason is originally due to the models themselves. The difficulties of defining AMTM can be divided into four aspects.

- *The complexity of the models* Models are always built to represent complex systems, thus the models themselves become complex. Analyzing complex models is a difficult task.
- *Various kinds of content are conveyed in the models* Models represent systems and focus on different functional point of views. Significant differences may exist among these systems, thus detecting the intrinsic links between them is difficult.
- *Heterogeneous modeling techniques* Models are built to represent systems, but at the same time they are built based on particular modeling techniques. Such techniques may be UML (Uniform Modeling Language), SysML (System Modeling Language), IDEF (Integrated Definition Modeling), BPMN (Business Process Model and Notation), etc. Different modeling techniques define their own standards, modeling rules and semantic representations. Transforming and merging these modeling techniques is a necessary, but time-consuming task.
- *Dynamic collaborative situations* Different partners will be involved in and dropping out of collaboration within a specific period. This requires that the

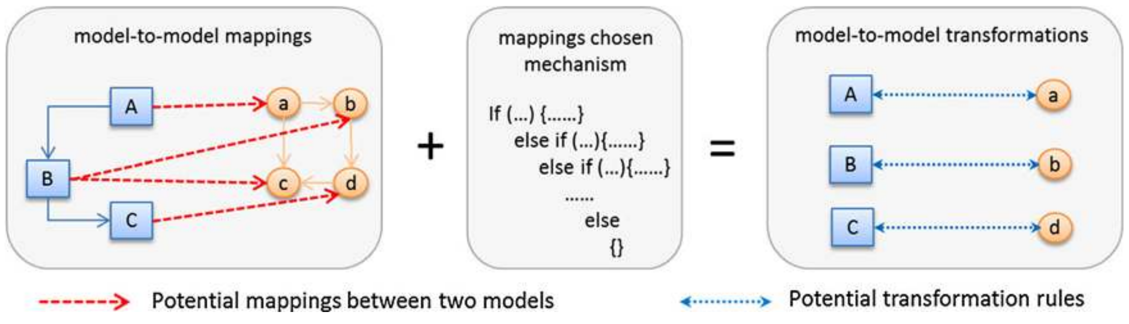

model-to-model mappings
A
B
C
a
b
c
d
+
mappings chosen mechanism
If (...) {......}
else if (...){......}
else if (...){......}
......
else
{}
=
model-to-model transformations
A
B
C
a
b
d
Potential mappings between two models
Potential transformation rules

Fig. 3 The relationship between model-to-model mappings and transformations

integration of EISs should also be dynamic. The collaborative situations might be highly complex due to the dynamic combinations involved, and thus modeling and model transformation work for such a collaborative situation requires huge efforts.

To solve all these problems, a powerful detection methodology and mappings selection mechanism is needed, which would be used to detect the potential transformation mappings and define transformation rules. The detection methodology should overcome the barriers of cross-domain (also cross modeling) techniques, and require as little manual effort as possible. Considering this point, semantic and syntactic checking measurements (S&S) are involved in AMTM as part of the detection methodology.

As a short conclusion to this section, the social problem that we are trying to solve is to improve enterprise interoperability and simplify the integration process of EISs. Within this social problem, a typical problem is to share and exchange data among EISs efficiently and effectively. To achieve this aim, we propose using model-based system engineering to address this problem. Specifically, an automatic model transformation methodology, "AMTM" is proposed. To overcome the weaknesses in traditional model transformation instances, AMTM includes S&S as part of the detection methodology.

3 State-of-the-art

This section presents the content in two aspects: (i) the aspect related to the model transformation domain, and (ii) the aspect related to enterprise integration and collaboration. For both of these aspects, we compare, contrast and position AMTM within the existing works.

3.1 Model transformation domain related principles and techniques

This subsection is divided into four parts. The first part describes basic concepts of model, meta-model and model transformation; these concepts are important because they give a better understanding of the model transformation domain. The second part concerns the category of model transformation instances and particularly focuses on the model-to-model category. In the third part, two prominent model-to-model mapping and transformation techniques are illustrated and compared. The fourth part presents and compares several model-to-model transformation practices and other research works that are relevant to AMTM.

3.1.1 Model, meta-model and model transformation

The terms "model", "modeling" and "model transformation" have been used in previous sections. This subsection presents clear definitions of these terms.

Since model transformation takes place between models, it is necessary to have a clear understanding of what models are and how they are built. As defined in

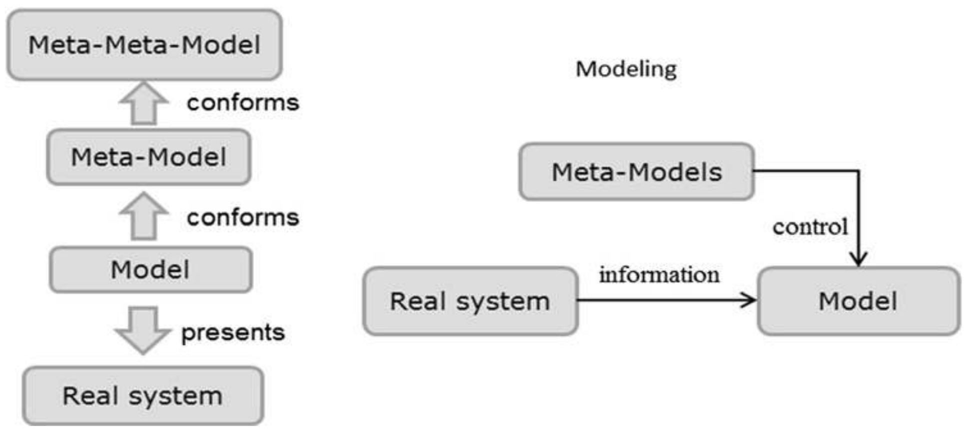


Fig. 4 The relationship between real system, model, meta-model and modeling

Bezivin (2006), “model is a simplification of the subject and its purpose is to answer some particular questions aimed towards the subject. Meta-model is a specific kind of model; it makes statements about what can be expressed in valid models.” In simple terms, a model represents a real subject by focusing on some of its characteristics. A meta-model, which is a special kind of model, defines the rules for building them. Other definitions about model and meta-model are also given in Vernadat (1999) and Terrasse et al. (2005). The activity of building models called modeling. The relationship between real system, model, meta-model and modeling is shown in Fig. 4.

Meta-models can exist on several abstract layers. The higher the layer, the more abstract the metamodels tend to be. A model is built to represent real systems and it may conform to one or several meta-models. With the help of meta-models, the models can be understood.

Since meta-models affect the process of modeling, they play a key role in detecting model transformation mappings and defining transformation rules. As defined 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 model-driven architecture (MDA) as “the process of converting a model into another model of the same system” (Miller and Mukerji 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”. Generally speaking, model transformation is a process of generating target models based on source models.

3.1.2 Model transformation categories

Because model transformation techniques and instances are developed for different purposes (to serve special situations), the category of model transformation is presented first.

In general, model transformation can be divided into three groups. A summary of these three groups is shown in Table 1. Normally, the content presented in models is described in abstract syntax, while the content presented in text is described in

Table 1 Categories of model transformations

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

concrete syntax. In some terms, text (code is a special kind of text) could also be regarded as a kind of model.

As defined in Czarnecki and Helsen (2003), there are two main model transformation approaches: model-to-code and model-to-model. For the model-to-code category, there are two kinds of approaches: "visitor-based" approaches and "template-based" approaches. For model-to-model categories, there are five approaches: the "direct-manipulation" approach, the "relational" approach, the "graph-transformation-based" approach, the "structure-driven" approach and the "hybrid" approach. The applicable situations, working mechanism and instances of these approaches have been explained in Czarnecki and Helsen (2003). For the model-to-model transformation category, there are some other approaches, such as: the marking and pattern approach, the automatic transformation approach, the meta-model based transformation approach, the model merging approach, etc.

AMTM belongs to the model-to-model transformation category. It is designed and implemented as a meta-model based approach and intended to be automatic. Furthermore, the meta-model based model-to-model transformation can also be divided into two situations, which are shown is Fig. 5.

In situation (a), the target meta-model is an evolved version (e.g. with new characteristics added) of the source meta-model. Source models that conform to the source meta-model are updated to target models that conform to the target meta-model. A large amount of research work focusing on this situation has been carried out, such as the methodology named "COPE", which is defined in Herrmannsdoerfer et al. (2009). In situation (b), the source meta-model and the target meta-model are two different models. There is no relationship (e.g. evolutionary relation) between them. In order to transform source models to target models, mappings should be built on the meta-model level and used on the model level.

Considering the context of enterprise interoperability and EIS integration, situation (a), shown in Fig. 5, suits the situation of improving EIS interoperability

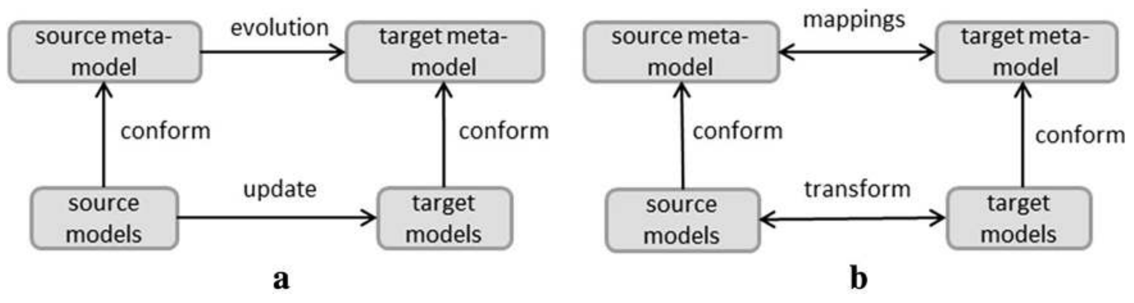


Fig. 5 Two meta-model based model transformation situations

within a specific enterprise. For collaboration that is cross-enterprise, situation (b) could simulate the process of integrating the information systems of different enterprises.

AMTM is able to serve both of these two situations, and furthermore, the precondition of applying AMTM to solve real problems requires less manual effort.

3.1.3 Model transformation techniques

Focusing on different model transformation situations, different model transformation categories and approaches have been defined. To support these approaches, many kinds of model transformation techniques have been developed. This subsection briefly presents two prominent model-to-model transformation techniques: "ATL" (ATLAS transformation language) described in Jouault et al. (2008), Jouault et al. (2006), Jouault and Kurtev (2005), etc. and "QVT" (Query/View/Transformation) defined by Omg (2008).

ATL is a model transformation language and toolkit. Since ATL provides both declarative and imperative constructs, it is regarded as a hybrid model transformation language. Its architecture is composed of three layers: ATLAS Model Weaving (AMW), ATL and ATL Virtual Machine (ATL VM). AMW may optionally be used as a higher abstraction level transformation specification language. Transformation programs are mainly written with ATL, and ATL VM is in charge of compiling ATL programs. ATL has its advantages: it is module independent and easy to maintain, supported by executable software tools, etc. ATL provides ways to produce a set of target models from a set of source models.

QVT is a standard set of languages for doing model transformation defined by "Object Management Group". It is capable of expressing queries, views, and transformations over models. The QVT standard defines three model transformation languages: relations, core, and operational mappings. Moreover, it integrates the "Object Constraint Language OCL 2.0" standard and also extends it with imperative features. It is also a hybrid transformation language with declarative and imperative constructs. Some of the advantages of QVT can be summarized as: strong theoretical support, self-explained and full-featured.

Table 2 shows comparisons of some characteristics of the two model transformation techniques.

Table 2 Comparisons between ATL and QVT

Name	Hybrid	Rule scheduling	M-to-N mappings	Interactive transformation	Software tool support	Note
ATL	Yes	Implicit internal explicit	Yes	No	Yes	Using AMW
QVT	Yes	Implicit internal explicit	Yes	No	No	Based on MOF 2.0

Both ATL and QVT serve general purpose. They have been used or adapted to solve real problems from different engineering domains. However, one of the common weaknesses of the two techniques is that they are complex to learn and use properly. Since they aim to serve general purposes, they provide huge function sets. Both of them define some strategies to do automatic or semi-automatic model transformation. However, a large amount of manual effort is still required to perform them well.

Thus, model transformation techniques can be divided into two groups: one to serve general purposes and one to serve specific purposes. Normally, specific-purpose model transformation techniques focus on particular problems. Therefore, the use of these techniques is limited to narrow situations, and they are not flexible for some special cases. On the other hand, these kinds of techniques can be executed automatically or semi-automatically in some aspects. General-purpose model transformation techniques, which could serve in a cross-domain context, are always complex to learn and use. Furthermore, large amount of user' effort is required to implement these kinds of techniques.

As indicated in previous sections, AMTM uses S&S as a detecting technique as it has both the advantages of specific-purpose and general-purpose techniques and is supported by a specific software tool.

3.1.4 Model transformation instances

To solve real problems, a large amount of model transformation instances have been developed, based on specific model transformation techniques. Like model transformation techniques, model transformation instances can also be divided into two groups: specific-purpose instances and general-purpose instances.

Focusing only on the category of model-to-model transformation, this subsection presents several model transformation instances from this category. To select these instances, we consider the following conditions.

- The instances should be developed based on mature model transformation techniques.
- The instances should be supported by software tools (not just theoretical solutions).
- The instances should apply to automatic or semi-automatic execution issues.

The main characteristics are summarized here, as well as the weaknesses that exist in these model transformation instances. As a comparison, we illustrate how AMTM overcomes these weaknesses.

Table 3 shows ten particular research techniques that are relevant to AMTM and compares their characteristic with AMTM. The comparison aspects focus on: the technique used, its applicability, and its main features. More details about these instances can be found in their references, as shown in the table.

Different model transformation theories and techniques have been used to create model transformation instances that serve various purposes, such as enterprise integration, software development, etc.

Models are becoming more and more complex and heterogeneous because of the numerous and heterogeneous modeling techniques and the complex systems that are being modeled. Thus, more and more model transformation instances, which focus on particular problems (or provide solutions to a set of special problems), are being developed. Normally, these instances are developed based on specific model transformation techniques and their usage is limited (with strict conditions). However, with the emergence of new collaborative situations (heterogeneous partners involved in achieving a common goal) general-purpose model

Table 3 Related works of model transformation related instances

Name	Technique	Domain specific	Note
Generic and meta-transformations for model transformation engineering (Varró and Pataricza 2004)	VPM metamodeling framework	No	Model transformations are also regarded as models; supported by VIATRA tool
Metamodel matching for automatic model transformation generation (Falleri et al. 2008)	Similarity Flooding algorithm	No	Aims at transforming automatically; work well only between similar meta-models
Transformation of decisional models into UML: application to GRAI grids (Grangel et al. 2010)	ATL	Yes	Focusing only on transforming GRAI Grids to UML model
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
Engineering model transformations with transML (Guerra et al. 2013)	MDE philosophy-based	No	Cohesive support for model transformations without providing transformation languages
Applying MDE to the (semi-) automatic development of model transformations (Bollati et al. 2013)	MeTAGeM (Bollati 2011)	No	Applying MDE principles to define model transformation
Model transformation co-evolution: a semi-automatic approach (García et al. 2013)	ATL	No	A meta-model based model transformation approach; specially focus on co-evolution situation (e.g. COPE)
Matching metamodels with semantic systems-an experience report (Kappel et al. 2007)	Semantic checking measurements	Yes	Integration of modeling languages via metamodels
A survey of schema-based matching approaches (Shvaiko and Euzenat 2005)	Semantic checking measurements	Yes	A new classification of schema-based matching techniques is presented
A survey of exploiting WordNet in ontology matching (Lin and Sandkuhl 2008)	Semantic checking measurements	Yes	Using concrete semantic thesaurus to do ontology matching

transformation practices are needed. Furthermore, these general-purpose model transformation practices need to be supported by general-purpose model transformation techniques and software tools. The less manual effort-involved, the better model transformation practices would be. Semantic (& syntactic) checking measurements can be used as a general-purpose technique of this kind, and several domains have already adopted them to achieve automation and replace manual effort. For the model transformation domain, where models are used as the first-class citizen, the theories and practices of using S&S are still immature. AMTM is an attempt to fill this gap.

3.2 Workflow management for enterprise integration and collaboration

The Business Process Management approach (BPM) can be carried out using several tools applied in the different states of the studied system. Several BPM lifecycles have been defined in the literature. Some are oriented more towards a business perspective and others also take into account the technical level i.e. the level that involves IT engineers to implement and execute the process. Wetzstein et al. (2007) proposes a decomposition through four phases:

1. Process Modelling is about "drawing" the business process according to modelling languages and using specific graphical modelling tools.
2. Process Implementation consists in transforming and enriching this business process model into an executable model. In the context of Service Oriented Architecture (SOA) the executable model could be a Business Process Execution Language (BPEL) model that states which web service should be invoked for each task of the business process.
3. Process Execution deals with the execution of the process with a process execution engine.
4. Process Analysis, whose goal is to monitor the process as it is running, is an analysis of the business process through specifically chosen key performance indicators to evaluate and enhance it.

Van der Aalst (2013), Weske (2012) and Jung et al. (2007) agree on a rather high-level decomposition of BPM phases that can be summarized through four steps:

1. *Model* relates to the previous process modelling step (Wetzstein et al. 2007).
2. *Enact* includes the previous implementation and execution steps (Wetzstein et al. 2007).
3. *Analyse* embedded in previous analysis step (Wetzstein et al. 2007).
4. *Manage* also included in previous analysis step (Wetzstein et al. 2007).

In order to retain the business/IT level consideration, Benaben et al. (2015) split the cycle into two overall parts: the design-time (when the process is modelled) and the run-time (when the process is executed). In addition, Jung et al. (2007) discusses semantic BPM and proposes an integration of knowledge management within the BPM lifecycle, which brings an interesting third point of view on BPM.

The results presented in this article mainly focus on implementation and execution of workflow [steps 2 and 3 of (Wetzstein et al. 2007)] or enactment of BPM [from Van der Aalst (2013), Weske (2012), Jung et al. (2007)].

4 Overview of AMTM and S&S comparisons

This section presents an overview of AMTM. AMTM aims to complete the process of detecting transformation mappings and defining transformation rules automatically (without or with little user effort). This section contains two main parts: the first part includes three main subsections: (i) objective and precondition of using AMTM, (ii) basic theories of building AMTM and (iii) AMTM working mechanism. The second part details the syntactic and semantic checking measurements involved.

To make AMTM understandable to a non-expert audience, a simple case of transforming date formats (coming from different cultures), called "DFT-UC", is used to help explain it. This case is the description of the date "First of June, 2015". It is written as: "First of June 2015 or '6/1/2015'" in America, "Le premier juin 2015 or '01/06/2015'" in France and "二零一五年六月一日 or '2015-06-01'" in China. The same concept is presented in different semantic and syntactic representations.

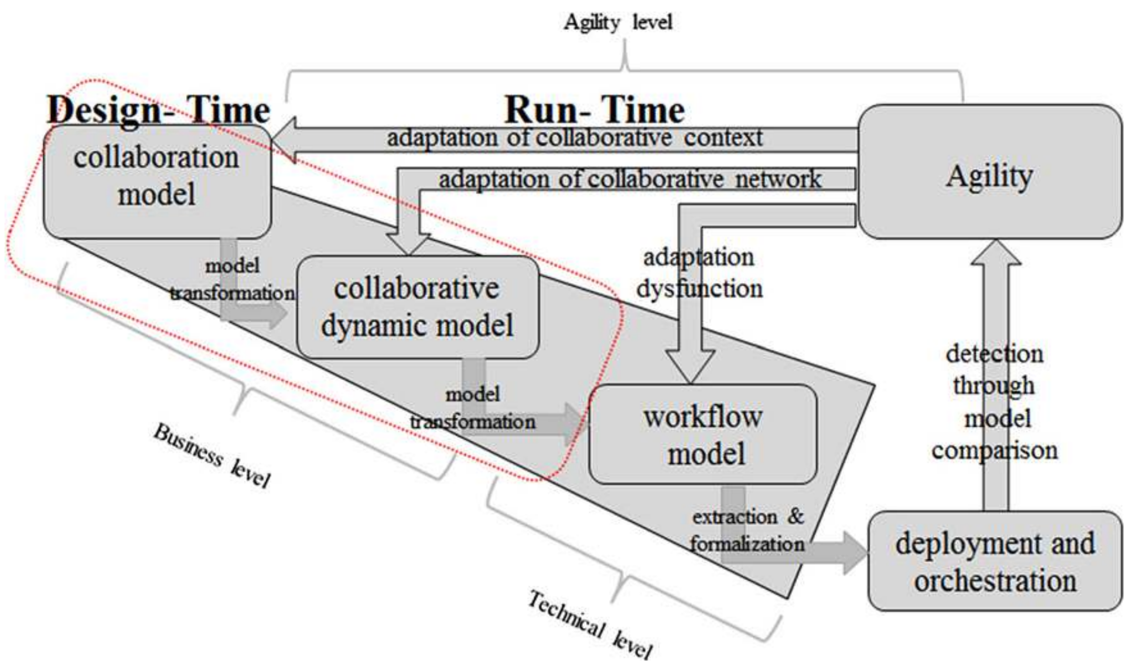

Agility level
Design- Time
Run- Time
adaptation of collaborative context
adaptation of collaborative network
collaboration model
Agility
model transformation
collaborative dynamic model
adaptation dysfunction
detection through model comparison
model transformation
workflow model
Business level
extraction & formalization
deployment and orchestration
Technical level

Fig. 6 The architecture of MISE and the working part of AMTM

4.1 Objective and precondition of using AMTM

4.1.1 Objective of AMTM

The initial objective of developing AMTM is to serve model-based systems engineering. The project of building AMTM is a part of a large research project named “Mediation Information System Engineering (MISE)”. The latest work on MISE can be seen in Benaben et al. (2012). MISE has a four-layer architecture, and model transformations are needed within each layer and between adjacent layers. Figure 6 shows a simple illustration of MISE and the working part of AMTM within it.

MISE contains four main implementation steps: design of the collaboration model, deduction of the collaborative behavior model, design of collaborative workflows and deployment and orchestration of the MIS. In the first, second and third steps, numerous models will be built; model transformations are needed to connect the models built within the same layer. Furthermore, between adjacent layers, model transformations help to automatically generate models for the following step. AMTM aims to improve the agility of MISE by removing the manual effort involved.

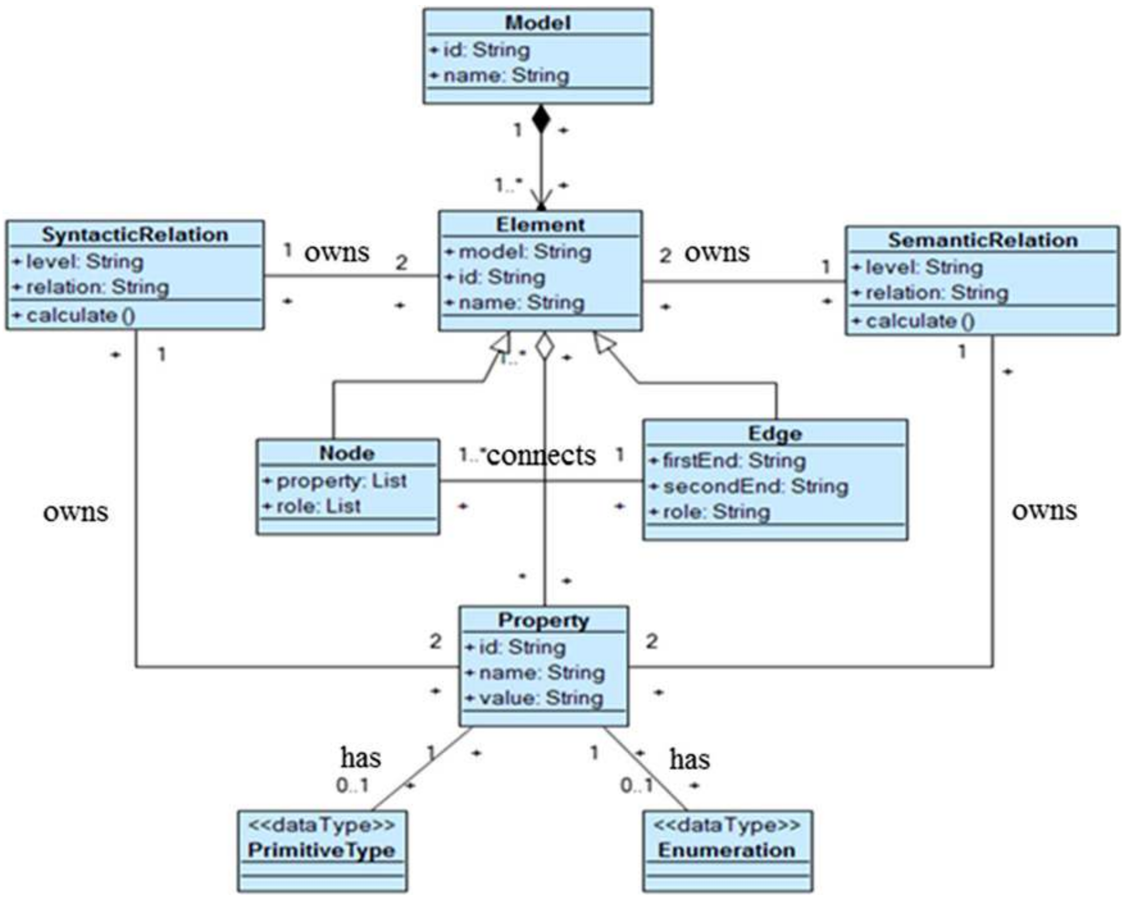


Fig. 7 The structure and content of MMM

4.1.2 Precondition of using AMTM

Like other model-to-model transformation instances, some preconditions are required to use AMTM. These preconditions mainly concern the detecting techniques involved in AMTM. As stated in previous sections, S&S is used to detect the model-to-model mappings, and in AMTM the mapping and transformation rules are defined on the meta-model level. However, according to Jouault et al. (2008), semantic checking measurements can not be applied to two models conforming to different meta-models. Therefore, one of the preconditions of using AMTM is that "the source and target meta-models should conform to one meta–meta-model". Based on this cognition, a specific meta–meta-model (MMM), which works on a high abstract level, is defined in AMTM.

The meta–meta-model defines the rules for meta-modeling. There are several meta-modeling architectures, two of the prominent ones being "MOF: Meta-Object Facility" (OMG 2008) and "ISO/IEC 24744" defined in Henderson-Sellers and Gonzalez-Perez (2008). However, such architectures always aim to serve general purposes and provide solutions to various domains. They define their own semantic and syntax representations and provide a wide range of functions to build meta-models. For AMTM, which focuses particularly on model-to-model transformation, these meta-modeling architectures are huge, complex and unsuitable for this purpose. Figure 7 shows the MMM defined in AMTM.

This MMM is designed particularly to serve model-to-model transformation in the context of EIS integration and interoperability. This MMM works on a high abstract level; it defines nine core items.

- "Model", stands for all kinds of models: e.g. EIS design models, collaborative situation simulation models and integration models. Models are made of elements.
- "Element", represents all the possible items contained in the models (elements are self-contained). In other words, "Elment" is made up of properties. It has two inheritances: "Node" and "Edge".
- "Node", stands for a subject or a concept; it is used to stand for subjects that exist in the world.
- "Edge", describes the relationship between "Nodes". Every "Edge" links two roles (every node belongs to at least one role). An "Edge" is a special kind of "Node".
- "Property", works as an identifier of the "Element". An "element" has a group of properties, and all these properties explain the element as a whole. Each "Property" has a "Data Type", which identifies the property itself. The "Data Type" can be either a "Primitive Type" or an "Enumeration type".
- "Primitive Type", stands for the formal property types: string, integer, double and Boolean, etc.).
- "Enumeration type", stands for the user-defined type.

Two special items in this MMM are “Semantic Relation” and “Syntactic Relation”. They are built among elements and properties to detect potential mappings and transformations.

- “Semantic Relation”, exists between “Element” pair, “Property” pair and also across “Element” and “Property”; it helps to define the transformation mappings automatically.
- “Syntactic Relation” is similar to “Semantic Relation”; it is used as an adjacent to “Semantic Relation” to detect model transformation mappings and transformations.

For the example of DFT-UC, the model involved could be the rules for describing the date (time) defined in different cultures. Each model contains elements like: year, month, day, hour, minute, second, etc. Each element contains a group of properties, such as: year-leap year, number of days in each month, etc. Thus, relations (Edge) can be built between different Nodes within one model. Between different models, semantic relations and syntactic relations can be built among their elements.

With the help of this MMM, semantic checking measurements can be used on meta-models. Users are only required to provide meta-models that conform to this MMM (since the MMM works on a really high abstract level, it is easy to modify meta-models to conform to it). Within AMTM, some algorithms are implemented to deduce from meta-models provided by users the new versions that conform to this MMM. In other words, the deduction process can be done automatically.

4.2 Basic theories of building AMTM

Unlike many other model transformation methodologies, in AMTM model transformation is regarded as an iterative. Figure 8 is a simple illustration of this iterative process.

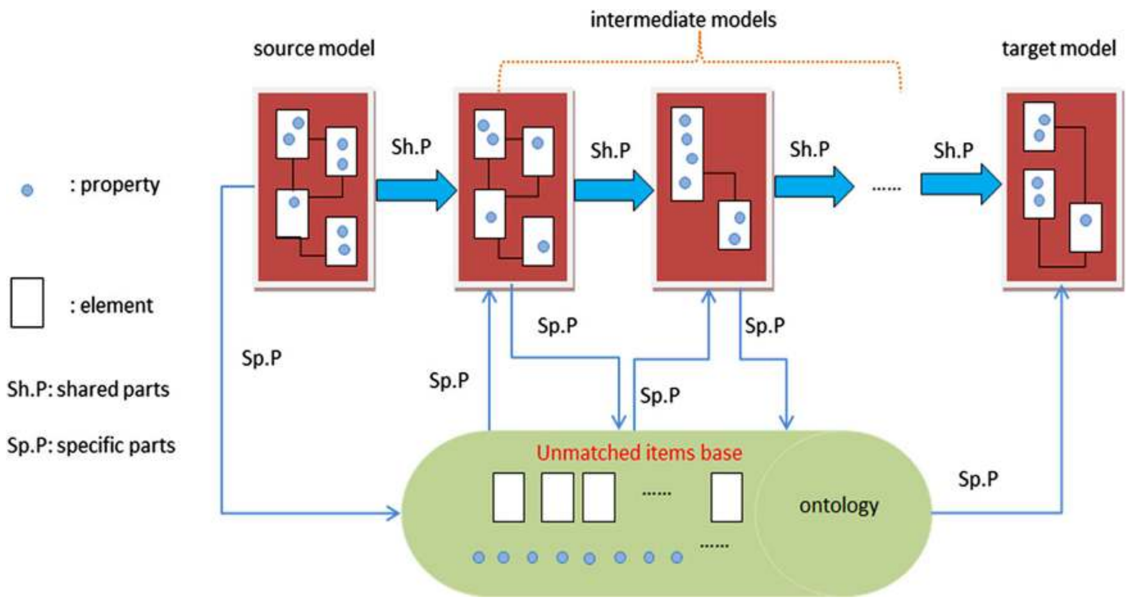


Fig. 8 The iterative model transformation process

Between source and target models, several intermediate models can exist. In each iteration phase, model transformation mappings and rules are built among elements and properties (as illustrated with the MMM). Normally, during the model transformation process, the source and target models contain shared parts and specific parts. The items between source model and target model that can be transformed are shared parts, while the items that cannot be matched are regarded as specific parts. The mapping and transformation rules are built between the shared parts (same or similar concepts) of the two adjacent models. In DFT-UC, the American format for describing dates could be used as an intermediate target model between the French one and Chinese one.

Within each iteration phase, the specific parts from the source model need to be stored and the specific parts of the target model need to be enriched. The iterative process allows the specific parts stored from former transformation phases to be used to enrich the specific parts of the target models that are generated in the later transformation phases. In AMTM, a special ontology is created to store and reuse these specific parts generated during the iterative transformation process. This ontology is designed with the same structure as the MMM and is called "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 Fig. 9.

As explained in the third section, AMTM builds model-to-model mappings and transformation rules on the meta-model level among elements and properties. The detection process contains four main steps: (i) applying S&S measurements on the

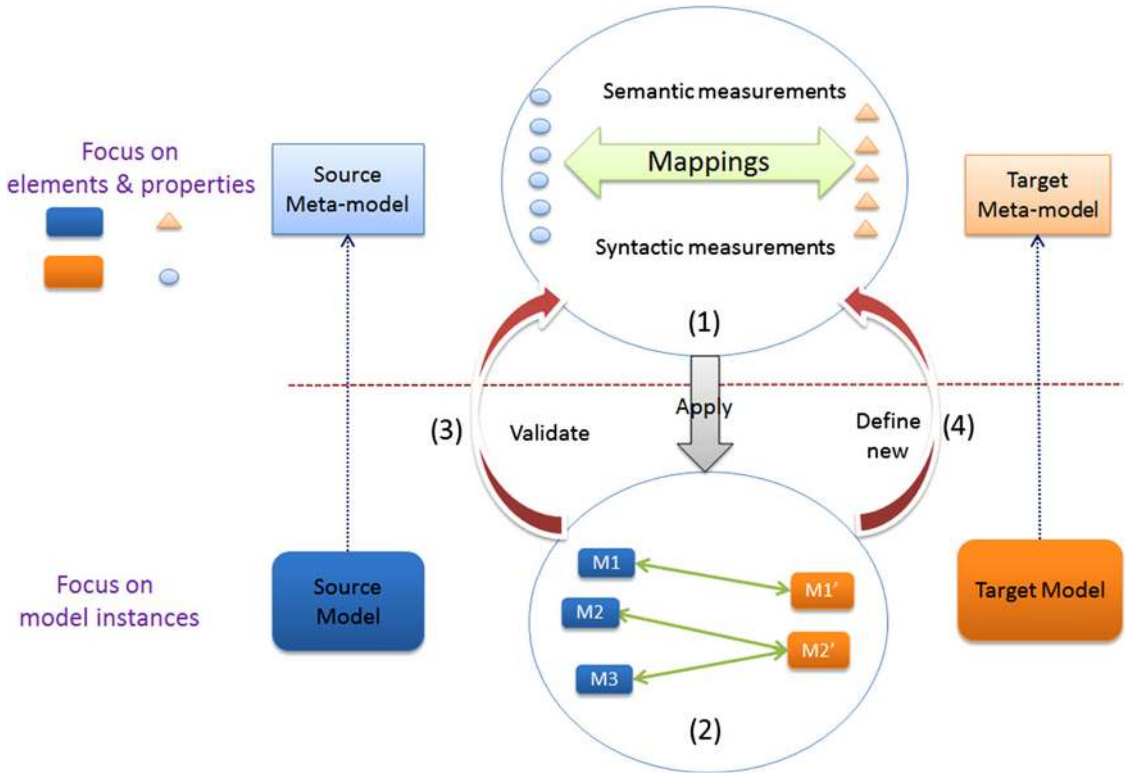


Fig. 9 The process of building model transformation mappings and rules within each iteration

meta-model level to generate potential mappings and transformation rules, (ii) using the generated mappings on the model level to test transformation results, (iii) according to the transformation results, validation of the potential mapping and transformation rules and (iv) examination of the validation results to define new mappings and transformation rules on the meta-model level.

The idea of regarding model transformation as an iterative process and dividing models into shared parts and specific parts during the transformation process is adapted from Bénaben et al. (2010). At this moment, AMTM focuses only on the first step: using S&S to detect the potential mapping and transformation rules. The last three steps are more concerned with the model level, and this paper does not detail the relevant content.

4.3 AMTM working mechanism

The granularity issue, as one of the main stubborn problems in model transformation domain, concerns about mixing matching among elements and properties. In order to solve it, AMTM applies S&S in three matching steps: matching within elements, hybrid matching and auxiliary matching. All of the three matching steps are detailed in the following subsections.

4.3.1 Matching within elements

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 matching pairs of elements. If two elements coming from the source model and the target model stand for the same (or similar) concept(s) (shared parts on the meta-model level), a mapping can be built between them. In DFT-UC, “June”, “六月” and “juin” stand for the same concept on the element level; they should thus share several similar properties.

How to detect if two elements stand for the same concept or not? AMTM tests the semantic and syntactic relations between two elements’ names and their property groups. Figure 10 shows a simple illustration of the detection process in the first matching step.

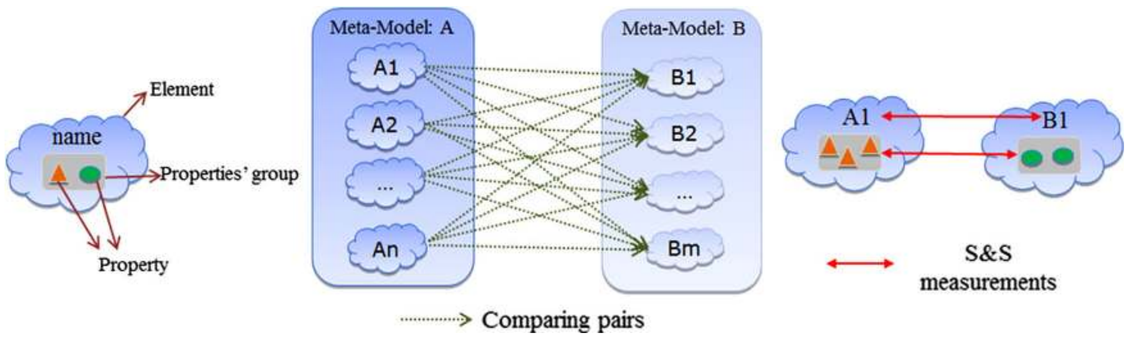


Fig. 10 Building mappings on element level

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 in Fig. 10). An element from the source meta-model should be compared with all the elements from the target meta-model until a matching one (or no matching element) is found. So, in this example, every element of meta-model A will be compared at least once and at most ‘m’ times with the elements from meta-model B.

A specific value “Ele_SSV” is calculated for each compared 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 Eq. (1).

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

“Ele_SSV” is the sum of two independent parts: elements’ names and elements’ groups of properties; 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 can assign values to the two factors to determine the mutual importance between the two parts: elements’ names and elements’ groups of properties.

“S_SSV” stands for “semantic and syntactic value between two words (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 how to calculate this value is presented later in this section.

“P_SSV” stands for “semantic and syntactic value between a pair of properties”. In Eq. (1), “x” stands for the number of properties of a specific element from the source meta-model. To match a pair of elements, all of the properties from the source element should be considered to make a match with the ones from the target meta-model elements. But how can calculate “P_SSV” be calculated? An example is shown below.

When comparing element “A1” and element “B1”, their property groups are taken into consideration. Assuming that “A1” has “x” properties and “B1” has “y” properties; the maximum number of comparisons on their property level would be “x*y”. A “P_SSV” exists in each pair of compared properties’. Equation (2) shows the calculation rule for “P_SSV”.

$$P\_SSV = pn\_weight * S\_SSV + pt\_weight * Id\_type \tag{2}$$

The calculation rule for “P_SSV” is similar to the one for “Ele_SSV”. It is also the sum of two parts: properties’ names and property types; and also two impact factors “pn_weight” and “pt_weight”, which are used to determine the weight of the two parts. The rules for assigning values to the two impact factors are the same as the ones in Eq. (1).

In Eq. (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 with value "1"; otherwise, this value could be 0.5 (e.g. double and float) or 0 (e.g. integer and string).

With the help of Eqs. (1) and (2), every element from the source meta-model could get one (or several) potential matching element(s) from the target meta-model. However, the detected pair is only a potential mapping. To become a real transformation rule, a mapping selection mechanism, which is presented later in this section, is needed. Within potential matching pairs of elements, their properties are also mapped based on the "P_SSV" values calculated by using Eq. (2).

4.3.2 *Hybrid matching*

Some of the elements from source meta-models are left unmatched after the first matching step. Even for the matched elements, some of their properties might still be unmatched. The hybrid-matching step focuses on these unmatched elements and properties. In DFT-UC, a property of element 'day' could be a special festival such as: Easter: Easter in American is Pâques in French, but this festival does not exist in Chinese.

The hybrid matching step focuses on properties; all the matching pairs would be built among properties. The idea of building mappings in this step is simple: comparing all the unmatched properties from the source meta-model with all the properties from the target model, in order to find similar pairs. Figure 11 is an illustration of the hybrid matching step.

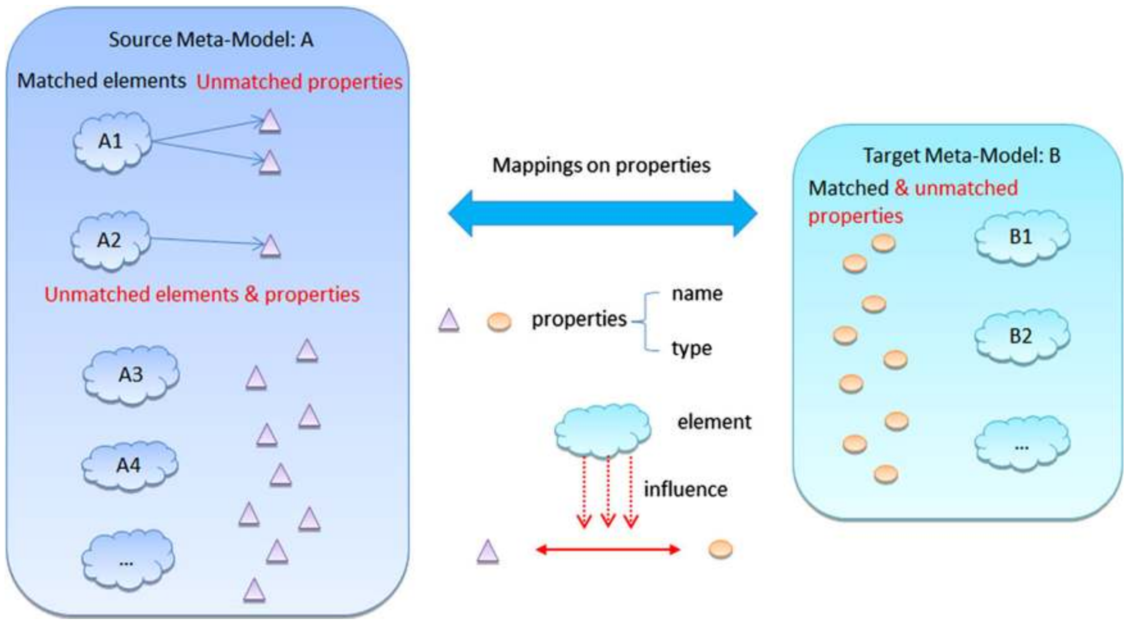


Fig. 11 Hybrid matching step illustration

This step aims to break one of the main granularity constraints: property-matching pairs only exist within matched element pairs. This step implements many-to-many mappings on the element level (Properties from one element can be transformed to several target elements while one target element can be generated by combining properties that come from several source elements). When comparing two properties, this step also considers the influence of an element's level. The matching mechanism of this step is shown in Eq. (3).

$$HM\_SSV = en\_weight * S\_SSV + pl\_weight * P\_SSV \tag{3}$$

"HM_SSV" stands for "hybrid matching semantic and syntactic value"; the idea of calculating this value is similar to those 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 roles as "name_weight" and "property_weight" in Eq. (1). The influence of the element level mainly depends on the element names. In Eq. (3), "S_SSV" calculates the semantic and syntactic value between two element's names.

4.3.3 Auxiliary matching

All the shared parts between source meta-model and target meta-model are considered to have been found after the first and second matching steps. The specific parts of source and target meta-models are still left untreated. The auxiliary matching step focuses on these specific parts.

In DFT-UC, consider again the special festival issue while transforming the date from American to Chinese (first iteration) and then transforming Chinese date to French date (second iteration). Some special festivals will be lost during the first iteration, and the lost part is needed as additional knowledge to enrich the target model (French date format) generated in the second iteration (French and American share some festivals while Chinese does not).

The auxiliary matching step defines the mechanism of storing and reusing the specific parts of source and target meta-models. After the first two matching steps, all the unmatched items from the source meta-model are stored in AMTM_O. For a complete model transformation process, the specific parts from former transformation iterations should be reused to enrich the specific parts of the target model.

Furthermore, the content in AMTM_O could also be enriched by other ontologies and knowledge from other domains, because the specific parts from the source meta-models might not be enough to generate all the specific parts needed by the target meta-models that are in the same iterative process.

4.4 Semantic and syntactic checking measurements involved

AMTM uses semantic and syntactic checking measurements as techniques to detect model-to-model mappings. In AMTM, the two checking measurements always work together to define a relationship, which has a value ranging between 0 and 1, between two words (names of elements' and properties'). The higher the value, the more similar the two words are.

Both in Eqs. (1) and (2), the "S_SSV" is used as a parameter to determine the semantic and syntactic value between two words. Now, the calculating rule for "S_SSV" is shown in Eq. (4).

$$S\_SSV = sem\_weight * S\_SeV + syn\_weight * S\_SyV \tag{4}$$

Equation (4) follows the same design idea as the former three equations. The values of two impact factors "sem_weight" and "syn_weight" are left to users to be assigned (to determine the mutual importance between semantic and syntactic relations between two words). "S_SeV" stands for the semantic value, while "S_SyV" stands for the syntactic value. Based on different application situations, the two aspects "semantic" and "syntactic" may have different effects (e.g. in enterprise engineering, the semantics may be more important but in a medical field, the syntactic meaning may be more important).

This subsection contains four parts: (i) presents the syntactic checking measurements, (ii) shows the semantic checking measurements, (iii) illustrates the matching pair selection mechanism based on the semantic and syntactic checking results and (iv) is a short conclusion of this subsection.

4.4.1 Syntactic checking measurements

An element has two main identifiers: name and properties; to differentiate two properties, their names and types need to be taken into consideration. So, the names (represented by natural words) of both element and property play a key role in identifying them.

Syntactic checking measurements are used to calculate the syntactic similarity between two words. In AMTM, syntactic checking measurements are regarded as an adjunct to semantic checking measurements. If two words have high syntactic similarity, they might stand for the same thing. Furthermore, the same two words always convey the same meaning. Since detecting semantic meanings and relations is a time-consuming process, it is better to do syntactic checking first (to replace semantic checking in some aspects).

The syntactic checking measurements involved in AMTM contain two steps: (i) test if two words that are in different forms nevertheless stand for the same word (with the same meaning), (ii) calculate the syntactic similarity between two different words.

4.4.1.1 Predefined syntactic checking To test if two words, which are in different forms, stand for the same word, we adopt the ideas defined in the "Porter stemming algorithm" (Willett 2006). Table 4 shows several typical pairs of words to be detected. They have different forms (e.g. tense, morphology) but stand for the same word.

There are many such situations where two words in different forms stand for the same meaning. The Porter stemming algorithm tries to detect all of them

Table 4 Several situations that Porter stemming algorithm focuses on

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	…	…	…

automatically since it can remove the more common morphological and inflexional endings from words in English. The main use of this algorithm is as part of a term normalization process, which is usually done when setting up information retrieval systems. The detail of this algorithm can be found in Willett (2006). Another similar stemming algorithm is illustrated in Porter (2001).

4.4.1.2 “Levenshtein distances” algorithm In AMTM, to detect the syntactic similarity between two different words, the “Levenshtein distances” (Gilleland 2009) algorithm is applied.

Syntactic checking methods are defined to calculate syntactic similarity between two words. Among all the syntactic checking methods, classic similarity metrics is used as one of the prominent methodologies. A conclusion on the existing syntactic checking measurements is given in Cohen et al. (2003).

Because “syntactic checking measurements are used as an adjunct to semantic checking measurements in AMTM”, a simple syntactic checking measurement, “Levenshtein distances”, is chosen to complete this step (this helps to improve the efficiency of the whole detecting process). The “Levenshtein distances” algorithm focuses on the occurrences of the letters involved in words and calculates the number of operations that are needed to transform one word to another. The “Levenshtein distances” is equal to the number of operations needed to transform one string (word) to another. There are three kinds of operations: insertions, deletions and substitutions. The basic theory and exact execution process of this algorithm is detailed in Heeringa (2004).

A simple example is given below to illustrate the mechanism of using the “Levenshtein distances” algorithm. This example is to calculate the “Levenshtein Distance” between two words: “sun” and “son”.

The two words are listed in Table 5, and the letters (ignore cases) involved are listed and marked with their positions in the words. The “Levenshtein distances” algorithm defines the rules to calculate the values to fill in the blank. The value in position “ABS” is calculated based on the value above it: “1”, the value to the left of it: “1” and the value in the upper left corner: “0”. The exact rule is: the value from upper and left should add “1”, thus getting “2”; the two corresponding letter (from upper and left) of “ABS?” are the same “s”, so the value in the upper left corner

Table 5 Initiating calculation matrix of "Levenshtein distances"

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

Table 6 The "Levenshtein distances" calculation result of this use 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

should add "0" (otherwise, add "1"). Then, the minimum value from the three (2, 2 and 0): "0" is chosen to fill in "ABS".

Based on the calculating rules, Table 5 can be completed. The final result is shown in Table 6. The final value used is from the lower right corner of the table "1"; it means only one step is needed to transform the word "sun" to the word "son".

Based on "Levenshtein distances", Eq. (5) is defined to calculate the syntactic similarity between two words.

$$S\_SyV = 1 - LD/max(word1.length, word2.length) \tag{5}$$

"S_SyV" stands for the syntactic similarity value between two words; "LD" means "Levenshtein distance" between the two words. The value of "S_SyV" should always be in the range of 0–1; the higher this value, the higher the syntactic similarity between the two compared words.

4.4.2 Semantic checking measurements

As the main clue of to detecting potential model-to-model mappings, the meanings carried by models (elements and properties contained) and semantic relations between the models need to be detected and defined. This subsection is divided into two parts: the first part illustrates the basis of semantic checking and the second part presents the process of carrying out semantic checking. The original idea of doing semantic checking is inspired by the work described in Boissel-Dallier (2012).

4.4.2.1 Basis of semantic checking As opposed to syntactic checking measurements (which focus only on the occurrences of the letters involved in compared words), semantic checking measurements focus on the meanings carried by the words and the relationships between these meanings. A huge semantic thesaurus, containing large amount of words, their meanings and the semantic relations between them, is required to do semantic checking. In AMTM, a semantic thesaurus has been created, called AMTM _ST. AMTM _ST was created on the basis of

"WordNet" (Huang 2007) and adapted its to serve the model transformation domain.

Figure 12 shows the structure of AMTM _ST.

The items stored in AMTM _ST are divided into three categories.

- *Word base* contains the majority of normal English words (nouns, verbs and adjectives) that have a high possibility of being used in the model transformation context.
- *Word-sense base* contains all the word senses that belong to the words stored in "Word Base"; a word can have "one to several" senses. Taking "star" as an example, in total it has six senses; as a noun, it has four senses; as a verb, it has another two senses.
- *Synset base* contains many groups of word senses. The word senses, which in a synset (a group of word senses), have synonymous meanings; semantic relations are built among the different synsets.

Considering the context of making model-to-model mappings, seven kinds of semantic relations are defined and maintained between synsets in AMTM _ST. In order to use these semantic relations to define mappings (used in the equations presented above), a specific value (between 0 and 1) is assigned to each of the semantic relations. Table 7 shows these semantic relations and their value pairs.

For each of the semantic relations, an example of word pairs is shown in Table 7. The corresponding "S_SeV" value [first introduced in Eq. (1)] for a particular semantic relation stands for the similarity of the word pairs from a semantic viewpoint. The higher this value, the closer the two words are semantically.

At present, all these "S_SeV" values are assigned directly (based on experience); in the conclusion section of this paper, another possible method of assigning these values is illustrated.

Thanks to "WordNet", huge amount of words, word-senses and synsets have been collected and defined. AMTM _ST adapts the majority of its word sets and word sense sets and a small part of the semantic relations that are built between synsets. Table 8 shows the quantity of content stored in AMTM _ST.

With the huge semantic-related content stored in AMTM_ST, AMTM provides strong semantic checking measurements to detect model-to-model mappings.

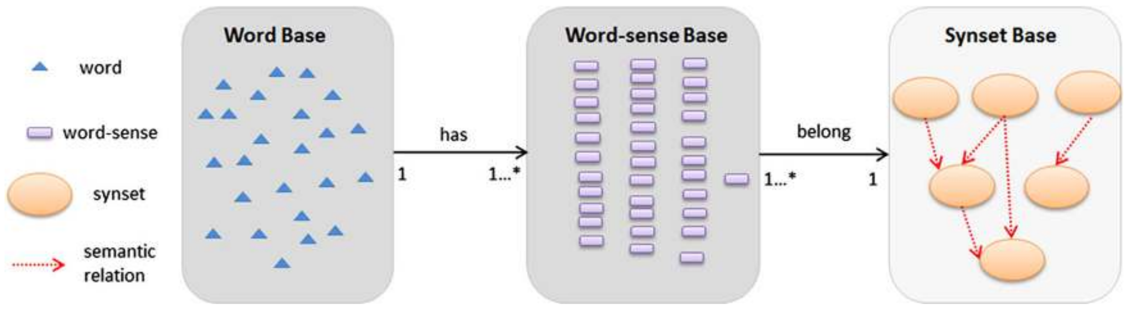


Fig. 12 The structure of AMTM _ST

Table 7 Semantic relations built in AMTM _ST and their value pairs

Semantic relation	S_SeV	Example
Synonym	0.9	shut & close
Hyponym	0.6	creator-person
Hypernym	0.8	person-creator
Similar-to	0.85	perfect & ideal
Antonym	−1	good & bad
Iterative hypernym	0.8^n	person-creator-maker-author
Iterative hyponym	0.6^n	author-maker-creator-person

Table 8 Content stored in AMTM _ST

Items	Number
Words	147306
Word senses	206941
Synsets	114038

4.4.2.2 Semantic checking process Semantic checking measurements take place between elements' and properties' names; the semantic relations between a pair of names are detected. AMTM defines a process of detecting semantic relations between two words. Figure 13 shows the location step of this process.

A word may have several meanings (word senses in AMTM _ST), and thus belong to several synsets. When checking two compared words, the first step is to locate them in AMTM _ST then find all of their word senses. Next these word senses are traced to all their synsets. Finally, the semantic relations between two groups of synsets are traced (one for "Word 1" and one for "Word 2").

After obtaining two synsets groups, the final step is to detect the semantic relations that exist among all the possible synset pairs (one from the word1 side, the other from the word2 side). In Fig. 13, the red dash lines show these possible pairs. The semantic relationships between two words are not limited to 0 and 1. There may be several semantic relations between a specific pair of words.

As illustrated in Table 7, seven semantic relations are defined and maintained among all synsets. These semantic relations can be divided into two groups: simple semantic relations and iterative semantic relations.

Figure 14 shows the mechanism of detecting semantic relations that belong to the simple semantic relation group. Five kinds of semantic relations are involved in this group: synonym, similar-to, hypernym, hyponym and antonym.

The detection principle of this semantic relations group is: for one word, search all the synsets that have the five kinds of semantic relations with the synsets the word belongs to, then compare to see if one same synset exists in the other word's synset group. Figure 14 shows five comparison loops; each loop tries to detect a specific semantic relation between the two words.

For the other group of semantic relations, the detection process of "iterative hypernym" and "iterative hyponym" semantic relations is a little complex. The detection principle of the two, "iterative hypernym" and "iterative hyponym"

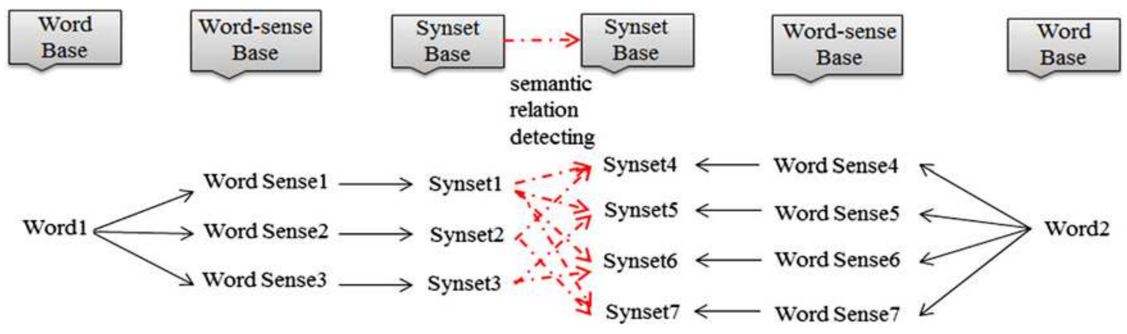


Fig. 13 The location step of semantic relations detection process

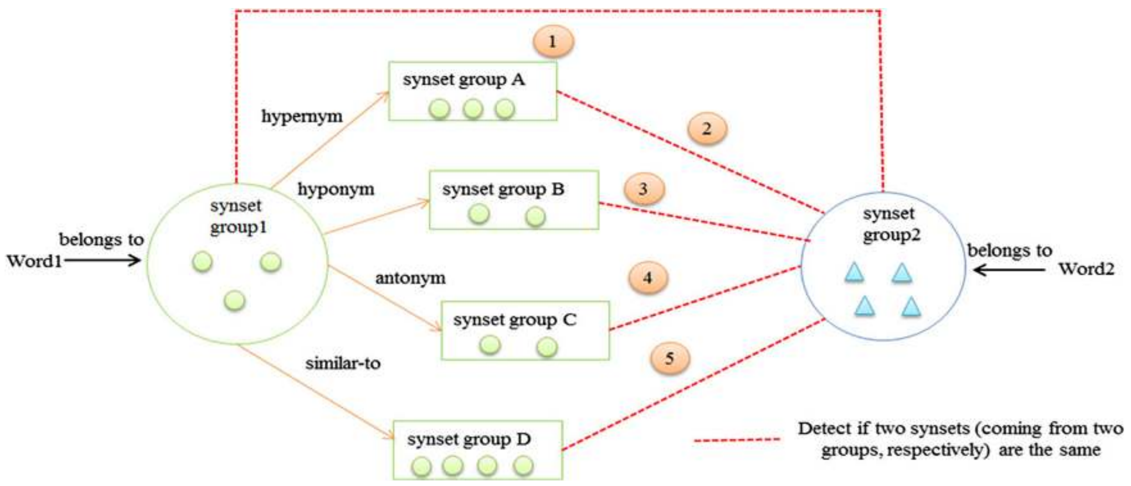


Fig. 14 The detecting process of simple semantic relations

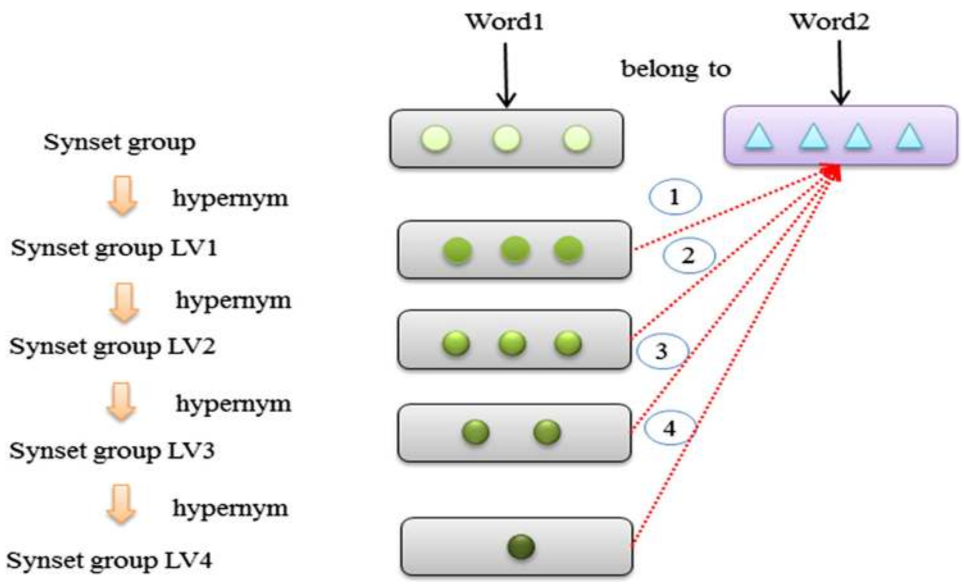


Fig. 15 An illustration of the iterative hypernym semantic relations detection process

semantic relations, is the same. To illustrate this principle, Fig. 15 shows the process of detecting if "iterative hypernym" semantic relations exist between "Word1" and "Word 2".

The iterative semantic relations detection process will be carried out only after the failure of the simple semantic relations checking. The principle of detecting semantic relations for this semantic relations group is: locating iteratively the synsets that have hypernym relations with word1's synsets and comparing them with the synsets that word-2 belongs to, in order to find two synsets that are the same. The iterative loop will be executed no more than four times, given the efficiency of this algorithm and the strength of semantic relations between two words. If there is no semantic relation found after four iterative comparisons, the two words are regarded as having no semantic relations between each other. Then the "S_SeV" between them is "0".

4.4.3 Matching pair selection mechanism

As illustrated in Fig. 3, some selection mechanisms are required to transform model-to-model potential mappings to model-to-model transformation rules. The mechanism of detecting potential mappings has been presented above; this subsection focuses on the matching pair selection mechanism.

The relation between two potential matching elements (coming from source and target models, respectively) is represented by a value "Ele_SSV" between 0 and 1. "Ele_SSV" is calculated based on semantic and syntactic comparisons among two elements' names and property groups. The mechanism of selecting matching element pairs depends particularly on the range of this value. Figure 16 shows the selection mechanism.

As shown in Fig. 16, in situation (a), two threshold values: 0.5 and 0.8 are defined to choose the element's matching pairs. 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 leave to user to decide whether to make the transformation or not). Otherwise, if this value is in region 3, no mappings will be built between the two elements. Situation (b) shows the mechanism of choosing potential matching word pairs. Considering the context of EIS integration and

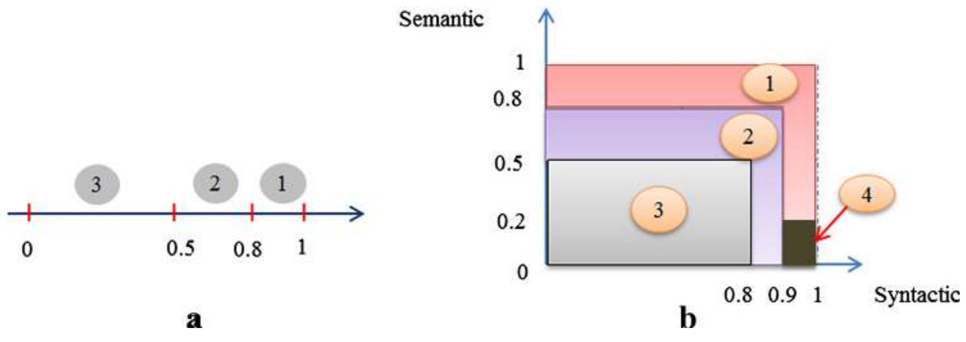


Fig. 16 An illustration of matching pair selection mechanism

interoperability, strong semantic relations means a high potential for making mappings between two words. Region 1 stands for two words that have a close relationship (strong syntactic relation and high semantic relation). The two words having such a relation can be transformed to each other. Region 2 stands for two words that have a high semantic and syntactic relation. Word pairs from here are potentially transformable pairs. Region 3 means two words have weak relationships, and thus a low possibility of being transformed to each other. The special part is Region 4. It stands for word pairs that have strong syntactic relations but very weak semantic relations. For example, the word pair "common" and "uncommon". Such a pair of words is regarded as having an antonymic semantic relation, and thus no transformation mappings would be built between them.

In AMTM, on both the element level and the property level, the same selection mechanism is used to define the final transformation rules. In this way, an element (or a property) may have from "0" to several potential matching items. So, from source meta-model to target meta-model, "many-to-many" model-to-model transformation rules are built on both element and property levels.

4.4.4 Short conclusion

This section presented the S&S measurements involved in AMTM. Semantic and syntactic relations are used together to define a relation (represented by a value "S_SSV") between a pair of words. Based on this "S_SSV" value, relations between different elements and properties can be defined. With the help of a matching pair selection mechanism, potential mappings can be used to define model-to-model transformation rules.

All the algorithms presented in this subsection (e.g. the "Levenshtein distance" algorithm, the "semantic relations detecting" algorithms) have been implemented. Since semantic checking measurements rely on a huge semantic thesaurus "AMTM _ST", this is a time-consuming process. So, syntactic checking measurements are always executed before semantic checking measurements in order to improve efficiency (to reduce the number of AMTM _ST searches).

By combining S&S measurements with AMTM (used differently in three matching steps), an automatic model transformation process is generated. AMTM provides an efficient model-to-model mapping and transformation solution, which it is possible to use to serve EIS integration and interoperability.

5 Supported software tool and use case

The theoretical solution illustrated in the previous sections requires information techniques to support it. One of the purposes of AMTM, to remove manual effort from the model transformation process, also implies a requirement to develop the artificial intelligence aspect. This chapter aims to give a brief illustration of the process of developing a software tool to meet this requirement, called "AMTM-SS". It also explains the working mechanism and describes how the working performance of AMTM is tested by a complete use case. Finally, a performance evaluation is implemented for the use case.

5.1 AMTM-SS design

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

Normally, software development contains four main steps (organized according to the previously mentioned development cycles): requirement analysis, system design, coding and testing. In this subsection, we only present briefly the requirement analysis step and the design step involved in the development process of AMTM-SS.

5.1.1 Requirement analysis

AMTM-SS contains six functional modules.

- *Model analysis module* this module aims to deal with user inputs: different kinds of model sets. The main task of this module is to analyze the input model sets, and, based on the results of this analysis, to deduce the two specific meta-models (between which potential model transformation mappings will be built).
- *Semantic relations detection module* this module focuses on the semantic checking part, which is applied on two deduced specific meta-models. AMTM_ST is implemented and used by this module. Since semantic relations detection is a time-consuming functional process, the algorithms (e.g. searching different items in AMTM_ST, adding new items to AMTM_ST) need to have a high degree of efficiency and performance.
- *Syntactic relations detection module* this module performs a similar role to the semantic relations detection module, but focuses on the syntactic representations carried by the specific meta-models. This module implements the "Levenshtein distances" (also, a part of the "Porter stemming") syntactic similarity checking algorithm.
- *Potential mappings selection module* this module aims to select potential model transformation mappings based on the syntactic and semantic relations checking results. The syntactic relations detection module and the semantic relations detection module generate an S&S value for each potential matching pair (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 defines threshold values to categorize the S&S values. It works as a bridge that crosses the gap between semantic and syntactic checking results and the building of model transformation mappings.
- *Transformation mappings generation module* this module builds potential model transformation mappings and rules based on the selection results. The mappings generated in the potential mappings selection module might be in conflict (e.g.

overlap, Mutex). This module tries to simplify the potential matching pairs (solve the overlap and Mutex problems). This module can reduce the interactions between AMTM-SS and its users.

- *User validation module* all the functional modules presented above aim to generate the potential model transformation mappings automatically. This module provides a solution to validate these automatically-generated mappings. The user interfaces provided by this module serve to connect with users: showing automatically-generated mapping rules, receiving and storing the modifications from the users.

All of the six functional modules focus on their own core functions, but they are not independent of each other. Data and information are passed among them as

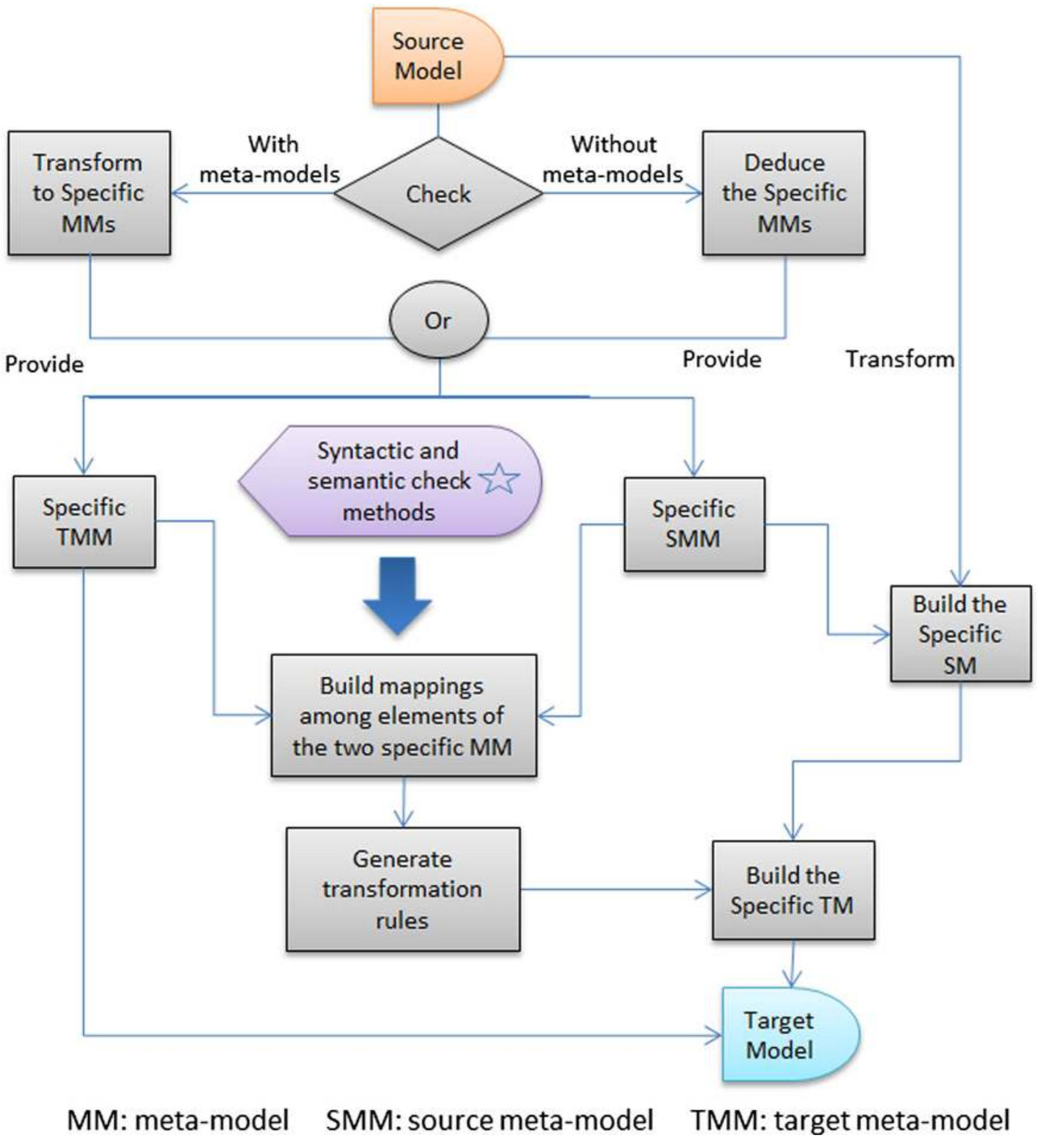

Source Model
With meta-models
Without meta-models
Check
Transform to Specific MMs
Deduce the Specific MMs
Or
Provide
Provide
Transform
Syntactic and semantic check methods
Specific TMM
Specific SMM
Build the Specific SM
Build mappings among elements of the two specific MM
Generate transformation rules
Build the Specific TM
Target Model
MM: meta-model SMM: source meta-model TMM: target meta-model

Fig. 17 A simple illustration of the design of AMTM-SS

parameters. They work together to complete the process of automatically detecting and generating model transformation mappings.

Besides these functional requirements, non-functional requirements are also defined in AMTM-SS. The non-functional requirements concern two main aspects: to remove user effort and to improve system execution efficiency.

As one of the common evaluation standards, execution efficiency for all software systems is important. The final purpose of AMTM-SS is to serve the data and information sharing (exchanging) that exists in collaborations. Thus, high efficiency is also an innate requirement for AMTM-SS. The efficiency of executing AMTM-SS is mainly determined by the execution of syntactic and semantic checking measurements.

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

5.1.2 System design

This subsection briefly presents the system design part of AMTM-SS. The information flow in AMTM-SS is divided into three states: information contained in the input models, information conveyed by the specific meta-models and information carried by the target model. Figure 17 is based on these information states and shows the design part of AMTM-SS.

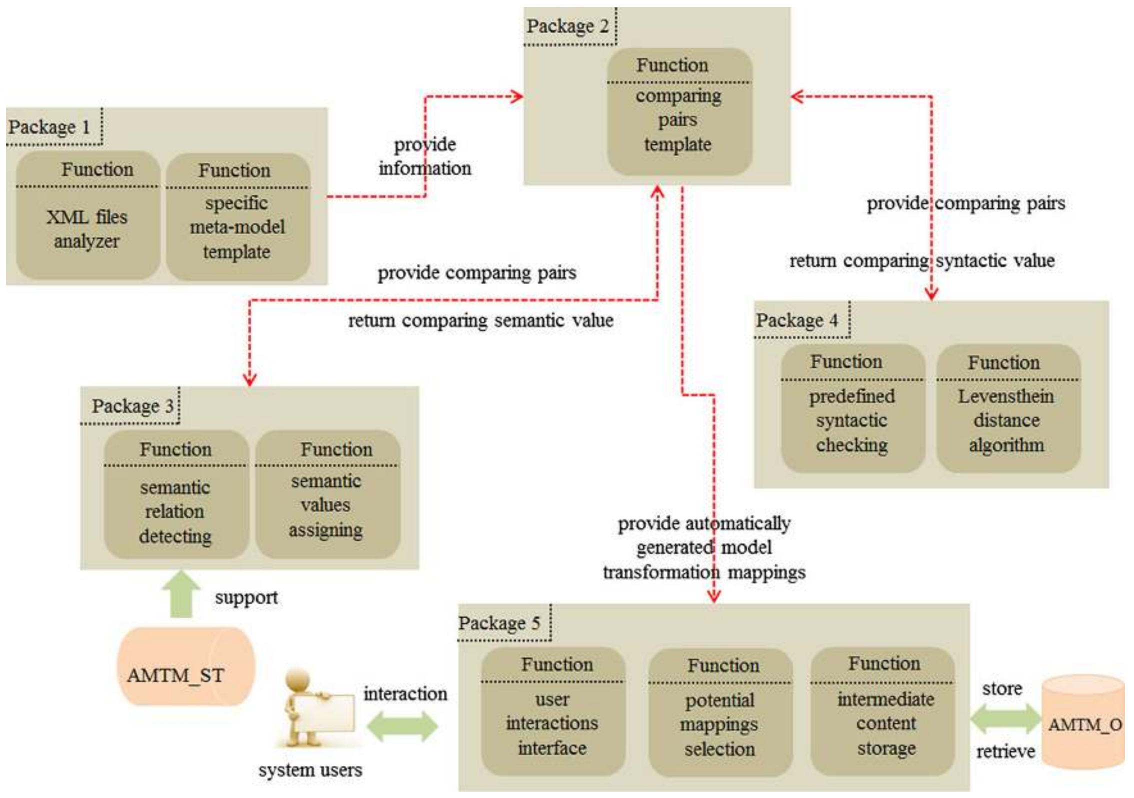

Package 2
Function
comparing pairs template
Package 1
Function
XML files analyzer
Function
specific meta-model template
provide information
provide comparing pairs
return comparing semantic value
provide comparing pairs
return comparing syntactic value
Package 4
Function
predefined syntactic checking
Function
Levensthein distance algorithm
Package 3
Function
semantic relation detecting
Function
semantic values assigning
support
AMTM_ST
provide automatically generated model transformation mappings
Package 5
Function
user interactions interface
Function
potential mappings selection
Function
intermediate content storage
interaction
system users
store
retrieve
AMTM_O

Fig. 18 Package design illustration

In AMTM, models are categorized by different layers (model and meta-model). Most of the mapping rules need be built on the meta-model layer. In order to obtain the specific meta-models that conform to the MMM, two approaches can be applied.

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

Both these approaches can generate the specific meta-models. The specific part, which is marked with a star in Fig. 17, concerns the core functions in AMTM-SS. It indicates out that the syntactic and semantic checking methods are applied on the specific meta-models. The mechanism of incorporating the two checking methodologies into the model transformation process has been illustrated in previous sections.

Based on the requirements analysis, the package design composition is created and shown in Fig. 18.

As shown in Fig. 18, five main packages are defined in the design step. Within each package, the important classes are indicated. The name of each class constitutes its main task. Here, a simple illustration is provided of each package and its main classes.

- *Package 1* this package contains several functional classes to analyze the input model sets, and extract the information contained in these input models to the template that conforms to the MMM. The classes in this package also provide an interface to users to upload their inputs into AMTM-ST. In simple terms, this package has the following main tasks: (i) receive inputs from system users, (ii) analyze the inputs and extract the useful information, and (iii) store and adjust the extracted information in a special format (specific meta-models conforming to MMM). The outputs of this package are two specific meta-models.
- *Package 2* this package works as a bridge which crosses the gap between the model transformation domain and the syntactic & semantic checking measurements. The inputs of this package are the two specific meta-models generated by package 1. It provides several templates to store the compared potential 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 for calling syntactic and semantic checking measurements, and also define different mechanisms for using the two checking measurements based on different compared item pairs (i.e. element to element, property to property). The returned values of these functions are used to assign the comparing value. Normally, for each potential matching pair, a comparing value will exist, which can be used to select the final matching pairs.
- *Package 3* this package contains the semantic checking measurements. The classes (functions) used to gain access to the 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, a comparing value is assigned to this compared pair of names. The output (return value) of this package is this semantic comparison value.

- *Package 4* this package performs the similar role to package 3. It focuses on the syntactic checking measurements. The syntactic checking methodologies, the predefined treatment (part of Porter stemming algorithm) and the “Levenshtein distances” algorithm, are defined in this package. The inputs of this package are also a compared pair of two names; the return value (ranging from 0 to 1) is the syntactic comparison result: the degree of syntactic similarity. As illustrated in previous sections, syntactic checking results can have an influence on the semantic checking part. The calling mechanism of the semantic and syntactic checking measurements is determined by package 2.
- *Package 5* this package provides the function of generating the final model transformation mappings and rules. The inputs of this package come from package 2, and concern all the potential mapping pairs and their compared relational values. The main tasks of this package are: (i) to select the useful potential mapping pairs (remove overlap, Mutex), (ii) to provide connecting interfaces to system users, and allow them to validate the automatically-generated mappings and define new mapping rules, (iii) to store the final mapping rules into ontologies and apply these rules on the source model to generate the target model. In order to complete these tasks, this package needs gain access to AMTM_O (the 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, so a friendly and efficient user interface is needed here.

All these five packages make up AMTM-SS. In Fig. 18, only the main functional classes are presented. Two outside resources for AMTM-SS: the semantic thesaurus “AMTM_ST” and the ontology “AMTM_O” are also shown. AMTM-SS provides three main interfaces: (1) connecting with system users, (2) obtaining access to AMTM_ST, and (3) obtaining access to AMTM_O.

The coding part and the testing part of AMTM-SS are given with a use case shown in the following subsection.

5.2 Use case

AMTM-SS was developed with a personal computer. The basic technical parameters of this computer are: (i) operating system: Windows 7 Professional N, (ii) CPU: Inter Core i7-3770, 3.4 GHz, (iii) main memory (RAM): 8.0 GB, etc. The main programming language is “Java”; the version used is “1.8.0_20”. The relevant java developing environment: the version of runtime environment is “1.8.0_20-b26”, and the java virtual machine is a 64-Bit server, 25.20-b23 mixed mode. The integrated developing environment (IDE) is “Eclipse”; the chosen version is “eclipse-java-luna-SR1-win32-x86_64”. This was the new and stable version when AMTM-SS was launched. The other developing environment techniques are: Maven 2.0 and GitHub, etc.

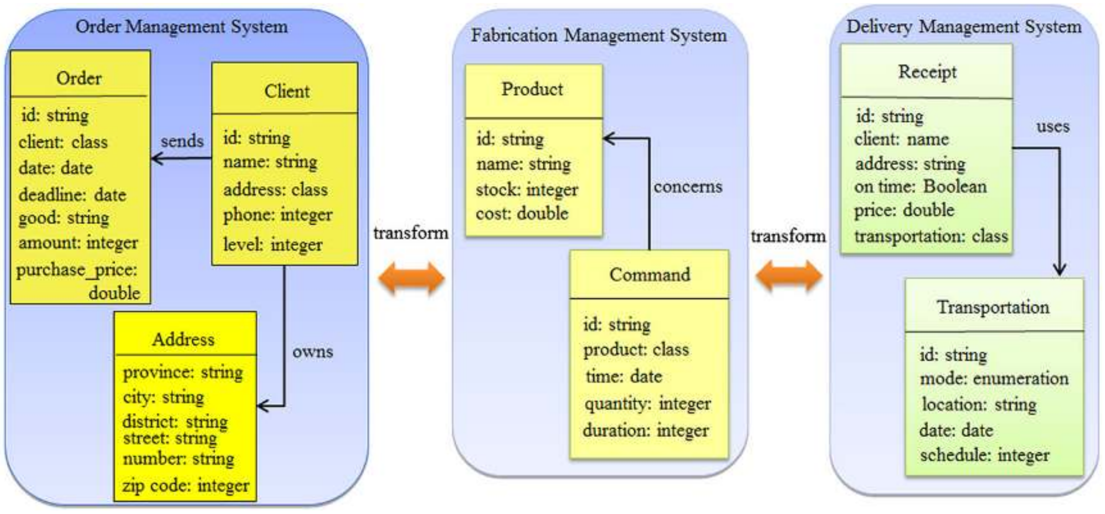


Fig. 19 AMTM testing use case

The semantic thesaurus "AMTM_ST" is stored in MongoDB (a new version of AMTM_ST is being developed with the graph database "Neo4j"). Several existing java jar packages are also involved. Two of the main jar packages are: (i) the jar packages released in order to support the algorithms presented in previous sections, and (ii) the jar package "JDOM".

This subsection presents a complete use case to test the functions and performance of AMTM-SS. This use case aims to cover all the matching theories defined in AMTM using all the main algorithms implemented in AMTM-SS. This use case simulates a simple "order-fabrication-delivery" process in the enterprise engineering domain. For the three phases: order, fabrication and delivery, three meta-models are created. AMTM tries to build model transformations among the three meta-models. Figure 19 shows the main idea of this use case.

This use case contains three meta-models: order management system, fabrication management system and delivery management system. The use case tries to make model-to-model mappings and transformations in two iterations: between the meta-models "order management system" & "fabrication management system", and between "fabrication management system" & "delivery management system". The "fabrication management system" could be regarded as an intermediate target model to test the "specific parts store and reuse" mechanism. This use case focuses on showing the mechanism of using AMTM-SS to automatically detect mappings among the elements (e.g. "Node" defined in MMM) coming from source meta-model and target meta- model.

5.2.1 First transformation iteration of this use case

In the first model transformation process iteration, the source meta-model contains three main "nodes": order (with seven properties), client (with five properties) and address (with six properties). All of the three nodes (concepts) will be compared with the two nodes (concepts): product (with four properties) and command (with five properties) in the target meta-model.

Table 9 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	SeV_weight	0.9	SyV_weight	0.1	4

There are several uncertain impact parameters within the equations that are defined in AMTM. The first step in executing this use case should therefore be to assign concrete values to these uncertain parameters. Table 9 shows the impact parameters and their value pairs defined for this use case. Actually, these are the default values assigned to these impact parameters; AMTM-SS also provides the mechanism that allows users to modify them.

With all these assigned values, the comparing mechanism defined in AMTM is executable in AMTM-SS. In the first model transformation iteration "from source meta-model 'order management system' to target meta-model 'fabrication management system'", in order to show the AMTM-SS working mechanism in detail, the detection process of comparing two nodes (concepts) "order" and "command" is shown step by step.

The first comparing step is to calculate the "S_SSV" between two elements' names (order and command).Eq. (4) is used to do this step.

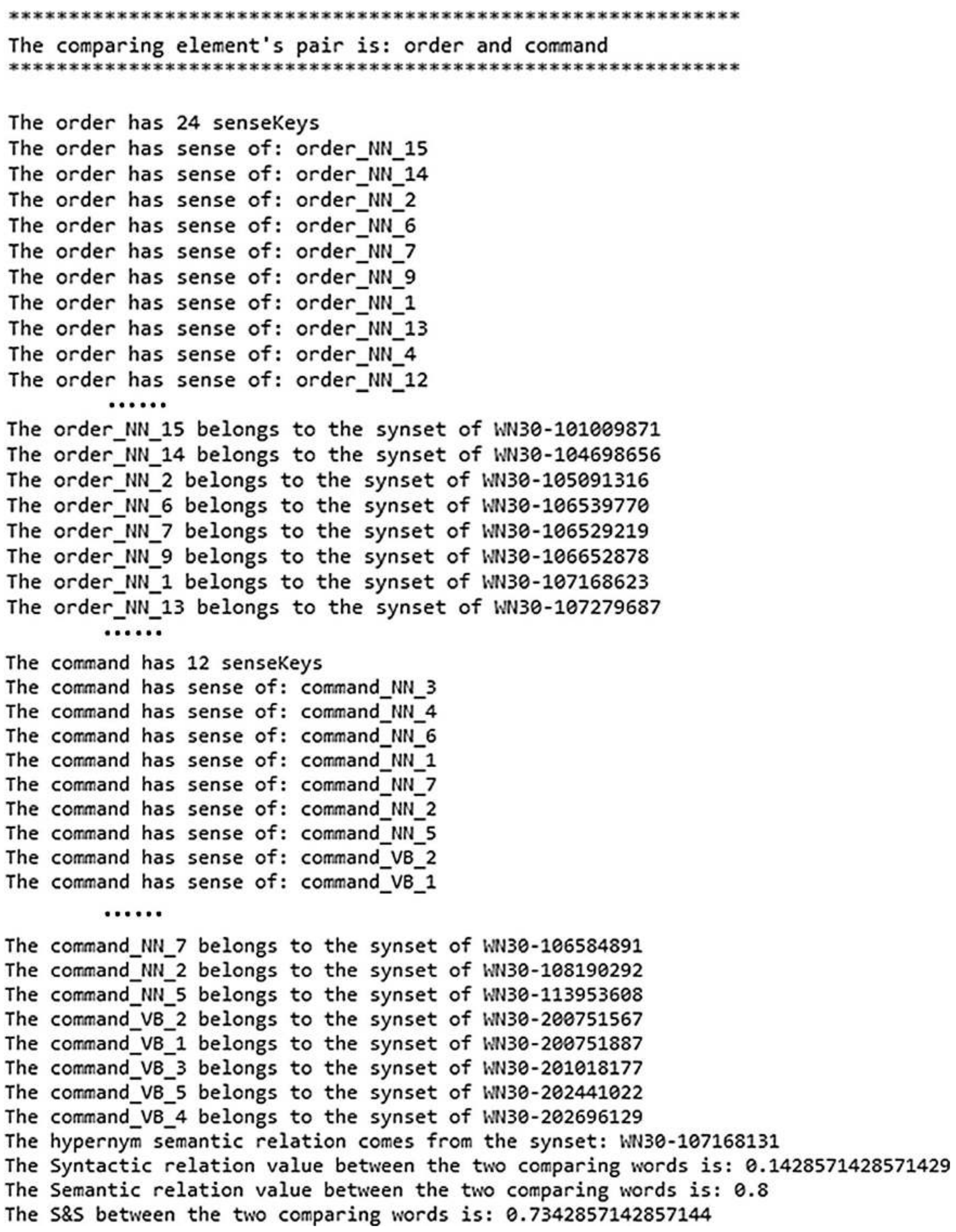
Figure 20 is the screenshot of executing Eq. (4) in AMTM-SS with "order" and "command" as two compared words.

According to Fig. 20, the word "order" has twenty-four semantic meanings (belongs to twenty-four synsets) and the word "command" has twelve semantic meanings (belongs to twelve synsets). The semantic relation between the two words is "hypernym", and the semantic value between them is "0.8"; the syntactic similarity value between them is: 0.1428. In this use case, the semantic relation is assumed to be more important than the syntactic relation, so two impact factors: "SeV_weight" and "SyV_weight" in Eq. (4) are assigned with values as 0.9 and 0.1, respectively. The final S&S value between the two words is: 0.7343.

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

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

When calculating S&S values on the property level, both the properties' names and their types are taken into account. In this use case, the property's name is



```
************************************************************
The comparing element's pair is: order and command
************************************************************

The order has 24 senseKeys
The order has sense of: order_NN_15
The order has sense of: order_NN_14
The order has sense of: order_NN_2
The order has sense of: order_NN_6
The order has sense of: order_NN_7
The order has sense of: order_NN_9
The order has sense of: order_NN_1
The order has sense of: order_NN_13
The order has sense of: order_NN_4
The order has sense of: order_NN_12
        ......
The order_NN_15 belongs to the synset of WN30-101009871
The order_NN_14 belongs to the synset of WN30-104698656
The order_NN_2 belongs to the synset of WN30-105091316
The order_NN_6 belongs to the synset of WN30-106539770
The order_NN_7 belongs to the synset of WN30-106529219
The order_NN_9 belongs to the synset of WN30-106652878
The order_NN_1 belongs to the synset of WN30-107168623
The order_NN_13 belongs to the synset of WN30-107279687
        ......
The command has 12 senseKeys
The command has sense of: command_NN_3
The command has sense of: command_NN_4
The command has sense of: command_NN_6
The command has sense of: command_NN_1
The command has sense of: command_NN_7
The command has sense of: command_NN_2
The command has sense of: command_NN_5
The command has sense of: command_VB_2
The command has sense of: command_VB_1
        ......
The command_NN_7 belongs to the synset of WN30-106584891
The command_NN_2 belongs to the synset of WN30-108190292
The command_NN_5 belongs to the synset of WN30-113953608
The command_VB_2 belongs to the synset of WN30-200751567
The command_VB_1 belongs to the synset of WN30-200751887
The command_VB_3 belongs to the synset of WN30-201018177
The command_VB_5 belongs to the synset of WN30-202441022
The command_VB_4 belongs to the synset of WN30-202696129
The hypernym semantic relation comes from the synset: WN30-107168131
The Syntactic relation value between the two comparing words is: 0.1428571428571429
The Semantic relation value between the two comparing words is: 0.8
The S&S between the two comparing words is: 0.7342857142857144
```

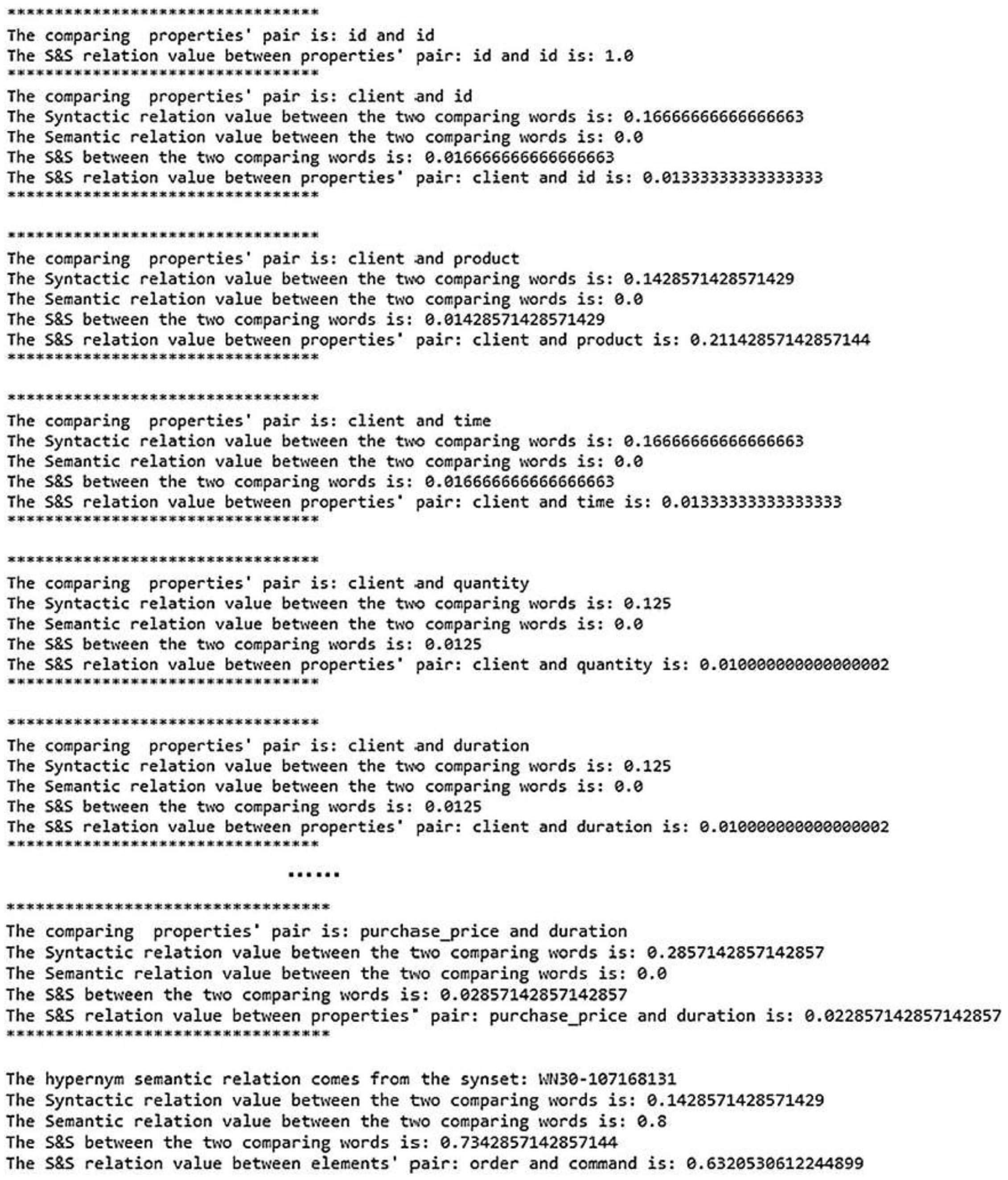
Fig. 20 S&S detection between elements' names "order" and "command"

regarded as more important than its type for the process of making transformation mappings. So in Eq. (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 Fig. 21.

As shown in Fig. 21, to compare a pair of elements, an S&S relation value "P_SSV" is generated for each potential matching pair of properties. In this test case, there are in total 7 (source properties) * 5 (target properties): 35 potential property matching pairs.

To improve 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 compared with other properties from the target element. To be more readable, Table 10 was created to store all these "P_SSV" values.

With all these "P_SSV" values and the "S_SSV" value "0.7343" calculated in the first step (between two elements' names: order and command), the "Ele_SSV" between elements "order" and "command" could then be calculated by using Eq. (1). In this use case, two impact factors: "name_weight" and "property_weight"



```
*********************************
The comparing  properties' pair is: id and id
The S&S relation value between properties' pair: id and id is: 1.0
*********************************
The comparing  properties' pair is: client and id
The Syntactic relation value between the two comparing words is: 0.16666666666666663
The Semantic relation value between the two comparing words is: 0.0
The S&S between the two comparing words is: 0.016666666666666663
The S&S relation value between properties' pair: client and id is: 0.013333333333333333
*********************************

*********************************
The comparing  properties' pair is: client and product
The Syntactic relation value between the two comparing words is: 0.1428571428571429
The Semantic relation value between the two comparing words is: 0.0
The S&S between the two comparing words is: 0.01428571428571429
The S&S relation value between properties' pair: client and product is: 0.21142857142857144
*********************************

*********************************
The comparing  properties' pair is: client and time
The Syntactic relation value between the two comparing words is: 0.16666666666666663
The Semantic relation value between the two comparing words is: 0.0
The S&S between the two comparing words is: 0.016666666666666663
The S&S relation value between properties' pair: client and time is: 0.013333333333333333
*********************************

*********************************
The comparing  properties' pair is: client and quantity
The Syntactic relation value between the two comparing words is: 0.125
The Semantic relation value between the two comparing words is: 0.0
The S&S between the two comparing words is: 0.0125
The S&S relation value between properties' pair: client and quantity is: 0.010000000000000002
*********************************

*********************************
The comparing  properties' pair is: client and duration
The Syntactic relation value between the two comparing words is: 0.125
The Semantic relation value between the two comparing words is: 0.0
The S&S between the two comparing words is: 0.0125
The S&S relation value between properties' pair: client and duration is: 0.010000000000000002
*********************************
                              ......
*********************************
The comparing  properties' pair is: purchase_price and duration
The Syntactic relation value between the two comparing words is: 0.2857142857142857
The Semantic relation value between the two comparing words is: 0.0
The S&S between the two comparing words is: 0.02857142857142857
The S&S relation value between properties' pair: purchase_price and duration is: 0.022857142857142857
*********************************

The hypernym semantic relation comes from the synset: WN30-107168131
The Syntactic relation value between the two comparing words is: 0.1428571428571429
The Semantic relation value between the two comparing words is: 0.8
The S&S between the two comparing words is: 0.7342857142857144
The S&S relation value between elements' pair: order and command is: 0.6320530612244899
```

Fig. 21 Test results of matching on element level between elements "order" and "command"

Table 10 Comparisons on property level for this use case

Order	Command				
	id	product	time	quantity	duration
id	1	–	–	–	–
client	0.013	0.2114	0.013	0.010	0.010
date	0	0.0114	0.6808	0.4808	0.030
deadline	0.010	0.010	0.220	0.4808	0.020
good	0.220	0.4548	0	0	0.01
amount	0	0.0343	0.1555	0.8580	0.3655
purchase_price	0.0058	0.0171	0.0114	0.0171	0.0229

Table 11 Detection results of potential matching pairs on element level

Order MS	Fabrication MS	
	Product	Command
order	0.1748	0.632
client	0.2499	0.1953
address	0.1574	0.1363

were assigned with values 0.5 and 0.5, respectively. The “element’s name” and “property group” are regarded as having the same importance in making mappings between elements. In this use case, the final “Ele_SSV” between “order” and “command” is: 0.632. The test results shown in Fig. 21 are stored and presented in Table 10.

In this model transformation iteration, six of this kind table were created (on the element level, there are six potential matching pairs, e.g. order-product, client-command and address-product) to store the intermediate transformation results. Following the same calculating rules and steps, all the “Ele_SSV” values of each potential matching pair of elements (coming from source model and target model, respectively) could then be generated. Table 11 shows the final result.

Based on the mechanism of choosing potential matching pairs, the potential matching element pairs could then be made. Furthermore, the property matching pairs (that exist within the matching element pairs) could also be defined with the help of contents stored in Table 10 (a set of this kind of table). Figure 22 shows the comparing and matching results.

In the first matching step of the first model transformation iteration, the elements “order” and “command” are matched. The matching results of their property group are: id–id, date-time, deadline-time, good-product and amount-quantity.

Each model transformation iteration phase defined in AMTM contains three matching steps: matching on element level, hybrid matching, and auxiliary matching. For the second matching step “hybrid matching”, the matching ideas and techniques used are the same as the ones used in the first matching step. “Hybrid matching” focuses on the unmatched properties left from the first matching step.

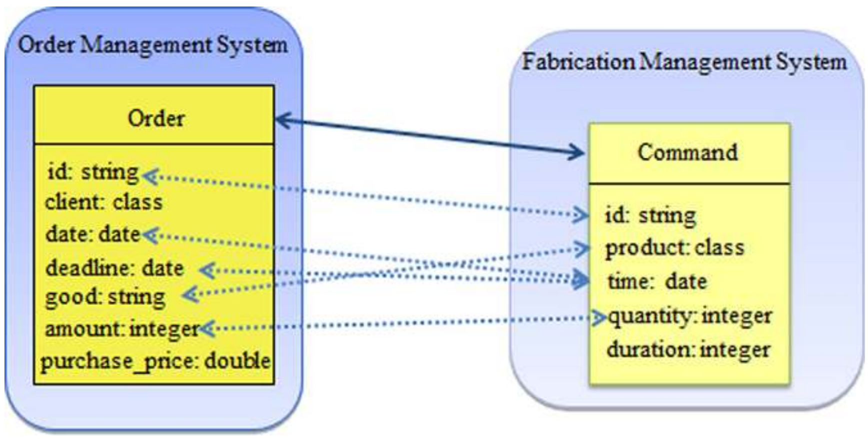


Fig. 22 Matching results on element level in the first transformation iteration of this use case

Table 12 Hybrid matching results in this use case

Element: property	Product: id	Product: name	…	Command: duration
Order: client	0.021	0.021	…	0.3721
Order: purchase_price	0.0171	0.02	…	0.3785
Client: id	0.5071	0.1071	…	0.0193
Client: name	0.1071	0.5071	…	0.0193
…	…	…	…	…
Address: zip_code	0.11	0.105		0

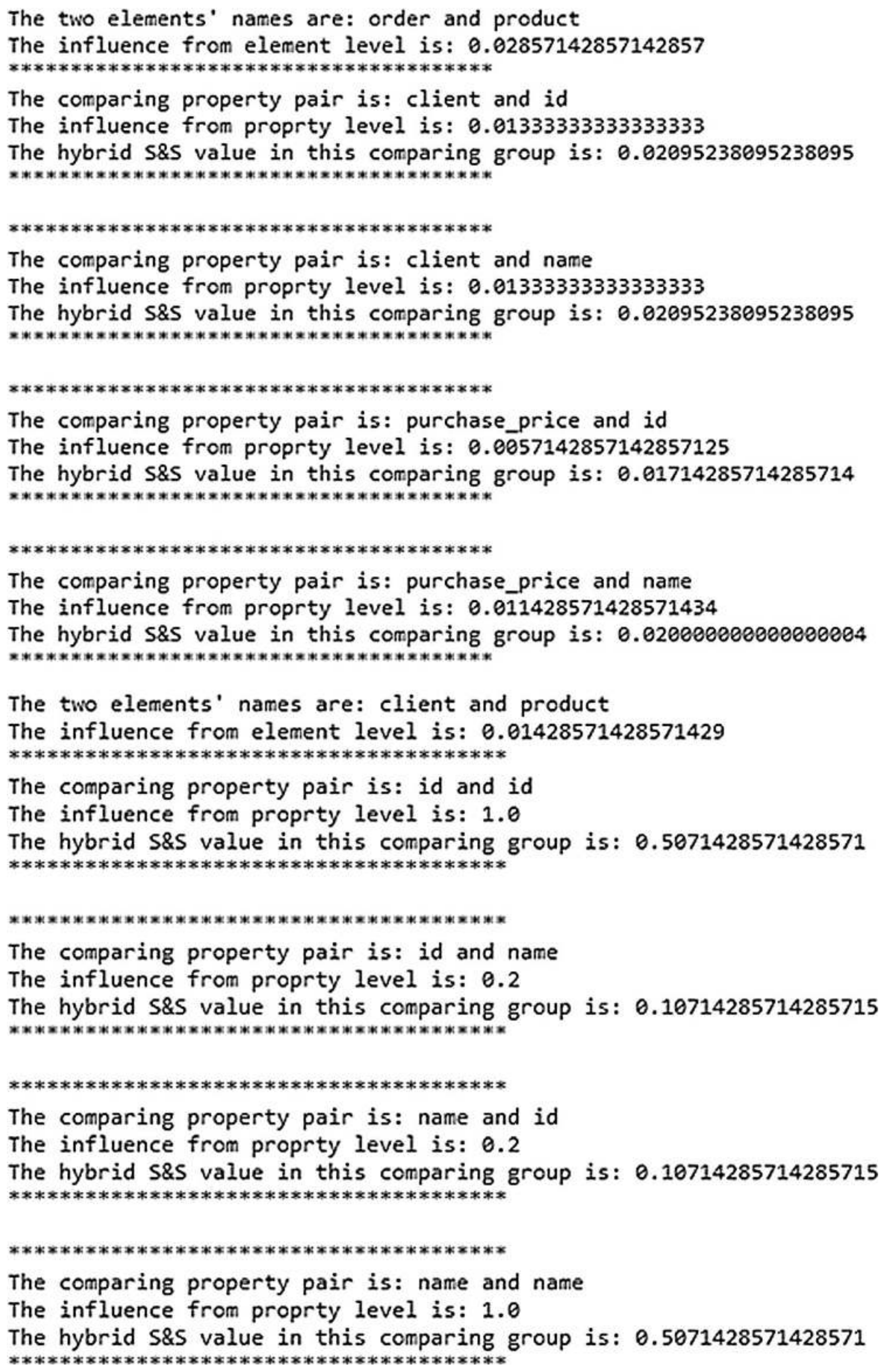
Table 12 was thus created and filled automatically by AMTM-SS to detect potential mappings in this step.

The two impact factors: "en_weight" and "pl_weight" are assigned with values 0.5 and 0.5, respectively. This means when making hybrid matchings, the influence of the element level and the property level are regarded as equally important.

All the unmatched properties from the source meta-model are compared with all the properties of the target meta-model. Parts of the final result of this step have been shown in Table 12.

Since the comparing process of this step is complex, we have just taken the comparing pairs (element: property): "Order: client–Product: id", "Order: client–Product: name", "Order: purchase_price–Product: id", "Order: purchase_price–Product: name", etc. as an example. The execution result is shown in Fig. 23.

As shown in Fig. 23, in this use case, the influence of the element level "between two elements' names, "order" and "product" is 0.02857; the influence of the property level (between property "client" with type "class" and property "id" with type "string") is: 0.0133. The final hybrid S&S relation value between this pair of properties is: 0.021. According to the potential matching pair selection mechanism, this pair of items would not be regarded as a model transformation mapping. Within the first model transformation iteration of this use case, there are a total of "13



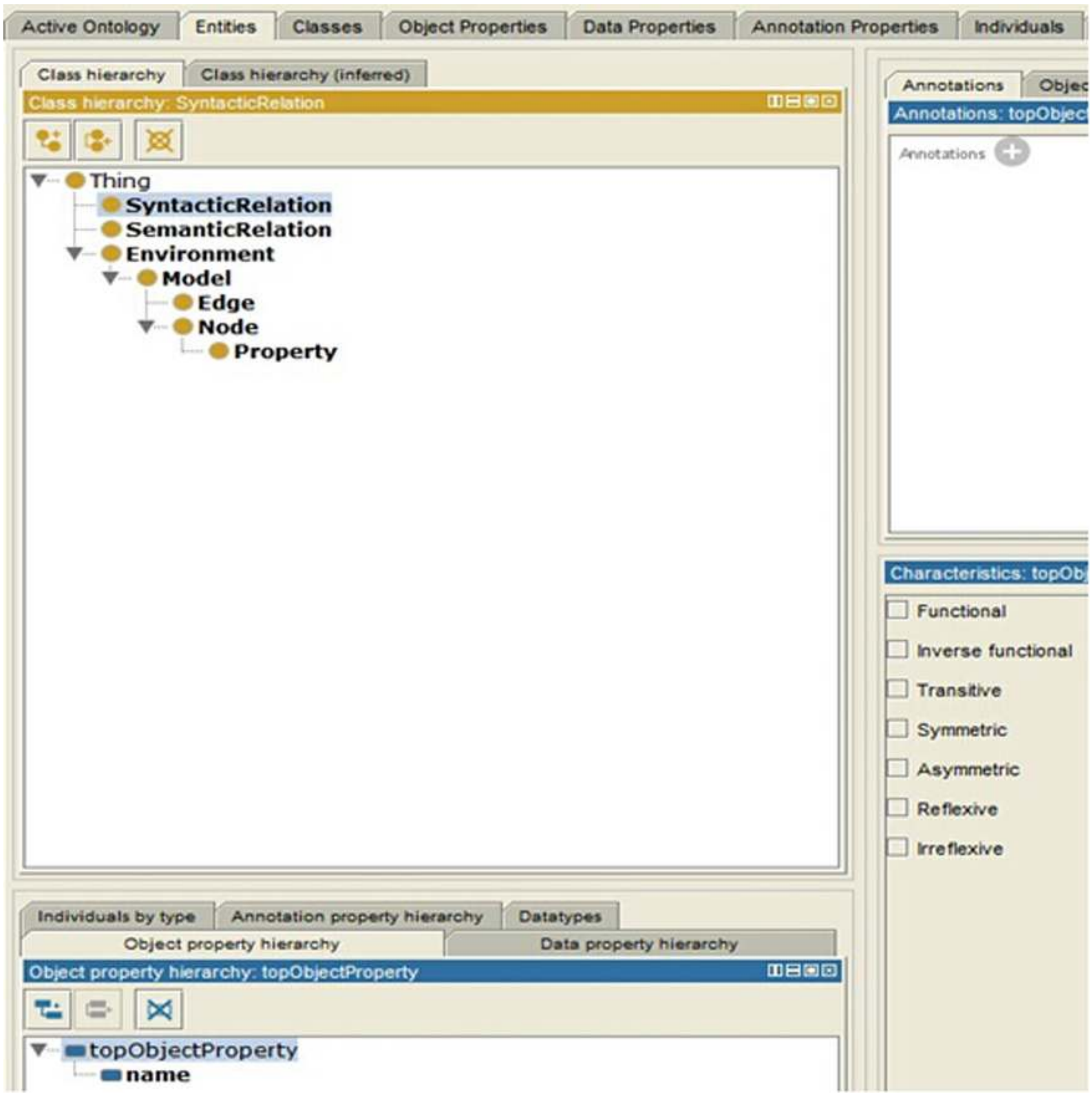
```
The two elements' names are: order and product
The influence from element level is: 0.02857142857142857
**************************************
The comparing property pair is: client and id
The influence from proprty level is: 0.013333333333333333
The hybrid S&S value in this comparing group is: 0.02095238095238095
**************************************

**************************************
The comparing property pair is: client and name
The influence from proprty level is: 0.013333333333333333
The hybrid S&S value in this comparing group is: 0.02095238095238095
**************************************

**************************************
The comparing property pair is: purchase_price and id
The influence from proprty level is: 0.0057142857142857125
The hybrid S&S value in this comparing group is: 0.01714285714285714
**************************************

**************************************
The comparing property pair is: purchase_price and name
The influence from proprty level is: 0.011428571428571434
The hybrid S&S value in this comparing group is: 0.020000000000000004
**************************************

The two elements' names are: client and product
The influence from element level is: 0.014285714285714229
****************************************
The comparing property pair is: id and id
The influence from proprty level is: 1.0
The hybrid S&S value in this comparing group is: 0.5071428571428571
****************************************

****************************************
The comparing property pair is: id and name
The influence from proprty level is: 0.2
The hybrid S&S value in this comparing group is: 0.10714285714285715
****************************************

****************************************
The comparing property pair is: name and id
The influence from proprty level is: 0.2
The hybrid S&S value in this comparing group is: 0.10714285714285715
****************************************

****************************************
The comparing property pair is: name and name
The influence from proprty level is: 1.0
The hybrid S&S value in this comparing group is: 0.5071428571428571
****************************************
```

Fig. 23 An illustration of hybrid matching testing results

(unmatched source properties) * 9 (all target properties): 117" compared pairs that need to be tested in the hybrid matching step. The mechanism of comparing them is the same. Based on the comparison results and the potential matching pairs selection mechanism, the potential matching pairs between properties "client: id–product: id", "client: name–product: name", etc. should be built as potential matching pairs. The complete matching result is shown in the next subsection: the use case evaluation.

The third matching step included in AMTM is "auxiliary matching". This step stores specific parts (unmatched items) of the source model in AMTM_O and tries to enrich the specific parts (unmatched items) in the target model by extracting content from AMTM_O. In this transformation phase of the use case, only the specific parts from the source meta-model are stored in AMTM_O. Since this is the first model transformation iteration phase, there is no content stored in AMTM_O that could be used to enrich the specific parts of the target meta-model. AMTM_O is implemented with the software tool "protégé"; the structure of AMTM_O is shown in Fig. 24.

The working mechanism of the auxiliary matching step is to "repeat the 'matching on element level' and 'hybrid' matching steps". But the two steps work



Thing
- SyntacticRelation
- SemanticRelation
- Environment
  - Model
    - Edge
    - Node
      - Property

topObjectProperty
- name

Fig. 24 The structure of AMTM_O

on specific source meta-models (specific parts of the former source meta-models) and target meta-models (specific parts of target meta-models).

As a brief conclusion, within the first model transformation iteration phase, all the mappings are built in the first matching step, “matching on element level”. There are no mappings being built in the hybrid matching step. The specific parts of the source meta-model were stored in AMTM_O in the auxiliary matching step.

5.2.2 Second transformation iteration of this use case

In the second model transformation iteration, the source meta-model is the target meta-model of the first model transformation iteration, “Fabrication Management System”, and the target meta-model is “Delivery Management System”, which has two nodes (concepts): receipt (with six properties) and transportation (with five properties).

Compared 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 the target meta-model. The illustration of this model transformation iteration is shown in Fig. 25.

The process of detecting potential model transformation matching pairs in this iteration phase is the same as that of the first iteration. It follows the three matching steps: matching on element level, hybrid matching, and auxiliary matching. In this iteration, the test results of the first matching step are shown in Table 13.

The test results show that no matching could be built in the first matching step (no element-to-element mappings).

The second matching step, “hybrid matching”, compares all the properties of the source meta-model with all the properties of the target meta-model. Equation (3) is used here. Between each potential matching pair, a “HM_SSV” is calculated and assigned.

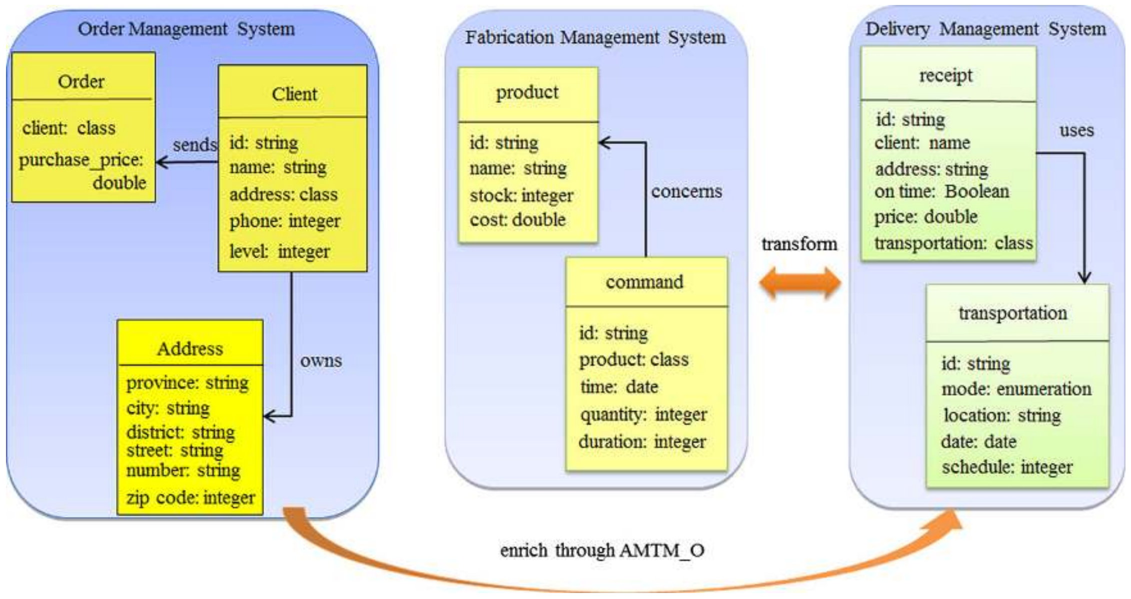


Fig. 25 An illustration of the second model transformation matching iteration

Table 13 Potential matching pairs on element level of second iteration in this use case

Fabrication MS	Delivery MS	
	Receipt	Transportation
Product	0.2387	0.1387
Command	0.1387	0.2427

The third matching step, "auxiliary matching", tries to enrich the specific parts (unmatched items) in the target meta-model. In this test case, the property "location" of element "transportation" could be enriched with the specific part (property "address" of element "client" in meta-model order management system) stored in AMTM_O. Also, other specific properties of the meta-model "delivery management system" could be enriched by the properties stored in AMTM_O.

To conclude, 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.

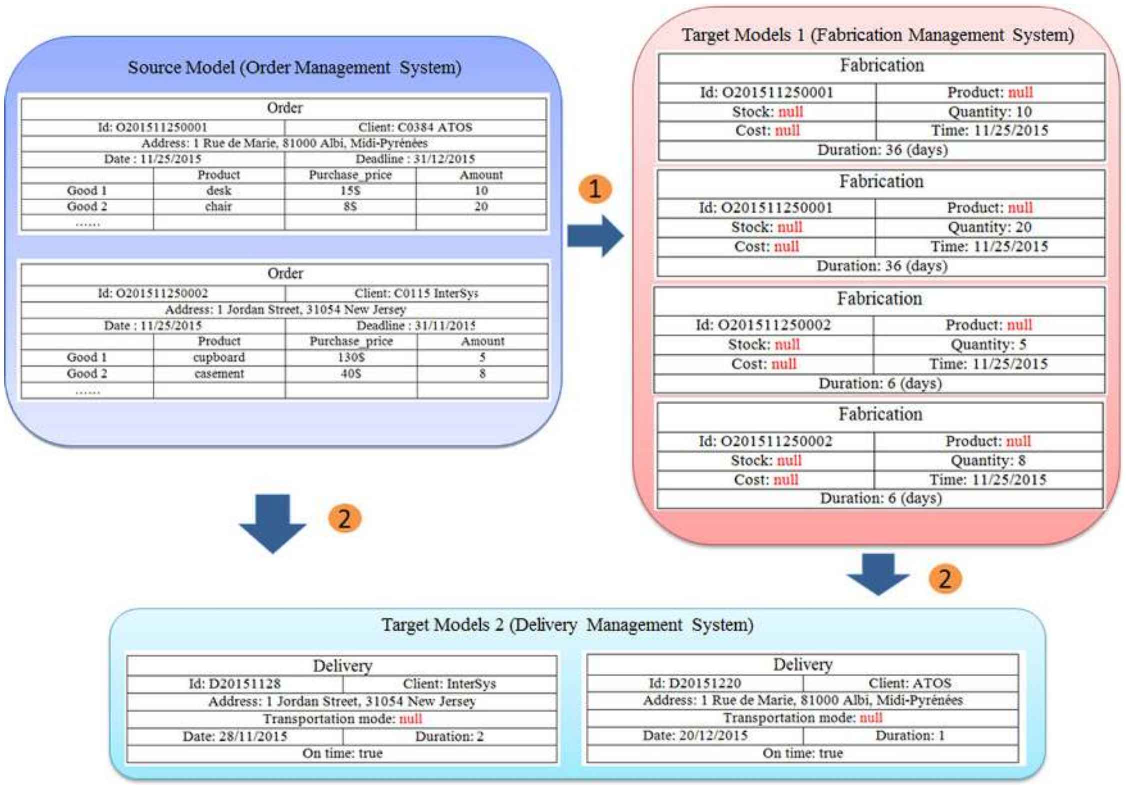
5.2.3 Applying the automatically-generated model transformation rules on models

The use case simulates the detection process of model transformation mappings. All the model transformation mappings and rules are built on the meta-model level (between source meta-models and target meta-models). Finally, these automatically-generated mappings and rules can be used on specific source models to generate target models. Here, we provide a source model which conforms to the order management system meta-model, and then apply the automatically-generated model transformation mappings and rules on this source model to get the target models (conforming to the fabrication and delivery meta-models, respectively). Figure 26 shows the source model and two automatically-generated target models.

As shown in Fig. 26, the "Source Model" has two instances of orders from clients. The two orders conform to the order management system meta-model presented above. By applying the potential transformation mappings, automatically-generated by the first transformation iteration phase, to "target models 1", four fabrication notes can be built. Some "form items" in "target models 1" cannot be filled directly by using the values in the source models, so some predefined rules need to be applied to help the transformation process. Also, some "form items"" in "target models 1" could not be filled in (no mappings are built on the meta-model level).

To generate the final target models, "target models 2", both "target models 1" and "source models" have to be taken into consideration. Here, the instances of "target models 2" shown in Fig. 26 are only ideal ones.

In conclusion, these test results show that some elements and properties in the target model could not be transformed from the source model (or enriched by AMTM_O). The remaining unmatched parts need to be dealt with manually. AMTM could therefore in some respects remove manual effort when defining the same (or similar) concepts between source and target meta-models.



Source Model (Order Management System)
1
2
Target Models 1 (Fabrication Management System)
2
Target Models 2 (Delivery Management System)

Order			
Id: O201511250001		Client: C0384 ATOS	
Address: 1 Rue de Marie, 81000 Albi, Midi-Pyrénées			
Date : 11/25/2015		Deadline : 31/12/2015	
	Product	Purchase_price	Amount
Good 1	desk	15$	10
Good 2	chair	8$	20
......			

Order			
Id: O201511250002		Client: C0115 InterSys	
Address: 1 Jordan Street, 31054 New Jersey			
Date : 11/25/2015		Deadline : 31/11/2015	
	Product	Purchase_price	Amount
Good 1	cupboard	130$	5
Good 2	casement	40$	8
......			

Fabrication	
Id: O201511250001	Product: null
Stock: null	Quantity: 10
Cost: null	Time: 11/25/2015
Duration: 36 (days)	

Fabrication	
Id: O201511250001	Product: null
Stock: null	Quantity: 20
Cost: null	Time: 11/25/2015
Duration: 36 (days)	

Fabrication	
Id: O201511250002	Product: null
Stock: null	Quantity: 5
Cost: null	Time: 11/25/2015
Duration: 6 (days)	

Fabrication	
Id: O201511250002	Product: null
Stock: null	Quantity: 8
Cost: null	Time: 11/25/2015
Duration: 6 (days)	

Delivery	
Id: D20151128	Client: InterSys
Address: 1 Jordan Street, 31054 New Jersey	
Transportation mode: null	
Date: 28/11/2015	Duration: 2
On time: true	

Delivery	
Id: D20151220	Client: ATOS
Address: 1 Rue de Marie, 81000 Albi, Midi-Pyrénées	
Transportation mode: null	
Date: 20/12/2015	Duration: 1
On time: true	

Fig. 26 Target models generated by using the automatically detected mappings

5.3 Methodology evaluation

AMTM is still in the stage of work in progress. It is necessary to study its performances to verify the viability of this methodology and to guide future improvements. This subsection presents the evaluation result of AMTM based on the use case presented above. The evaluation method is inspired by Wieringa and Daneva (2015).

5.3.1 Evaluation of performance

The use case contains 7 nodes (concepts) and 38 properties. The quantitative evaluation mostly focuses on the property–property comparison due to the fact that the different matching approaches are based on this comparison. The performance evaluation was performed on a personal computer with 2.9 GHZ, i7 CPU and 8 Go RAM (with Mac OS X and Java 8).

All performance measurements are based on three indicators: precision, recall and time-consuming. Precision (noted as P), evaluates the quality of results (percentage of relevant mappings among retrieved ones) whereas Recall (noted as R), evaluates the sufficiency of the results (percentage of retrieved mappings among relevant ones). Time-consuming evaluates the required time to compare two properties.

$$P = \frac{|retrived \cap relevant|}{|retrived|} \quad R = \frac{|retrieved \cap relevant|}{|relevant|} \tag{6}$$

In the use case, the relevant mappings (property pairs) that are supposed to be made are presented in Table 14.

The retrieved mappings (automatically detected results by AMTM-SS) are presented in Fig. 27. The figure presents the retrieved mappings on the basis of two-dimensional (semantic and syntactic) coordinates: the mappings selection mechanism that is illustrated in Sect. 4.4.3.

Based on the mappings selection mechanism, the potential matching pairs (property–property) can be selected.

The data provided by Table 14 and Fig. 27 work together to be used by Eq. (6) to evaluate the "precision" and "recall" of AMTM. Figure 28 shows the "precision" and "recall" issues, respectively.

Figure 28 "Precision" shows three evaluation results: only using syntactic checking only, using semantic checking only, and using both checking-methods (hybrid). It shows the retrieved potential mapping pairs in Area 2 (within the mappings selection mechanism framework with 'two-dimension coordinate'). It can be seen that, using semantic checking only can generate a high percentage of potential mapping pairs.

Figure 28, "Recall", shows that using syntactic and semantic checking measurements together could build high quality potential mappings (high percentage of relevant mapping pairs). Also, semantic checking has better performance in detecting potential mapping pairs (pairs appearing in Area 2) than syntactic checking.

Table 14 Relevant matching property pairs

No.	Element: property	Element: property
1	Order: id	Command: id
2	Order: deadline	Command: time
3	Order: good	Command: product
4	Order: purchase_price	Product: cost
5	Order: id	Receipt: id
6	Order: id	Transportation: id
7	Order: deadline	Transportation: date
8	Order: amount	Command: quantity
9	Order: deadline	Transportation: schedule
10	Order: purchase_price	Receipt: price
11	Client: name	Receipt: price
12	Address: city	Receipt: address
13	Address: street	Receipt: address
14	Address: number	Receipt: address
15	Command: duration	Transportation: date
16	Command: duration	Transportation: schedule

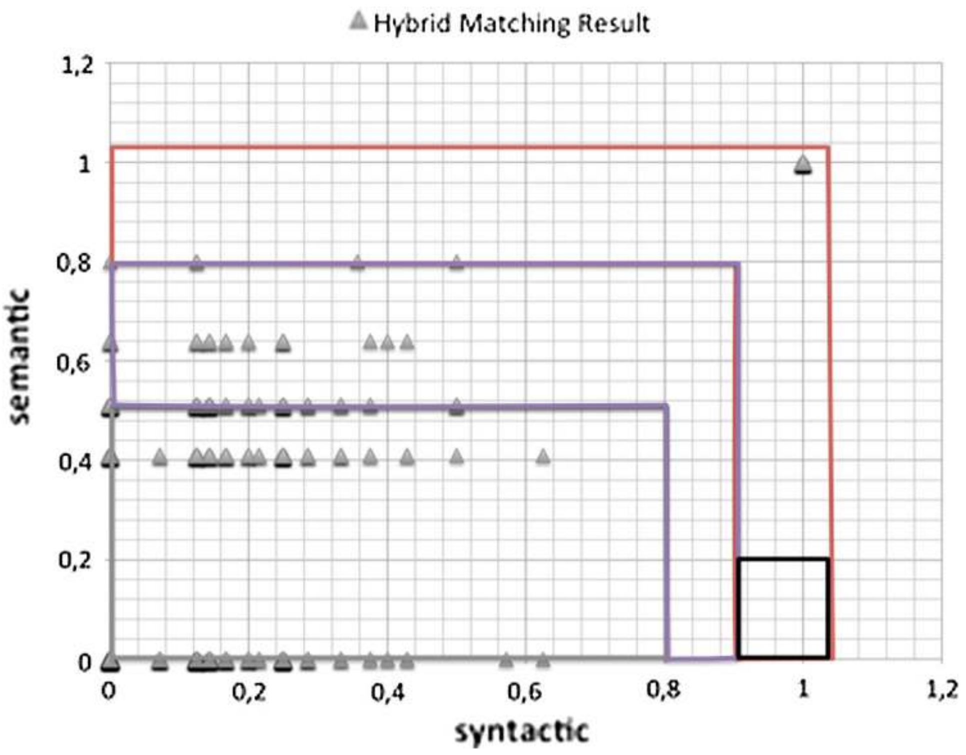
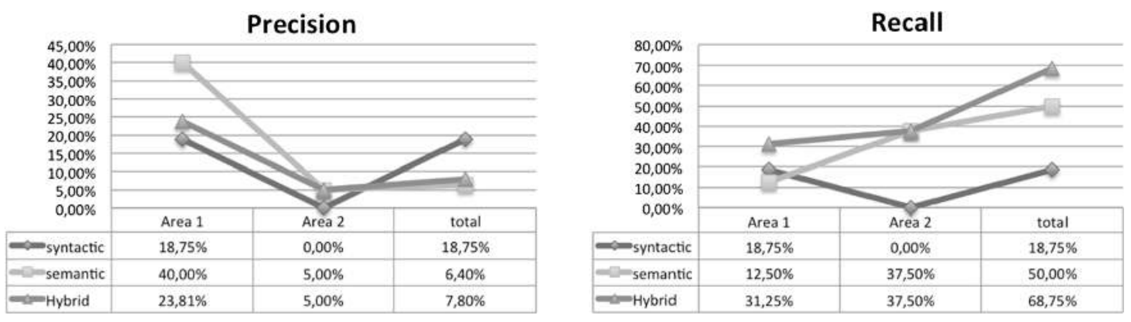


Fig. 27 Automatically-generated comparison of results for “property–property”



Precision

	Area 1	Area 2	total
syntactic	18,75%	0,00%	18,75%
semantic	40,00%	5,00%	6,40%
Hybrid	23,81%	5,00%	7,80%

Recall

	Area 1	Area 2	total
syntactic	18,75%	0,00%	18,75%
semantic	12,50%	37,50%	50,00%
Hybrid	31,25%	37,50%	68,75%

Fig. 28 The evaluation of “precision” and “recall” issues

As a short conclusion, the evaluation results show that both hybrid and semantic checking measurements provide interesting results (represented by the Precision and Recall graphs). It seems that it is possible to define more relevant mapping as well as more non-relevant mapping with the hybrid approach than when only using the semantic checking measurements. Since AMTM aims to help users to define model transformation automatically, we suppose that it is better to have as many relevant mappings as possible. Nevertheless, this assumption is acceptable when the time-consuming issues are similar for the two approaches. Figure 29 shows the time-consuming aspect in this use case.

Figure 29 illustrates that both hybrid and semantic approaches require the same time to deduce the mappings. The syntactic approach is instantaneous, and the points representing semantic and hybrid approaches are overlaid. It is possible to identify two groups of points: (i) points around the value of 1 (s) and (ii) points

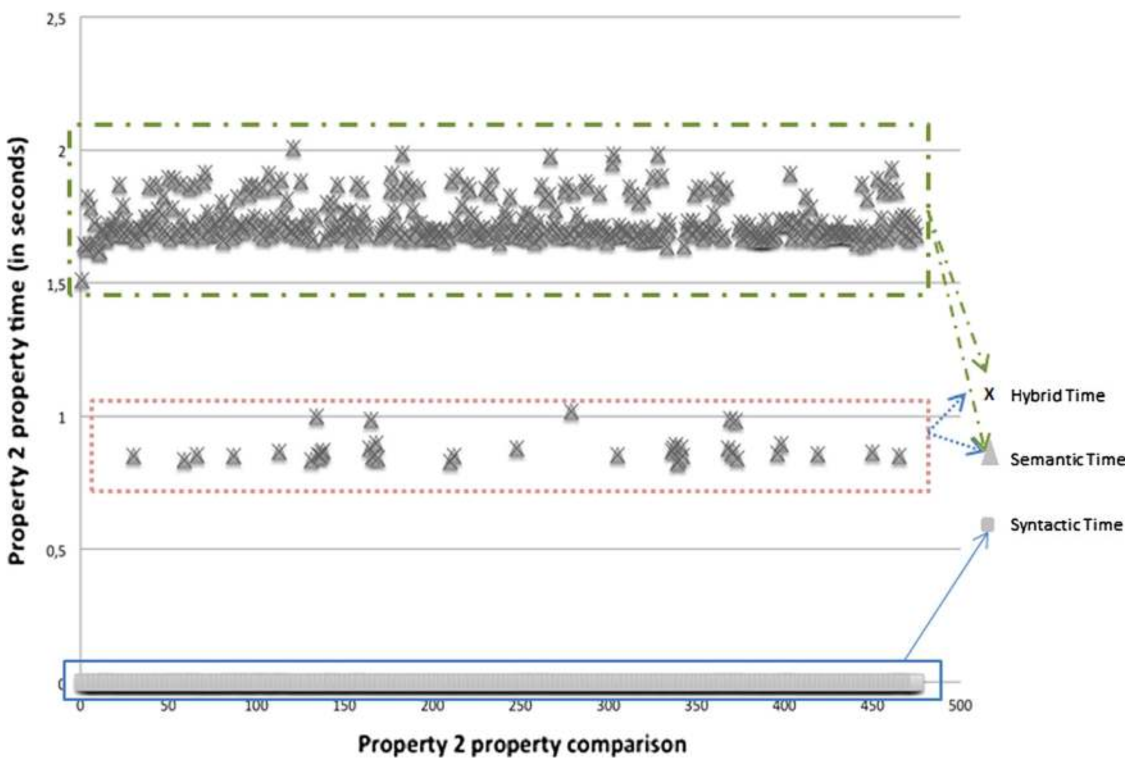


Fig. 29 Evaluation of the "time-consuming" issue

between 1.5 and 2 (s). These two groups of points are separated because of the semantic relations that exist between properties' names with the first group of points corresponding to direct semantic relations and the second group of points corresponding to iterative semantic relations (the two kinds of semantic relations are described in Sect. 4.4.2.2).

The evaluation shows that semantic and syntactic checking measurements should be used together (hybrid). Finally, in order to improve the "precision" and "recall" of AMTM, the threshold values defined in the mappings selection mechanism should be modified.

6 Conclusion

This paper presents an automatic model-to-model mapping and transformation methodology (AMTM), which could serve model-based EIS integration and interoperability by solving the data sharing and exchange problem involved.

Compared with the existing model transformation methodologies, the main feature of AMTM is that it serves cross-domain and defines the mappings and transformations automatically. To achieve this, AMTM combines semantic and syntactic checking measurements into a refined meta-model based model transformation process. A meta–meta-model, which defines the mechanism of applying S&S measurements on the model transformation process, is created. In order to solve the granularity and mismatch issues involved in the model transformation process, AMTM regards model transformation as an iterative process and three matching steps are created within each iteration phase.

Different algorithms have been developed to carry out the S&S measurements. For syntactic checking, part of the "Porter stemming" algorithm and the "Levenshtein distance" algorithm are used to calculate the syntactic similarity between two words. For semantic checking, a huge semantic thesaurus, AMTM_ST, has been created on the basis of "WordNet". In order to combine these algorithms and semantic relations into model transformation processes, five equations have been defined. A use case has been presented to show the working mechanism of AMTM, and based on the test results, a partial evaluation of AMTM has been also illustrated.

In AMTM, there are several points still need to be improved:

- The validation phase of the automatically-generated model transformation mappings should be added to the whole transformation process. The possible validating methods could be "redo the mappings and transformations (from target meta-model to source meta-model)" or could involve users to carry out manual validation.
- Some uncertain values (e.g. impact factors in equations and thread values in the matching pair selection mechanism) need to be reconsidered. A better way to assign them might be to use mathematical strategy [e.g. the "Choquet" integral; one of the uses of the "Choquet" integral is explained in Abril et al. (2012)].
- The "S_SeV" values defined in Table 7 need to be modified to come within a reasonable scope. Furthermore, more semantic relations and their corresponding semantic values need to be defined and maintained within AMTM_ST.
- The efficiency of semantic checking measurements needs to be improved. As semantic checking measurements rely on a huge semantic thesaurus, it is a time-consuming process to detect semantic relations between two words.
- The semantic thesaurus built for AMTM, AMTM_ST, was created on the basis of WordNet. At the moment, AMTM_ST is stored in MongoDB. Considering the need for efficiency and visibility when undertaking semantic detecting, other databases (e.g. Neo4j) might replace "MongoDB" to store AMTM_ST.
- Improving the focus on serving specific domains. Since AMTM_ST was created on the basis on "WordNet", only general words (with their general meanings) are stored. In order to use AMTM to serve specific domains, some specific content should be included to enrich AMTM_ST and AMTM_O. For example, to serve EIS integration and interoperability, the knowledge in the MIT process Handbook (Malone et al. 2003) would be extremely helpful.

There is a broader vision of how AMTM might be used: it could be employed as a methodology to share and exchange information and knowledge between different domains (or within the same domain). Figure 30 shows a general idea of this usage.

Today, many data collectors (e.g. sensors, smart equipment and computers) are used to gather rough data from a particular region or domain. The collected data serve various purposes and reflect different views of a system. Moreover, different collectors store data in their own structures, which might be heterogeneous with

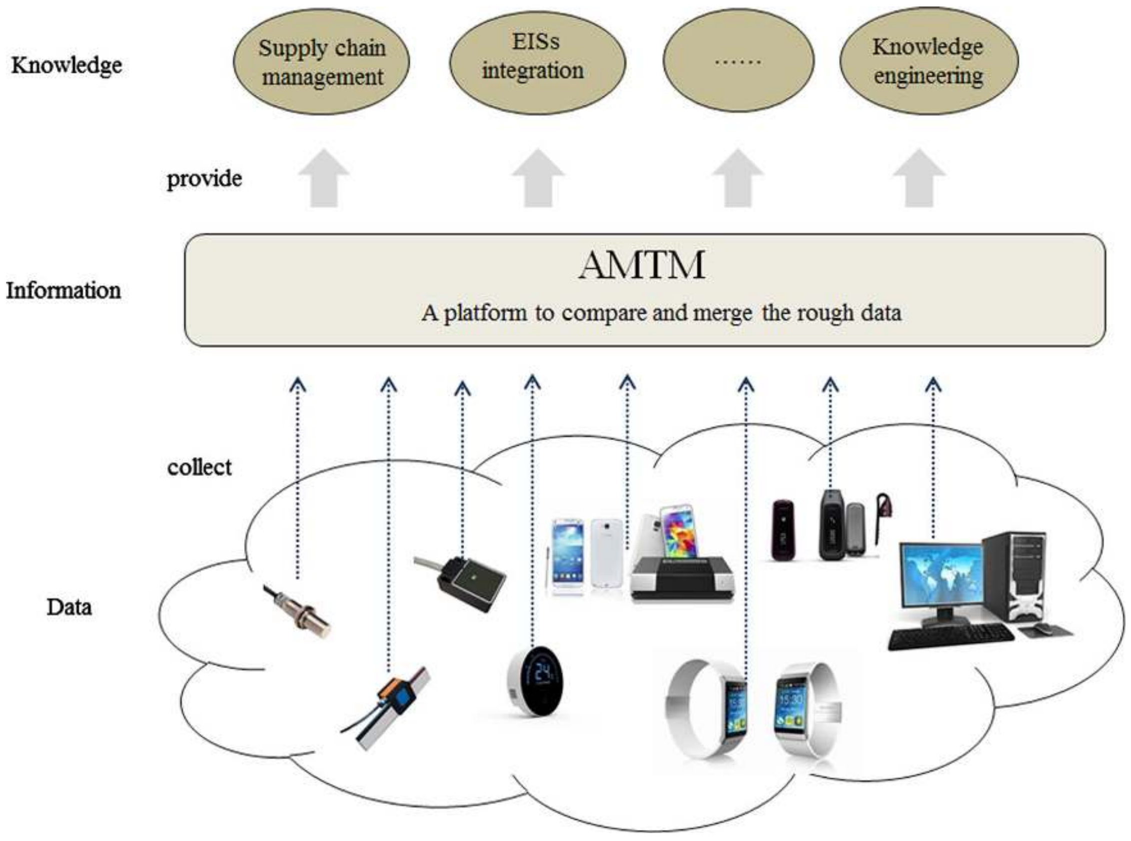


Fig. 30 A broader vision of the use of AMTM

regard to each other. So, it is difficult to make use of this kind of data as a whole. In the context of AMTM, all these collected data are regarded as single models. Thus, AMTM could use semantic and syntactic checking measurements to detect the intrinsic links between these models. After comparing and transforming these data, a final target model (an overview of a specific system) could be generated. Different domains could use this final target model, and with domain specific rules, the information contained in the model could be used as knowledge.

Converting rough data into information, then sharing and exchanging information and knowledge within the same domain (or between different domains) might be another use of AMTM. As stated in Panetto and Molina (2008): "recent advances in information and communication technologies have allowed manufacturing enterprise to move from highly data-driven environments to a more cooperative information/knowledge-driven environment." Focusing only on one data level is not enough to solve real engineering domain problems. Thus, AMTM is an attempt to build connections between data, information and knowledge. The use of AMTM in the web service composition domain and the information and knowledge engineering domain have been described in Wang et al. (2015a, b), respectively.

References

Abril D, Navarro-Arribas G, Torra V (2012) Choquet integral for record linkage. Ann Oper Res 195 (1):97–110

Benaben F, Touzi J, Rajsiri V, Pingaud H (2006) Collaborative information system design. In: AIM conference, pp 281–296

Benaben F, Lauras M, Truptil S et al (2012) Mise 3.0: an agile support for collaborative situation. In: Camarinha-Matos LM, Xu L, Afsarmanesh H (eds) Collaborative networks in the internet of services. Springer, Berlin, pp 645–654

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). Enterp Inf Syst 9(5–6):556–582

Bénaben F, Mu W, Truptil S et al (2010) Information systems design for emerging ecosystems. In: 2010 4th IEEE international conference on digital ecosystems and technologies (DEST). IEEE, pp 310–315

Bezivin J (2006) Model driven engineering: an emerging technical space. In: Lämmel R, Saraiva J, Visser J (eds) Generative and transformational techniques in software engineering, International Summer School, GTTSE 2005, Braga, Portugal, July 4–8, 2005. Revised Papers, Part I. Lecture Notes in Computer Science, vol 4143. Springer, Berlin, Heidelberg, pp 36–64. doi:10.1007/11877028_2

Boissel-Dallier N (2012) Réconciliation sémantique des données et des services mis en oeuvre au sein d'une situation collaborative. Ph.D. thesis. Les thèses en ligne de l'INP

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

Bollati VA, Vara JM, Jiménez Á et al (2013) Applying MDE to the (semi-) automatic development of model transformations. Inf Softw Technol 55(4):699–718

Camarinha-Matos LM, Afsarmanesh H (2008) Classes of collaborative networks. In: Putnik GD, Cunha MM (eds) Encyclopedia of networked and virtual organization, vol 1. Information Science Reference, Hershey, pp 193–198

Chen D, Doumeingts G, Vernadat F (2008) Architectures for enterprise integration and interoperability: past, present and future. Comput Ind 59(7):647–659

Cohen W, Ravikumar P, Fienberg S (2003) A comparison of string metrics for matching names and records. In: Kdd workshop on data cleaning and object consolidation, vol 3, pp 73–78

Czarnecki K, Helsen S (2003) 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 V, Marcos E, Vara JM (2011) Applying CIM-to-PIM model transformations for the service-oriented development of information systems. Inf Softw Technol 53(1):87–105

Del Fabro MD, Valduriez P (2009) Towards the efficient development of model transformations using model weaving and matching transformations. Softw Syst Model 8(3):305–324

Falleri JR, Huchard M, Lafourcade M, Nebut C (2008) Metamodel matching for automatic model transformation generation. In: Czarnecki K, Ober I, Bruel J-M, Uhl A, Völter M (eds) Model driven engineering languages and systems. Springer, Berlin, pp 326–340

García J, Diaz O, Azanza M (2013) Model transformation co-evolution: a semi-automatic approach. Softw Lang Eng 7745:144–163

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

Grangel R, Bigand M, Bourey JP (2010) Transformation of decisional models into UML: application to GRAI grids. Int J Comput Integr Manuf 23(7):655–672

Guerra E, de Lara J, Kolovos DS, Paige RF, Dos Santos OM (2013) Engineering model transformations with transML. Softw Syst Model 12(3):555–577

Heeringa WJ (2004) Measuring dialect pronunciation differences using Levenshtein distance. University Library Groningen, Host

Henderson-Sellers B, Gonzalez-Perez C (2008) Standardizing methodology metamodelling and notation: an ISO exemplar. Springer, Berlin

Herrmannsdoerfer M, Benz S, Juergens E (2009) COPE-automating coupled evolution of metamodels and models. In: ECOOP 2009—object-oriented programming. Springer, Berlin, pp 52–76

Huang X (2007) An OWL-based WordNet lexical ontology. J Zhejiang Univ Sci A 8(6):864–870

IEEE (1991) IEEE standard computer dictionary: a compilation of IEEE standard computer glossaries. doi:10.1109/IEEESTD.1991.106963

Ide N, Pustejovsky J (2010) 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

Jouault F, Kurtev I (2005) Transforming models with ATL. In: Satellite events at the MoDELS 2005 conference. Springer, Berlin, pp 128–138

Jouault F, Allilaire F, Bézivin J, Kurtev I, Valduriez P (2006) ATL: a QVT-like transformation language. In: Companion to the 21st ACM SIGPLAN symposium on object-oriented programming systems, languages, and applications. ACM, pp 719–720

Jouault F, Allilaire F, Bézivin J et al (2008) ATL: a model transformation tool. Sci Comput Program 72 (1):31–39

Jung J, Choi I, Song M (2007) An integration architecture for knowledge management systems and business process management systems. Comput Ind 58(1):21–34

Kappel G, Kargl H, Kramler G, Schauerhuber A, Seidl M, Strommer M, Wimmer M (2007) Matching metamodels with semantic systems—an experience report. In: BTW workshops, pp 38–52

Kleppe AG, Warmer JB, Bast W (2003) MDA explained: the model driven architecture: practice and promise. Addison-Wesley, Reading

Konstantas D, Bourrieres J-P, Léonard M, Boudjlida N (2005) Interoperability of enterprise systems and applications. In: Proceedings of the international conference on interoperability for enterprise systems and applications (I-ESA) 2005, Geneva, Switzerland. Springer

Li L (2012) Effects of enterprise technology on supply chain collaboration: analysis of china-linked supply chain. Enterp Inf Syst 6(1):55–77

Lin F, Sandkuhl K (2008) A survey of exploiting wordnet in ontology matching. In: Bramer M (ed) Artificial intelligence in theory and practice II. Springer, Berlin, pp 341–350

Malone TW, Crowston K, Herman GA (2003) Organizing business knowledge: the MIT process handbook. MIT Press, Cambridge

Miller J, Mukerji J (2003) MDA guide version 1.0.1. Object Management Group. http://www.omg.org/cgi-bin/doc?omg/03-06-01

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

Panetto H, Molina A (2008) Enterprise integration and interoperability in manufacturing systems: trends and issues. Comput Ind 59(7):641–646

Porter MF (2001) Snowball: a language for stemming algorithms. http://snowball.tartarus.org/texts/introduction.html

Pressman RS (2005) Software engineering: a practitioner's approach. Palgrave Macmillan, New York

Ramirez R, Melville N, Lawler E (2010) Information technology infrastructure, organizational process redesign, and business value: an empirical analysis. Decis Support Syst 49(4):417–429

Scheer A-W (1992) Architecture of integrated information systems: foundations of enterprise modelling. Springer, Berlin. doi:10.1007/978-3-642-97389-5

Shvaiko P, Euzenat J (2005) A survey of schema-based matching approaches. In: Spaccapietra S (ed) Journal on data semantics IV. Springer, Berlin, pp 146–171

Terrasse MN, 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, pp 17–28

Touzi J, Lorré J-P, Bénaben F et al (2007) Interoperability through model-based generation: the case of the collaborative information system (CIS). Enterprise Interoperability, Part VII. Springer, London, pp 407–416. doi:10.1007/978-1-84628-714-5_38

Tratt L (2005) Model transformations and tool integration. Softw Syst Model 4(2):112–122

Van der Aalst W (2013) Business process management: a comprehensive survey. ISRN Softw Eng 2013:1–37

Varró D, Pataricza A (2004) Generic and meta-transformations for model transformation engineering. In: Baar T, Strohmeier A, Moreira A, Mellor SJ (eds) «UML» 2004—the unified modeling language. Modeling languages and applications. Springer, Berlin, pp 290–304

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

Wang T, Truptil S, Benaben F (2015a) An automatic model transformation methodology to serve web service composition data transforming problem. In: 2015 IEEE world congress on services (SERVICES). IEEE, pp 135–142

Wang T, Truptil S, Benaben F (2015b) 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). The steering committee of the world congress in computer science, computer engineering and applied computing (WorldComp), p 3

Weske M (2012) Business process management: concepts, languages, architectures. Springer, Berlin

Wetzstein B, Ma Z, Filipowska A, Kaczmarek M, Bhiri S, Losada S, Lopez-Cob J-M, Cicurel L (2007) Semantic business process management: a lifecycle based requirements analysis. In SBPM

Wieringa R, Daneva M (2015) Six strategies for generalizing software engineering theories. Sci Comput Program 101:136–152

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

Zdravković M, Noran O, Panetto H, Trajanović M (2015) Enabling interoperability as a property of ubiquitous systems for disaster management. Comput Sci Inf Syst 12(3):1009–1031