An Approach to Automatically Generated Model Transformations Using Ontology Engineering Space

Stephan Roser, Bernhard Bauer

Programming of Distributed Systems,
Institute of Computer Science, University of Augsburg, D-86135 Augsburg, Germany
[roser|bauer]@informatik.uni-augsburg.de

Abstract. Integration of systems across various enterprises to realize cross-organisational collaborations is complex. The application of model-driven software development facilitates faster and more flexible integration by separating system descriptions in models to different levels of abstraction. However, interoperability problems in modelling can be hardly overcome by solutions operating essentially at syntactical level. This paper presents an approach using the capabilities of semantic web technology in order to improve cross-organisational modelling by automated generation of model transformations.

1 Introduction

With the development of more and more complex systems across enterprises new challenges arise. While new interoperability issues in modelling enterprises and application systems have to be overcome, model-driven software development has to be fostered to enable an efficient development of flexible cross-organisational information and communication systems. Unfortunately, most current interoperability solutions, addressing the problems of different representation formats, modelling guidelines, modelling styles, and methodologies at syntactical level, focus on metamodels' abstract and concrete syntax. However interoperability can only be achieved on a semantical level. Thus there is a need to integrate and adapt ontologies in future architectures and infrastructures to the layers of enterprise architectures and to operational models. This can be done by applying mappings between different enterprise model formalisms based on an enterprise modelling ontology and by enriching heterogeneous business models semantically by ontologies to achieve a shared understanding of the enterprise domain [12].

In this work we propose the approach of *ontology-based model transformations* (ontMT), which integrates ontologies in modelling by utilising different technological spaces [14] (namely MDA and Ontology technological space) for automated generation of model transformations and mappings between metamodels. Interoperability in modelling is fostered by employing automated reasoning technology from ontology engineering technological space to the generation of model transformations. It is shown, how the ontMT approach can be realized as a *semantic-enabled model transformation tool (Sem-MT-Tool)* in a semantic-enabled IDE (SemIDE [1]). This tool

applies technology bridging MDA and Semantic Web approaches like the Ontology Definition Metamodel (ODM) [19] and makes use of the capabilities and benefits of both approaches.

The paper is organized as follows: After introducing background information to our work in chapter 2, chapter 3 comprises the problem statement which is the origin of our work. In section 4 the overall approach of ontMT is explained, before the procedure and concepts of automated model transformation generation are introduced in chapter 5. Chapter 6 provides a detailed look on the components of semantic-enabled IDE tool realizing the ontMT approach. Finally after extracts of a case study about ontMT in section 7, this paper concludes with a discussion and outlook in chapter 8.

2 Background and Context

Models: Definitions of models vary according to the purpose they are used for. A nice general definition is provided by a mega-model presented in [6], describing 'a model as a system that enables us to give answers about a system under study without the need to consider directly this system under study'. In short a model is *representationOf* a system, where systems can be physically observable elements like models or, more abstract concepts like modelling languages. A modelling language is a set models. Models are *elementsOf* a modelling language, if they *conformTo* a model of the modelling language (i.e. a metamodel).

Model-driven Software Development: Model-driven software development (MDSD), as a generalization of OMG™'s Model-driven Architecture paradigm (MDA®), is an approach to software development based on modelling and automated transformation of models to implementations [7]. In MDSD models are more than abstract descriptions of systems, as they are used for model- and code generation – they are the key part of the definition of a software system. Largely automated model transformations refine abstract models to more concrete models (*vertical model transformations*) or simply describe *mappings* between models of the same level of abstraction (*horizontal model transformations*). As model transformations play a key role in MDSD, it is important that transformations can be developed as efficiently as possible [8]. With the MOF 2.0 Query, Views, and Transformation (QVT) specification [20] the OMG provided a standard syntax and execution semantics for transformations used in a MDSD tools chain. Beneath of commercial products facilitating MDA[1] there exist open source projects dedicated to MDSD. The Eclipse Generative Modeling Tools project (GMT) [9] provides a set of research tools illustrating operations applicable to abstract models. Those tools range from code generation (oAW, MOFScript) over model transformation and weaving (ATL, AMW) to model management (AM3). The MODELWARE project [17] aims to close the gap between the end-users and solutions of currently used software development methods by using models for the construction of software. It contributes to the Eclipse Model Driven Development integration project (MDDi) [16]. MDDi is dedicated to offer a platform

[1] OMG's list of MDA companies: http://www.omg.org/mda/committed-products.htm

the integration facilities needed for applying a MDSD approach. It aims to provide the ability to integrate modelling tools to create a customizable MDSD environment.

Ontology: Ontologies are considered a key element for semantic interoperability and act as shared vocabularies for describing the relevant notions of a certain application area, whose semantics is specified in a (reasonably) unambiguous and machine-processable form [4]. According to [18] an ontology differs from existing methods and technologies in the following way: (i) the primary goal of ontologies is to enable agreement on the meaning of specific vocabulary terms and, thus, to facilitate information integration across individual languages; (ii) ontologies are formalized in logic-based representation languages. Their semantic are thus specified in an unambiguous way. (iii) The representation languages come with executable calculi enabling querying and reasoning at run time. *Application ontologies* contain the definitions specific to a particular application [10], while *reference ontologies* refer to ontological theories whose focus is to clarify the intended meaning of terms used in specific domains.

Technological Spaces: Kurtev et al. [14] introduce the concept of *technological spaces* (TS) aiming to improve efficiency of work by using the best possibilities of different technologies. A technological space is in short a zone of established expertise and ongoing research. It is a working context with a set of associated concepts, body of knowledge, tools, required skills, and possibilities. Initially five technological spaces (MDA TS, XML TS, Abstract Syntax TS, Ontology TS, DBMS TS) have been presented in [14], of which the *MDA TS* and the *Ontology TS* are important for our work. In the MDA TS models are considered as first-class citizens, representing particular views on the system being built. The Ontology TS can be considered as a subfield of knowledge engineering, mainly dealing with representation and reasoning. The ontology engineering space performs outstanding in traceability, i.e. in the specification of correspondences between various metamodels, while the MDA TS is much more applicable to facilitate aspects or content separation. With the Ontology Definition Metamodel (ODM) [19] the OMG issues a specification defining a family of independent metamodels, related profiles, and mappings among the metamodels corresponding to several international standards for ontology definition, as well as capabilities supporting conventional modelling paradigms for capturing conceptual knowledge. It is based on a grounding in formal logic, through standards-based, model-theoretic semantics, sufficient to enable reasoning engines to understand, validate, and apply ontologies developed using ODM. ODM includes a set of metamodels which are grouped logically together according to the nature of the representation formalism that each represents; ODM comprises metamodels for RDF(S), OWL, common logic (CL), topic maps (TM), and as a non normative part description logic (DL). Metamodels for RDF(S) and OWL represent more structural or descriptive representations that are commonly used in the semantic web community. ODM further defines transformations between the UML2 metamodel and different metamodels defined in ODM (e.g. OWL and RDF(S)).

Semantics: Unfortunately the notion of the term semantics differs in the context it is used and by the people using it. As the root of the problem Harel and Rumpe [11] identify insufficient regard for the crucial distinction between syntax and true semantics. Thus we clarify a few terms that have particular significance to this work.

- **Syntax:** *Syntax* N_L is the notation of a language *L*. It is distinguished between the concrete syntax, the textual or graphical representation of the language, and an abstract syntax or metamodel, being the machine's internal representation. A metamodel is a way to describe a language's syntax [11].
- **Semantic:** Semantic is the meaning of language, which is expressed by relating the syntax to a semantic domain. The description of a *semantic domain S* (its notation is N_S) can vary from plain English to mathematics. Semantics is defined by a *semantic mapping M:* $L \rightarrow S$ from the language's syntax to its semantic domain [11].
- **Ontological:** According to [19] 'an ontology defines the common terms and concepts (meaning) used to describe and represent and area of knowledge'. Talking about 'ontological' we mean technology of the Ontology TS, i.e. technology based on logics like RDF(S) or OWL used by the semantic web community to describe e.g. vocabularies or ontologies. Instead of semantic web enabled we also use the term *semantic-enabled* as synonym in this work.

3 The Problems

Model-driven software development is getting more sophisticated, by using more powerful tools and languages for modelling enterprises and developing application systems. As a natural course of things a huge diversity of often specialized methodologies, modelling languages and representation formats has been evolved, serving the purposes of the particular application domains. Thus we can find heterogeneity of models, especially in the syntax they use but also in the semantics of the used concepts that is associated with by different people and organisations.

- **Different versions of metamodels:** Time and again new versions of metamodels or domain specific languages are released, e.g. the metamodels for UML 1.x and UML 2.x. New model transformations have to be developed and existing model transformations have to be adjusted, wherever new versions replace the old ones.
- **Different abstract and concrete syntax:** Apart from different visual representations (concrete syntax) of modelling languages, metamodels in many cases also differ in their abstract syntax (as internal representation formats could be used e.g. different MOF™ implementations like EMF or MDR, OWL, etc.) though they were developed describing the same application domain.
- **Different semantics:** Since the semantics of a modelling language's concepts is rarely formally specified, different people and organisations often associate different semantics with the same concepts used in the metamodel. Often this is done consciously by applying special model styles.
- **Different modelling views:** In distributed systems development and application integration across collaborating organisations development processes have to be integrated by e.g. providing views of models which are compliant to the representation guidelines of the respective partner. Specification and implementation of those views has to be supported by an automated model transformation generation in order to improve seamless inter-organisational system development.

4 The Approach

To overcome those problems, ontMT facilitates methods to generate model transformations despite of structural and semantic differences of metamodels by applying Semantic Web technology of Ontology TS.

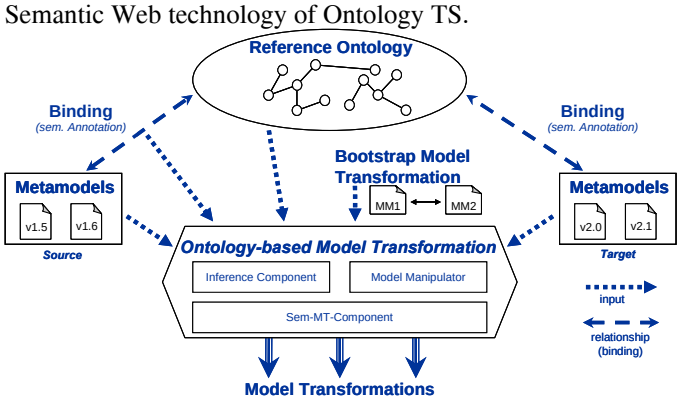

Figure 1: Ontology-based model transformation – overall approach

Different versions of metamodels are bound to a *reference ontology* (see definition in chapter 2) of a certain domain (see figure 1). Bindings (sem. Annotation) specify the semantic mapping from metamodels to the semantics of their concepts, i.e. to the reference ontology. To generate model transformations for various model transformation languages ontMT makes use of reasoning mechanisms. The metamodels and the reference ontology are given, while the bindings of the metamodels to the reference ontology have to be specified. Finally a bootstrap model transformation is needed, which is either given or generated automatically. The bootstrap model transformation is an initial model transformation (e.g. from metamodel v1.5 to metamodel v2.0) in which the rules for model transformation (and especially the semantics of the model transformations to generate) are encoded. If e.g. a new model transformation from metamodel v1.5 to metamodel v2.1 has to be generated, only the delta between metamodel v2.0 and v2.1 has to be considered. The new model transformation is generated by substituting the concepts of metamodel v2.0 with the concepts of metamodel v2.1 in the initial model transformation. Details are given in the next section.

5 Automated Generation of Model Transformations

Model transformations between various modelling languages can be automatically derived and generated by the ontMT approach (see figure 1). In this section we describe the procedure to generate mappings (i.e. semantically identical model transformations) between a modelling language *A* and a modelling language *B*.

For both languages exists abstract syntax N_A/N_B in various technological spaces: A has (like B) an abstract syntax in the MDA TS $N_{A\text{-}mda}$ and the Ontology TS $N_{A\text{-}ont}$ which are synchronized. Thus we can work with the syntax and the capability of that

technological space better suited for solving a problem (see figure 2a). Semantics of the concepts is described by the means of the semantic domain *SD* and its notation in a reference ontology N_{RO} (e.g. OWL) respectively. Semantics of languages is defined by semantic mappings to the semantic domain M_A: $A \rightarrow SD$ and M_B: $B \rightarrow SD$. The ontological grounding[2] is a notation of the semantic mapping from $N_{A\text{-}ont}$ to N_{RO}. The goal of the transformation to generate is to define 'identity' relationships between the concepts of *A* and *B*. The model transformation MT_{mapAB}: $A \leftrightarrow B$ between *A* and *B* has the following semantics $M_{MTmapAB}$: $MT_{mapAB} \rightarrow id$, where *id* is the identical mapping.

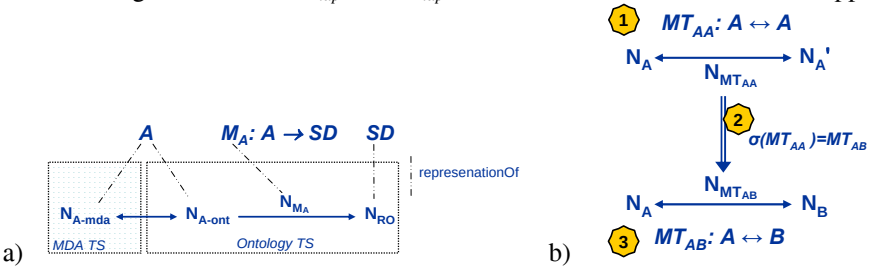

Figure 2: a) Modelling language, semantic mapping, semantic domain and their representations; b) Procedure of automated mapping generation

The generation procedure works on the model of the model transformation and the models of the modelling languages, and exploits the ontological grounding to the reference ontology. On the basis of reasoning results gained in the Ontology TS, modification operations are called to obtain the new model transformation working solely on the model of the model transformation and the metamodels. To generate the model transformation MT_{mapAB} the following steps are performed (see figure 2b):

- ① A bootstrap model transformation MT_{mapAA}: $A \leftrightarrow A$ is generated, mapping *A* on itself. This bootstrapping step is necessary to obtain a first model of the model transformation (transforming N_A to N_A')[3], which only has to be adjusted by modifications operators. Assuming the same ontological grounding for N_A and N_A', the bootstrap model transformation is an *id*: $M_{MTmapAA}$: $MT_{mapAA} \rightarrow id$.
- ② The inference engine derives interrelationships in between N_A' and N_B in the Ontology TS. This is possible, since both N_A' and N_B are mapped to the same reference ontology N_{RO}. It is automatically computed, how the concepts of N_A' can be substituted by semantically identical concepts of N_B ($\sigma(MT_{mapAA})=MT_{mapAB}$). Those interrelationships can be transferred to the MDA TS as the modelling languages *A* and *B* have synchronous representations in both MDA TS and Ontology TS.
- ③ Finally the concepts of N_A' are substituted with the concepts of N_B in the model of MT_{mapAA} and we obtain a model of the model transformation MT_{mapAB} with $M_{MTmapAB}$: $MT_{mapAB} \rightarrow id$. The substitution is performed via modification operations on the abstract syntax (model) of the model transformation MT_{mapAA} in MDA TS.

[2] The definition of the ontological grounding is a semantic annotation comprising static semantics of the metamodels, i.e. the semantics of the concepts, i.e. an ontology.

[3] A simple version of such a mapping can easily be generated on basis of a metamodel in the MDA TS. By traversing the metamodel via its containment relationships the appropriate mapping rules can be generated.

The first (bootstrapping) step helps to extend our approach to scenarios in which given model transformations have to be adjusted to modelling languages and meta-models for which they initially have not been designed. The bootstrap transformation is simply replaced by a given transformation and step 2 and 3 can be performed like described above. Avoiding to derive model transformations directly from ontologies results in a more flexible and well-structured architecture of the Sem-MT-Tool (see chapter 6). Issues concerning the model transformation, like checking if its model conforms to the QVT metamodel or considering the cardinality of associations' ends, are all dealt with in the MDA TS. The Sem-MT-Component invokes modifications operations on the basis of the results of the reasoning over the reference ontology and the application of heuristics (more see section 6.4).

6 Components of an Sem-MT-Tool

This section presents in detail the components of ontMT realized as a tool for a semantic-enabled IDE, namely the Sem-MT-Tool, is called its parts and functionality.

6.1 Overview

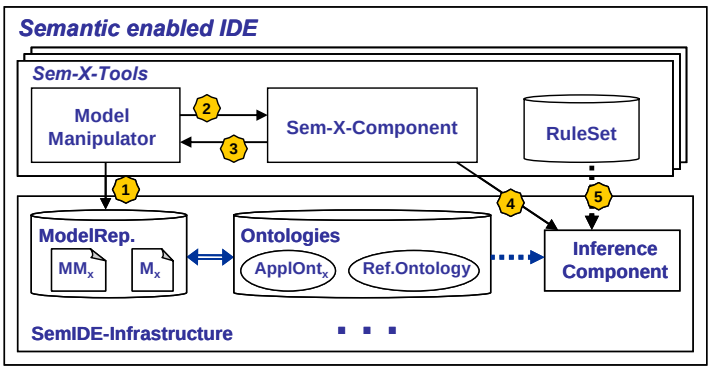

Figure 3: Ontology-based model transformation as part of a semantic-enabled IDE

OntMT is realized as Sem-X-Tool as part of a semantic-enabled IDE (SemIDE) (see figure 3) [1]. The infrastructure of the SemIDE provides basic functionality including a bridge between models of MDA TS and application ontologies of Ontology TS (like it is described in [3]) and an inference component, which can be individually configured and used by Sem-X-Tools registered at the infrastructure. Sem-X-Tools, like the Sem-MT-Tool presented in this paper, are built on top of the SemIDE infrastructure and consist of a model manipulator, a Sem-X-Component and a rule set. The model manipulator reads, creates, modifies and deletes models of the model repository ①. It delivers information about models to the Sem-X-Component ② and provides interfaces for model manipulation ③.

The Sem-X-Component implements the core functionality of a Sem-X-Tool. It makes use of the reasoning results gained by inferring ontologies and computes a

queue of model adjustment operators ④. Model adjustment operators are tasks that specify which modification steps have to be performed by the model manipulator. The queue of model adjustment operators is processed by calling the model manipulator's interface. Since Sem-X-Tools are based on different relationships between the ontologies' elements, each Sem-X-Tool has its own set of reasoning rules.

6.2 Inference Component

Figure 4 depicts a detailed architectural view on the inference component of ontMT. The inference component consists of a knowledge base and a reasoner. The base graph contains all facts of the knowledge base before the reasoning, i.e. the reference ontology, application ontologies[4] and the ontological groundings. The reasoner is triggered by rules specific to the Sem-MT-Tool, and computes the inference graph on the basis of the base graph. As the result of the reasoning the knowledge base contains information about all relationships important for the ontMT. These are especially relationships between the application ontologies.

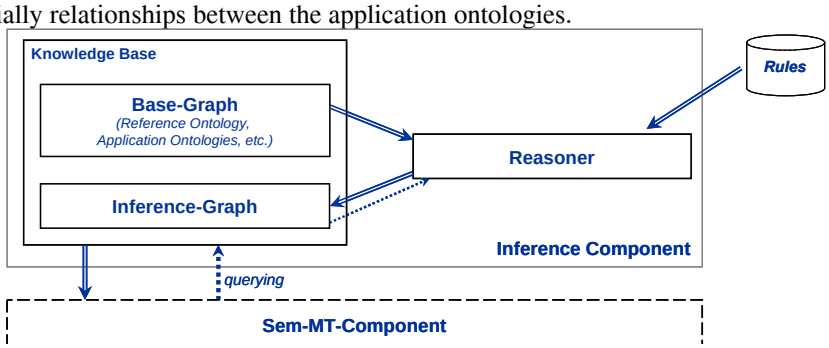

Figure 4: Inference component

6.3 Model Manipulator

The model manipulator provides modification operations on model transformations (the model transformations themselves are models) and the respective metamodels. It solely works on the abstract syntax of the (meta)models in the MDA TS. The model manipulator component is divided in a front and a back end, similar to established compiler technology (see figure 5). The front end primarily conducts tasks that only dependent on the source language, while the back end deals with all issues specific to the target language. The metamodels and the bootstrap model transformation are brought into an intermediate representation format by the scanner and the parser. The abstract syntax tree analyser checks, whether the modifications proposed by the inference component can sensibly be applied to the bootstrap model transformation. Those modifications are performed by the transformation manipulator.

[4] An application ontology corresponds to a metamodel in the Ontology TS.

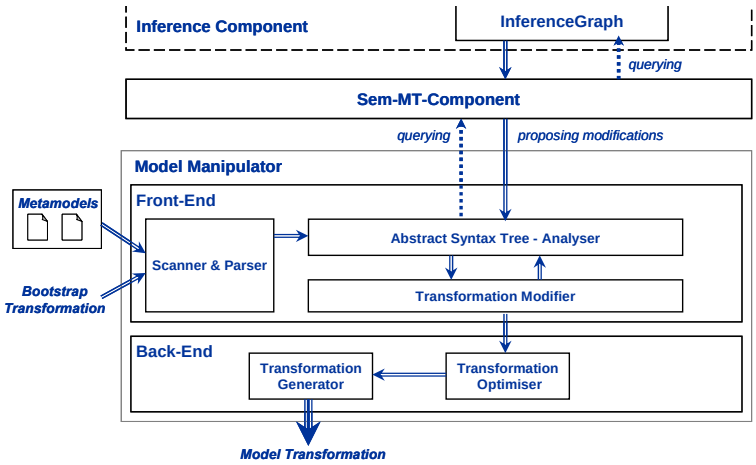

Figure 5: Model manipulator

6.4 Sem-MT-Component

The Sem-MT-Component (a specific Sem-X-Component) implements the core part of ontMT approach. It realizes the main functionality of the Sem-MT-Tool by using inference results of the Ontology TS to gain a queue of adjustment operators for the modification and generation of model transformations in the MDA TS.

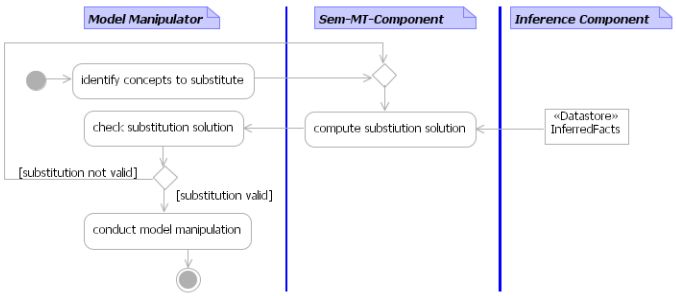

Figure 6: Activity diagram of ontology-based model transformation procedure

Figure 6 shows how the Sem-MT-Component and model manipulator interact in order to generate model transformations. Firstly, the model manipulator identifies metamodel's concepts, which have to be substituted in the model transformation. Secondly, the Sem-MT-Component queries the inferred fact base and computes the 'best possible' substitution of the metamodel's concepts (as a queue of adjustment operators) using heuristics. Before the adjustment of the bootstrap model is conducted, the model manipulator checks whether the substitution can be applied to the model transformation and is sensible. In the case that the substitution cannot be applied to the model transformation, the model manipulator states to the Sem-MT-Component which concepts need a different substitution and the severity of the ex-

ception. On the basis of the history of the previously proposed substitutions the Sem-MT-Component uses its heuristics to compute a 'next-best' substitution. When a proposed substitution finally passes the model manipulator's checks, the queue of modification operators is applied to adjust the model transformation's model.

The Sem-MT-Tool generates QVT model transformations. Therefore its model manipulator works on an EMF-based implementation of QVT's metamodel. The inference component uses a semantic web framework which allows loading, storing and modifying ontologies, reasoning over ontologies as well as conducting queries over ontologies. In a first prototype we use the Jena Semantic Web Framework [13], but also other projects like EODM [5] are an option for further implementations.

7 Case Study about Automated Mapping Generation

This chapter comprises a short example of ontMT and how the Sem-MT-Tool works. Therefore a mapping between two metamodels (figure 7 and 8) for process modelling is generated automatically. The first metamodel *MM1* is an excerpt of a metamodel for process orchestration in a service-oriented environment. The second metamodel *MM2* is also for process modelling.

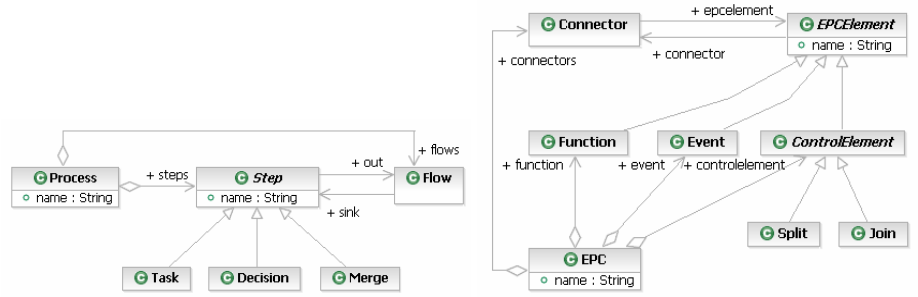

Figure 7 and 8: Metamodel MM1 and Metamodel MM2

The reference ontology in this example (see figure 9) is an excerpt of Web Ontology Language for Services (OWL-S). For the ontological grounding we use a notation similar to SMAIL [15] (Semantic Mediation and Application Interoperability Language). '=:' stands for a lossless annotation, where the annotation fully captures the intended meaning. '>:' denotes an overspecification, where the level of refinement of the annotated element is greater than the level of refinement of the concepts in the reference ontology.

Process[name,steps,flows]	=:	ServiceModel[name,composedOf,composedOf]
Step[name,out]	=:	ProcessComponent[name,connected]
Flow[sink]	=:	FollowedBy[followed]

EPC[name,connectors]	=:	ServiceModel[name,composedOf]
EPC[name,function,connectors]	>:	ServiceModel[name,comp.Of,composedOf]
EPC[name,event,connectors]	>:	ServiceModel[name,comp.Of,composedOf]
EPC[name,controlelement,connectors]	>:	ServiceModel[name,comp.Of,composedOf]

EPCElement[name,connector]	=:	ProcessComponent[name,connected]
Connector[epcelement]	=:	FollowedBy[followed]

Table 1 and 2: Ontological Grounding of MM1 and MM2

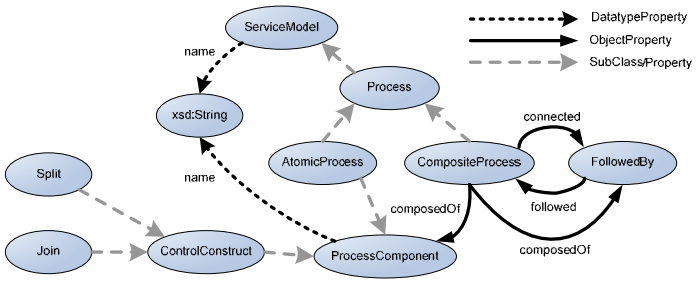

Figure 9: Reference Ontology

As a first step the bootstrap model transformation is generated by traversing metamodel *MM1*. Its model (see table 3) serves as an input for the model manipulator.

```
transformation process_bootstrap(mm1:MM1, mm1':MM1) {
 ...
 relation R1 {
   n: String;
   checkonly domain mm1 prc:Process {name=n, steps=stps:mm1.Step,
                            flows=flws:mm1.Flow};
   enforce domain mm1' prc':Process {name=n, steps=stps':mm1'.Step,
                            flows=flws':mm1'.Flow};
   where { R2(stps, stps'); R3(flws, flws'); }
 }
 relation R2 {
   n: String;
   checkonly domain mm1 stp:Step {name=n, out=flw:mm1.Flow};
   enforce domain mm1' stp':Step {name=n, out=flw':mm1'.Flow};
 }
 relation R3 {
   checkonly domain mm1 flw:Flow {sink=stp:mm1.Step};
   enforce domain mm1' flw':Flow {sink=stp':mm1'.Step};
 }
}
```

Table 3: Excerpt of bootstrap model transformation written in QVT [20] syntax

With this input (metamodels, reference ontology, ontological groundings and bootstrap transformation) the computation of the substitution of N_A' with N_B can start:

- For each concept to substitute the Sem-MT-Component queries the inference component for possible substitutions, which searches the fact base for triples of the kind <*Step:MM1*> <*?*> <*?:MM2*>[5]. The result is:

  <*Step:MM1*> <*equal*> <*EPCElement:MM2*> ; <*Step:MM1*> <*general*> <*Function:MM2*> ;
  <*Step:MM1*> <*general*> <*Event:MM2*> ; <*Step:MM1*> <*general*> <*ControlElement:MM2*> ;
  <*Step:MM1*> <*general*> <*Split:MM2*> ; <*Step:MM1*> <*general*> <*Join:MM2*>

[5] The triples of in the fact base are of the form <*subject*> <*predicate*> <*object*>. The Sem-MT-Component currently supports the following predicates: equal, meaning the subject is equal to the object; special, meaning the subject is a specialization of the object; general, meaning the subject is generalization of the object.

In the first substitution proposal only facts with the predicate being <*equal*> are considered, in order to find the best possible substitution. Since e.g. for the ObjectProperty <*steps(Process,Step): MM1*> no 'equal' substitution is possible, this ObjectProperty is omitted in the substitution in the hope that this does not affect the model transformation. Thus the proposed substitution is:

Process[name,flows]	→	EPC[name,connectors]
Step[name,out]	→	EPCElement[name,connector]
Flow[sink]	→	Connector[epcelement]

- Before the substitution is performed, the model manipulator checks whether the substitution can be applied and produces sensible results. In our example a new valid model transformation would be generated by the proposed substitution. However the new model transformation would loose connecting between its rules *R1* and *R2*, since the *steps*-Attribute is removed from the transformation.
- Thereon the Sem-MT-Component searches for an alternate substitution, also considering facts with predicates other than <*equal*>. For a substitution decision it applies a hierarchy, in which the predicate <*equals*> is better than <*special*> and <*special*> is better than <*general*>. The facts provided by the inference component are:

  <*steps:MM1*> <*general*> <*function:MM2*> ; <*steps:MM1*> <*general*> <*event:MM2*> ;
  <*steps:MM1*> <*general*> <*controlelement:MM2*>

  Based on its history of previously proposed substitutions[6] and the fact, that no facts with the predicates <*equals*> or <*special*> exist, the Sem-MT-Component proposes the following concept substitution:

Process[name,steps,flows]	→	EPC[name,function&event&controlelement, connectors]
Step[name,out]	→	EPCElement[name,connector]
Flow[sink]	→	Connector[epcelement]

- The model manipulator again checks the substitution, which would lead to the following new model transformation:

```
relation R1 {
  n: String;
  checkonly domain mm1 prc:Process {name=n,steps=stps:mm1.Step,
                          flows=flws:mm1.Flow };
  enforce domain mm2 epc:EPC {name=n,
                          function=fct:mm2.EPCElement, ...,
                          connectors=cnnts:mm2.Connector};
  where { R2(stps, fct); ... ; R3(flws, cnnts); }
}
```

  This transformation is invalid, since e.g. the attribute *function* is of the type *Function* and not of the type *EPCElement* (see *MM2*).
- Thus the Sem-MT-Component calculates an alternative subsitution for *Step*:

Process[name,steps,flows]	→	EPC[name,function&event&controlelement, connectors]
Step[name,out]	→	Function[name,connector]& Event[name,connector]& ControlElement[name,connector]

[6] The Sem-MT-Component has a history of its previous substitution proposals, so that it will not make the same proposal a second time and the search for substitutions terminates. If the Sem-MT-Component cannot find a 'better/other' substitution, though requested by the semantic analyser, the semantic analyser chooses the last substitution proposal producing a valid model transformation.

Flow[sink]	→	Connector[epcelement]

- The check of the model manipulator shows, that this concept substitution will generate a valid model transformation and will have no major side-effects on the model transformation.
- Finally substitution is used to generate the new model transformation via model manipulation. The following transformation rules are an excerpt of the automatically generated mapping between metamodel N_A to N_B:

```
relation R1 {
  n: String;
  checkonly domain mm1 prc:Process {name=n, steps=stps:mm1.Step,
                           flows=flws:mm1.Flow };
  enforce domain mm2 epc':EPC {name=n, function=fct:mm2.Function,
      event=evt:mm2.Event, controlelement=cntel:mm2.ControlElement};
                           connectors=cnnts:mm2.Connector};
  where { R2a(stps, fct); ...; R3(flws, cnnts); }
}
relation R2a {
  n: String;
  checkonly domain mm1 stp:Step {name=n, out=flw:mm1.Flow};
  enforce domain mm2 fct:Function {name=n,
                           connetor=con:mm2.Connector};
}
```

8 Discussion and Outlook

The ATLAS Model Weaver (AMW) tool implements the model weaving approach introduced in [2]. It enables the representation of correspondences between models, in so-called weaving models, from which model transformations can be generated. Nevertheless, though model weaving is to improve efficiency in the creation and maintenance of model transformations, creating weaving links is not automatic.

In [21], the authors introduce model typing as extension of object-oriented typing and propose an algorithm for checking the conformance of model types. It is presented, how model typing permits more flexible reuse of model transformations across various metamodels while preserving type safety. This approach improves reuse of model transformations, but does not provide automatic mapping generation.

The approach of ontMT integrates ontologies in MDSD and makes use of the reasoning capabilities of the Ontology TS. In the case study we have only considered subClassOf and equivalentClass relationships. In further versions we will apply expressions like intersectionOf or unionOf to enable more complex annotations and ontology specifications; in this context our reasoning rule set will be extended.

OntMT supports interoperability between enterprise models by an automated generation of horizontal transformations. This offers new possibilities for the integration of domain specific languages (DSLs) and ('legacy') models. Effort for adjusting model transformations to new versions of metamodels is reduced. Generic 'bootstrap' model transformations encode knowledge e.g. about software or platform architecture independent. Changes to model transformation rules have to be adapted in generic transformations only once. Thus ontMT improves collaboration of distributed development partners using different development environments, modelling guidelines and model transformations of methodologies.

Nevertheless our approach uses additional information, which has to be provided by the people developing metamodels and domain specific languages. Hopefully these ontological groundings can also be used by other semantic-enabled tools. Problems also arise, when no appropriate reference ontology exists. In those cases techniques for matching and merging ontologies, like linguistic, schema-based or probabilistic approaches, have to be applied to obtain a suitable reference ontology.

We have experimented with selected prototypes to show the feasibility of the ontMT approach, but a Sem-MT-Tool still has to be developed and integrated in a semantic-enabled IDE. On this basis the next goals will be to develop better heuristics for the computation of concept substitutions and to provide more sophisticated model manipulation operations on model transformations.

References

1. B. Bauer, S. Roser: Semantic-enabled Software Engineering and Development, 1st International Workshop on Applications of Semantic Technologies, 2006.
2. J. Bézivin, F. Jouault, P. Valduriez: First Experiments with a ModelWeaver, OOPSLA & GPCE Workshop, 2004.
3. J. Bézivin et al.: An M3-Neutral Infrastructure for bridging model engineering and ontology engineering, I-ESA Conference, 2005.
4. S. Borgo et al.: OntologyRoadMap. WonderWeb Deliverable D15; http://wonderweb.semanticweb.org, 2002.
5. EODM - Eclipse project, www.eclipse.org/emft/projects/eodm/
6. J. M. Favre: Foundations of Meta-Pyramids: Languages vs. Metamodels, Episode II: Story of Thotus the Baboon, Dagstuhl, Germany, 2004.
7. D. S. Frankel: Model Driven Architecture – Applying MDA™ to Enterprise Computing, Wiley, 2003.
8. T. Gardner et al.: A review of OMG MOF 2.0 QVT Submissions and Recommendations towards the final Standard, MetaModelling for MDA Workshop, 2003.
9. Generative Modeling Tools (GMT) - Eclipse project, http://www.eclipse.org/gmt/
10. N. Guarino: Understanding, Building, and Using Ontologies. Proceedings of Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop, 1996.
11. D. Harel, B. Rumpe: Meaningful Modeling: What's the Semantics of "Semantics"?, IEEE Computer, Volume 37, No. 10, pp. 64-72, IEEE, 2004.
12. IDEAS: A Gap Analysis, www.ideas-roapmap.net, 2003.
13. Jena 2 A Semantic Web Framework, http://jena.sourceforge.net
14. I. Kurtev, J. Bézivin, M. Aksit: Technological Spaces: An Initial Appraisal, Int. Federated Conference (DOA, ODBASE, CoopIS), Industrial Track, Irvine, 2002.
15. M. Missikoff et al.: A Controlled Language for Semantic Annotation and Interoperability in e-Business Applications, Workshop on Semantic Integration, 2003.
16. MDDi - Eclipse project, http://www.eclipse.org/mddi/
17. MODELWARE project, http://www.modelware-ist.org/
18. D. Oberle: Semantic Management of Middleware, Springer, 2005.
19. OMG: Ontology Definition Metamodel, ad/2006-05-01.
20. OMG: Revised Submission for MOF 2.0 QVT RFP (ad/2002-04-10), ad/2005-03-02.
21. J. Steel, J.-M. Jézéquel: Model Typing for Improving Reuse in Model-Driven Engineering, 8th International Conference MoDELS/UML'05, LNCS, 2005.