10-2008

ON THE CHALLENGE OF A SEMI-AUTOMATIC TRANSFORMATION PROCESS IN MODEL DRIVEN ENTERPRISE INFORMATION SYSTEMS

Slimane Hammoudi
ESEO, Angers, France, slimane.hammoudi@eseo.fr

Wajih Alouini
ISACM, Institut Supérieur Agronomique de Chott Meriem, I.R.E.S.A, Tunisia, wajih.alouini@gmail.com

Mohammed Mohsen Gammoudi
ESSAIT, Ecole Supérieure des Statistiques et d'Analyse d'Information de Tunis, Université 7 Novembre, Tunisia, mohamed.gammoudi@fst.rnu.tn

ON THE CHALLENGE OF A SEMI-AUTOMATIC TRANSFORMATION PROCESS IN MODEL DRIVEN ENTERPRISE INFORMATION SYSTEMS

Hammoudi, Slimane, ESEO, Angers, France, slimane.hammoudi@eseo.fr

Alouini, Wajih, ISACM, Institut Supérieur Agronomique de Chott Meriem, I.R.E.S.A, Tunisie, wajih.alouini@gmail.com

Gammoudi, Mohammed Mohsen, ESSAIT, Ecole Supérieure des Statistiques et d'Analyse d'Information de Tunis, Université 7 Novembre, Tunisie, mohamed.gammoudi@fst.rnu.tn

Abstract

Recently, Model Driven Engineering (MDE) approaches have been proposed for supporting the development, maintenance and evolution of software systems. Model driven architecture (MDA) from OMG (Object Management Group), "Software Factories" from Microsoft and the Eclipse Modelling Framework (EMF) from IBM are among the most representative MDE approaches. Nowadays, it is well recognized that model transformations are at the heart of these approaches and represent as a consequence one of the most important operations in MDE. However, despite the multitude of model transformation languages proposals emerging from university and industry, these transformations are often created manually. In this paper we present in the first part our previous works towards automation of the transformation process in the context of MDA. It consists on an extended architecture which introduces mapping and matching as first class entities in the transformation process, represented by models and metamodels. Our architecture is enforced by a methodology which details the different steps leading to a semi-automatic transformation process. In the second part, we propose the illustration of the architecture and methodology to the main case of transforming a PIM into PSM.

Keywords: Model Driven Architecture, Transformation Language, Mapping Metamodel, Matching Metamodel, Semi-Automatic Transformation, Transformation Architecture and Transformation Methodology.

1 INTRODUCTION

With the constant advances in computing, networking technologies and changes in organizational structures, the current information systems have become increasingly complex. To respond to the new technological requirements and the ever more exigencies of users, software engineering heightened its level of abstraction in an emerging trend called "Model Driven Engineering" (MDE). Model Driven Architecture (MDA) of the Object Management Group (OMG) is among these approaches of software development transferring the focus of work from programming to modeling by placing models in a central position of the life cycle of Software. Nowadays, it is well recognized that model transformations are at the heart of model driven architecture (MDA) approach and represent as a consequence one of the most important operations. These transformations consist of creating a set of rules involving, and at the same time merging both mapping and transformation techniques between two metamodels. However, despite the multitude of model transformation languages proposals emerging from university and industry, these transformations are often created manually. Thus, they are still fastidious and error-prone tasks, and therefore it is an expensive process. Semi-automating the transformation task is of paramount importance to decrease the errors, to reduce the effort required in a manual task and to improve the quality of the obtained mappings. In a previous work (Hammoudi, 2005-1), we initiated a first attempt towards this semi-automation. An approach separating mapping specification from transformation definition has been introduced, and this approach has been implemented in a tool called MMT (Mapping Modeling Tool). In this first approach, a mapping specification was created manually to define the relationships between metamodels (i.e. equivalent metamodel elements), while transformation definition was generated automatically and contained the operational description of the transformation rules between models. This work continues our research towards a semi-automation of the transformation process in Model Driven Enterprise Information Systems. We continue conceptualizing our framework including two main components: architecture and methodology (Hammoudi, 2005-2). This architecture introduces mapping and matching as first class entities in the transformation process represented by models and metamodels. This methodology details the different steps involved in a semi-automatic transformation process. The main idea of the given work is to increase automation into transformation process by considering matching techniques being actively studied in database and ontology areas to provide automatically mappings between two metamodels. Furthermore, beyond this architecture we propose to take advantage of the potential benefits of the progress of machine learning techniques to reuse the previously determined and accepted mappings and to take into account the manual adaptation done by the user and applied on the automatically obtained mappings.

This paper is organized as follows: The first part of section 2 presents our extended architecture for a semi-automatic transformation process and discusses the matching and mapping metamodels as two important components in this process. The second part of section 2 presents the methodology which in the first part, starts with the presentation of the main users involved in an MDA project. This section details the steps of our methodology distinguishing two main activities: preparation and execution activities. The section 3 reviews the main steps of the transformation process according to our architecture and methodology and introduces a comparison between ontology and meta-modeling techniques, a proof that various ontology matching techniques are available in the MDA context. Finally, section 4 concludes our work and presents some final remarks and future perspectives.

2 ARCHITECTURE AND METHODOLOGY

Before presenting our architecture for a semi-automatic transformation process, we would like to recall the two main problems concerning the main scenario of the MDA transformation process and that have motivated our previous and current works:

The first problem concerns the creation of "transformation rules" between metamodels which are often created manually, generally a fastidious and error-prone task, and therefore expensive process.

The second problem concerns the specification of these "transformation rules", which merge together techniques of mappings and transformations without explicit distinction between them. That is to say, the specification of correspondences between elements of two metamodels and the transformation between them are grouped in the same component at the same level. As we have already discussed in (Hammoudi, 2005-1), an explicit distinction between techniques of mapping and transformation could be very helpful in the whole MDA process of transformation. Moreover, the separation between the mappings and transformations parts is a first step towards a semi-automatic process, since mappings could be automatically generated by a matching process.

2.1 An architecture for the transformation process

Figure 1 illustrates our proposal of an extended architecture for the transformation process in MDA, allowing a semi-automatic generation of transformation rules and the semi-automatic generation of a target model from a source model. The three main operations of our approach are: Matching, Mapping and Transformation. All the components linked to these operations, and their relationships, are presented in figure 1 based on the four level MDA metamodeling architecture.

The matching operation is the process that produces the mappings between two metamodels. Generally, this task implies a search of equivalent or similar elements between two metamodels. In the database domain, this task is called schema matching. In our context, a matching model (Matching M) takes two metamodels designed by source and target (representing respectively a PIM and a PSM metamodel), and produces a mapping model (Mapping M).

The matching model conforms to a metamodel of matching (Matching MM) which implements techniques that consist of finding semantically equivalent modeling concepts between two metamodels. Thus, different kinds of relationships between metamodel elements are discovered using the metamodel of matching.

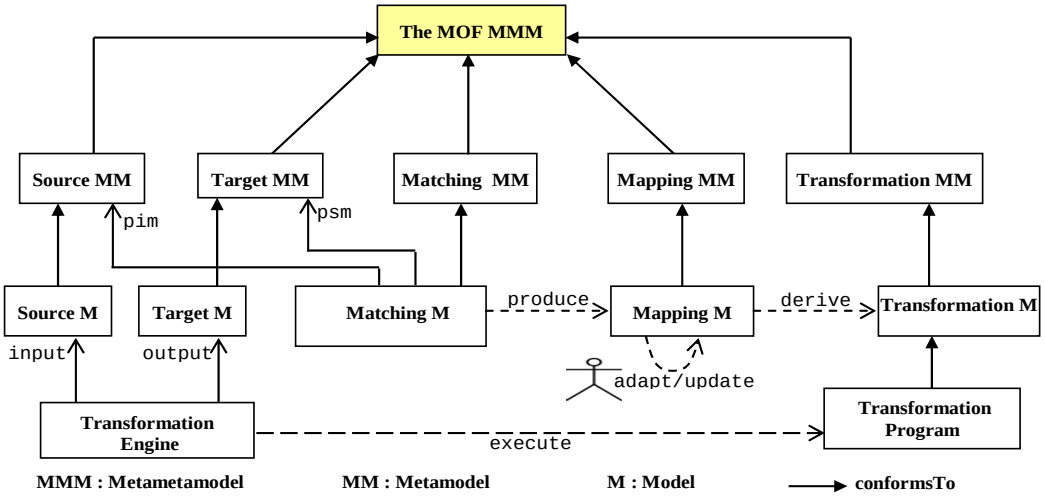


Figure 1. Architecture for a semi-automatic transformation process in MDA.

The relationships between metamodel elements are saved in a mapping model which conforms to a mapping metamodel *(Mapping MM***)**. This metamodel defines the different kinds of links (relationships) that could be generated by the matching model. Each kind of link corresponds to one transformation pattern specified in the transformation model described hereafter. Given that no generic matching solution exists for different metamodels and application domains, it is recommended to give the human expert the possibility to check the obtained mappings, and, if necessary, update or adapt it. This is the only step in the whole process, in which the expert intervenes to complete and/or validate the obtained results. Finally, a transformation model *(Transformation M)*, in conformance to its

transformation metamodel (*Transformation MM*), is derived automatically from a mapping model. A transformation model is basically represented by a set of rules that states how elements from source metamodel are transformed into elements of target metamodel. These rules are expressed in a transformation language based on MDA standards (OCL, MOF). This language, such as the standard QVT is described by a metamodel as a general formalism and abstract syntax for model transformation in MDA. Generally, the transformation model is completed by some information such as those concerning the execution environment, and produces a transformation program ready for the execution. This last part is often achieved by a designer (or software engineer) who implements a business model in a specific platform. Finally, a transformation engine takes a source model as input, and executes the transformation program to transform this source model into the target model.

According to our approach and architecture, the matching and transformation components are executable programs that take models or metamodels as parameters, while the mapping component is a set of relationships between elements of source and target metamodels. Concerning the mapping component, we have proposed in a previous work a generic metamodel and implemented it in a tool called MMT (Lopes, 2005-1). In this first approach, the mapping model between two metamodels, was supposed to be defined manually by an expert. From this mapping model, a transformation model represented by a set of rules is generated automatically.

2.2 A methodology for a semi-automatic transformation process

We intended through our methodology to enforce the new architecture for the transformation process presented above and to discuss the features of the different steps and users involved. A methodology should define guidelines to be used in a real project, in terms of the necessary activities, roles, and work products. For this purpose, we classified the users of MDA in two main categories :

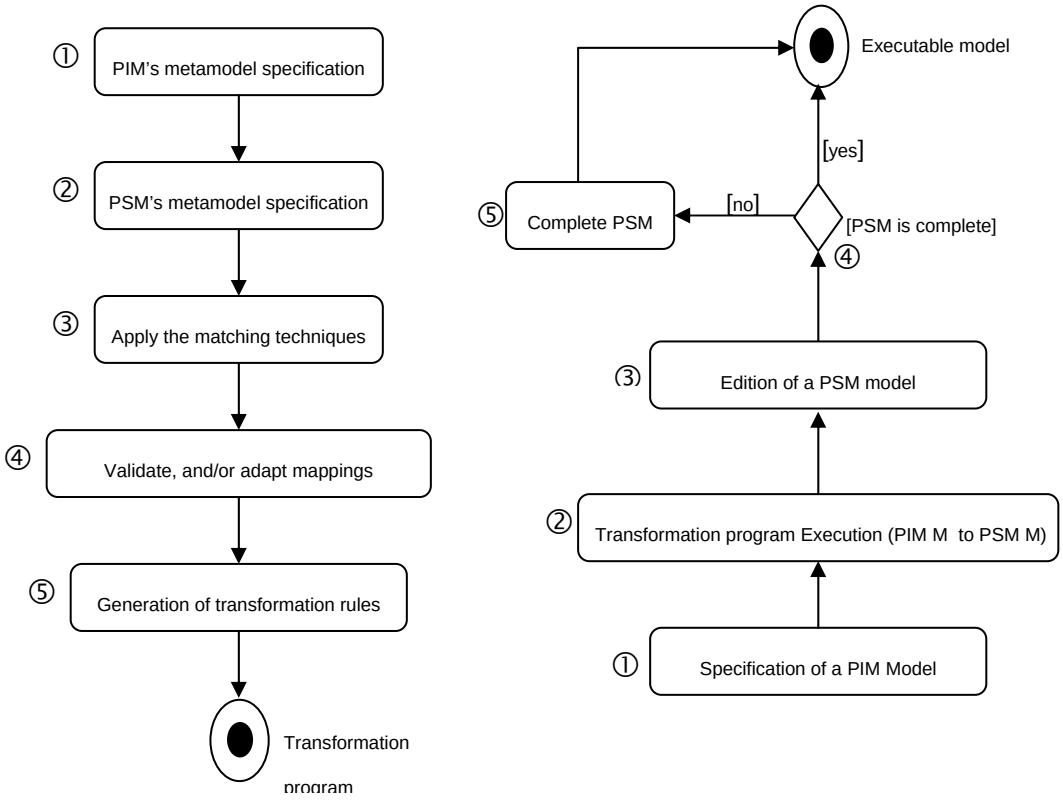
Expert users, which groups mainly knowledge builders, people who build knowledge (repositories) to be used in multiple different MDA-based projects, and occasionally knowledge facilitators, people who assemble, combine, customise and deploy knowledge for each specific MDA-based project,

Knowledge users: people who apply the knowledge built and facilitated by the other user categories, respectively. This category includes designers and software engineers.

The main steps of our methodology are represented by two activities diagrams. The first activity (a) shows the steps followed by an expert user who starts with the specification of the two metamodels source ① and target ② and follows the process until the generation of transformation rules ⑤ and an executable transformation program. The second diagram (b) illustrates the steps of a knowledge user who specifies a business model of a given application based on a PIM metamodel ① and generates automatically, by using a transformation program ②, an implementation of this business model on a given specific platform③.

The first goal within such a methodology is to introduce the matching process into the OMG's Model-Driven Architecture (MDA) approach in order to increase the degree of automation of the transformation process. This requires the reduction of human expert manual tasks by the rational choice among the plethora of existing works on matching algorithms. These algorithms have a high applicability to the problem of useful automatic mapping production.

Our methodology is based on MDA standards. All the metamodels, source and target, as well as transformations, are based on the same metametamodel "Meta Object Facility" ("MOF 2.0).



(a) Preparation activities **(b) Execution activities**

Figure 2. Methodology for a semi-automatic transformation process in MDA.

2.2.1 Preparation activities phase : transformation rules generation

In the preparation activities phase we have 5 activities (figure 2 (a)). It is mainly the charge of the expert-user.

- *PIM metamodel Specification* ①*:* This activity aims to define the appropriate PIM metamodel for a given application domain. PIMs metamodels are mainly specified by the whole or a part of, standards MDA metamodels such as UML or EDOC defined from the same metametamodel MOF. UML profiles could be used to take into account particular semantics of a given system.
- *PSM metamodel Specification* ②*:* This phase aims to define and specify the appropriate PSM metamodel. PSM are specified in the same formalism as PIM, but are more low levels than PIM as they must adhere to specific constraints imposed by the target platform, i.e. the platform in which the application will be implemented.
- *Matching techniques* ③*:* This phase aims to select the relevant match algorithms or matchers in order to generate matches (mappings). The Matching process could be represented by a model management operation called *match* (Bernstein, 2003). This operation takes two metamodels M1 and M2 as input and produces a first version of a Mapping model Mm as output. M1 and M2, respectively, conform to the corresponding metametamodels MM1 and MM2. Mm conforms to the metamodel of mapping MMm. The "→" operator represents the "conformsTo" relationship.

```
Mm → MMm = match (M1 → MM1, M2 → MM2)
```

- *Validate and/or update Mappings* ④*:* The human expert is in charge of this phase. Given that it is extremely optimistic to assert that all the mappings are obtained as a result of the previous phase or that all the matching techniques exists and are utterly effective, it is fairly rational to provide the human expert with interactive tools in performing the required correcting task. These tools, allow the

expert to accept, discard or modify the obtained mappings, furthermore, to specify correspondences which the matcher was unable to find. We call this part "Adaptation" technique which is essential in the transformation process. Loosely speaking, the mapping and matching techniques (models) could be defined with the following intuitive formula :

Mapping = Matching + Adaptation

- *Generation of Transformation Rules ⑤:* This phase aims to generate automatically transformation rules from mappings and formatting them into a transformation model in order to be used by the transformation program which transforms the PIM model into the PSM model. The mapping model obtained in the previous step after adaptation by the expert user should be completely defined to allow an automatic generation of transformation model. This transformation model, which consists of a set of transformation rules, is a model structured in conformance to the OMG's standard MOF2.0-QVT. In the same way as above, and loosely speaking, transformation and mapping models could be defined with the following intuitive formula (the main difference here, is that the "Derivation" technique is completely automatic while the "Adaptation" technique is the responsibility of the expert user):

Transformation = Mapping + Derivation

2.2.2 Execution activities : model generation on a target platform

This phase is mainly achieved by a knowledge user who, after defining his business model would like to implement it on a specific platform. In this phase we have 5 activities (figure 2 (b))which we could reduce to three. In fact, the three last activities concerning the PSM model are grouped together as they concern the same model.

- *PIM model Specification①:* Using a PIM metamodel, a knowledge user defines his business model focusing only on the business logic without taking into account implementation considerations. In this step the user may use, for example, different UML diagrams which will lead to a final class diagram ready for the implementation on a given specific platform.
- *Transformation program execution②:* The transformation program obtained in the first part is used here. It takes a PIM model as input and produces the equivalent PSM model as output. The transformation engine, which implements the transformation program, reads the source model, applies the rules to the source model and produces the corresponding target model.
- *From the edition to an executable PSM model* ③, ④, ⑤ *:* Here, we group the last three activities of the second phase, starting with a first binding of a PSM model and leading to an executable model on a given platform. The PSM model produced from the previous step represents a first version of a platform specific model which usually should be completed by information very specific to the target platform to produce a final executable model. So, the completeness of the PSM obtained is to be verified. In the case of effective completeness the transformation task is successfully accomplished, otherwise, the knowledge user will complete it manually.

3 TOWARDS A SEMI-AUTOMATIC TRANSFORMATION PROCESS

In the context of MDA, the common scenario of transformation consists on obtaining a PSM from a PIM. In the first part we explain how to achieve this task in conformance to our architecture and methodology. In the second part, we focus on the matching process. Presenting it as as a separate entity grants the advantage of making profit from the matching techniques actively studied in the ontology field.

3.1 From a PIM to a PSM

In our context, the first input is a PIM and the final output is a PSM (Figure 3). A matching model takes two metamodels designed by PIM-MM and PSM-MM (representing respectively a PIM and a PSM metamodel), and optionally a previous mapping (MAP1) and produces a new mapping model. MAP1 is the most relevant mappings formerly obtained for comparable metamodels. The matching

model is a relevant match algorithm selected in order to generate mappings. It conforms to a metamodel of matching which implements techniques that consist of finding semantically equivalent modeling concepts between two metamodels. Thus, different kinds of relationships between metamodel elements are discovered using the metamodel of matching. The relationships between metamodel elements are saved in a mapping model which conforms to a mapping metamodel. The human expert may check the obtained mappings, and, if necessary, update or adapt it. This is the only step in the whole process in which the expert intervenes via a friendly user-interface to complete and/or validate the obtained results. The produced mappings should be adapted and validated by an expert for the automatic derivation of a transformation model, as a set of transformation rules. Adaptation is the responsibility of the expert user who should accept, discard or modify the obtained mappings, more than that, to specify the correspondences which the matcher has been unable to find. The mapping model obtained in the previous step after adaptation by the expert user (MAP 2) should be completely defined allowing an automatic generation of a transformation model. This operation is called derivation.

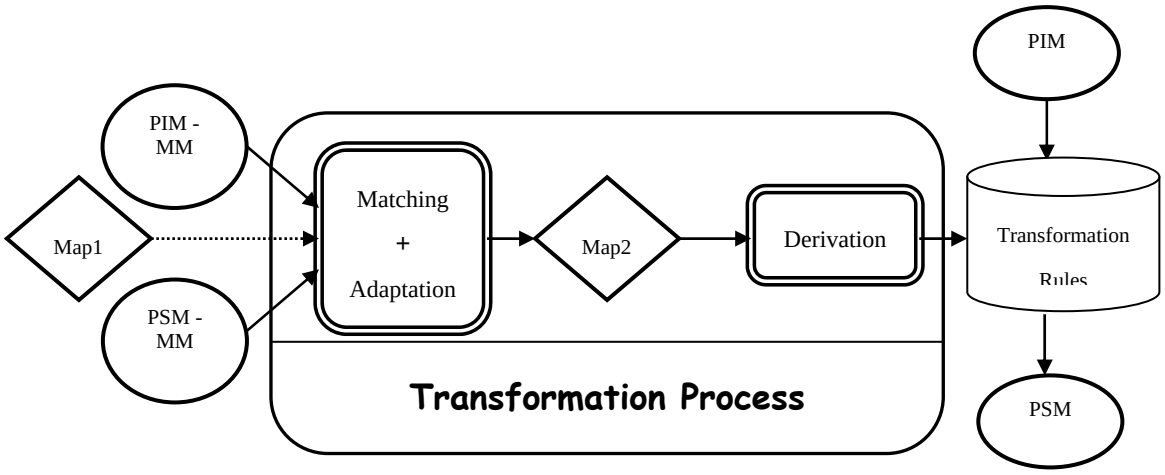


Figure 3. Semi-automatic Transformation Process.

3.2 Ontology matching vs Metamodel matching

The Ontology and Metamodeling technologies have many concepts and techniques in common and will soon converge. In fact, these two technical spaces, from an abstract point of view, are closely related. A clear similarity appears in their elements and features e.g., ontology - package, Class - Class, Association, Attribute - Property, and so on.

The matching process is under our focus and we propose to adapt and adopt schema and ontology matching techniques in order to apply them to MDA's Metamodels. Our proposal is founded on the convergence, currently seen, between models such as database schemas, XML message formats, UML diagrams and ontologies. Our architecture typically exploits the advances seen in schema and ontology matching techniques in order to automatically explore semantic correspondences between elements of two given metamodels. Many matching tools enabling automatic matching of various kinds of schemas confirmed their availability in many applications such as databases (data integration, schema integration, etc.), Artificial intelligence (knowledge bases, ontology merging, information gathering agents, etc.) or Web services (e-commerce, semantic web, etc.).

Considering two metamodels as two different concept hierarchies, the result of the adapted ontology matching technique, called Alignment, is a set of pairs of entities from two metamodels that are supposed to satisfy a given relation with a given confidence. The problem is easier in MDA's technical space than in ontology, since metamodels are well-defined and conform all to one metametamodel (MOF). Thus, the task of metamodel matching is converted into finding pairs of

elements from different metamodels that have an equivalent meaning otherwise, if impossible, the closest meaning.

In our case, we apply ontology matching techniques that yield a mapping between two metamodels, which is then the basis for a code generation process that derives model transformations defined between the metamodels.

The resemblance of the ontological space and the metamodel space is clearly visible in the objects they manipulate and the process they use. The Ontology matching is the process of finding the relations between ontologies; metamodel matching is the process of finding the relations between metamodels. The result of this process is called Alignment in the case of ontology, and Mapping in the case of metamodels. Various ontology matching techniques are available finding the correspondences, e.g., equivalence or subsumption, holding between sets of discrete entities (classes, properties, rules, predicates, formulas), belonging to two ontologies. In the state of the art, it is well recognized that ontologies matching are not circumscribed to one area of ontology, but applied to any application that communicates through ontologies. The main features of ontologies that are usually used for providing matching are: Terminological techniques, structural techniques and semantic techniques. We propose to combine these techniques and apply them in order to improve the results. On the other hand, we propose to store obtained mappings in our architecture so as a new matching process may search some certified existing mappings in mapping store.

4 CONCLUSION

In this paper, we have presented our approach for a semi automatic transformation process in MDA through an architecture and a methodology. We argue that a semi-automatic transformation process will be a great challenge in MDA approach as there is not yet a complete solution that automates the development of model transformation. A semi-automatic process will bring many advantages: it accelerates the development time of transformations; it reduces the errors that may occur in manual coding; it increases the quality of final transformation code. The key principle for this process is to consider mapping and matching metamodels as first class entities in MDA. In our present work, under the presentation of the matching process as a separate entity we make possible the use of the well advanced ontology matching techniques to the metamodels of the MDA context. In future work, we will implement metamodel matching algorithm, which takes its source from an ontology matching algorithm and apply it to the input and output metamodels without leaving the metamodel technical space.

References

Almeida, A.J.P., (2006). Model-driven design of distributed applications. PhD thesis, University of Twente. ISBN 90-75176-422

Bernstein, P.A, (2003). Applying Model Management to Classical Meta Data Problems. In CIDR'03, Proceedings of the Conference on Innovative Data Systems Research. CIDR.

Bézivin, J., (2005). On the Unification Power of Models. In Software and Systems Modeling, 4(2):171-188,

Bézivin, J., Dupé, G., Jouault, F., Pitette, G., Rougui, J.E., (2003). First Experiments with the ATL Model Transformation Language: Transforming XSLT into Xquery. In 2nd OOPSLA, Workshop on Generative Techniques in the context of Model Driven Architecture.

Bézivin, J., Hammoudi, S., Lopes, D., Jouault, F., (2004). Applying MDA Approach for Web Service Platform. In EDOC 2004, 8th IEEE International Enterprise Distributed Object Computing Conference.

Bézivin J., Heckel, R., (2004). Language Engineering for Model-driven Software Development. In Dagstuhl Seminar.

Bézivin, J., Lammel, R., Saraiva, J. Visser, J., (2006). Model Driven Engineering: An Emerging Technical Space. In GTTSE 2006, Generative and Transformational Techniques in Software Engineering.

Blanc, X., (2005). MDA en action, Ingénierie logicielle guidée par les modèles, EYROLLES. Paris, 1st edition

Budinsky, F., Steinberg, D., Merks, E. , Ellersick, R., Grose, T. J., (2003). Eclipse Modeling Framework: A Developer's Guide, Addison-Wesley Pub Co, 1st édition.

Dimitris, M., Dimitris, P., (2006). A Tool for Semi-Automated Semantic Schema Mapping: Design and Implementation. In Caise'06, International Workshop Data Integration and the Semantic Web.

Dominguez, K., Pérez, P., Mendoza, L., Grimán, A., (2006). Quality in Development Process for Software Factories According to ISO 15504 , In CLEI electronic journal, [http://www.clei.cl, Vol. 9 Num. 1 Pap. 3: June 2006].

Dou D, McDermott D, and Peishen Qi, (2003). Ontology Translation on the Semantic Web. In Proc.Int'l Conf. on Ontologies, Databases and Applications of Semantics (ODBASE2003). LNCS 2888**.**

Eclipse Tools Project. Eclipse Modeling Framework (EMF) version 2.0, (2004). Available on http://www.eclipse.org/emf.

Gavras. A, Belaunde.M, Pires L.F, Almeida.J.P, "Towards an MDA-based Development Methodology for Distributed Applications" , Proceedings of the 1st European Workshop on Model-Driven Architecture with Emphasis on Industrial Applications (MDA-IA 2004)

Hammoudi, S., Janvier, J., Lopes, D., (2005). Mapping Versus Transformation in MDA: Generating Transformation Definition from Mapping Specification, In VORTE 2005, 9th IEEE International Enterprise Distributed Object Computing Conference.

Hammoudi, S., Lopes, D., (2005). From Mapping Specification to Model Transformation in MDA: Conceptualization and Prototyping. In MDEIS'2005, First International Workshop.

Hammoudi, S., Alouini, W., Lopes, D., (2008). Towards a Semi-Automatic Transformation Process in MDA: Architecture and Methodology In ICEIS 2008, International Conference on Enterprise Information Systems. Barcelona 12-16, June 2008

Jouault, F., (2006). Contribution à l'étude des langages de transformation de modèles, Ph.D. thesis, University of Nantes.

Kappel, G., Kargl, H., Kramler, G., Schauerhuber, A., Seidel, M., Strommer, M., Wimmer, M., (2007). Matching Metamodels with Semantic Systems – An Experience Report. In BTW 2007, Datenbanksysteme in Business, Technologie und Web.

Kleppe, A., Warmer, J., Bast, W., MDA Explained: The Model Driven Architecture: Practice and Promise. Addison-Wesley, 1st édition, August 2003.

Lopes, D., (2005). Study and Applications of the MDA Approach in Web Service Platforms, Ph.D. thesis (written in French), University of Nantes.

Lopes, D., Hammoudi, S., Bézivin, S., Jouault, F., (2005). Generating Transformation Definition from Mapping Specification : Application to Web Service Platform. In CAISE'05, Proceedings of the 17th Conference on Advanced Information Systems Engineering.

Lopes, D., Hammoudi, S., Abdelouahab, Z., Schema Matching in the context of Model Driven Engineering: From Theory to Practice. Editores Tarek Sobh and Khaled Elleithy, Advances and Innovations in Systems, Computing Sciences and Software Engineering, Springer, 2006

Lopes, D., Hammoudi, S., De Souza, J., Bontempo, A., (2006). Metamodel matching : Experiments and comparison. In ICSEA'06, Proceedings of the International Conference on Software Engineering Advances.

OMG, 2001. Model Driven Architecture (MDA)- document number ormsc/2001-07-01. (2001).

OMG, 2003. UML Profile for Enterprise Distributed Object Computing Specification. OMG Adopted Specification ptc/02-02-05.

OMG, 2005. MOF QVT Final Adopted Specification, OMG/2005-11-01.

OMG, 2007. MDA Specifications. Available in http://www.omg.org/mda/specs.htm#MDAGuide.

Sun, X. L. and Rose, E.. Automated Schema Matching Techniques: An Exploratory Study. Research Letters in the Information

Unified Enterprise Modeling Language (UEML), 2003. Available on http://www.ueml.org.