A QVT-BASED APPROACH FOR MODEL COMPOSITION
Application to the VUML Profile

Adil Anwar[1,3], Sophie Ebersold[1], Mahmoud Nassar[2], Bernard Coulette[1] and Abdelaziz Kriouile[2]
[1] *University of Toulouse, IRIT ;UT2 ; 5 allées A. Machado, F-31058 Toulouse, France*

[2] *SI2M laboratory, ENSIAS, BP 713 Agdal, Rabat, Maroc*

[3] *LRIMIARF laboratory, Université Mohammed V-Agdal, Faculté des sciences, Rabat, Maroc*

Keywords: Model composition, VUML Profile, Viewpoints, Transformations, Correspondences, translation and composition rules, QVT-Core standard.

Abstract: With the increasing importance of models in software development, many activities such as transformation, verification and composition are becoming crucial in the field of Model Driven Engineering (MDE). Our main objective is to propose a model-driven approach to compose design models. This approach is applied to the VUML profile that allows to analyse/design a system on the basis of functional points of view. In this paper we first describe a transformation-based composition process and then we specify transformations as a collection of QVT-Core rules implemented in ATL. The proposal is illustrated by a simple example.

1 INTRODUCTION

In the context of increasing complexity of Information Systems, the composition of models is a challenging and recurring activity. To cope with composition of UML models developed separately, we have been using the VUML (View Based UML) profile which allows to put into action a view-based analysis/design. Our global objective is to formalise and implement the model composition activity (called "model merging" in VUML) by means of the Model Driven Architecture (MDA) (Soley et al., 2000) features, and to apply it to VUML.

Model driven approaches have been focusing mainly on the definition of languages and tools allowing the implementation of operations on models, such as transformations (Jouault et al., 2005) (OMG, 2007) or verification (Nébut et al., 2006), etc. Several research works deal more specifically with model composition in such domains as Aspect Oriented Modelling (Reddy el al., 2006) (Baniassad et al., 2004) or Requirements Engineering (Sabetzadeh et al., 2005) (Chitchyan et al., 2007). An emerging approach has proposed generic operators and mechanisms to reuse design knowledge of composition operators (Fleurey et al., 2007). These approaches do not consider composition as a MDA transformation.

However, composition should concern MDA approach since it is addressed in the context of Domain Specific Languages (DSL) and implemented in programming languages such as Java, or with transformation languages such as ATL (Jouault et al., 2005). We think that a more abstract and generic approach for specifying models composition (and specially *merging*) as a MDA activity is needed. Such models composition should be independent of any specific DSL. Therefore, we strongly believe that it lacks a shared definition of *composition* as a MDA operation for combining models.

In the present paper, we show how a composition operation can be specified using the transformation technology based on the QVT (Queries, Views, Transformations) standard (OMG, 2007). More precisely, we define the composition transformation as a set of declarative rules described with the QVT-Core language, which is the basic infrastructure of the declarative part of QVT. There are significant benefits to use QVT-Core for this purpose: (i) the QVT-core language allows the rules developer to declaratively specify transformation rules (which, also, can be bidirectional), and to check the structure of the involved models. Thus, the developer can put the emphasis on rules specification, rather than on rules execution and sequencing ; (ii) model patterns

Anwar A., Ebersold S., Nassar M., Coulette B. and Kriouile A. (2008).
A QVT-BASED APPROACH FOR MODEL COMPOSITION - Application to the VUML Profile.
In *Proceedings of the Tenth International Conference on Enterprise Information Systems - ISAS*, pages 360-367
DOI: 10.5220/0001710203600367

can be described in a graphical way similarly to UML object diagrams; this kind of representation makes the transformation rules quite easy to understand and to edit (Lohmann et al., 2007).

This paper addresses the following contributions:

- a generic composition process for merging UML design models. This process is applied to VUML profile;
- a models composition activity described in QVT-Core to graphically specify model transformations based on declarative rules.

The paper is organised as follows: section 2 introduces the VUML methodology to analyse and design model software systems with a user centred approach. Section 3 describes main steps of a composition process. Section 4 illustrates our approach with a simple example. In section 5, we specify the transformation rules in QVT-Core thanks to a graphical notation. Section 6 considers related works, and the last section discusses the contributions of this paper and some perspectives.

2 ANALYSIS/DESIGN APPROACH IN VUML

2.1 Principles of VUML Profile

The VUML profile (*V*iew based *U*nified *M*odelling *L*anguage) was developed to meet the needs of modelling complex information systems with UML according to various points of view. A point of view (called *viewpoint* in VUML) on the system represents an actor's requirements and rights. Such viewpoints may be considered as functional aspects. The main concept of VUML language is the *multiview class,* which is composed of a *base class* (shared by all viewpoints), and a set of view classes (extensions of the base class), each view class being specific of a given viewpoint. VUML's semantics is described by a metamodel, a set of well-formed rules expressed in OCL (OMG, 2003b), and a set of textual descriptions in natural language. On the methodological level, a process allows us to analyse and design software systems with respect to viewpoints. A code generator was developed to derive Java code from VUML class diagrams.

2.2 Analysis and Design Process Associated to VUML

The VUML analysis/design process is led by a set of designers who have specific viewpoints. The process is decomposed into five phases (figure 1):

- **Analyse Actor's Requirements.** During this phase, actors are identified and their requirements are described in terms of activities and use cases. Each actor is associated with a viewpoint.
- **Define a Common Domain Glossary.** This phase is a general and preliminary analysis aiming at defining the basic concepts which represent the business domain of the application. At the end of this phase, a common glossary of the system is established and shared by all the designers.
- **Design Viewpoint Models.** The aim of this phase is to produce a set of design models according to specific viewpoints. Each viewpoint model may be designed in parallel with the others.
- **Analyse Conflicts.** During this phase, representation conflicts, such as naming conflicts and structural conflicts are detected and resolved. While naming conflicts are solved through renamings into the design models, structural conflicts are solved by applying heuristics allowing a semi-automatic resolution method.
- **Compose Viewpoint Models.** The last phase is a composition operation; it consists in merging design models developed separately in order to obtain a global VUML design model. In the following sections, we will focus on this phase.

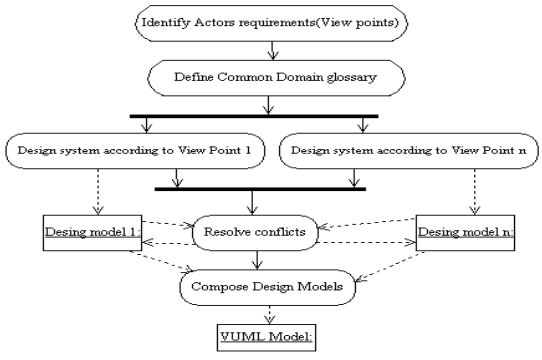

Figure 1: VUML Analysis and Design Process.

3 COMPOSITION PROCESS

3.1 Composition Steps

In order to provide a complete automatic solution for model composition, we reuse existing researches (Kolovos et al., 2006) (Fleurey et al., 2007) which have demonstrated that designing a model composition operator can be done through two separate operators: a correspondence operator and a merging operator. Thus, our composition process is composed of three main steps called *correspondence*, merging *and analysis*.

Correspondence. It consists in identifying the set of correspondences between models to be composed. This activity is governed by correspondence rules which specify the strategy of comparison between model elements. The comparison of elements is based on syntactic properties defined at the metamodel level. For example, the syntactic properties of a UML class are represented by the set {name, isAbstract, ownedAttribute,ownedOperation} that are properties of the metaclass *Class* in the UML metamodel (OMG, 2003a). A correspondence rule, applied to two elements describing the same concept, creates a link between those elements called *correspondence relationship*. Correspondence relationships are stored into a separate model called *correspondence model*. The correspondence step is summarized in Figure 2.

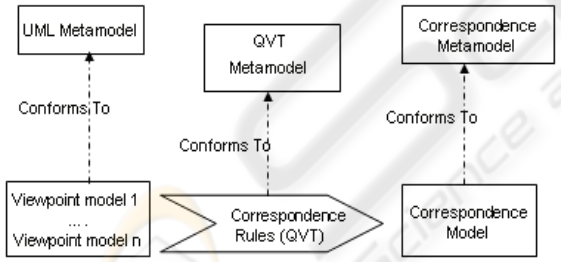

Figure 2: Step 1 of Composition: Elaboration of the correspondence model.

Merging. The merging step of the process depends on the target metamodel. In our application context, this step produces a VUML model. It creates VUML elements from source elements of the correspondence model according to the type of correspondence relationships which relate them. The strategy for merging two model elements is specified by a merging rule. Elements which have no correspondent in the opposite model are simply translated into the VUML model with respect to translation rules. The merging step is summarized in Figure 3.

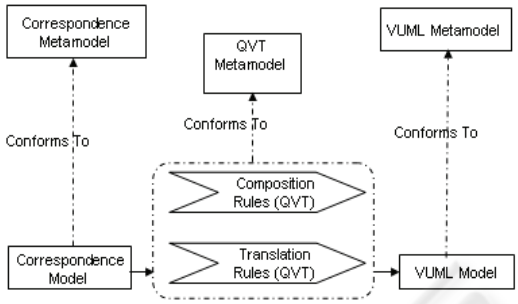

Figure 3: Step 2 of Composition: Creation of the resulting VUML Model.

Analysis. The last step of the composition process is an analysis activity that aims to supress composition errors, Thus the produced VUML model can be analyzed according to well-formedness rules defined in the VUML profile. The analysis strategy is similar to the one defined in (Bézivin et al., 2005); it allows to generate a diagnosis model which conforms to a diagnosis metamodel.

3.2 MDA-based Models Architecture

MDA proposal is a conceptual framework for system modelling organized in four levels of abstraction. We use this framework to describe the interactions among models and metamodels involved in our composition process. In Figure 4 below, we show only the three higher levels. The M1 level is dedicated to models that are conceptual abstractions of real word objects (M0). QVT transformations and VUML class diagrams are, thus, considered as models. At the M2 level, we find the concept of metamodel. Metamodels are models of languages which allow us to describe the models involved in M1 level. Four metamodels are to be considered in our approach: UML2 metamodel, Correspondence metamodel, VUML metamodel, and QVT metamodel. We proposed in a precedent work (Anwar et al., 2007b) a kernel of a Correspondence metamodel. It contains an abstract metaclass called *CorrespondenceRelationship* which is specialized according to the semantics of bindings between elements. Due to space limitation, we do not describe here this correspondence metamodel.

The M3 level of this framework is composed of the metametamodel MOF2.0 which is able to describe itself (meta-circularity).

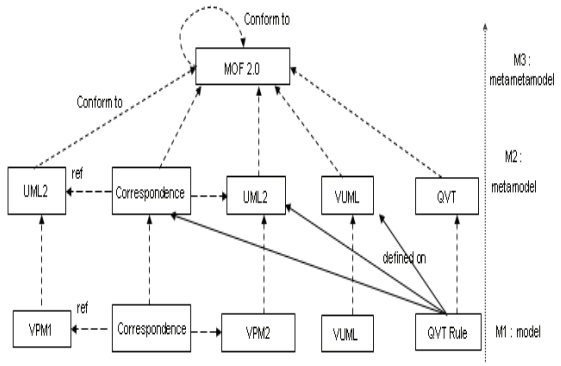

Figure 4: MDA-based models architecture.

4 EXAMPLE

To illustrate how the transformation rules are applied on models, we consider a simple example which particularly describes a weighted point through the WPoint class designed according to two viewpoints. The first viewpoint represents a weighted point with respect to its Cartesian coordinates. We consider also the class Segment which is composed of two weighted points and a length method to calculate the Euclidian distance between the two points. Figure 5 shows the resulting design model elaborated with the Cartesian viewpoint.

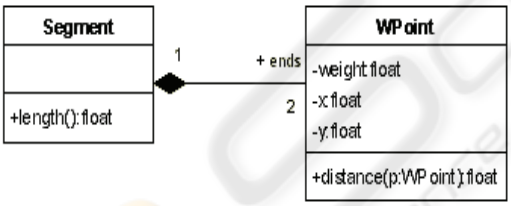

Figure 5: The Cartesian Viewpoint design model.

Let us consider now another viewpoint that identifies a weighted point object by its polar coordinates. A set of points forms a Disk which is represented by a radius and a method to calculate for example the perimeter of the Disk. Figure 6 shows the resulting design model elaborated with the Polar viewpoint.

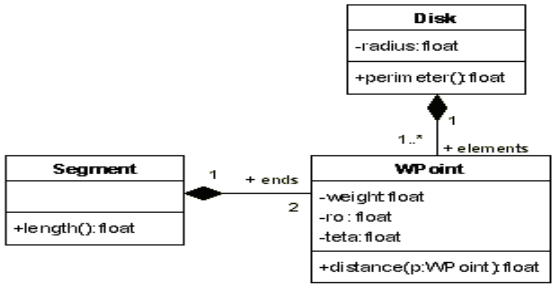

Figure 6: The Polar Viewpoint design model.

The design models described above have to be composed to build the global view of the system. This operation is necessary for checking the global consistency of the system's model, or for analyzing interactions across the composed views (Fleurey et al., 2007).

The VUML merging scenario, illustrated in Figure 7, can be achieved as follows: the class Segment is the same in both design models. Therefore, it is merged as one class Segment in the VUML model. The class Disk exists in the Polar viewpoint model only; so it is translated as the multiview class Disk which has a base and one Polar view. The class WPoint appears in both design models with distinct descriptions; thus it is translated as the multiview class WPoint: the shared elements like weight and distance are placed into the base, whereas specific coordinates are put into Cartesian and Polar views.

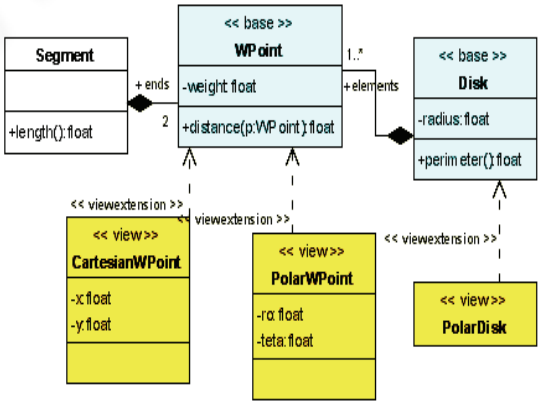

Figure 7: The VUML Design model.

In the following section, we will use those two viewpoint design models and the VUML resulting model to illustrate the application of transformation rules in the composition process.

5 QVT TRANSFORMATION RULES

Our model composition strategy is governed by three categories of transformation rules: correspondence, composition and translation rules. This enables us: first to establish correspondences between input models, second to merge these models in order to produce the global model.

As justified in the introduction (Section 1) of this paper, we have adopted the QVT standard to describe the transformation rules that govern our composition process. More precisely, we specify rules in the QVT-Core language (OMG. 2007) with a graphical representation defined in (Greenyer et al., 2007). This notation is similar to that of UML object diagrams, that is convenient to identify both source and target patterns of transformation rules.

This section presents these three kinds of rules. To illustrate them, we consider the two design viewpoint models of the example described in Section 3 above.

5.1 Correspondence Rules

This category of rule focuses on identifying all possible relationships between source model elements. In this aim, the compares the meta-properties values defined in metamodel constructs. This operation is showned in the guard condition of the rule. That means that the relationship between source elements holds (creating a correspondence relationship which relates the two source model elements) only when the guard condition is evaluated to true.

According to the semantics of correspondence which exists between two model elements, there are different types of relationships. For example, for class elements, we have identified two correspondence relationships: similarity and conformity. Conformity holds when two classes appear with the same name and are semantically equivalent (represents two views of the same concept), and when they have the same properties (attributes, operations, associations, etc.).

The transformation rules in QVT-Core language are called mappings; a single mapping is composed of several types of patterns. For example, the mapping *AttributesToEquality* (Figure 8) is structured in four columns. The first three columns contain the patterns of the involved models, and the last one contains the mapping nodes, called also *trace objects* that reference nodes of the domain patterns.

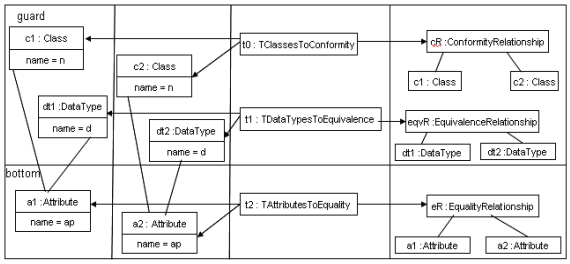

Figure 8: The AttributesToEquality mapping.

This rule shows the case where two attribute elements of source models having two corresponding types and defined in two distinct classes sharing the same name are related by an EqualityRelationship.

The mappings *ClassesToConformity* and *DataTypesToEquivalence* are called top mappings because they specify the context of the mapping *AttributesToEquality*. Hence, they are placed in the guard pattern of this mapping. They are considered as preconditions of the rule.

To start the transformation, the guard patterns are matched in the source models, and then the target model elements are created. In this case, a *ConformityRelationship* element and an *EquivalenceRelationship* element are created in the target model. Later, the pattern in the bottom area of the mapping *AttributesToEquality* is searched in the source models when the required precondition pattern is true, the context of the mapping can be applied. Finally, the *EqualityRelationship* element is created in the target model.

Given our example about weighted points, an *EqualityRelationship* is created in the Correspondence model to link both weight attributes of viewpoint models (Figures 5 and 6).

5.2 Composition Rules

The second category of transformation rules regroups the composition rules; it consists in building new model elements from elements related by correspondence relationships.

To start this composition activity, the composition operator looks for the set of elements that match in the correspondence model and creates a new model element in the target model with respect to the semantics of the correspondence relationship. Indeed, the composition strategy depends on the nature of relationships. For example, if two class elements are related by a *ConformityRelationship*, the composition strategy consists of merging these classes that is to create in

the target model a class that represents a deep copy of the sources classes.

Figure 9 shows a QVT-Core rule that describes how two classes related by a similarity relationship are composed to build a multiview class in the VUML model.

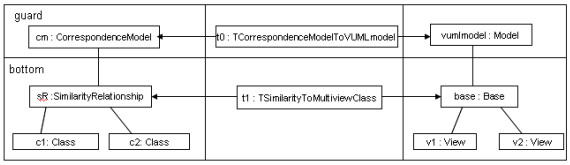

Figure 9: The QVT-Core SimilarityComposition rule.

To apply this rule, the top mapping described in the guard pattern must hold first so that a VUML model could exist. When the guard is true, the context of the rule *SimilarityToMultiviewClass* is available and so, the bottom pattern is searched in the source model and matched. Then, a multiview class is created in the target model for each similarity relationship found in the correspondence model.

Let us consider our example about weighted points. A SimilarityRelationship was created about WPoint during the Correspondence step. Henceforth, a multiview class is created with a base and two views in the VUML target model (figure 7).

5.3 Translation Rules

As composition rules, translation rules produce also elements in the target model. The main difference between them is that translation rules consider only elements for which no correspondent has been found in the opposite model. In case where the default translating strategy is applied, the source elements are deeply copied into the target model.

Alternate translation strategies must be used in some situations depending on the target metamodel. For example, if we apply the default translation strategy to the Disk class (Figure 6), this class will appear in the VUML model as a normal UML class which may be lead to an inconsistency because this class should be accessible only from the polar viewpoint. To resolve this problem, the translation rule has to create a multiview class as target element with a base and one Polar view (Figure 7). In this way, it is possible to access to the WPoint class only through the PolarDisk view, that consistent with the Polar viewpoint model (Figure 6). The rule described below expresses this translation strategy.

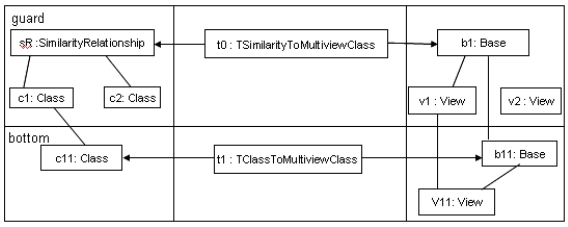

Figure 10: The QVT-Core Class2MultiviewClass Translation rule.

Let us consider again our example about weighted points. Class Disk appears only in the Polar viewpoint model, thus, it is translated into the VUML model as a multiview class of same name with one Polar view (figure 7).

6 RELATED WORKS

A number of researchers have developed model composition known also as model merging approaches in different application domains: requirements engineering (Sabetzadeh et al., 2005), Aspect Oriented Modelling (Baniassad et al., 2004) (Reddy et al., 2006), Schema integration (Eder et al., 1994), Ontology merging (Noy et al., 2000). We will focus in the rest of this section on works that are close to our approach.

Reddy et al (Reddy et al., 2006) propose directives for model composition in aspect oriented design. The objective is to compose a set of aspect models which encapsulate crosscutting concerns (persistence, security, etc) with a primary model which represent the functionalities of a software system. This approach can be classified as imperative because it describes the operation of composition in an algorithmic way. Therefore, it is not easily compatible with our approach based on declarative rules which rather specify what should be transformed instead of how it should be done.

The EML language (Epsilon Merging Language) proposed by (Kolovos et al., 2006) is a rule based language for merging models. The tool support of EML was developed as an Eclipse plugging. EML belongs to the Epsilon platform (Epsilon, 2006), which is a model driven framework for developing integrated languages for model management tasks such as model comparison, model transformation, model validation, etc. The comparison strategy in EML is similar to ours. But contrary to EML, our approach considers the result of the correspondence step as a model that is the input of the merging step.

The Atlas Model Weaver (AMW) (Del Fabro et al., 2005) is a model composition framework that uses model weaving and model transformation to define and execute composition operation. The tool support is available as an open source deliverable of Eclipse GMT (EclipseGMT. 2005). This technique has the advantage of being generic and flexible thanks to the extension mechanism of the weaving metamodel; however, the manual definition of the links between model elements is a tedious work.

Other works such as (Yahyaoui et al., 2005) (Romero et al., 2006) focus on the definition of viewpoints correspondences in the context of the RM-ODP standard (ISO, 2005). These authors relate viewpoints through explicit links for change management and propagation. But contrary to them, we also use such links to merge viewpoint models as shown above in this paper.

7 DISCUSSION AND CONCLUSIONS

In this paper, we have presented a QVT-based approach to specify rules to compose models developed separately. We adopted QVT because it is the OMG standard recommended for specifying MDE transformations. We used the QVT-Core language which represents the basic infrastructure of the declarative part of QVT. In addition, we adopted the graphical notation proposed by (Greenyer et al., 2007).

Compared to our previous works (Anwar et al., 2007b) the main benefit of this study is to use a standardized language to express composition as declarative transformation rules at a high level of abstraction. Thus, the rules designer may concentrate himself on the definition of rules rather than on their sequencing.

Our approach has been applied to the VUML profile which allows to produce view-based models separately and merge them into a unique multiview model. We experimented it on a significant case study describing the view-based design of a Shared Medical File Sytem (Anwar et al., 2007a).

To validate our work, we have implemented the operations (QVT transformations) in ATL (Jouault et al., 2005) which is considered here as an implementation of the QVT standard. It is also a standard component in Eclipse, now integrated into the M2M project (Eclipse, 2007).

In the near future, we intend to extend our approach in several ways:

- Enrich our composition rules metamodel (Anwar et al., 2007a) so as to separate target model-independent rules from target model-dependent rules. Concretely, we have already identified generic composition rules which are reusable in any model composition strategy and specific rules for the VUML DSL. We intend to integrate this hierarchy into a well-defined metamodel.
- Enlarge the kinds of source models: we should compose not only UML models resulting from decentralized modelling with UML, but also VUML models previously composed through the VUML process.
- Integrate behavioural diagrams of UML (mainly state charts or activity diagrams) into the VUML profile so as to cover dynamic aspects of system design.
- Cope with compositional conflicts as follows: (i) semantics conflicts due to homonymy and synonymy at the analysis level. To solve them in automatic ways, one can reuse dedicated techniques coming from database community (Geller et al., 1992) or ontology-based works (Noy et al., 2000); (ii) structural conflicts at the design level. These types of conflicts may be solved through refactoring techniques but so far they are difficult to automate.

ACKNOWLEDGEMENTS

This work was partially supported by the COMPUS Project MA/06/151, PAI VOLUBILIS 2006.

REFERENCES

Anwar, A., Ebersold, S., Coulette, B., Nassar, M., Kriouile, A., 2007a. Une approche MDA pour produire un modèle VUML par intégration de modèles par points de vue. *In Proceedings of (IDM'2007)*, Toulouse France, Hermès Science, pp 41-58.

Anwar, A., Ebersold, S., Coulette, B., Nassar, M., Kriouile, A., 2007b. Vers une approche à base de règles pour la composition de modèles : application au profil VUML. *Revue RSTI-L'Objet*. vol.13- n°4/2007, pp 73-103.

Baniassad, E., Clarke, S., 2004. Theme: An Approach for Aspect-Oriented Analysis and Design. In Proceedings of *ICSE'04 (Int. Conference in Software Engineering)*, p. 158-167. Edinburgh, Scotland.

Bézivin J., Jouault F., 2005. Using ATL for Checking Models. *In proc. of the International Workshop on*

Graph and Model Transformation (GraMoT), Tallinn, Estonia. 2005.

Del Fabro, MD., Bézivin, J., Jouault, F., Breton, E., Gueltas, G., 2005. AMW: a generic model weaver. Dans 1ère Journée IDM'05. Paris, France, p. 105-114.

Eclipse GMT Project Web Page. http://www.eclipse.org/gmt/amw/, 2005.

Eclipse/M2M Project Web Page . http://www.eclipse.org/m2m/, 2007.

Eder, J., Frank, H., 1994. Schema Integration for Object Oriented Database Systems. Published in: Tanik M., Rossak W., Cooke D. (eds.): Software Systems in Engineering, ASME, PD-Vol. 59, pp. 275-284.

EpsilonSubProject2006. http://www.eclipse.org/gmt/epsilon/.

Fleurey, F., Baudrey, B., France, R., Ghosh, S., 2007. A Generic Approach For Model Composition. In Proceedings of *the Aspect Oriented Modeling. Workshop at Models 2007*. Nashville USA.

Geller, J., Perl, Y., Neuhold, E., Sheth, A., 1992. Structural Schema Integration with full and partial correspondence using the Dual Model, *Information Systems*, Vol. 17, No. 6, 1992, pp. 443-464.

Greenyer, J., 2006. A Study of Model Transformation Technologies - Reconciling TGGs with QVT. University of Paderborn, Department of Computer Science, Master/Diploma thesis.

Greenyer, J., Kindler, E., 2007. Reconciling TGGs with QVT. G. Engels et al. (Eds.): *MoDELS*, LNCS 4735, pp. 16-30.

ISO/IEC CD 19793, ITU-T Rec. X.906 (2005). Information technology – Open distributed processing – Use of UML for ODP system specifications. ISO & ITU-T.

Jouault, F., Kurtev, I., 2005. Transforming Models with ATL. In Proceedings of *the Model Transformations in Practice, Workshop at Models*. Montego Bay, Jamaica 2005.

Kolovos, DS., Paige, RF., Polack, FAC., 2006. Merging Models with the Epsilon Merging Language (EML). *In Proc. ACM/IEEE 9th International Conference on Models/UML)*, Genova, Italy, October.

Lohmann, C., Greenyer, J., Jiang, J., Systâ, T., 2007. Applying Triple Graph Grammars For Pattern-Based Workflow Model Transformations. In Journal of Object Technology, Special Issue: Tools EUROPE, October 2007. pp 253-273. http://www.jot.fm/issues/issue_2007_10/paper13/.

Nassar M., Coulette B., Crégut X., Ebsersold S.., Kriouile A., 2003. Towards a View based Unified Modeling Language. Proc. of *5th International Conference on Enterprise Information Systems (ICEIS'2003*), Angers, France.

Nébut C., Fleurey, F., Le Traon, Y., Jézéquel, JM., 2006. Automatic test generation: A use case driven approach. *IEEE Transactions on Software Engineering*, 32(3):140--155, March.

Noy, N., Musen, M., 2000. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In proc. Of AAAI/IAAI, pp 450–455.

OMG 2003a, UML 2.0 Superstructure Final Adopted specification,Document-ptc/03-08-02, http://www.omg.org/docs/ptc/03-08-02.pdf.

OMG 2003b, UML 2 OCL Final Adopted Specification, 2003. http://www.omg.org/docs/ptc/03-10-14.pdf.

OMG 2007. OMG: (MOF2).0 QVT Final Adopted Specification.http://www.omg.org/docs/ptc/07-07-07.pdf.

Reddy, Y. R., Ghosh, S., France, R. B., Straw, G., Bieman, J. M., McEachen, N., Song, E., Georg, G., 2006. Directives for Composing Aspect-Oriented Design Class Models. *Transactions of Aspect-Oriented Software Development*, Vol.1, No. 1, LNCS 3880, p75-105, Springer.

Romero, R., Moreno, N., Vallecillo, A., 2006. Modeling ODP Correspondences using QVT. *In MDEIS 2006*, Paphos, Cyprus, pp. 15-26.

Sabetzadeh, M., Easterbrook, S., 2005. An Algebraic Framework for Merging Incomplete and Inconsistent Views. *In 13th IEEE International Requirements Engineering Conference*, September 2005.

Soley et al., 2000. MDA Model Driven Architecture, Object Management Group White Paper, Draft 3.2 - November 27.

Yahiaoui, N., Traverson, B., and Levy, N. 2005. Adaptation management in multi-view systems. I*n Proc. of the 2nd International Workshop on Coordination and Adaptation Techniques for Software Entities (WCAT'05)*, pages 99–105, Glasgow, Scotland, UK.