AMW: A Generic Model Weaver

Didonet Del Fabro Marcos[1], Bézivin Jean[1], Jouault Frédéric[1], Breton Erwan[2], Gueltas Guillaume[2]

1 ATLAS Group, INRIA & LINA - 2, rue de la Houssinière - BP 92208 - 44322, Nantes Cedex 3

2 Groupe SODIFRANCE - 4 rue Château de l'Eraudière, BP 72438 - 44324, Nantes Cedex 3

Abstract

In the MDA approach proposed by the OMG (Object Management Group) as well as in other MDE (Model Driven Engineering) approaches such as Microsoft "Software Factories", model transformation is a central operation. However it is often necessary to create links between models, where distinct operations will be executed based on the link semantics. These links are captured by weaving models. One of the possible application domains is data mapping. We have been engaged in building a model weaver prototype as part of the construction of AMMA, a MDE platform. The basic assumption in such platforms is to consider models as first class entities. In the present work we go one step beyond to consider also model weaving as models, and thus as manageable entities. This paper suggests that several situations in software engineering could be captured by model weaving techniques. Present achievements (mainly output generation of a weaving model, extensible weaving metamodels, partial GUI generation, etc.) and future potential extensions are discussed.

1. Introduction

A model is an artifact that conforms to a metamodel and represents a given aspect of a system. The relations of "conformance" between a model and its metamodel and of "representation" between a model and a system are central to model engineering [5]. A model is composed of model elements. All this means that the metamodel describes the various kinds of contained model elements and the way they are arranged, related and constrained. A language intended to define metamodels is called a metametamodel.

A basic principle in MDE (Model Driven Engineering) is to consider models as first class entities. It has not only the advantage of conceptual simplicity; it also leads to clear architecture, efficient implementation, high scalability and good flexibility, allowing to operate directly on the mapping elements and to attach semantics to them. There are currently several approaches based on MDE principles, the most known being "Software Factories" by Microsoft [16] or MDA™ (Model Driven Architecture) by OMG [17].

One of the most important operations in MDE is model transformation. A model transformation language is used to define how a set of source models is visited and transformed in a set of target models. A transformation is a typed operation, but it has also some invariants, pre and post-conditions. They are all expressed in term of the metamodels and constraints applying to the

metamodels or to the models. However, we realized with our experiments that it is not possible to reduce all kind of operations to executable model transformations. We propose here that model weaving is other important operation in MDE and we present some arguments for this kind of facility. Its primary objective is to handle fine-grained relationships between elements of distinct models, establishing links between them. These links are captured by a weaving model. It conforms to a metamodel that specifies the link semantics. Typical application domains of model weaving are database metadata integration.

We have undertaken the building and evaluation of a model-weaving prototype using the Eclipse EMF [8] [9], in order to progressively define the position of this kind of tool in the standard model-engineering workbench. It allows the visualization of models and the correspondences between them (a weaving model) and it has a flexible extension mechanism where developers can add new visualization facilities according to the application needs. Distinct visualization forms (such as class diagrams, Petri networks, ontology representation) are plugged as needed. We created a platform with a minimal metamodel that can be extended following the current platform requirements, for example using Domain Specific Languages (DSLs) instead of an overall design language such as UML. In [15] the new idea that was proposed was to use the Unified Modeling Language (UML) to express mappings between two given models. In the present work we update and extend this approach in several ways. First we consider, instead of UML, a variety of small dedicated mapping languages, each one based on a specific metamodel. The necessary variability of these mapping metamodels seems essential. Furthermore, we base our approach on the possible extensibility of these mappings metamodels. Also one important step forward is that we have build, through successive iterations, a prototype named AMW (ATLAS Model Weaver) that allows us to experiment, validate and improve the approach. From these experimentations, we have already learnt several lessons:

1) The notion of a weaving session begins with loading the weaving metamodel, then the models to be linked. Then the mapping operations may start. At the end of a session a weaving model is produced. The linked models are said to be woven.

2) It is possible later to work again on this produced weaving model, in order for example to perform incremental mappings.

3) When the weaving metamodel is loaded in a session, it is possible to partially generate, corresponding to this metamodel, a suitable GUI for supporting the session.

4) A resulting weaving model may be used for many operations, for example for generating a transformation in the ATL language [2].

5) Contrary to a transformation, a weaving is usually a non automatic operation. It is usually performed by a human operator. However it may be guided by heuristics. These heuristics will often be domain specific. Taking into account heuristics on the basis of metamodels may be done on a modular basis.

This paper is organized as follows. Section 2 presents weaving operations on models and their implications in the AMMA platform. Section 3 describes the weaving metamodel and it addresses the issue of metamodel extensibility. In section 4 we present the current prototype. The related work is presented in section 5 and the plans for future work in section 6. Section 7 concludes the presentation.

2. Motivating examples

In software engineering practices, the "Y organization" (see Figure 1) has often been proposed as a methodological guide. The OMG has promoted this idea in the MDA proposal where a Platform Independent Model (PIM) should be merged with a Platform Definition Model (PDM) to produce a merged Platform Specific Model (PSM). Let us suppose we have a PIM for a bank containing the

class *BankAccountNumber*; we have also a PDM designed for a specific implementation platform containing classes *LongInteger* and *String*. Both models should be woven into a new platform specific model (PSM), but we must decide if *BankAccountNumber* should be *LongInteger* or *String*. Design decisions like that are one of the most important events in the software development chain. We will not discuss here the validity of this decision. However, we would like to ensure that this decision is well recorded, with the corresponding author, date, rationale, etc. Furthermore this decision is probably based on previous decisions and further decisions will be based on it. What we see here is that a metamodel for design decisions would be more useful with several properties and links associated to each decision. We can understand also that it would be very improbable to have an automatic weaving algorithm since this is most often a human decision based on practical know-how. The user deciding the weaving actions should be guided by intelligent assistants proposing her/him several choices. These assistants may be sometimes based on design patterns.

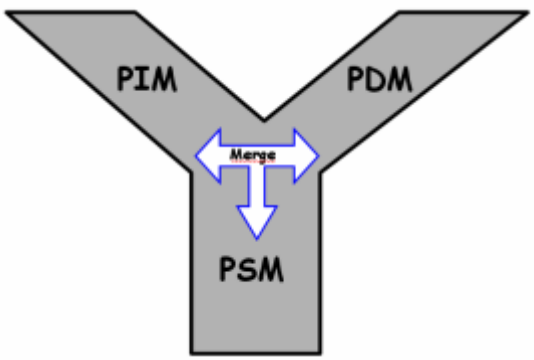

Figure 1. MDA and the Y-shaped organization

Another use of weaving operation is data integration from heterogeneous or legacy data sources. We take as inspiration an example from the work of Ph. Bernstein [3], [11]. We have two address books to merge and we get both metamodels MM_1 and MM_2. In MM_1 we have the class *Name* and in MM_2 the classes *FirstName* and *LastName*. Here we need to establish a more complex link stating that they are related by an expression of concatenation to produce a merged metamodel *MergeMM*. Bernstein proposes a Merge operator (among other operations for generic model management) for this kind of problems in the domain of database schemas.

3. Model Weaving

Metamodeling is a convenient way for isolating concerns of a system. A metamodel acts as a filter. It specifies the set of concerns that should be taken into account while creating a model. Separate metamodels may address distinct domains (information, processes, organization) or different levels of abstraction. Once these concerns have been separately expressed, they may have to be reassembled. This may be fulfilled by an ad hoc mapping, specific for each couple of metamodels. However, we found many similarities between mapping cases, for instance the creation of correspondences with semantic meaning between model elements. These correspondences are used as a base to perform different operations. Thus we think that a generic weaving operation must be defined.

In order to provide a description of a model weaving operation, let us suppose we have two metamodels *LeftMM* and *RightMM*. We need to establish links between their elements. Some issues have to be considered concerning the set of links between elements of both metamodels:

- The set of links cannot be automatically generated because it is often based on design decisions or heuristics.
- It should be possible to save this set of links as a whole, in order to use it later in various contexts.

- It should be possible to use this set of links as an input to automatic tools.

As a consequence, we come to the conclusion that a model weaving operation produces a precise weaving model *WM* representing the mapping between these metamodels. Like other models, this should conform to a specific weaving metamodel *WMM*. The produced weaving model relates with the source and target metamodels *LeftMM* and *RightMM* and thus will remain linked to these metamodels in a global model registry. Weaving operations may be applied to models instead of metamodels. In Figure 2 we illustrate the conformance relations of *LeftMM*, *RightMM* and *WM*.

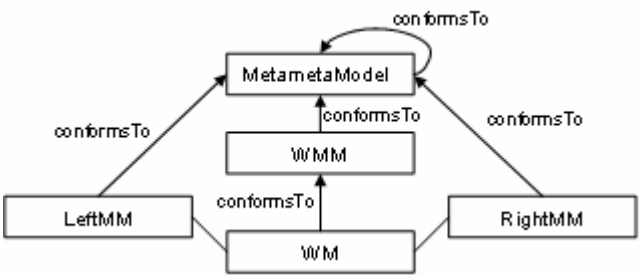

Figure 2. Weaving conformance relations

Each link instance has to be typed conforming to the given metamodel *WMM*. Links should provide useful semantic information to the designer. Even if some links contain only textual descriptions, these are valuable for tools supporting documentation or performing heuristics. Obviously more abstract constraints information, for example expressed in OCL, may be attached to a link.

3.1 Weaving vs. transformations

One question often asked is why we need model weaving operations in addition to model transformations. This question raises at least the following issues:

- Issue of "arity": Usually a transformation takes one model as input and produces another model as output, even if extensions to multiple inputs and outputs may be considered. A model weaving takes basically two models as input and one weaving metamodel.
- Issue of "automaticity": A transformation is an automatic operation while a weaving may need the additional help of heuristics or guidance to assist the user to perform the operation.
- Issue of "variability": A transformation conforms to a fixed metamodel (the metamodel of the transformation language) while there is no canonical standard weaving metamodel, since for every different application a new metamodel should be created.

Although one may argue that there may be several levels of abstraction in transformations (e.g. specifications and implementations of transformations), these three mentioned issues allow concluding that transformation and weaving are different problems and this has been confirmed by our first experiments with our transformation language and AMW.

However in some particular cases a weaving model may be itself transformed into a transformation model. The QVT RFP [12] defines mappings and relations transformations. Relation transformations are multi directional transformation specifications between two models. They are mainly used to validate two models and serve as a basis for implementing a transformation mapping.

3.2 Weaving metamodel

From what has been said until now, we conclude that there is no standard weaving metamodel *WMM* capable of capturing all weaving semantics. The two examples mentioned in section 2 are obviously different. Each application domain has different needs that must be considered by the developers. The design of a base metamodel of which most weaving metamodels may be seen as extension is a delicate compromise between expression power and minimality. To be able to use a

simple core in a wide range of applications, we also need a coherent extension mechanism that will allow the addition of new metamodel constructs.

Basically, a weaving model stands between a set of metamodels or models. It defines a set of links between elements from these metamodels through a reference mechanism which may be metamodel- or implementation-specific. Associations may be specified between links. Below we explain the main structures of our base abstract weaving metamodel (see Figure 3).

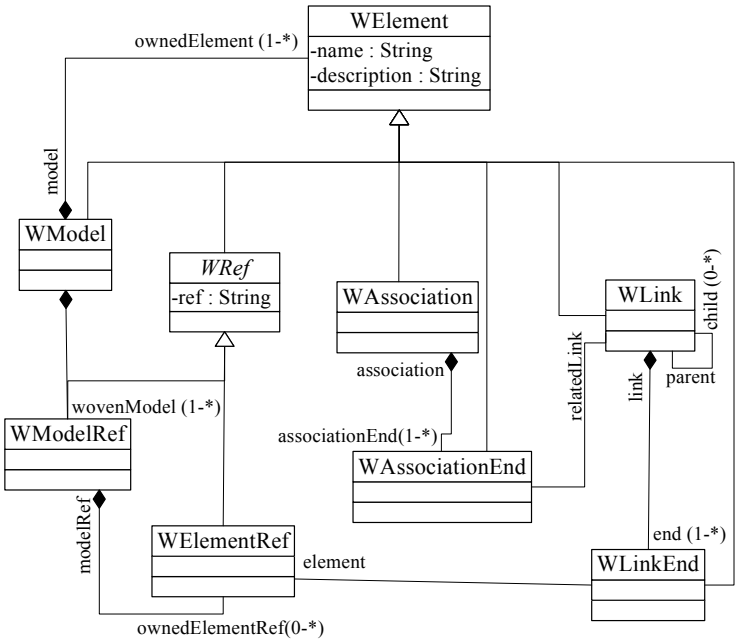

Figure 3. The weaving metamodel

- *WElement*: it is the base element of all metamodel elements. All the others elements extend it. It has two attributes: *name* and *description*.
- *WModel*: the weaving metamodel root element. It is composed of the weaving elements and the references to woven models.
- *WLink*: represents the link between model elements. The reference *end* enables linking between arbitrary numbers of elements. Weaving links can also relate with other weaving links to create a containment relation. This element should be extended to add different linking semantics to the weaving metamodel.
- *WLinkEnd*: indicates the extremity of a link, referencing the woven model elements through a *WElementRef*.
- *WRef*: abstract class that represents the references.
- *WElementRef*: all referenced elements of a woven model. The attribute *ref* contains the identifier of the woven elements. This element should be extended to add different identification mechanisms, for example using XPointer or XMI-IDs.
- *WModelRef*: references a (meta)model being woven. This allows keeping track of woven (meta)models. It is composed by element references.
- *WAssociation*: used to create association relationships between links.
- *WAssociationEnd*: similar to *WLinkEnd*, specifies the extremities of an association.

3.2.1 Extensible metamodels

A weaving metamodel can be expressed as an extension of another weaving metamodel. This allows creating a generic weaving tool with a standard, basic metamodel that may be extended as necessary. Our base metamodel has a minimal set of semantic constructs representing correspondences and links between models and associations between links, but there is no other indication of what these links actually mean. We intend that any other weaving semantics are expressed by other weaving elements extending the base metamodel.

Consider we have a weaving metamodel *WMM*. It contains concrete elements representing the abstract metamodel described above. We have another metamodel *AMM* with an element representing equality (we call it *Equals*). We want to extend the weaving metamodel to be able to represent equality between woven models. We thus say that the element *Equals* extends *WLink*. This way we have a link with equality semantics. The 1-to-many multiplicity from *WLink* to *WLinkEnd* indicates we may have equality between many elements.

We made a first assumption that new metamodels must have an association with at least one element of the base weaving metamodel. It prevents from creating a mal-formed weaving metamodel with two sets of elements without any relation between them.

However the existing weaving metamodel must not change in a way it interferes with existing weaving models or metamodels. For example we may have an element *e1* that is woven with an element *e2* by the means of an equality element *equals*. The equality semantic should not be excluded or modified from the weaving metamodel, otherwise the current weaving becomes invalid.

4. Model weaving tool

After defining the base weaving metamodel we built the AMW weaving tool (it will be available in the GMT project [14]). The AMW tool defined in AMMA reuses part of the infrastructure of the ATL IDE [1] based on the Eclipse Platform. The three notions on which we based the design are metamodel extensions, Eclipse Plugins and the Eclipse EMF platform for models' manipulation. The prototype borrows engineering concepts from the Eclipse Platform: to build a solid base workbench that is extensible to a wide range of applications. The Eclipse architecture is based on contributions: we contribute to the platform with a new plugin (component) and we also define extension points (an entry point for adding new contributions).

Regarding implementation details, AMW is built as a plugin to the Eclipse EMF (Eclipse modeling framework) [9]. It provides an API to access models and metamodels that are based on the Ecore [9] metametamodel. The AMW workbench defines itself different extension points to contribute to the main editor (see Figure 4). The workbench is responsible for controlling the interaction between the different plugged components. The main idea of the implementation is to have a simple user interface of the weaving tool that might be partially generated, without having to build a specific tool for each weaving task or use case.

The woven models extension point enables creating different user interfaces for representing models, for instance three-like panels or a graphical interface with boxes to represent elements and lines to represent links. The only constraint is that the components might implement a previously defined interface to return all model elements in a way they can be accessed by the weaving component.

The weaving panel component handles the actions over the weaving model, such as the implementation of the identification mechanism and creation of weaving elements. It is tightly coupled with the metamodel extension plugins, since it should adapt its interface to handle different weaving metamodels. For this we implemented a standard weaving component using the EMF reflective API to access model elements and to generate user interfaces. It provides standard functionalities for creating and modifying weaving elements, such as a property editor, creation of

compositions and references. This adds flexibility since all new metamodel extensions that do not need extra functionalities can be added without a line of code, just by specifying a metamodel extension file.

The weaving metamodel input format is KM3 [2], as well as for metamodel extensions. KM3 is a language created to facilitate the textual representation of metamodels. For the weaving model and woven models we use XMI [10] (XML Meta Data Interchange), the OMG format for model and metamodels interchange. As output we have a weaving model in XMI format. New input and output formats can be obtained plugging new input and output components. The output weaving model may be used by different transformation engines, such as ATL or XSLT.

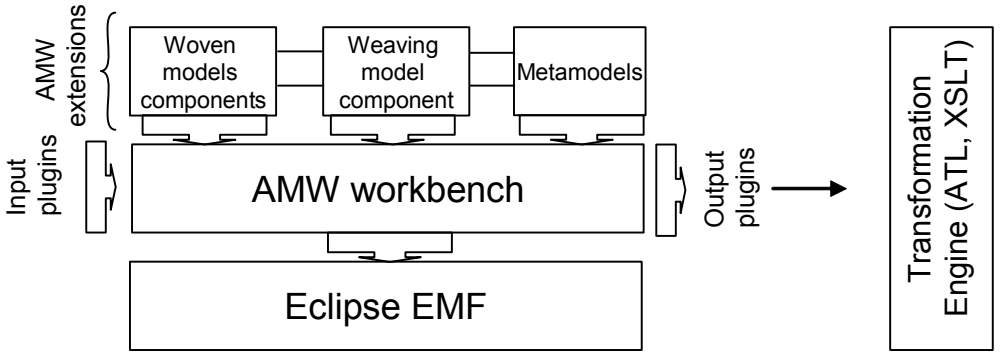

Figure 4: AMW pluggable architecture

The prototype handles the base weaving metamodel described in the previous section and it can be extended incrementally. It is initially composed of three containers: a left metamodel, the weaving metamodel and a right metamodel. The standard functionalities are themselves added as an extension to the base tool, validating our generic approach.

In Figure 5, we illustrate a weaving model in the AMW prototype in a data exchange scenario, where we want to translate data from a relational schema into a XML schema. We have the woven models in the left and right panels and the weaving model in the middle. We want to translate data from *leftModel* to *rightModel*. The weaving model contains an element indicating equality between the *ISBN* attribute from the left panel with the *BookID* attribute from the right panel. Other weaving elements indicating nested semantics are also shown. In the bottom part we may visualize and modify the properties of the selected elements. The created weaving model may be used as a base for generating transformations in different transformation languages, such as ATL or XSLT, to actually perform the data transformation.

5. Related work

Clio [4] generates semi automatic mappings between schemas based on value correspondences. It is focused on database and XML schemas. Our approach allows creating mappings with different kinds of structures as defined in the weaving metamodel. Rondo [7] uses algebraic operators to manage mappings and models. It solves many problems, however the simple mapping language cannot express complicated mappings semantics. Considering weavings as models enables representing much different kind of mappings, though adding complexity while using weavings in a transformation engine. In [15] they propose an extension of the Unified Modeling Language (UML) for expressing mappings between models using diagrams. The main limitation of this approach is the fixed metamodel for mappings the models. In [13] a rich mapping meta-ontology is defined to map between XML DTDs and RDF schemas concentrating on business integration. We have rich mapping representations as well, however our extensible weaving metamodel may be applied to a wider family of problems.

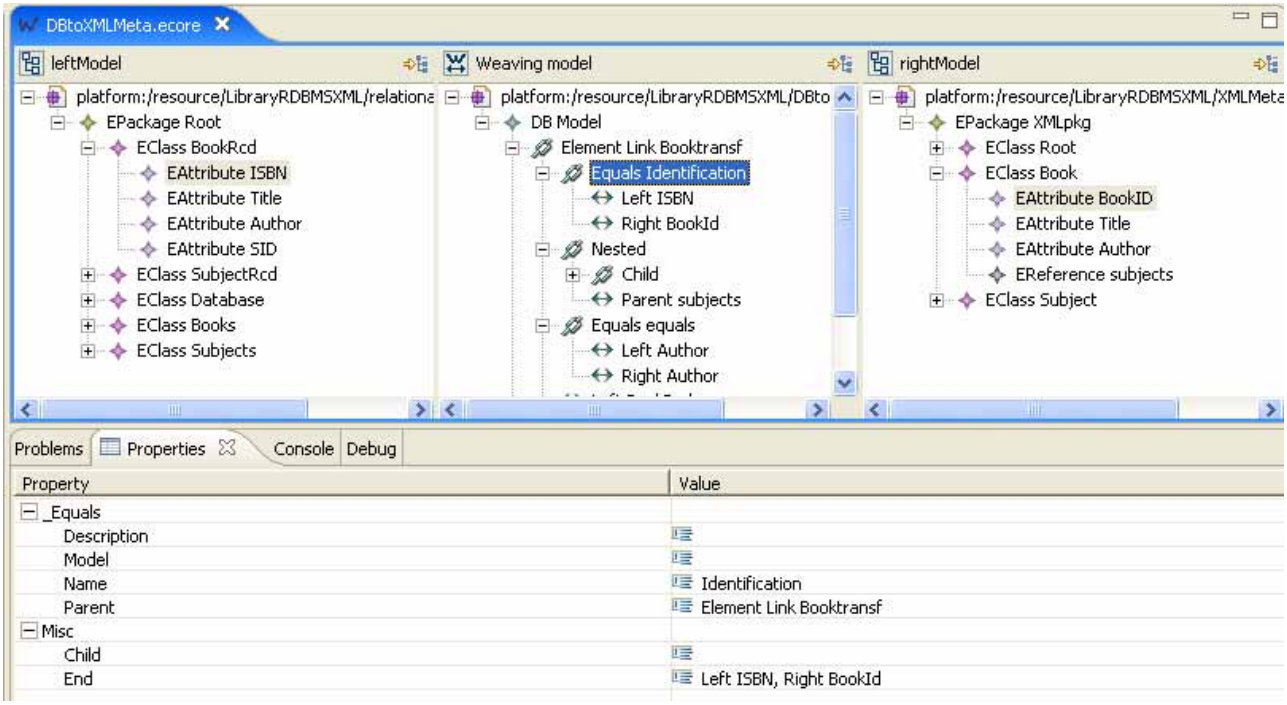

Figure 5. A weaving section in the AMW prototype

6. Future work

A weaving model is now considered as a concrete software artifact that may be stored, retrieved, viewed, extended and more generally handled as any other model. This opens a completely new ways of thinking since we may now consider very abstract situations. We already knew how to handle higher order transformations in ATL, i.e. building transformations that take transformations as input and/or produce transformations as output. Now we can envision weaving two transformation models, transforming a weaving model into another weaving or transformation model or even weaving two weaving models. The special case of transforming the result of a weaving session into an executable ATL program has already been handled.

We realize also that other communities could be inspired by this work to create their own user interface facilities, for example to create aspect weaving tools. There are certainly many similarities between model weaving and aspect weaving as supported in AOP (Aspect Oriented Programming). An investigation is being conducted in this area, based on the current version of the AMW prototype to better understand the commonalities and differences between both approaches.

One interesting property of model engineering is the unification of any concern in a software system, its architecture or its development and maintenance process as a model. In [6] such advanced model-based process organization has been described. Among the basic building blocks of these processes, one may consider automatic model transformations and semi-automatic model weaving. Further more in [6], the interesting case of weaving the SPEM process metamodel with the UML object oriented artifact metamodel.

7. Conclusions

In this paper we have presented a novel approach for handling mappings between models. Several software problems may be abstracted as management of links between two or more representations. The underlying idea here is to consider not only these representations as models, but also the mapping itself as a model. We have pushed this idea as far as possible and we have provided a significant implementation. The corresponding AMW tool is used jointly with the ATL model transformation tool.

We have found this generic approach to have applications in many domains regarding metadata management; we cite for example database schema migration, XML schema integration, and merging a PIM with a PDM to create a platform specific model. Other tools envisioned are model-editors and model-checkers. We are constantly learning from our experimental work and constantly adapting our views on these notions.

We presented our motivations and the design decisions to create a model weaving prototype, as well as the standard weaving metamodel. The implementation as Eclipse plugins allowed a good visibility and extensibility to the prototype. We achieved to have a generic solution for model manipulation, enabling its use by other communities with different application needs.

8. Acknowledgments

We thank Patrick Valduriez for continuous inspiration on this work and Denivaldo Lopes who designed an initial prototype with a fixed metamodel. The current prototype is built as cooperation between the ATLAS group and the Sodifrance Company. Work partially supported by a grant from Microsoft Research, Cambridge, UK, and ModelWare. IST European project 511731.

9. References

[1] Allilaire, F., Idrissi, T. ADT: Eclipse Development Tools for ATL. EWMDA-2, Second European Workshop on Model-Driven Architecture with an Emphasis on Methodologies and Transformations, September 7th-8th 2004, Canterbury, England.

[2] ATL, ATLAS Transformation Language Reference site: http://www.sciences.univ-nantes.fr/lina/atl/

[3] Bernstein, P.A., Levy, A.L., Pottinger, R.A. A Vision for Management of Complex Systems, MSR-TR-2000-53, ftp://ftp.research.microsoft.com/pub/tr/tr-2000-53.pdf

[4] Miller, R. J., Hernandez, M., Haas L, Yan L., Howard Ho, C. T., Fagin, R., Popa, L. The Clio Project: Managing Heterogeneity, SIGMOD Record, Vol. 30, No. 1, March 2001

[5] Bézivin, J., Jouault, F., Rosenthal, P., Valduriez, P. The AMMA platform support for modeling in the large and modeling in the small. LINA research report, November 04

[6] Bézivin, J., Breton, E. Applying the basic principles of model engineering to the field of process engineering. CEPIS, UPGRADE, The European Journal for the Informatics Professional Vol. V, No. 5, 2004

[7] Melnik, S., E. Rahm, P. A. Bernstein, "Rondo: A Programming Platform for Generic Model Management," Proc. SIGMOD 2003, pp. 193-2

[8] Eclipse Modeling Framework, EMF, http://www.eclipse.org/emf

[9] Eclipse Project, http://www.eclipe.org

[10] Object Management Group, http://www.omg.org - 2000

[11] Pottinger, Bernstein, P.A. Merging models Based on Given Correspondences, Proc. 29th VLDB Conference, Berlin, Germany, 2003

[12] Revised submission for MOF 2.0 Query/Views/Transformations - RFP (ad/2002-04-10) – QVT Merge Group - version 1.6 (2004/08/16)

[13] Omelayenko B. RDFT: A Mapping Meta-Ontology for Business Integration, In: Proceedings of the Workshop on Knowledge Transformation for the Semantic Web (KTSW 2002) at the 15-th European Conference on Artificial Intelligence, 23 July, Lyon, France, 2002, p. 76-83

[14] Eclipse GMT project: http://www.eclipse.org/gmt/

[15] Hausmann, J.H., Kent,S. Visualizing Mappings in UML SoftVis'2003, pp.169-178

[16] GreenField J., Short K. with Cook S., Kent S., (foreword by Crupi J.) – Software Factories, Assembling Applications with Patterns, Models, Frameworks and Tools, Wiley Publishing, 2004

[17] Model Driven Architecture, by Richard Soley and the OMG Staff Strategy Group, Object Management Group White Paper, Draft 3.2 - November 27, 2000