First Experiments with a ModelWeaver

Extended Abstract

Jean Bézivin (1), Frédéric Jouault (1), (2) and Patrick Valduriez (1)

(1) ATLAS Group (INRIA & LINA) (2) TNI-Valiosys Company

Abstract

As part of AMMP (ATLAS Model Management Platform), we are currently building several model management tools to support the tasks of modeling in the large and modeling in the small. For the first one (modeling in the large), a companion paper describes our current vision of a "megamodel", i.e. a local or global registry for handling references to models and metamodels and for managing their metadata. Currently the most advanced AMMP tool for modeling in the small is the IDE and basic engine for ATL (ATLAS Transformation language). In the present paper we motivate the need for a different tool called "ModelWeaver". It has the primary objective to handle fine grained relationships between elements of different metamodels. We have undertaken the building and evaluation of a series of model weaving prototypes in the context of the AMMP, in order to progressively define the precise need and the position of this kind of tool in the standard model engineering workbench. This paper is the first recording of our preliminary work in this area of model weaving.

1 Introduction

The basic principle of model as first class entities has not only the advantage of conceptual simplicity. It also leads to clear architecture, efficient implementation, high scalability and good flexibility. As part of several ongoing projects, mainly in Europe and North-America, open source platforms are being built with the intention to provide high interoperability not only between recent model-based tools, but also between and with other legacy tools as well.

Similarly to the difference that was noticed in 1976 by DeRemer and Kron [13] between programming in the large and programming in the small, we are today experimenting with the tasks of modeling in the large and modeling in the small. As part of our local model management platform AMMP (ATLAS Model Management Platform), we are currently building two main set of tools for modeling in the small (model transformation and model weaving) and a set of tools for modeling in the large based on what we call megamodels [12]. The most advanced part of AMMP is an IDE and basic engine for ATL (ATLAS Transformation Language) described in [11] and [1]. The ModelWeaver described in this paper is the latest tool intended to complement the previous ones.

One of the most important operations in MDE is model transformation. Since a transformation program ***Mt*** is considered as a model, it must conform to a given metamodel ***MMt***. The transformation ***let Mt Ma:MMa*** $\rightarrow$ ***Mb:MMb*** produces a model ***Mb*** conforming to metamodel ***MMb*** from a model ***Ma*** conforming to metamodel ***MMa***. As we can see the metamodels act as types.

A transformation is a typed operation, but it has also some invariants, pre conditions and post conditions. All these are expressed in term of the metamodels, of constraints applying to the metamodels or to the models.

As a side effect, a transformation may also produce a trace model of the transformation execution. Since there is no standard trace, this model is also based on some trace metamodel, either generic or specific to a given transformation. There are many different usages for traces (code synchronization, incremental transformations, etc.)

and different tools will deal with these traceability data in our model management platform.
Although a model transformation language is probably the most important tool for programming in the small, it may not be the only one. There are probably some other operations that are not reducible to executable model transformations. We postulate here that one such operation is model weaving, and we present some arguments on the need for such a facility.

2 Model weaving

In order to provide a naive description of the ModelWeaver, let us suppose we have two metamodels ***LeftMM*** and ***RightMM***. We often need to establish links between these. There are many occasions when we need such functionality in a MDE platform as will be discussed later.

The set of links between elements of both metamodels should have several properties:

- The set of links cannot be automatically generated because it is often based on human decisions or heuristics.
- It should be possible to record 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 model ***WM***. Like other models, this should be based on a specific metamodel ***WMM***. The produced weaving model relates to the source and target metamodels *LeftMM* and *RightMM* and thus remains linked to these metamodels in a megamodel registry.
Now an important consideration is that links between the two metamodels should be somewhat typed. There is no unique kind of link. ***WMM*** defines the types of link used in a given weaving operation.
What we have just said could also be applied to the weaving of models instead of metamodels. The generic term of model weaving covers both these situations. This will be detailed in a discussion section of the definitive paper.

3 Motivating examples and Related Work

3.1 Motivating examples

In software engineering practices, the "Y organization" has often been proposed as a methodological guide. The OMG has promoted this idea with the notion of MDA where a Platform Independent Model (PIM) should be combined with a Platform Definition Model (PDM) to produce a merged Platform Specific Model (PSM).

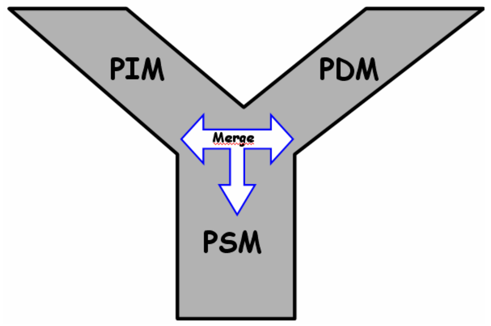


Figure 1 The Y organization and MDA

Let us suppose we have a PIM for a bank containing the class *BankAccountNumber*. Suppose we have a PDM for an implementation platform containing classes *LongInteger* and *String*. One of the most important events in the software development chain is to take design decisions. One such design decision here would be for example to establish that the *BankAccountNumber* should be implemented using a *String* instead of a *LongInteger*. We will not discuss here the validity of this decision because we know that if no concatenation will be performed on such entities neither would addition operations be performed. However what we would like to ensure is 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. It should be possible to undo it, to check it w.r.t. certain criteria, to explain it, etc.
What we see here is that a metamodel for design decisions would be most useful with several properties and links associated to each design decision. We can understand also that it is not very likely that an automatic weaving algorithm could be defined since this is most often a sequence of human decisions based on practical know-how. Of

course the user deciding of the weaving actions should be guided and helped by intelligent assistants that may propose her/him several choices. These helpers may be sometimes based on design patterns.
Let us take another example inspired by the work of Ph. Bernstein [3], [17]. We have two address books to merge and we get both metamodels *LeftMM* and *RightMM*. In *LeftMM* we have the class *Name* and in the second one the classes *FirstName* and *LastName*. Here we need to establish a more complex link stating that these are related by an expression of concatenation.

3.2 Related work

There are plenty of application areas that have touched similar problems for example DBMS schema matching, visual programming, specification of transformation, etc.
As an example of a specification of a transformation between, UML and BPEL4WS, the example of Figure 2 was provided in [10].

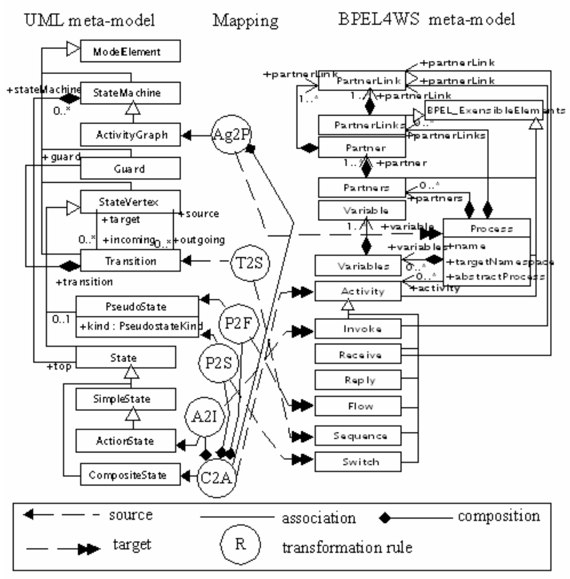
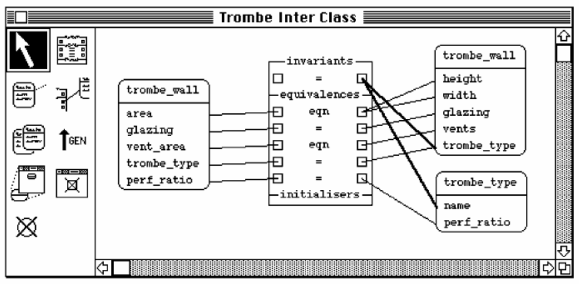


Figure 2UML to BPEL4WS transformation specification

Many examples of model mappings tools are provided in [15], as for example the one illustrated in Figure 3. What is starting to appear is the advantage of treating many different mapping situations with the help of model engineering techniques. The more we look at various different situations, the more we are struck by the possible factorizations resulting from the increase in abstraction.



```
inter_class([trombe_wall],[trombe_wall, trombe_type],
        invariants(trombe_wall.trombe_type = name),
        equivalences(area = height * width,
                     glazing = glazing,
                     vent_area = sum(vents=>(height * width)),
                     trombe_type = trombe_type,
                     perf_ratio = perf_ratio)
                                          ).
```

Figure 3 VML (View Mapping Language)

Another example extracted also from [15] is RVM (RIMU Visual Mapper) as illustrated in Figure 4. We can guess that the basic underlying problems addressed there are very similar to the ones addressed in [10] for example.

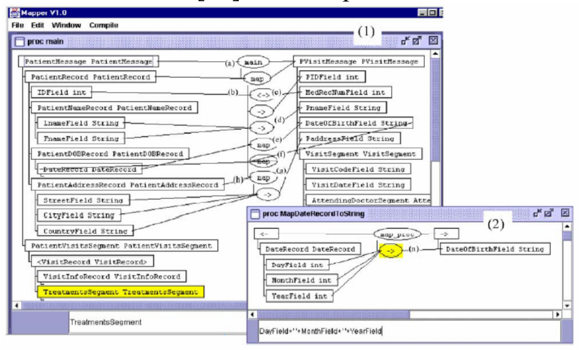


Figure 4 The RVM (RIMU Visual Mapper)

We could not discuss the area of model weaving without strong references to aspect weaving. Many results available from AOP obviously are of relevance to our research.
Finally the domain of ontology mapping and alignment is also presently very active and narrowly related to the problems discussed here.
In the full paper we will extend this survey to these areas and show more clearly how model engineering may allow to take a more abstract and generic view on all these problems of mapping, weaving, alignment, association, etc.

4 Extensible metamodels

From what has been said until now, we may conclude that there is no standard metamodel for weaving operations. Each one should be able to define its own specific one. However, most often a given weaving metamodel will be expressed as an extension of another weaving metamodel.

Defining a clean and powerful metamodel extension mechanism is beyond the scope of this paper. However this is absolutely necessary if we aim to provide powerful model weaving tools. In the first prototypes, we will use an approximation of such an extension operator.

5 First prototype

We provide in this section just a few details on our first implementation on top of the Eclipse platform. More will be provided later in the definitive paper.

The ModelWeaver tool in AMMP reuses part of the infrastructure of the ATL IDE based on the Eclipse Platform [1]. We suppose there is a stub weaving metamodel and this is extended by a specific metamodel extension. The important goal is not to have to build a specific tool for any new weaving specific task. The two notions on which we are basing the design are metamodel extensions and Eclipse Plugins.

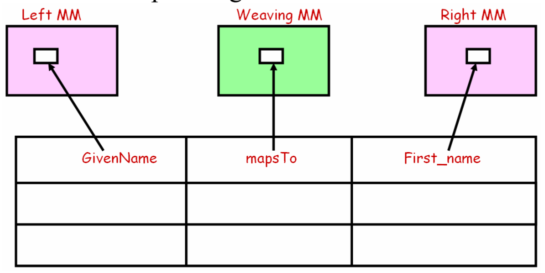


Figure 5 Rough sketch of the Weaver

The main idea of the implementation is that the GUI of the weaving tool is roughly similar to the bottom part of Figure 5. From the left part we can select any class or association of the left metamodel and from the right part we can similarly select any class or association of the right metamodel. In the central part appear all the classes of the weaving metamodel. Selecting a triple thus means creating a weaving link in the resulting weaving model.

Proceeding in this way we get a generic weaving tool, adaptable to any kind of left, right and weaving metamodels. Of course many alternatives design are being explored in the actual building of this tool.

6 Conclusion

We have tried to summarize in this paper some work done in the context of the AMMP project. The three main parts of this platform we are currently developing are a transformation language (ATL), its engine and its IDE, a “model weaving” facility that may be used as a front-end to the ATL model transformer and a global support for megamodels. Several of these tools are intended to be proposed as part of the Eclipse/GMT open source project [14] in 2004/2005. Other tools envisioned are model-editors and model-checkers for example. We are constantly learning from the AMMP experimental work and constantly adapting our views on these notions. The model weaver is an important tool to define individual relations between model and metamodel elements. We are convinced that it is going to take a growing importance in the basic tools populating any model management platform.

Acknowledgements

This work is being done in the context of the **Modelware** 6th European Framework integrated project. We would like to acknowledge the help of many people of the ATLAS team in this work, especially Denivaldo Lopes and Nicolas Dronet.

About the Authors

Patrick Valduriez is director of research at INRIA, heading the newly created ATLAS group at the University of Nantes. Jean Bézivin is professor at the University of Nantes, leading the model management project in the ATLAS group. Frédéric Jouault is a PhD student at the University of Nantes and the main designer of the ATL engine and environment. Frédéric is supported in this work by the TNI-Valiosys company, Brest, France. Authors can be reached at the following address: LINA, Université de Nantes, 2, rue de la Houssinière, 44072 Nantes cedex 03 and by e-mail at {bezivin, jouault, valduriez}@lina.univ-nantes.fr.

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] Bézivin, Breton, Dupé & Valduriez The ATL Transformation-Based Model Management Framework, LINA Technical Report available from http://www.sciences.univ-nantes.fr/lina/vie/rapports/rr2003/rr0308

[5] Bézivin, J. From Object Composition to Model Transformation with the MDA TOOLS'USA 2001, Santa Barbara, August 2001, Volume IEEE publications TOOLS'39.

[6] Bézivin, J. In search of a Basic Principle for Model Driven Engineering, Novatica/Upgrade, Vol. V, N°2, (April 2004), pp. 21-24, http://www.upgrade-cepis.org/issues/2004/2/up5-2Presentation.pdf

[7] Bézivin, J., Gérard, S. Muller, P.A., Rioux, L. MDA Components: Challenges and Opportunities, Metamodelling for MDA, First International Workshop, York, UK, (November 2003), http://www.cs.york.ac.uk/metamodel4mda/onlineProceedingsFinal.pdf

[8] Bézivin, J., Hammoudi, S., Lopes, D., Jouault, F. Applying MDA Approach for Web Service Platform, The 8th International IEEE Enterprise Distributed Object Computing Conference – EDOC 2004, 20-24 September 2004, Monterey, California, USA.

[9] Bézivin, J., Hammoudi, S., Lopes, D., Jouault, F. Applying MDA Approach to B2B Applications: A Road Map, Workshop on Model Driven Development (WMDD 2004), Tuesday, June 15, 2004 ECOOP 2004, Oslo, Norway (June 14–18, 2004).

[10] Bézivin, J., Hammoudi, S., Lopes, D., Jouault, F. B2B Applications, BPEL4WS, Web Services and dotNET in the context of MDA, ICEIMT 2004, Toronto, Canada, October 2004.

[11] Bézivin, J., Jouault, F., Valduriez, P. An Eclipse-Based IDE for the ATL Model Transformation Language. Submitted for publication, OOPSLA Workshop, 2004.

[12] Bézivin, J., Jouault, F., Valduriez, P. On the Need for Megamodels. Submitted for publication, OOPSLA Workshop, 2004.

[13] Deremer, F. and Kron, H. Programming in the Large versus Programming in the Small, IEEE Trans. On Software Eng. June 1976 http://portal.acm.org/citation.cfm?id=390016.808431

[14] GMT, General Model Transformer Eclipse Project, http://www.eclipse.org/gmt/

[15] Grundy, J.C., Hosting, J.G., Amor, R.W., Mugridge, W.B., Li, Y. Domain-Specific Visual Languages for Specifying and Generating Data Mapping Systems. Journal of Visual Languages and Computing, 15(2004), 207-209.

[16] Popa, L., Velegrakis, Y., Miller, R.J., Hernandez, M.A., Fagin, R. Translating Web Data, VLDB'02.

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

[18] Soley, R. & the OMG staff MDA, Model-Driven Architecture, (November 2000), http://www.omg.org/mda/presentations.htm