

Missing Meta-Modeling-In-The-Large Concepts in EMOF

Carsten Amelunxen, Andy Schürr and Ingo Weisemöller

Darmstadt University of Technology, Real-Time Systems Lab,
Merckstrasse 25, 64283 Darmstadt, Germany
[amelunxen|schuerr|weisemoeller]@es.tu-darmstadt.de
<http://www.es.tu-darmstadt.de>

Abstract. Nowadays, a typical software development process involves a great number of developers which participate in the development process by using a wide variety of development tools. As a consequence, the data representing the project as a whole is separated over the different development tools. For the purpose of consistency, maintainability and traceability it is an essential task to be aware of the relationships between semantic equivalent data in different tool repositories. The Real-Time Systems Lab and the Darmstadt University of Technology performs research in the area of tool and metamodel integration to provide solutions to overcome this gap. We will explain why EMOF is not the right choice for meta-modeling-in-the-large activities as sketched above and show why we urgently need features provided by CMOF for that purpose. Furthermore, we will list some properties for a true meta-modeling-in-the-large language that even go beyond the capabilities of CMOF (and UML).

1 Introduction

The integration of development tools, or rather of the metamodels which are implemented by the development tools, requires a metamodeling language which provides sufficient features for clearly structured and well formed metamodels. Since, known from experience, metamodels of development tools are usually quite big and complex, only a metamodeling language which is able to deal with large and complex specifications is suitable for the purpose of metamodel integration. In other words, the potential of an integration approach is directly dependent on the used metamodeling language's capacity.

The most promising approach concerning the size of specifications is provided by MOF 2.0 [Obj06], or, to be more precise, by the most sophisticated part of MOF 2.0 which is denoted as CMOF. Thus, the integration approach which is developed at the Real-Time Systems Lab [KS06] [KS05] is conceptually and technically [AKRS06] [MOF] based on CMOF. In the following we will present a short example of metamodel integration and briefly present important meta-modeling features before we come up with a short conclusion. We concentrate on conceptual issues rather than on implementation issues (like concurrent access, configuration control etc.) which we gained from several industrial projects [SDG⁺07], [A⁺03].

2 Integration of metamodels

Figure 1 shows an example from the area of automotive systems development processes. It is limited to the application of the requirements engineering tool Doors [Doo07], the system design and simulation tool Matlab/Simulink [Sim07] and a standard UML tool, though the number of tools used in a development process of automotive systems is typically larger. The improvement of automotive systems development processes by means of tool integration is currently addressed not only by research organizations but also by industrial partnerships [Aut06].

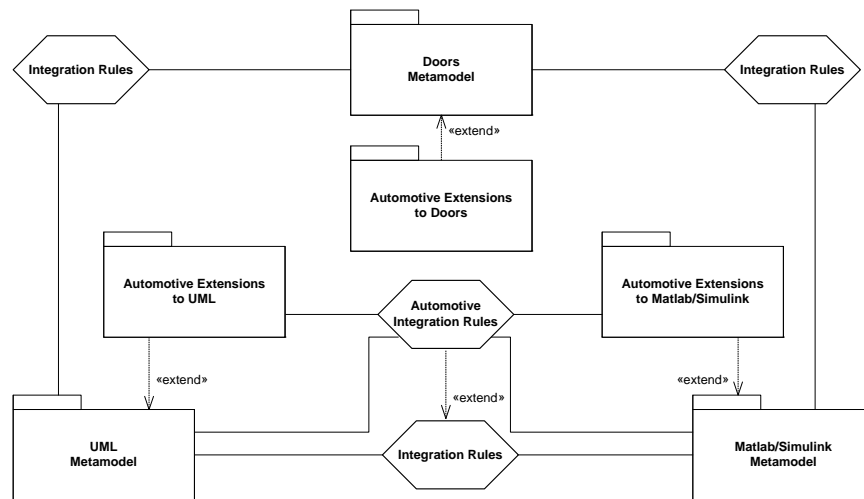


Fig. 1. Example of integrated metamodels

For each tool, besides the metamodel as provided by the tool developer there exists an extension of that metamodel specific to the development of automotive systems. A variety of integration rules is required to keep semantic equivalent data represented in multiple tools consistent. Again, these integration rules can be subdivided into those which are applicable for all users of the tools listed above and those which are specific to the development of automotive systems. For the latter we can further distinguish between rules that apply to the standard metamodels of the tools and those that deal with domain specific extensions.

Both the basic metamodels provided by the tool developers as well as the basic integration rules need to be refined to be appropriate for the domain of automotive system development. In the following, we point out a set of impacts these refinements and the size of the metamodels have on the requirements to metamodeling and integration tools. Most of these requirements have been implemented in the MOFLON Tool Suite [AKRS06], an implementation of MOF

2.0, including an implementation of MOF QVT [Obj05] based on Triple Graph Grammars [KS05] for model-to-model transformations. However, the semantics of the concepts introduced by MOF to support metamodeling-in-the-large are defined in an unprecise manner in certain aspects, which makes the implementation of a MOF-compliant tool rather hard and needs to be improved in future versions of the specification.

2.1 Modularization

To provide appropriate support for the definition of large metamodels and their integration, a modularization concept in both, the metamodeling and the integration language, is required, since the number of integration rules grows with the size of the metamodels. To keep metamodels of that size manageable, a hierarchical concept for namespaces is required. A global namespace (as in EMOF) prevents the designers of the metamodel and the integration rules from introducing a clear structure and results in problems concerning overview and maintainability.

From the introduction of namespaces follows the necessity to provide a concept to access elements from different namespaces, which is required to be able to reuse and refine these elements. As a matter of convenience, the import of single elements as well as of all elements from a different namespace should be supported. The possibility to reuse parts of the metamodel described by other domain experts is not only essential for the definition of clearly structured and understandable metamodels. It also allows developers to share knowledge specific to their area of expertise with each other. Reusability of components has proven to be of value in the design of large software systems for years, and it should be self-evident when designing large metamodels as well.

2.2 Relations

One of the absolute essential features for the purpose of metamodel integration is an advanced relation concept. Since, the integrated metamodels must not be affected in any kind, it must be possible to model a relationship between elements of different metamodels without modifying the involved classes. It is essential that the involved metamodels on the one hand and the integration construct on the other hand can be separately compiled¹. If the integration causes changes in the integrated metamodels, a separated compilation would be impossible. For this demand, the concept of a reference consisting of two mutual referencing class attributes simply fails.

In fact, associations as first class modeling constructs are indispensable to allow an integration which does not enforce the modification of the integrated metamodels. The association concept applied in CMOF introduces associations as packagable classifiers next to classes. This concept, which treats associations

¹ In an ideal world, the compiled tool metamodels are provided by the tool manufacturer and, therefore, have to be treated as read only.

as relations rather than mutual pointers, offers the possibility to maintain association instances independent from their involved class instances and thus, in turn, allows to implement the integration issues separately from the integrated metamodels. Finally, a metamodeling language which is suitable for integration has to provide real associations instead of simple references.

2.3 Refinement

For the extension of metamodels and integration rules as shown in Figure 1, a concept allowing to reuse and refine existing elements in the metamodel must be present in a metamodeling language. These refinements must be possible with respect to the immutability of the metamodels provided by tool developers. The necessity to introduce refinements does not only apply to classes, but also to associations and entire namespaces, which need to be introduced as stated in the previous sections. Besides the generalization of classes, CMOF therefore introduces subtyped and redefined association ends and package merges. These concepts also allow to define several views of the same element in different namespaces, each of which may abstract from certain aspects and introduce new features as required.

2.4 Implementation

One last aspect, which is indeed a matter of implementation, is the subject of qualified associations which offer the possibility to query related objects based on attribute values. For an efficient implementation of a model repository, queries based on attribute values are extremely helpful. We had to gain this experience during the work on our metamodeling framework MOFLON. Thus, we would like to emphasize the relevance of qualified associations although they are dispensable from a conceptual point of view. In this case the practical point of view prevails, which triggered us to introduce qualified associations in MOFLON although they are neglected in CMOF as well.

3 Conclusion

Our mentioned integration approach highly benefits from the presented features of CMOF. In fact, they are an essential part. Although, CMOF meets our demands, we also notice that in the meantime EMF is the more prominent alternative. Since, we cannot abandon the powerful features of CMOF but are also not averse to a more popular and well known base technologie, we keep an eye on the discussion about the future of EMF and are curious in which way EMF is evolving. The main issues which should be taken into account from our point of view are stated throughout this position paper.

References

- [A⁺03] F. Altheide et al. An Architecture for a Sustainable Tool Integration. In Dörr and Schürr, editors, *TIS 2003 Workshop on Tool Integration in System Development*, pages 29–32, 2003.
- [AKRS06] C. Amelunxen, A. Königs, T. Röttschke, and A. Schürr. MOFLON: A Standard-Compliant Metamodeling Framework with Graph Transformations. In A. Rensink and J. Warmer, editors, *Model Driven Architecture - Foundations and Applications: Second European Conference*, volume 4066 of *Lecture Notes in Computer Science (LNCS)*, pages 361–375, Heidelberg, 2006. Springer Verlag.
- [Aut06] AUTOSAR Enabling Technology for Advanced Automotive Electronics, 10 2006. http://www.autosar.org/download/AUTOSAR_long_en.pdf.
- [Doo07] Telelogic DOORS - Requirements Management for Advanced Systems and Software, 2007. <http://www.telelogic.com/Products/doors/doors/index.cfm>.
- [KS05] A. Königs and A. Schürr. Multi-Domain Integration with MOF and Extended Triple Graph Grammars. In J. Bezivin and R. Heckel, editors, *Language Engineering for Model-Driven Software Development*, number 04101 in Dagstuhl Seminar Proceedings. IBFI, Schloss Dagstuhl, Germany, 2005. <http://drops.dagstuhl.de/opus/volltexte/2005/22>.
- [KS06] A. Königs and A. Schürr. MDI - a Rule-Based Multi-Document and Tool Integration Approach. *Special Section on Model-based Tool Integration in Journal of Software and System Modeling*, 5(4):349–368, December 2006.
- [MOF] MOFLON Homepage. <http://www.moflon.org>.
- [Obj05] Object Management Group. *MOF QVT Final Adopted Specification*, November 2005.
- [Obj06] Object Management Group. *Meta Object Facility (MOF) 2.0 Core Specification*, January 2006. formal/06-01-01.
- [SDG⁺07] I. Stürmer, H. Dörr, H. Giese, U. Kelter, A. Schürr, and A. Zündorf. Das MATE Projekt visuelle Spezifikation von MATLAB Simulink/Stateflow Analysen und Transformationen. Dagstuhl Seminar Modellbasierte Entwicklung eingebetteter Systeme, Januar 2007.
- [Sim07] Simulink - Simulation and Model-Based Design, 1994–2007. <http://www.mathworks.com/products/simulink/>.