Model-Based Tool Integration - State of the Art and Future Perspectives[1]

Elisabeth Kapsammer[2], Thomas Reiter[2], Wieland Schwinger[3]

[2]Information Systems Group (IFS)
Johannes Kepler University Linz, AUSTRIA

[3]Department of Telecooperation
Johannes Kepler University Linz, AUSTRIA

Abstract

The quality and effectiveness of software development heavily depends on the underlying tools used for different phases of the development lifecycle. With the rise of model-driven development, a proper integration of modeling tools represents a crucial success factor. At the same time, the usage of models as the major artifact in software development allows a new form of interoperability in terms of model-based tool integration. Model-based tool integration focuses on the integration between the tool's languages metamodels, providing a proper basis for a later (semi-)automatic model integration step. The contribution of this paper is to investigate the current state of the art in model-based tool integration, to identify and elaborate on promising concepts and technologies and to point the way to next-generation tool integration solutions, based on semantic technologies.

Keywords: Tool Integration, Model, Metamodel, Semantic Integration, Ontologies

1. Introduction

Effective software development processes resulting in high-quality software products require proper tool support for different phases of the software lifecycle. The model-driven software development paradigm as, e.g., propagated by the OMG with its Model-Driven Architecture (MDA) places models instead of pure code in the center of the development process, being used for different kinds of development tasks. For example, one could use a special purpose modeling tool for requirements definition, another one for UML modeling and a third one for test case generation and simulation thereof. Employing a series of special-purpose modeling tools is beneficial, on the one hand since it avoids the dependency from a huge general purpose tool and its vendor and on the other hand since it allows to employ those tools best suited for the special modeling task at hand. One major pre-requisite is, however, that these tools are properly integrated, allowing the seamless exchange of models in-between.

What we are looking for is *model-based tool integration*, enabling to facilitate any tool appropriate for the modeling task at hand (cf. [32], [33], [34]). Model-based tool integration means that tools are integrated on basis of metamodels defining syntax and semantics of the modeling languages supported by the tools. This would allow to integrate the metamodels once and to apply this integration solution to any models subsequently, conforming to these so-called *tool metamodels*. Thus, the repetitive effort of ad-hoc model-integration, which is a cumbersome and error-prone task, could be avoided.

To provide a comprehensive solution for model-based tool integration, three crucial requirements have to be fulfilled (cf. also Fig. 1). First, several integration scenarios should be supported, comprising, e.g., translation, modularization, alignment, and merge of models. Second, it should be possible to integrate models based on arbitrary metamodels, having only a certain meta-metamodel in common, such as MOF (Meta Object Facility) [29]. Third, and probably most important, transformations between concrete models, which realize the integration scenarios should be (semi-)automatically derived from the integrations defined between the corresponding metamodels. Although still in its infancy, there are already a few approaches dealing with model-based tool integration, fulfilling more or less one or the other requirement (cf., e.g., [35]).

The goal of this paper is to investigate the current state of the art in model-based tool integration, to identify and elaborate on promising concepts, approaches, and technologies and to point the way to next-generation tool integration solutions, based on semantic technologies. In Section 2, tool integration is considered from a historical point of view. Section 3 discusses approaches most relevant for the purpose of model transformation in the context of model-driven development. Section 4 investigates promising approaches for the purpose of semantic integration, especially focusing on ontologies. Section 5, finally, points to future research by briefly reporting on a model-based tool integration approach

[1] This work has been partly funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) under grant FIT-IT-810806.

called ModelCVS, which we are currently realizing in the course of an industrial project.

2. History of Tool Integration

Research in tool integration goes back to the so-called Stoneman Model[2] which was proposed at the end of the 70's and summarized by Brown [6] in two categories, the conceptual level ("what is integration?") and the mechanical level ("how do we provide integration?").

Conceptual level of integration. Concerning conceptual integration, Wasserman [41] has suggested a categorization to describe the integration of tools from a *functional point of view* comprising integration in terms of *platforms*, *GUIs*, *data*, *control*, and *processes*. Other categorizations used for characterizing tool integration comprises *depth of integration*, varying from exchanging byte streams to semantics-preserving integration, and the *universal applicability* of the integration approach. Commercial of the shelf (COTS) tools, for example, are meant to be integrated if they function coherently and effectively in an environment as a whole, as is the case in an integrated development environment (IDE).

Mechanical level of integration. The research efforts at the mechanical level of tool integration (cf., e.g., [22] for an overview) include (1) a series of *standardization efforts* and *middleware services* like CAIS [27], PCTE [1], CDIF [14], CORBA[3], and OMG's recent RFP OTIF[4] (Open Tool Integration Framework) to support tool interoperability, (2) *architecture models*, *infrastructures*, and *tool suites* like the ECMA toaster model [11], the ToolBus architecture [3], and finally (3) basic *tool integration mechanisms* such as data sharing, data linkage, data interchange, and message passing [35].

Some of these efforts were often grounded in large initiatives but have not been widely accepted. The European standardization effort PCTE, e.g., supporting data integration with a common repository was not widely adopted in industry, not least because of its heavyweight architecture and high usage costs. Another example, CDIF, a standard for model exchange has been in the meanwhile replaced by MOF. Regarding, e.g., tool suites, they are often incomplete with respect to the various development activities requiring tool support, and most often do not allow to select between "best of class" tools (apart from promising exceptions like Eclipse[5]) [35].

Despite of all these important efforts, tool integration is still a challenging task, leading most often to hand-crafted bilateral integration solutions [35]. These "solutions" suffer from high maintenance overheads not least in case of evolutions of the underlying data or tools themselves, are often strongly technology-dependent and, most importantly, *do not scale*. With the advent of *model-driven development (MDD)* and in particular the introduction of OMG's *model-driven architecture* (*MDA)* [28] new possibilities have been opened up to cope with these challenges, as described in the next Section.

[2] http://www.adahome.com/History/Stoneman/stoneint.htm
[3] http://www.omg.org/corba
[4] http://www.omg.org/docs/mic/04-08-01.pdf
[5] http://www.eclipse.org

3. MDA and Model Transformations

The key idea of MDA is to focus on models instead of code as the major artefact in software development. This allows modeling tools to be integrated on basis of the metamodels of modeling languages supported by the tools (i.e., the tool metamodels), thus paving the way for another generation of *(meta)model-based tool integration approaches* and providing a basis to overcome the above mentioned limitations of existing integration approaches. For this, MDA includes a set of interrelated standards[6], comprising a language for metamodel definition (*Meta Object Facility – MOF*), and the MOF-compliant languages for constraint specification (*Object Constraint Language – OCL*), metadata interchange (*XML Metadata Interchange – XMI*), and last but most important model transformation (*QVT*).

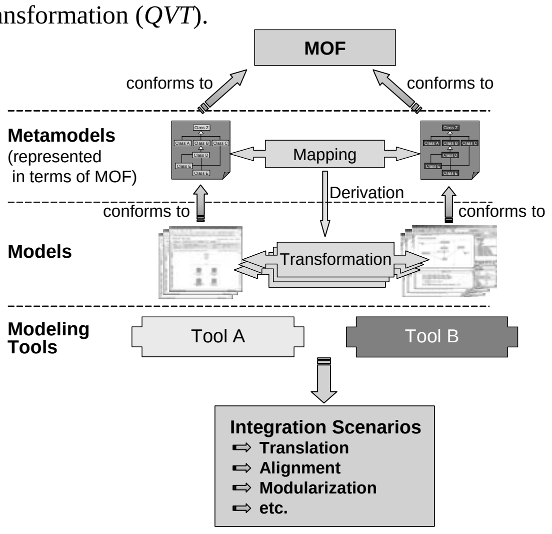

Figure 1. Model-based Tool Integration – The Big Picture

Model Transformation Languages

Model transformation is one of the major building blocks in the context of model-based tool integration and a very active research area. Existing approaches in this area having been either submitted to OMG's QVT request for proposals or being already part of existing MDA tools range from *algorithmic* and *imperative approaches*, via *graph-transformation-based approaches* to *template rule-driven*, and *hybrid approaches* [9]. *Tratt et al.* [38], e.g., provide an extensible, imperative model transformation language with some rule-based elements for pattern matching purposes, whereas *Becker et al.* [2]

[6]http://www.omg.org/technology/documents/modeling_spec_catalog.htm

use purely rule-based mechanisms employing graph-transformations and generates wrapper for tool integration following a kind of programming-by-example approach. Transformation languages such as BOTL[7], allow the definition of modular, rule-based transformations, with independent rules for sets of metamodel elements. For a more detailed overview of such transformation languages cf. [9].

Infrastructures based on Transformation Languages

Based on these several kinds of QVT-like transformation language proposals, infrastructures and frameworks have been built for tool integration [35]. For example, *WOTIF (Web-based open tool integration framework)*[8] uses a graph-transformation mechanism and realises different tool integration scenarios (e.g., direct tool integration and integration via a common metamodel), but requires that every tool to be integrated supports certain APIs for installing plugins. *GeneralStore* [31] being in fact a MOF-based repository, allows bi-directional transformations between models, but uses XSLT or ad-hoc approaches for model transformation, only. Finally, *MDDi, (Model-driven Development Integration Project of Eclipse*)[9], although providing some interesting ideas for model integration in terms of a bus architecture similar to AMMA (cf. below), is still in its draft proposal phase.

Deficiencies of pure model transformations. Although QVT-like model transformation languages are a core technology for model-based tool integration, existing proposals are too generic and lack appropriate abstraction mechanisms for different kinds of *model integration scenarios*, which are highly needed in practice and well-known from other research areas such as *federated and multi database systems* [36]. Such integration scenarios would require a series of basic model transformations which will simply not scale up when manually specified for complex models.

Abstraction Mechanisms for Model Integration

There are only few approaches providing abstraction mechanisms in terms of, e.g., high-level integration operators or modularisation techniques in the areas of *model management and model integration* as well as in the area of *aspect-oriented modeling* which are described in the following in more detail.

Rondo. The *generic model management initiative* from Bernstein et al. [25] provides a prototypical implementation called *Rondo*, which aims at keeping the matching of large XML schemata scalable. An approach to matching is introduced that operates on fragments of a large schema to lower the complexity of matching tasks. Besides this modularisation, model management operators on relational and XML schemata are provided, comprising, e.g., the automatic derivation of semantic correspondences or differences, the merging of models, and the derivation of a mapping from other mappings.

Although set in the context of relational and XML schema matching, this idea seems to be transferable to tool metamodels. Model-based tool integration, however, is not only aimed at finding semantic correspondences between metadata, but also to support certain model integration scenarios, keeping a later code-generation step in mind in terms of deriving appropriate model transformation programs thereof. Another problem is that the focus of model-based tool integration should go beyond integrating XML and database schemata, by allowing the integration of arbitrary MOF-models in the sense of MDA.

AMMA / AMW. The *ATLAS Model Weaver (AMW)* which is part of the *AMMA model engineering platform* (soon to be released under the *Eclipse GMT* project[10]) proposed by Bézivin et al. [4], allows to perform a weaving operation in terms of establishing semantic correspondences between two metamodels, which are stored in a weaving model. Model weaving seems to be – different to Rondo – a manual operation, requiring an explicit specification of appropriate semantics for correspondences.

Aspect-orientated Approaches for Model Integration

The research efforts associated with *aspect-orientation* also deal with modularization in terms of factoring out cross cutting concerns into modules called *aspects*. This idea manifests in *aspect-oriented programming languages* [21], but also in *aspect-oriented modeling*, which allow to modularize cross-cutting-concerns in an implementation independent manner (cf. the approaches below).

Model-based tool integration focuses on tool integration, meaning that metamodels are, e.g., decomposed according to certain concerns they cover. *Weaving* as in aspect-orientation can be compared to the re-assemblage of models after modularization. In a tool integration setting, one can assume modularization to take place by detecting join points, e.g., in the form of meta-associations and point-cuts, e.g., in the form of links between model elements, to offer automatic support for a future re-assemblage. Most of the following approaches more or less use ideas from aspect-orientation for model integration purposes.

Model Composition Semantics. Clarke [8] introduces a *composition mechanism* for UML class diagrams, representing different separated concerns. Overlapping concepts are identified in these models and thus merged as specified by a composition relationship, following so-called *merge* and *override* strategies. Based on this basic integration behavior, *composition patterns* are introduced as an extension to UML templates.

[7] http://www4.in.tum.de/~marschal/botl/
[8] http://escher.isis.vanderbilt.edu/tools/get_tool?WOTIF
[9] http://www.eclipse.org/proposals/eclipse-mddi/
[10] http://www.eclipse.org/gmt/

This approach focuses on UML models, only, and does not allow, e.g., the deletion of obsolete model elements after an integration is performed, as required for model-based tool integration. In addition, model-based tool integration should focus on the derivation of model transformation programs during the integration stage, which are capable of automatically performing, e.g., the merging of models.

Model Composition Directives. Based on [8], Straw et al. [37] propose so called *composition directives* for composing UML class diagrams. These basically include name rewriting, adding and deleting of model elements, change of references, and control of execution order. Inspired by aspect-oriented programming, so-called primary models are composed with aspect models, which represent a crosscutting concern to be interwoven.

The primary focus of this approach seems to be on model weaving but not on meta-model weaving as required for model-based tool integration.

GME. The *Generic Modeling Environment (GME)* proposed by Karsai et al. [20] is a modeling and metamodeling toolkit based on UML notation and a GME specific meta metamodel. GME allows for the composition of metamodels. The composition mechanisms comprise an *equivalence operator* creating a union of two model elements, similar to the *merge* semantics in [8] and two different inheritance operators, realizing implementation inheritance and interface inheritance.

Unfortunately, GME is not based on the MOF standard. Furthermore, GME supports metamodel composition, only, neglecting further model integration scenarios.

C-SAW. C-SAW, developed as a plug-in for GME by Gray et al. [16] is a so called cross-cutting-concern weaver. Aspects are specified using the Embedded Constraint Language (ECL), a OCL superset, additionally providing imperative constructs for model manipulation.

The transformation capabilities of ECL are, however, limited to models of the same metamodel, it lacks support for abstract integration mechanisms and is, instead of MOF, based on a meta-metamodel specific to GME.

Domain Composition Approach. Estublier et al. [12] propose a *UML profile* allowing the composition of separately designed domain models, as required when facing the federation of immutable components off the shelf. UML associations and association classes are specialized by stereotypes to express feature correspondence and concept overlapping.

In principle, this approach supports an alignment integration scenario, but does not support other integration scenarios. In addition, only UML models are supported instead of arbitrary MOF-models.

Summarizing (cf. Table 1), although there are already few approaches targeting model-based tool integration from a meta-modeling point of view and providing some basic abstraction mechanisms each of them suffers from certain deficiencies with respect to the focus of model-based tool integration as outlined above.

	MOF-based	Meta-level Integration	Model-level Integration	Automatic Integration	Different Scenarios
Rondo	✗	✗	✓	✗	✗
AMMA	✓	✓	✓	~	✓
Composition Semantics	✗	✗	✓	✗	✓
Composition Directives	✗	✗	✓	✗	✓
GME	✗	~	✓	~	✗
C-SAW	✗	✓	✓	~	~
Domain Composition	✗	✗	~	~	✗

Legend: ✓ … supported ✗ … not supported ~ … not applicable

Table 1. Comparison of Integration Approaches

Nevertheless, several ideas and concepts of these approaches could be of high value for model-based tool integration.

4. SEMANTIC INTEGRATION AND ONTOLOGIES

Besides the support of model transformations and appropriate abstraction mechanisms as discussed in the previous section, semantic integration, i.e., the mediation between semantic heterogeneities constitutes another crucial challenge for model-based tool integration. The history of semantic integration goes back to the early 1980s, where Brodie et al. [5] addressed semantic relativism in data modeling, leading to a comprehensive taxonomy of semantic heterogeneities introduced by Shet et al. [36] in the early 1990s and an in-depth survey of automatic schema matching approaches in 2001, published by Rahm et al. [30]. Although the problem of semantic integration is tackled in various ways by different communities, as could be seen at the remarkable Dagstuhl workshop on semantic interoperability and integration in 2004[11], in recent years, *ontologies* became very popular to facilitate various semantic integration tasks. This is not least since, in comparison to other techniques, integration based on ontologies can rely heavily on the high expressive power of ontology languages and on appropriate reasoning techniques. In this respect related work in the area of *lifting metadata to ontologies*, *issues of integrating ontologies*, and the usage of *integration scenarios for ontologies* is highly relevant for model-based tool integration, as discussed in the following (cf. Fig. 2).

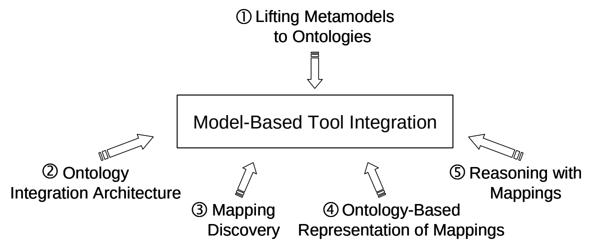
① Lifting Metamodels to Ontologies
Model-Based Tool Integration
② Ontology Integration Architecture
③ Mapping Discovery
④ Ontology-Based Representation of Mappings
⑤ Reasoning with Mappings

Figure 2. Semantic Integration Technologies

[11] http://www.dagstuhl.de/04391/

Lifting Metamodels to Ontologies

A basic question to be investigated is the *derivation of ontologies* from the tool metamodels, often referred to as *lifting*. Few existing work, although approaching the lifting problem from somewhat different angles, could be used as starting point to resolve this research question.

OntoLIFT. Lifting is, e.g., dealt with in the WonderWeb project in terms of the OntoLIFT prototype [40], which helps to semi-automatically create ontologies from database schemata by using syntactical patterns as employed for mapping database schemata to ER models. Although these ontologies have to be further refined to infer specific semantics, OntoLIFT provides a useful entry point for the establishment of ontologies.

Ferdinand et al. Another approach from Ferdinand at al. [13] proposes an automatic mechanism to lift XML Schema to the *Web Ontology Language (OWL)* via RDF and provide according mapping rules.

Although both approaches deal with the derivation of ontologies from structured sources, methods applied in these two approaches cannot be immediately reused, since model-based tool integration requires the derivation of ontologies from metamodels. Further research has to be put into the question of how to facilitate the creation of ontologies from MOF-based metamodels.

ODM. A way to bridge between model engineering and ontology engineering could be the *Ontology Definition Metamodel (ODM)*[12], an upcoming OMG standard for the definition of ontologies in terms of MOF models.

Guizzardi et al. [17] provide an evaluation framework to estimate the appropriateness and the comprehensibility of a modeling language for describing concepts in terms of domain knowledge captured in an ontology. Such considerations are relevant in the context of model-based tool integration to define ontologies for modeling languages or to estimate to what extent existing ontologies can be reused.

Integrating Tool Ontologies

As model-based tool integration is able to perform tool metamodel integration on basis of semantics covered by tool ontologies, these individual tool ontologies have to be integrated. One has to deal with different forms of *heterogeneity*, establish a certain *ontology integration architecture*, and provide appropriate mechanisms for *mapping discovery*, *representation* and *reasoning* [26]. Although having different goals in mind since in model-based tool integration, ontologies can be used as a basic vehicle for the integration of tool metamodels, we can benefit from a large body of literature which may provide useful input. For a comprehensive overview about this active research area compare, e.g., [19], and [26].

Ontology integration architecture. Concerning the architecture for ontology integration, one can basically distinguish three alternatives (cf. e.g., [26]): (1) a *direct mapping* between ontologies, (2) an *indirect mapping* via a *common, shared ontology* further on called *upper ontology* (sometimes also referred to as *toplevel, or reference ontology*), e.g., DOLCE [15] and (3) a mapping based on a *library of already mapped ontologies* [39]. This is again similar to database integration research, where peer-to-peer database systems are similar to the direct mapping approach, and federated database systems relying on a global schema are similar to the indirect mapping approach with the difference that an upper ontology is usually more general since it needs to encompass the top level for ontologies yet to be developed [26]. For model-based tool integration, a *hybrid approach*, involving all three architectures seems to be most beneficial.

Mapping Discovery

Based on a certain ontology integration architecture, *mappings between ontologies* have to be established, i.e., similar concepts have to be related to each other. *Mapping discovery techniques* deal with finding such *correspondences* (also called *matches*) between ontologies. This can be done either in a *fully manual* way or by utilizing *heuristic-based or machine learning techniques* that use various characteristics of ontologies, such as their schemata (*schema-based matching*), their instances (*instance-based matching*) as well as *lexical reference systems* [30], [26]. It has to be emphasized, that for the purpose of model-based tool integration, it seems not to be necessary to develop yet another mapping discovery technique. A selection of some of these approaches which may be useful for model-based tool integration are sketched out in the following.

Chimaera. Chimaera [24] provides support for *ontology merging by interactively relating concepts* that are identical or related by subsumption or instance relationships. Further, it supports to *manipulate the ontologies* as to improve alignment by suggesting modifications.

PROMPT. PROMPT [26] supports *interactive, guided ontology merging*, starting from *linguistic* and *structural similarity* matches. Merge operations can be performed, and based on the results and potential conflicts arising from the merge (e.g., name conflicts, dangling references, or redundancies in class hierarchies), further operations are proposed.

PUZZLE. The goal of PUZZLE [18] is to construct a *consensus* ontology, i.e., a common, shared ontology, from independently designed ontologies. Both, *linguistic* as well as *contextual features* of ontology concepts are considered, there is no need for a previous agreement on the semantics of the used terminology and WordNet is used to support, e.g., synonyms and homonyms. Reasoning rules are based on the relationships *subclass*, *superclass*, *equivalentclass*, and *sibling*, and on property lists of ontology concepts to find new relationships among concepts.

[12] http://www.omg.org/cgi-bin/doc?ad/2003-03-40

Representation of and Reasoning with Mappings

Having found appropriate mappings, they have to be properly represented in order to facilitate reasoning on mappings. Concerning the representation of mappings, several approaches can be found in literature [26]. Note that also combinations thereof are possible. First, similar to traditional data integration, *views* can be used to describe mappings, e.g., between upper ontology and local ontologies, either using the *global-as-view (GAV)* or the *local-as-view (LAV)* approach, well known from database integration research [30]. Second, mappings can be represented in terms of *bridging axioms* in first-order logic to express transformation rules, relating classes and properties of two ontologies, as it is done in the OntoMerge system [10]. Finally, mappings can be represented as instances in an *ontology of mappings*. The mapping ontology usually provides different ways of linking concepts from the source ontology to the target ontology, transformation rules to specify how values should be changed, and conditions and effects of such rules. An example is the *Semantic Bridge Ontology* of the MAFRA framework [23].

For the purpose of model-based tool integration, it can be concluded that specific mapping ontologies seem to provide a great potential for mapping representation as well as reasoning.

Finally, reasoning aims at drawing a conclusion, e.g. to perform semantic integration tasks. For model-based tool integration, reasoning over ontology mappings is required to facilitate metamodel integration. Reasoning hardly depends on the underlying representation form [26] and is therefore not further dealt with in this paper.

As outlined in this section, there is already a huge amount of work in the area of semantic integration dealing with ontologies, providing a proper basis for model-based tool integration. There is, however, to the best of our knowledge, no literature available, dealing with the usage of ontologies for metamodel integration, thus leaving open research questions in several directions. We advocate that a combination of existing techniques in the area of model management and integration with semantic technologies in terms of ontologies will allow to better exploit the potential of model-based tool integration.

5. OUTLOOK

Based on the state of the art described in this paper, we are currently realizing *ModelCVS*, a system which enables model-based tool integration based on semantic technologies. ModelCVS is developed in the course of an industrial project, involving, among others, a partner of Computer Associates (CA) and the Austrian Ministery of Defense. ModelCVS provides transparent transformation of models between different tools' modeling languages expressed as MOF-based metamodels, as well as versioning capabilities exploiting the rich syntax and semantics of models. It enables concurrent development by storing and versioning software artifacts that clients can access by a check-in/check-out mechanism, similar to a traditional CVS server. ModelCVS focuses on different integration scenarios, such as metamodel translation and modularization. Semantic technologies in terms of ontologies are used together with a knowledge base to store machine-readable, tool integration relevant information, thus allowing to minimize repetitive effort and partly automate the integration process.

The case study used to demonstrate the applicability of ModelCVS assumes the integration of three tools, CA's CASE tool *AllFusion Gen* (*Gen* for short), the UML tool Rational Software Modeler, and the Oracle BPEL Process Manager. Covering a wide range of modeling tasks, Gen is a legacy tool under which many existing applications have been developed. UML should be employed for new projects to link up with current technologies, and finally BPEL (Business Process Execution Language for Web Services) is required for developing certain web-enabled workflow applications.

REFERENCES

[1] M. J. Anderson, B. D. Bird: An evaluation of PCTE as a portable tool platform, Proc. of the Software Engineering Environments Conference, July 1993.

[2] S. Becker et al.: Model-Based A-Posteriori Integration of Engineering Tools for Incremental Development Processes, Journal on Software and Systems Modeling (SoSym), Springer, 4(2), 2005.

[3] J. A. Bergstra, P. Klint: The Discrete Time ToolBus - a software coordination architecture. Coordination Models and Language, LNCS# 1061, 1996.

[4] J. Bézivin et al.: First Experiments with a ModelWeaver, OOPSLA & GPCE Workshop, Vancouver, Oct. 2004

[5] M. L. Brodie: On Modeling Behavioural Semantics of Databases, 7th International Conference on Very Large Data Bases, Cannes, France, Sept. 1981.

[6] A. W. Brown, P. H. Feiler, K. C. Wallnau: Past and future models of CASE integration, Proc. of the 5th International Workshop on Computer-Aided Software Engineering, IEEE, July 1992.

[7] Engineering (ICSE), Toronto, Canada, 2001.

[8] S. Clarke: Extending standard UML with model composition semantics, Science of Computer Programming, Elsevier Science, 44(1), July 2002.

[9] K. Czarnecki, S. Helsen: Classification of Model Transformation Approaches, OOPSLA Workshop on Generative techniques in the context of MDA, Oct. 2003.

[10] D. Dou, et al.: Ontology translation on the semantic web, International Conference on Ontologies, Databases and Applications of Semantics, 2003.

[11] A. Earl: Principles of a Reference Model for Computer Aided Software Engineering

Environments, Proc. of the Int. Workshop on Software engineering environments, Springer-Verlag, New York, USA, 1989.

[12] J. Estublier, et al.: A Domain Composition Approach, Proc. of the International Workshop on Applications of UML/MDA to Software Systems (UMSS), Las Vegas, USA, June 2005.

[13] M. Ferdinand, et al: Lifting XML Schema to OWL, 4th International Conference on Web Engineering (ICWE), Munich, Germany, July, 2004.

[14] R. G. Flatscher: Metamodeling in EIA/CDIF - meta-metamodel and metamodels, ACM Transactions on Modeling and Computer Simulation, Oct. 2002.

[15] A. Gangemi et al.: Sweetening wordnet with DOLCE, AI Magazine, 24(3), 2003.

[16] J. Gray, et al.: An Approach for Supporting Aspect-Oriented Domain Modeling, Generative Programming and Component Engineering (GPCE), LNCS 2830, Germany, Sept. 2003.

[17] G. Guizzardi, et al: An Ontology-Based Approach for Evaluating the Domain Appropriateness and Comprehensibility Appropriateness of Modeling Languages, 8th Int. Conf. on Model Driven Engineering Languages and Systems, Jamaica, 2005.

[18] J. Huang et al.: A Schema-Based Approach Combined with Inter-Ontology Reasoning to Construct Consensus Ontologies, 1st Int. Workshop on Contexts and Ontologies: Theory, Practice and Applications, July, 2005.

[19] Y. Kalfoglou, M. Schorlemmer: Ontology Mapping: The State of the Art, Proc. of Dagstuhl Seminar on Semantic Interoperability and Integration 2005, Dagstuhl, Germany, 2005.

[20] G. Karsai et al: Composition and Cloning in Modeling and Meta-Modeling Languages, IEEE Transactions on Control System Technology, 2004.

[21] G. Kiczales et al.: Aspect-Oriented Programming, Proc. of the European Conference on Object-Oriented Programming (ECOOP), Springer LNCS 1241, Finland, 1997.

[22] X. Lin, G. Kappel: Model CVS: A Model Transformation and Versioning System, Technical Report, TU Vienna, 2005.

[23] A. Maedche, et al.: MAFRA – a Mapping Framework for Distributed Ontologies, 13th European Conf. on Knowledge Engineering and Knowledge Management, EKAQ, Spain, 2002.

[24] D. L. McGuinness, et al.: An Environment for Merging and Testing Large Ontologies, 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2000), USA, 2000.

[25] S. Melnik, E. Rahm, P. A. Bernstein: Rondo: a programming platform for generic model management, ACM SIGMOD international conference on Management of data, ACM Press, New York, USA, June 2003.

[26] N. Noy: Semantic Integration: A Survey Of Ontology-Based Approaches, SIGMOD Record, 33(4), Dec. 2004.

[27] P. A. Oberndorf: The Common Ada Programming Support Environment Interface Set, Software Engineering, IEEE Transactions, 14(6), June 1988.

[28] Object Management Group (OMG): MDA Guide. Version 1.0.1, June 2004

[29] Object Management Group (OMG): Meta Object Facility (MOF) 2.0 Core Specification, Oct. 2003.

[30] E. Rahm et al: A survey of approaches to automatic schema matching, VLDB Journal, 10(4), 2001

[31] C. Reichmann, et al.: GeneralStore - a CASE-tool integration platform enabling model level coupling of heterogeneous designs for embedded electronic systems, 11th Int. Conf. on Engineering of Computer Systems, May 2004.

[32] T. Reiter, E. Kapsammer, W. Retschitzegger, W. Schwinger: Model Integration Through Mega Operations, Workshop on Model-driven Web Engineering (MDWE), Sydney, July 2005

[33] T. Reiter, E. Kapsammer et al., Towards a Semantic Infrastructure Supporting Model-based Tool Integration, 1st Int. Workshop on Global Integrated Model Management, Shanghai, May 2006.

[34] T. Reiter, E. Kapsammer et al., A Generator Framework for Domain-Specific Model Transformation Languages, 8th International Conference on Enterprise Information Systems (ICEIS), Cyprus, May 2006.

[35] A. Schürr, H. Dörr: Introduction to the special SoSym section on model-based tool integration, Journal on Software and Systems Modeling (SoSym), Springer-Verlag, 4(2), May, 2005.

[36] A. P. Shet, J. A. Larson: Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases, ACM Computing Surveys, 22(3), Sep. 1990.

[37] G. Straw et al.: Model Composition Directives, Proc. of the 7th UML Conf., Lisbon, Oct. 2004.

[38] L. Tratt: Model transformations and tool integration, Journal on Software and Systems Modeling (SoSym), Springer-Verlag, 4(2), May, 2005.

[39] M. Uschold et al.: Ontologies and Semantics for Seamless Connectivity, SIGMOD Record, 33(4), Dec. 2004.

[40] R. Volz, et al.: OntoLIFT Prototype, IST Project 2001-33052 WonderWeb, Deliverable 11, 2003.

[41] A. Wasserman: Tool integration in software engineering environments, Proc. of the Int. workshop on SWEE, Springer, USA, 1989.