On the Need for Megamodels

Preliminary Draft

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

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

Abstract

This note presents a preliminary view of what we call a "megamodel" i.e. some kind of registry for models and metamodels. A megamodel is a model of which at least some elements represent and/or refer to models or metamodels. An initial characterization of these megamodels as well as a description of some of their potential usages is provided. We are presently experimenting with the concept of megamodel in our AMMP environment (ATLAS Model Management Platform). Some initial application of the concept of megamodel in the design of such model-driven software development platforms will be presented. Besides argumenting for the need to use megamodels in a variety of situations, the paper argues on the importance of this concept as an essential part of any model-driven software development platform.

1 Introduction

Similarly to the difference that was noticed in 1976 by DeRemer and Kron [16] between programming in the large and programming in the small, we are today experimenting with the need to consider separately the activities of modeling in the large and modeling in the small. As part of a local model management platform, we are 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. Megamodeling will be considered in this paper as the activity of modeling in the large, i.e. of taking some sort of global view on the considered artifacts, related to other possible local views handled in various other contexts. We propose here an initial investigation on these important topics.

In November 2000 the OMG proposed a new approach to interoperability named MDA™ (Model-Driven Architecture) [38]. MDA is one example of a much broader approach known as Model Driven Engineering (MDE), encompassing many popular research trends like generative programming, domain specific languages, model-integrated computing, model-driven software development, model management and much more [14].

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 projects, open source platforms[1] 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.

Connected to such an open platform, tools will exchange models. But tools will also be handled as models. A tool implements a number of services or operations. Each service or operation is also represented as a model. An operation for example may also have input and output parameters, each parameter being itself considered as a model. The interoperability platform may be organized as an intelligent model exchange system according to several principles and protocols like

[1] Through this paper, the word "platform" will mainly mean the basic framework allowing all model-management tools to interoperate. The MDA-based meaning of "platform" (as in PIM or PSM) mentioned for example in section 3, will be considered as disjoint in the context of this work.

the classical software bus or even more advanced P2P architectures. To facilitate this exchange, the platform may use open standards like XMI (XML model Interchange), CMI (CORBA Model Interchange), JMI (Java Model Interchange), etc. In AMMP (ATLAS Model Management Platform) three main components are being integrated: an environment for the ATL model transformation language[2], a model weaving tool and support for megamodels. We only report in this paper on this later capability. In a previous work [10] we identified the need for a megamodel. However the notion was only briefly mentioned there. Here we discuss broader applications of this concept. The issues of model transformation and model weaving (modeling in the small) are presented in other papers.

This paper is organized as follows. Section 2 recalls the main characteristics of the MDE approach. In particular we emphasize the idea that the notion of MDE component encompasses features related to typing, identification and encapsulation of these components as suggested in [10]. Section 3 provides several examples of megamodels intended for various different goals. Section 4 describes model management platforms and more particularly AMMP which defines the main application context of megamodels in this paper. Section 5 tackles the problem of defining metamodels for megamodels. Section 6 presents related work and further ideas.

2 MDA and MDE

The basic assumption in MDE is the consideration of models as first class entities. A model is an artifact that conforms to a metamodel and that represents a given aspect of a system. These relations of conformance and representation are central to model engineering [6]. When we say that a model is composed of model elements and conforms to a metamodel, 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 and models is called a metametamodel. The OMG/MDA proposes the MOF (Meta Object Facility) as such a language. The Eclipse/EMF metametamodel is called Ecore and is compatible with MOF 2.0. This language has the power of UML class diagrams complemented by the OCL assertion and navigation language.

There are other representation systems that may also offer, outside the MDA strict boundaries, similar model engineering facilities. We call them technical spaces [24]. They are often based on a three level organization like the metametamodel, metamodel and model of the MDA. One example is grammarware [21] with EBNF, grammars and programs, but we could also consider XML documents, Semantic Web, DBMS, ontology engineering, etc. A Java program may be viewed as a model conforming to the Java grammar. As a consequence we may consider strict (MDA)-models, i.e. MOF-based like a UML model but also more general models like a source Java program, an XML document, a relational DBMS schema, etc.

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 → 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 terms of the metamodels, possibly extended by 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.

3 Motivating examples

Just to illustrate our purpose, we may start by listing a number of situations where one could find useful to get macroscopic information on models. By macroscopic we mainly mean rela-

[2] The ATL engine and IDE will be available on the Eclipse GMT project [18].

tions that consider models as wholes and not their individual elements[3].

One of the classical situations in many areas, including the area of software development, is the relation between product and process models. The process models usually describe who is doing what, how and when. It is structured around some kind of actors that use or produce various artifacts. The product models usually defines the "types" of artifacts that can be used or produced by the actors mentioned in the process model. For example a SPEM model may explain how an engineer may take as input some UML class diagram and produce Java classes and interfaces, directly by hand or with the help of an automatic transformation tool. We see in this situation a relation between the SPEM, the UML and the Java metamodels.

Another well known example of global link is the conformance relation that holds between a model and its metamodel. This is often considered as an implicit link, but we suggest here that this could be also explicitly captured in a megamodel with many advantages. One interesting property of this global conformance relation between a model and its metamodel is that it may be viewed as summarizing a set of relations that holds between model elements and metamodel elements (a relation we often name the "meta" relation, see for example [7]). We clearly see here the coexistence between global model level relations and local element based relations [7]. In some situations we are not interested at all with the local element level relations because the global relation provides sufficient reliable information on what we practically need.

Another example is related to transformations [26]. Recall that in our MDA landscape, a transformation is a model, conformant to a given metamodel. So, if a model ***Mb*** has been created from model ***Ma*** by using transformation ***Mt***, then we can keep this global information in the megamodel. Suppose the transformation model ***Mt*** has a link to its reverse transformation $\underline{\boldsymbol{Mt}}$, then we see some interest that could be reaped from this explicit information.

There is a whole set of information that could be regarded as global metadata. For example, we could associate to a model the information of who created it, when and why it was created and may be what for, who has updated it and the history of updates, etc. To a metamodel we can associate its goal, the place where we can find its unique original definition, alternate definitions, its authors, its history, its previous and next versions, etc. These are just some examples, but the kind of associated information may obviously be much broader.

The idea of model driven architecture has sometimes been presented with megamodels like in [10] from where the diagram of Figure 1 and Figure 2 are extracted. What is conveyed there is that a PSM (Platform Specific Model) is a model, in relation with a given PIM (Platform Independent Model) and with a given PDM (Platform Description Model). From a PIM, it should be possible to know which PSMs are available and to what platform they correspond. In simple cases, the description of the process that produced the PSM from the PIM and the PDM could also be explicitly defined. Although somewhat idealized, these illustrations show how megamodels may be used beyond mere documentation of architectural and process approaches.

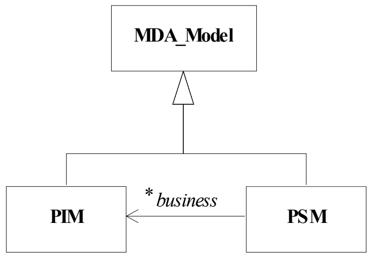

Figure 1 Basic MDA classification

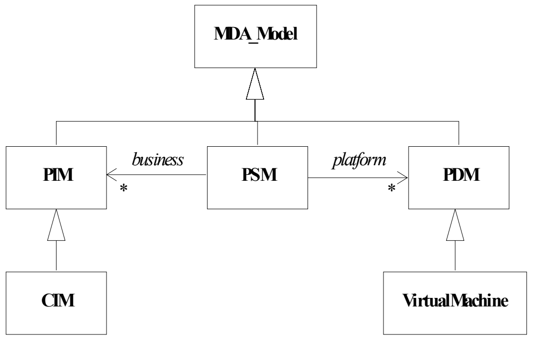

Figure 2 Extended MDA classification

Of course many other kinds of relations may exist between models, beyond PIMs, PDMs and PSMs, for example as described in Figure 3.

[3] Of course this is a relative consideration since we are talking also about elements of megamodels. We will try later to give a more precise characterization of this situation.

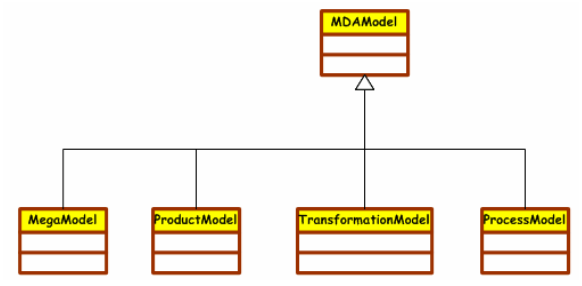

Figure 3 Other kinds of models

There is currently an important activity addressing the model-based modernization approach (ADM). As part of this initiative, a first request of proposal named KDM (Knowledge Discovery Metamodel) has been launched at OMG [42]. This may be viewed again as a step going in the direction of megamodels. As a matter of fact, the problem consists in making precise and operational inventories of various kinds of legacy artifacts in order to propose a model-based migration path supported by MDA-related tools. Another set of examples could be related to current OMG initiatives on model and metamodel portfolios.

4 Open Platforms for MDE

4.1 Tool interaction

The advantage of using a model-based platform is that it allows many economies of scale. For example model and metamodel repositories may handle efficient and uniform access to these models, metamodels and their elements in serialized or any other mode. Transactional access, versioning and many other facilities may be also offered, whatever the kind of considered model: executable or not, product or process, transformation, business or platform, etc.

In the late 90's it was suggested to organize the cooperation of tools working on models as a software bus on which could circulate such entities as UML models serialized in XMI for example ([4], [12]). This view was characterized at the time by an organization as described in Figure 4 for example where the middleware bus could be complemented by a semantically richer "Modelware" bus.

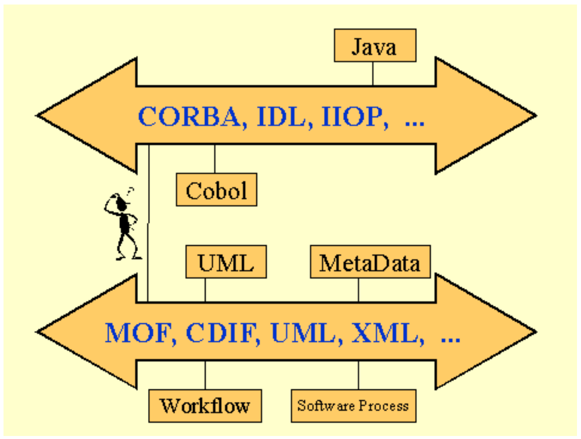

Figure 4 Communicating "Middleware" and "Modelware" buses

We use the notion of a MDE platform as a more abstract term than the mentioned buses because there are a number of ways to implement such a platform, one of them could be Modelware bus. For example one very promising different implementation for a MDE platform could be based on a distributed P2P architecture.

A MDE platform is primarily intended for tool integration. Several tools are usually available on such a platform, mainly:

- a repository,
- a transformation tool,
- a megamodel support,
- various generic editing tools,
- general browsing facilities,
- etc.

Each time a given tool registers or unregisters to the platform, the associated megamodel is updated. There are also plenty of other events that may change the megamodel.

For the sake of simplicity, we shall suppose that a platform is local and may be connected to other platforms. To each local platform, we suppose there is an associated megamodel defining the metadata associated to this platform.

4.2 Model components

To identify a given model we could use a three level scheme ***model:a/b/c*** where ***a*** is the metametamodel, ***b*** the metamodel and ***c*** the model. This could be seen either as an extension of MIME types or as a basis for a URI-naming style for MDE. This has the advantage of allowing several representation systems to be considered, each being identified by its own metametamodel. An example could be ***model:Ecore/UML/Banking***.

To be handled by the platform, a model component should be named and typed. Another issue for these model components is packaging. A model component should be encapsulated in a standard way so that the various tools may easily interoperate by sending and receiving these components. The current encapsulation standard for AMMP is based on the OMG RAS recommendation (Reusable Asset Specification) [28].

Finally the platform will also need to know the available tools and services. This is where comes the need for megamodels, i.e. models which elements represent models, metamodels, metametamodels, services, tools, etc. As any model, the megamodel is based on a precise metamodel. It records the main relations between its different entities and also the metadata that applies on them. One typical relation is the association of a metamodel to a given model or the fact that a metamodel is an extension or a newer version of another one. Applying a given transformation to a model may imply finding the metamodel for this transformation and an available transformation tool compatible with this metamodel. If none is found, one possibility would be to transform the transformation model itself, if this is possible.

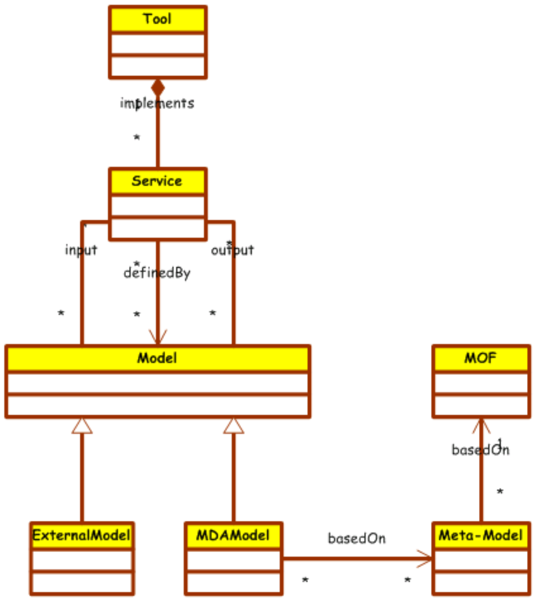

Figure 5 Tool representation in a megamodel

Each time a tool is plugged into a given platform, it registers for itself and for its supported services. This supposes we have a common view of what is a tool, which services it implements, etc. in a megamodel as illustrated in Figure 5.

If the tool is a legacy tool, there is an <u>adapter</u> to the platform in charge of interfacing and handling these tasks. The local megamodel records the local context. Global views of local megamodels may also be used by local platforms.

4.3 Use cases for MgM

There are plenty of possible use cases for megamodels, as has been suggested previously. However usually little attention is given to the necessity to define a global view. In order to illustrate possible usage scenarios, let us suppose we have the following situation.

We consider a situation where there are two MDE platforms installed in specific organizations, one in Nantes and one in Oslo for example. Within each organization there are different tools for capturing (Rational XDE, Poseidon, etc.) and storing/retrieving models and metamodels. Some tools may also be common. Each platform has been initialized in an initial neutral state. From there, each time a tool has been plugged or unplugged, the local megamodel has been correspondingly updated. Also, each tool has the possibility/responsibility to update the local megamodel upon achievement of specific operations. Examples of these operations may be for example:

- When a user has created a model with a model capture tool like Poseidon or Rational XDE
- When a user has created a metamodel in KM3[4] format with a textual editor
- When a user has modified a metamodel
- When a user has created a transformation in ATL
- When a user has created a new model by running an ATL transformation
- etc.

Let us suppose now a user in Oslo wishes to generate a list of contacts in a given database from a set of 5000 contacts contained in his/her local Microsoft Outlook system. The first action would be to look on the local Oslo platform if there is an available metamodel for Microsoft Outlook.

[4] KM3 is a notational convenience in the AMMP platform that allows rapid textual definitions for small metamodels. KM3 may be generated rapidly from tools like Microsoft Visio.

Let us first suppose there is one. Then the user will look for a corresponding injector able to act as a driver to transform Outlook data into MDE formats. If these correspond exactly to his/her needs, then they will be used as is, if not they will be adapted. After that the user will do the same for the target database (metamodel and extractor, for example SQL-based). Finally the local platform will be interrogated for a suitable transformation. If none corresponds, one will be built or adapted with the help of suitable browsing/editing tools.

In case the components are not available on the local Oslo platform, then the other platforms that have previously established links with the Oslo platform will be interrogated, for example the Nantes platform. The actions described before will be applied on this new platform. Alternatively, the user may consider his/her platform to virtually correspond to all components available on the Oslo or Nantes platform.

4.4 Platform architecture

The approach to defining the model management platform architecture in AMMP is iterative and incremental. Several investigations like the one reported in this paper allow us to elaborate an abstract functional view of the requirements.

From these abstract requirements, there are several possible implementations. The most classical and straightforward one is based on the conventional software bus or tool bus organization. The original proposal (see for example [4], [12], etc.) was to use a CORBA-like organization to organize this bus. In this view, what will circulate on this semantic bus will be essentially abstract models, for example serialized in XMI. The tool bus will consist of standards and protocols for connecting tools as well as a set of extended facilities. This bus will offer high level mechanisms for exchanging models and not code. New possibilities for handling operations, services, events, transactions, etc. may be easily added to the tool bus.

The software bus view corresponds to the present industrial state of the art. Much of the work done in the last 15 years on middleware technology (e.g. CORBA-like) may be relevant. The difference may lie in the need to have a more "intelligent" bus, able to handle the increase in abstraction corresponding to the move from code to models, from IDL to MOF, XMI and OCL. Care should be taken that this increase in functionalities does not bring an increase in complexity and execution costs. Instead of basing the architecture, in a CORBA-like way, on integrated functionalities, we prefer to externalize these functionalities as much as possible, with the help of metamodel driven tools. The approach suggested here for the handling of model and metamodel registries with megamodels goes in this direction. The first implementation of AMMP follows this conventional software bus approach.

Another possible implementation for a model-management platform is more speculative and research oriented. Starting from the same requirements, we may have a more decentralized architecture for this model-management platform. The second implementation of AMMP will follow this direction and mainly use technologies like Peer to Peer (P2P). In this vision we see users exchanging models, metamodels, transformations, injector and extractor drivers and other MDE components like others may exchange songs and videos on the Web today. The study of this organization is just starting, based on standards like JXTA [20], but many current investigations have shown initial feasibility in related domains [15], [27], [32], [35], [36], [37], [40], [43], etc. What can be said is that the proposed megamodel approach fits very well with such a decentralized organization.

5 Metamodels for megamodels

From what has been said until now, we may conclude that being a model, a megamodel conforms to a metamodel. However, there is no unique metamodel for all megamodels.

The important question is thus to define metamodels for megamodels. If megamodels are going to be used for distributed, shared or collaborative platforms, we cannot postulate that all these platforms will use the same metamodel.

The approach taken to this central question in our project is again experimental and iterative. In a first prototype, we make the assumption of a unique standard metamodel. Starting from there, we are following two paths for metamodel extension, one based on a common minimal metamodel with integrated extension capabilities and the other one on a more complete basis. The existence of a full-fledged metamodel extension mechanism, with possibilities for a given metamodel to extend several ones will be assumed in this work.

6 Related Work

There are currently plenty of activities going on in the domain of model engineering. In the motivating example section we have seen several examples of using megamodels. However these are often local examples and no significant effort is usually given in abstracting out the global view.

As we have also briefly mentioned earlier, there are several applications of megamodels in technical spaces outside the strict area of MDE. Probably the most important is related to XML, Web services and UDDI. Many important ideas can be borrowed from there.

In the full version of this paper a survey of related work will be provided, mainly in the fields of metadata registries, of ontologies and of UDDI registries ([25], [41], etc.)

7 Conclusion

We have tried to summarize in this paper some work done in the context of the AMMP project. The three main part of this platform we are currently developing are a transformation language (ATL) and its IDE, a "model weaver" facility that may be used as a front-end to the ATL model transformer and a global support for megamodels. Other tools envisioned are model-editors and model-checkers for example. One of our main objectives is to set up rich libraries of transformation components together with their metamodels and the corresponding extractors and injectors. The wide availability of such components is a prerequisite for demonstrating potential reusability in the domain of model transformation. In order to achieve this we progressively came to the conclusion of the need for a general abstract consideration of the notion of megamodels. The present paper has proposed our current vision on this. However we are constantly learning from the AMMP experimental work and constantly adapting our views on these notions. Going much beyond mere documentation, the notion of megamodel is still evolving, but what this work has shown is the absolute necessity to consider very seriously the task of modeling in the large.

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, and specially Gaetan Le Colleter.

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, September 2004, Kent. http://www.cs.kent.ac.uk/projects/kmf/mdaworkshop/

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

[3] Bézivin, J., Farcet, N., Jézéquel, J.-M., Langlois, B., & Pollet, D. Reflective Model Driven Engineering, UML Conference, 2003.

[4] Bézivin, J. Who's Afraid of Ontologies? Proceedings of OOPSLA 98, Vancouver, Canada, 18-22 Oct 1998 available from http://www.metamodel.com/oopsla98-cdif-workshop/bezivin1/

[5] Bézivin, J. From Object Composition to Model Transformation with the MDA TOOLS'USA 2001, Santa Barbara, August 2001, Volume IEEE publications TOOLS'39. http://www.sciences.univ-nantes.fr/info/lrsg/Recherche/mda/TOOLS.USA.pdf

[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., Lemesle, R. Towards a true reflective modeling scheme LNCS, ISSN: 0302-9743, Vol. 1826/2000, http://www.springerlink.com/media/3G267U4QVH5RRJ47VBFT/Contributions/2/8/4/W/284W7VGQC302VR5W.pdf

[8] Bézivin, J., Breton, E. Applying the basic principles of model engineering to the field of process engineering. Novatica N° 171, Software Process Technologies http://www.ati.es/novatica/infonovatica.html

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

[10] 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

[11] Bézivin, J., Gerbé, O. Towards a Precise Definition of the OMG/MDA Framework ASE'01, San Diego, USA, November 26-29, 2001 http://www.sciences.univ-nantes.fr/lina/atl/publications/ASE01.OG.JB.pdf

[12] Bézivin, J., Lemesle, R. The sBrowser: a Prototype Meta-Browser for Model Engineering. Proceedings of OOPSLA 98, Vancouver, Canada, 18-22 Oct 1998. available from http://www.metamodel.com/oopsla98-cdif-workshop/bezivin2/

[13] Bézivin, J. Meta-Model Technology: Concepts and Applications. XML'99 Conference, Philadelphia, Pennsylvania, Dec 1999. available from http://www.infoloom.com/gcaconfs/WEB/philadelphia99/biezivin.HTM

[14] Booch G., Brown A., Iyengar S., Rumbaugh J., Selic B. The IBM MDA Manifesto The MDA Journal, May 2004, http://www.bptrends.com/publicationfiles/05-04%20COL%20IBM%20Manifesto%20-%20Frankel%20-3.pdf

[15] Christoffel, M. Cooperations and Federations of Traders in an Information Market. In Proceedings of the AISB Symposium Intelligent Agents in Electronic Commerce, York, 2001.

[16] Deremer, F., 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

[17] D'souza, D. Model-Driven Architecture and Integration: Opportunities and Challenges Version 1.1. February 2001, Available from www.kinetiuym.com

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

[19] IBM Business Explorer for Web Services, http://www.alphaworks.ibm.com/tech/be4ws

[20] JXTA http://www.jxta.org

[21] Klint, P., Lämmel, R. Kort, J., Klusener, S., Verhoef, C., Verhoeven, E.J. Engineering of Grammarware. http://www.cs.vu.nl/grammarware/

[22] Kobryn, C. The Road to UML 2.0: Fast track or Detour. SD Magazine, April 2001.

[23] Kobryn, C. UML 2001: A Standardization Odyssey. Communications of the ACM, V.42, N.10, 1999

[24] Kurtev, I., Bézivin, J., Aksit, M. Technical Spaces: An Initial Appraisal. CoopIS, DOA'2002 Federated Conferences, Industrial track, Irvine, 2002 http://www.sciences.univ-nantes.fr/lina/atl/publications/

[25] Lee, J., Goodwin, R., Akkiraju, R., Doshi, P., Ye, Y. 2003 SNoBASE: A Semantic Network-based Ontology Ontology Management. http://alphaWorks.ibm.com/tech/snobase

[26] Lemesle, R. Transformation Rules Based on Meta-Modeling EDOC,'98, La Jolla, California, 3-5 November 1998, pp.113-122.

[27] Maedche, A., Staab, S. Services on the Move - Towards P2P-Enabled Semantic Web Ser-

vices. In: Proceedings of the 10th International Conference on Information Technology and Travel & Tourism, ENTER 2003, Helsinki, Finland, 29th-31st January 2003.

[28] OMG, Reusable Asset Specification, Final Adopted Specification, PTC/04/06/06, www.omg.org, (June 2004)

[29] OMG/MOF Meta Object Facility (MOF) Specification. OMG Document AD/97-08-14, September 1997. Available from www.omg.org

[30] OMG/RFP/QVT MOF 2.0 Query/Views/Transformations RFP, OMG document AD/2002-04-10. Available from www.omg.org

[31] OMG/XMI XML Model Interchange (XMI) OMG Document AD/98-10-05, October 1998. Available from www.omg.org

[32] Paolucci, M., Sycara, K., Nishimura, T. and Srinivasan, N. Using DAML-S for P2P Discovery, in Proceedings of the First International Conference on Web Services (ICWS'03), Las Vegas, Nevada, USA, June 2003, pp 203- 207 .

[33] Ploquin, N. Tooling the MDA framework: a new software maintenance and evolution scheme proposal http://www.sciences.univ-nantes.fr/info/lrsg/Pages_perso/Publications/joop01.pdf

[34] Schlosser, M., Sintek, M., Decker, S. Nejdl, W. A Scalable and Ontology-Based P2P Infrastructure for Semantic Web Services. Second International Conference on Peer-to-Peer Computing (P2P'02) September 05-07, Linkoping, Sweden 2002.

[35] Schmidt, C., Parashar, M. A Peer-to-Peer Approach to Web Service Discovery, World Wide Web, Internet and Web Information Systems, Kluwer Academic Publishers, August 2003.

[36] Sivashanmugam, K., Verma, K., Sheth, A. Discovery of Web Services in a Federated Registry Environment Proceedings of International Conference on Web Services, 2004, http://lsdis.cs.uga.edu/lib/download/MWSDI-ICWS04-final.pdf

[37] Sivashanmugam,K., Verma, K., Sheth, A., Miller, J. Adding Semantics to Web Services Standards, Proceedings of the 1st International Conference on Web Services (ICWS'03), Las Vegas, Nevada (June 2003) pp. 395-401.

[38] Soley, R. and the OMG staff Model-Driven Architecture. OMG document Available from www.omg.org (November 2000).

[39] Sun Java Community Process JMI Java MetaData Interface Specification Available from ftp://ftp.java.sun.com/pub/spec/jmi/asdjhfjghhg44/jmi-1_0-fr-spec.pdf

[40] Thaden, U., Siberski, W., Nejdl, W. A Semantic Web based Peer-to-Peer Service Registry Network, Technical Report, Learning Lab Lower Saxony,

[41] UDDI.org Evolution of UDDI, White Paper available at http://www.uddi.org/pubs/the_evolution_of_uddi_2002
0719.pdf

[42] Ulrich, W. A status on OMG Architecture-Driven Modernization Task Force. http://adm.omg.org/MELS_EDOC2004_Ulrich_Extended_Submission(pdf).pdf

[43] Verma, K. Sivashanmugam, K. Sheth, A. Patil, A., Oundhakar, S. And Miller, J. METEOR–S WSDI: A Scalable Infrastructure of Registries for Semantic Publication and Discovery of Web Services, Journal of Information Technology and Management (to appear, 2004).