Interoperability Issues in Metamodelling Platforms

Harald Kühn[1, #], Marion Murzek[2, *]
[1]BOC Information Systems GmbH, Platform Development
Rabensteig 2, A-1010 Vienna, Austria
[2]Women's Postgraduate College for Internet Technologies (WIT),
Vienna University of Technology, Austria
E-Mail: harald.kuehn@boc-eu.com, murzek@wit.tuwien.ac.at

Abstract

Metamodelling platforms are getting more and more base technology, therefore interoperability of metamodelling platforms becomes a crucial aspect in managing corporations' knowledge assets. This paper describes a generic metamodelling platform architecture and presents an overview of interoperability issues according to conceptual domains in metamodelling platform architectures. Some of these issues are illustrated by a case study from the insurance sector. The collection of interoperability issues can serve as a starting point to stimulate further research on interoperability problems in the metamodelling platform domain.

1. Introduction

Metamodelling platforms are software environments allowing the definition, usage and maintenance of a method's elements: (a) *metamodels* describing problem-specific modelling languages, (b) *mechanisms & algorithms* working on models and their underlying metamodels, and (3) *procedure models* representing process descriptions how to apply the metamodels and the corresponding mechanisms. Some of their functional and non-functional requirements are multi-product ability, web-enablement, multi-client ability, adaptability, and scalability [6].

Metamodelling approaches are an active research field since the past 15 years and since then have found serious *application areas* in the software and information technology industries. Some of them are Enterprise Model Integration (EMI) [9] in the context of Enterprise Application Integration (EAI) [12], Model Integrated Computing (MIC) [11], domain specific modelling languages such as the Unified Modelling Language (UML) [22] based on Meta Object Facility (MOF) [18], the Unified Enterprise Modelling Language [27], and model-driven development approaches such as Model Driven Architecture (MDA) [19]. Additionally, metamodelling approaches serve as valuable base technology to merge different modelling approaches into a problem specific modelling language.

Since widespread industrial and research usage of metamodelling technology such as ADONIS [1], MetaEdit+ [13] or METIS [14], the integration and *interoperability of metamodelling platforms* is moving into focus of applied research and product-quality implementations [16]. The goal of this paper is to provide an overview of interoperability issues in the domain of metamodelling platforms. This overview can serve as a starting point to stimulate further research on interoperability problems in this domain.

The remainder of the paper is organized as follows: chapter 2 presents a generic metamodelling platform architecture. Chapter 3 gives an overview of issues in metamodelling platform interoperability. These issues provide input for further research areas in metamodelling platform interoperability. Chapter 4 presents a case study in metamodelling platform interoperability. Related work is discussed in chapter 5. Finally, chapter 6 summarizes the paper and gives an outlook to future work.

[#] This work is partially supported by the Commission of the European Communities under the sixth framework programme (INTEROP Network of Excellence, Contract N° 508011, <http://www.interop-noe.org>).
[*] This research has been partly funded by the Austrian Federal Ministry for Education, Science, and Culture, and the European Social Fund (ESF) under grant 31.963/46-VII/9/2002.

2. Metamodelling Platform Architecture

Figure 1 presents a generic architecture of metamodelling platforms [6, 8]. An important element is the *meta-metamodel* (meta^2 model). The meta^2 model defines general concepts available for method definition and method usage such as "metamodel", "model type", "class", "relation", "attribute" etc. *Semantic schemas* are tightly coupled with the meta^2 model. They describe the semantics of each method element defined by using the meta^2 model. Semantic schemas can be described by using approaches such as ontology [5], semantic engines ("mechanisms") etc.

The *metamodel base* contains metamodels of concrete modelling languages. Metamodel editors are used for the definition and maintenance of metamodels. The metamodel base is based on the meta^2 model. The metamodel base forms the foundation of the *model base*, in which all models are stored. Models can be created, changed and visualized by using appropriate editors.

All mechanisms and algorithms used for evaluating and using models are stored in the *mechanism base*. Mechanism editors are used for definition and maintenance of mechanisms. The mechanism base is based on the meta^2 model.

Procedure models describe the application of metamodels and mechanisms. They are stored in the *procedure model base*. Procedure model editors are used for definition and maintenance of procedure models. The procedure model base is based on the meta^2 model.

Persistency services support the durable storage of the various bases. These services abstract from concrete storage techniques and permit filing of modelling information in heterogeneous databases, file systems, web services etc.

Access services serve two main tasks: on the one hand they enable the open, bi-directional exchange of all metamodelling information with other systems using API or file based. interfaces. On the other hand they cover all aspects concerning security such as access rights, authorization, en-/decryption etc.

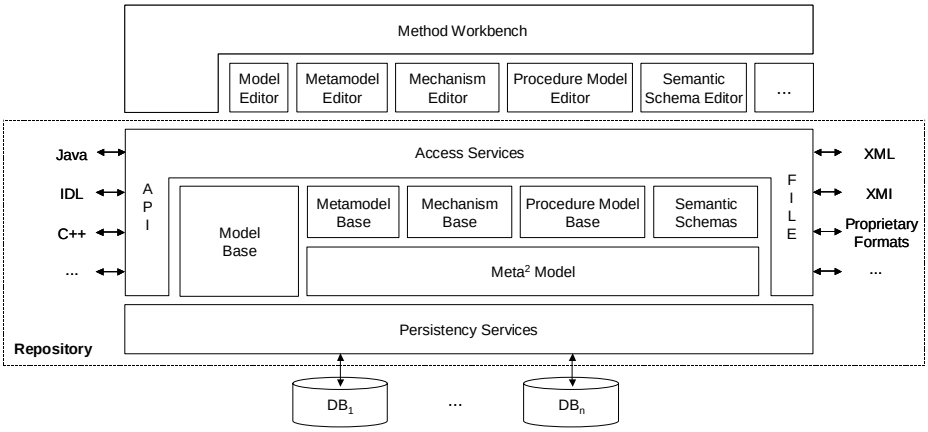

Figure 1: Generic Architecture of Metamodelling Platform

3. Interoperability Issues

OUSKEL AND SHETH identified two major categories of interoperability problems: information heterogeneity and system heterogeneity [24]. In the context of metamodelling platforms, information heterogeneity maps to the modelling hierarchy of meta^2 models, metamodels and models of each platform ("model heterogeneity"). System heterogeneity maps to the diversity of available access services, mechanisms, persistency services, and implementation technologies of each platform (see fig. 2). The further description of interoperability issues in metamodelling platforms will be structured according to the conceptual domains of the generic metamodelling platform architecture as described in fig. 1.

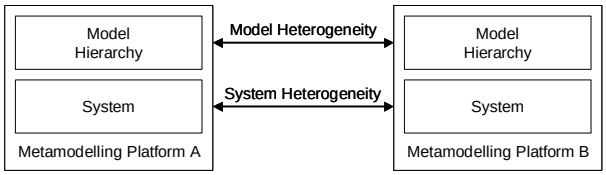

Figure 2: Model Heterogeneity and System Heterogeneity

3.1 Meta^2 Model Domain

The meta^2 model provides the basis for the other conceptual domains (see fig 1.). Interoperability problems in this domain may arise in the syntax, semantics and expressiveness of underlying metamodelling languages to define, integrate and represent a method's elements [8], and appropriate transformation mechanisms for metamodel transformation.

Some important aspects to be considered by metamodelling languages are:

- inheritance and meta-class features for metamodel definition.
- the expressive power and cardinality features of meta-relationships such as aggregation (part-of), object-orientation (is-a), pointer (link), and binary or n-ary relationships.
- the amount of available meta-attributes to define concrete attributes of a certain type.

Some important aspects to be considered by metamodel transformation mechanisms are:

- handling of different relationship concepts such as n-ary relationships, and circular and recursive dependencies.
- nesting of elements and their handling in flattened structures during metamodel transformation.
- uniqueness of element identification and the possibility of using model annotations to store information in the target metamodel to avoid information loss during transformation.

3.2 Metamodel Domain

Interoperability issues in the metamodel domain may occur in the definition, integration and representation of the syntax, semantics and notation of modelling languages. Additionally, model transformation mechanisms for the horizontal and vertical model transformation are aspects to be considered in interoperable metamodelling platforms.

Some important aspects to be considered on the metamodel level are:

- identification and consideration of syntactic and semantic mismatches among modelling languages (same name – different concept, different name – same concept etc.).
- identification and usage of domain-specific ontology to consider domain-specific aspects and knowledge to establish metamodel interoperability.
- measurements for analysis and evaluation of modelling languages and their underlying metamodels to identify interoperable and non-interoperable parts.
- definition of "hot spots" in participating metamodels to provide linking points for metamodel integration.

3.3 Model Domain

Models correspond to their underlying metamodel. Therefore, the interoperability problems on this level are influenced by the problems concerning metamodels (see 3.2).

Some important additional aspects to be considered in model interoperability are:

- existence of non-corresponding model fragments, i.e. their metamodels are partly not corresponding. This can result in information loss or in hidden information to avoid losing information in bidirectional model exchange.
- diversity of graphical representations and diversity of the underlying coordinate system to place and arrange modelling objects. In worst case, models cannot be understood after model exchange because of complete loss of graphical information.

- models are input or provide parameters for mechanisms such as simulation, analysis, reporting, and code generation. Even if models correspond to its metamodel, it may occur that mechanisms cannot be used because of incomplete models.
- existence of appropriate domain ontology to support a proper model interpretation in each platform.
- history logs to record model changes which can be necessary in model synchronisation in distributed model change scenarios.

3.4 Mechanism Domain

Mechanisms provide possibilities to generate added-value out of the different model bases. Typical examples for mechanisms are version management, multi language support, model analysis, and simulation.

Some important aspects to be considered in mechanism interoperability are:

- mechanisms can be implemented either on meta model level or meta^2 model level. Before exchanging mechanisms between metamodelling platforms, the interdependencies of a mechanism to these both levels have to be analyzed.
- the technology used to implement a mechanism (scripting, programming language, query language etc.) has strong influence on its interoperability. A possible way to implement interoperable mechanisms, is using wrapper concepts.

3.5 Procedure Model Domain

Procedure models describe the processes how to apply modelling languages and mechanisms to solve certain problem scenarios. This includes concepts such as phases, milestones, responsibilities, work steps, results etc.

Important aspects to be considered in interoperability of procedure models are:

- the availability and mismatch of special procedure model fragments such as contradictory work step descriptions.
- merging of procedure models into consolidated procedure descriptions.

3.6 Semantic Schema Domain

Semantic schemas describe the semantics of each method element. They are connected either to elements of the model level, meta level or meta^2 level. A semantic schema can be defined e.g. by semantic engines ("script libraries") or by using ontology.

Important aspects to be considered in interoperability of semantic schemas are:

- semantic similarity among semantic schemas and the measurement of the similarity.
- mismatches of ontological constructs used in the semantic schemas.
- merging and integrating semantic schemas into a consolidated and shared semantic schema.

3.7 Persistency Services Domain

Persistency services provide support for durable storage of the various bases. Some of the relevant interoperability issues in this domain are:

- heterogeneous structures of underlying data sources such relational DBMS, object-oriented storage systems, XML-based databases or file systems.
- different transaction systems to counter effect an interoperable commit strategy.
- heterogeneous user, user profiles and connect definitions to make consistent data access difficult or even impossible (single sign-on).

3.8 Access Services Domain

Interoperability issues in this domain are mainly caused by system heterogeneity. It can be separated into problems of direct (via API) or indirect (via files) exchange:

Direct exchange can be supported by metamodelling platform API. Some important interoperability issues are:

- the involved providers must agree on necessary interfaces regarding their *programming languages*, *method signatures* and in general about the *security handling* and the *access rights*.
- Agreement on standardized "protocols" as suggested in [2] by using a general "model bus" where each vendor could get attached by implementing one of the provided protocols.

In indirect exchange the supported file formats play an important role such as XMI [17], HUTN [20], XML, and proprietary formats:

- In case of *proprietary formats* a parser must be implemented to be able to interpret the file syntax. Then rules must be defined to convert the semantic content and the target file (format) must be generated.
- *Standard formats* and *languages* have the advantage that their syntax and partly their semantics are given. Also standardized script languages e. g. XSLT or XQuery for XML are provided.

4. Case Study

On the basis of an example from the financial services sector, namely the insurance sector, interoperability issues in the metamodel and the access service domain are demonstrated.

The example consists of three metamodels which are instances of the ADONIS [1] meta^2 model (quality management, business process management and ERP introduction metamodel). These and an additional metamodel of a fixed metamodelling platform should be integrated into a new metamodel (fig. 3). In the following the four metamodels and their integration on the metamodel and model level is described.

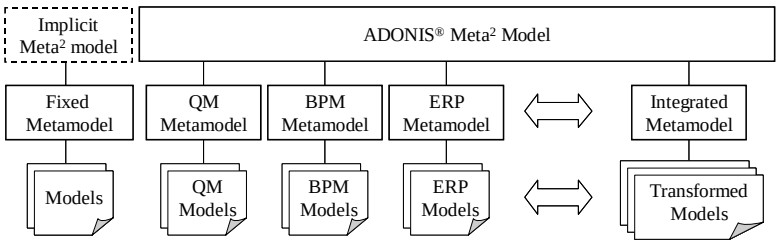

Figure 3: Integration and Interoperability of four Different Metamodels

The main part of the *ERP metamodel* (fig. 4 top right) consist of a process flow and additional process objects. The process flow generalizes the components event, function and the logical operators. There are two types of functions: basic functions and decomposition functions. Specializations of the additional objects are organizational unit, information system, and information object.

The *quality management metamodel* (fig. 4 top left) contains four different model types: process overview, business process model, organizational model and system model. The differences of the type business process model and the process model in the ERP metamodel are the missing class event and the missing operator XOR. Therefore it contains additionally classes such as start and end. The process overview contains processes which could refer to other processes or to business model processes. Additionally, a process could reference a document. The organizational model consist of organisational units and actors which have a role and which could be referenced by an activity from the business process model. The system model contains system components which could be connected via data flows. There

are two types of system components, systems and subsystems which could refer to another system model. The systems are referenced by input/output information classes.
The *business process metamodel* is mainly contained in the quality management model which is described above. The ERP metamodel and the *fixed metamodel* are very similar. Therefore and due to a lack of space only two of the four different source metamodels are illustrated in fig. 4.
To ensure syntactical interoperability, in the target metamodel all of these concepts must be integrated. To enable the transformation of the existing models from the old platform to the new one the mapping of the classes between each source metamodel and the new integrated metamodel must be defined.
Fig. 4 graphically illustrates the syntactical mapping between the metamodels. The new integrated metamodel is shown at the bottom of fig. 4. It contains seven model types, the same four as the quality management metamodel and additional three new pool model types. The pool models summarize all documents, all roles and all process owners. Due to the fact that the new system model does not provide the class subsystem, the structure of the ERP models has to be flattened what means a loss of information. Also for each ERP model a process start and an end must be newly created to match the syntax of the integrated metamodel. The events in the ERP models have to be eliminated and the logical operator XOR must be converted in a decision. Moreover, many classes have to be renamed, for example function in activity and decomposition function in sub process as shown in fig. 4.

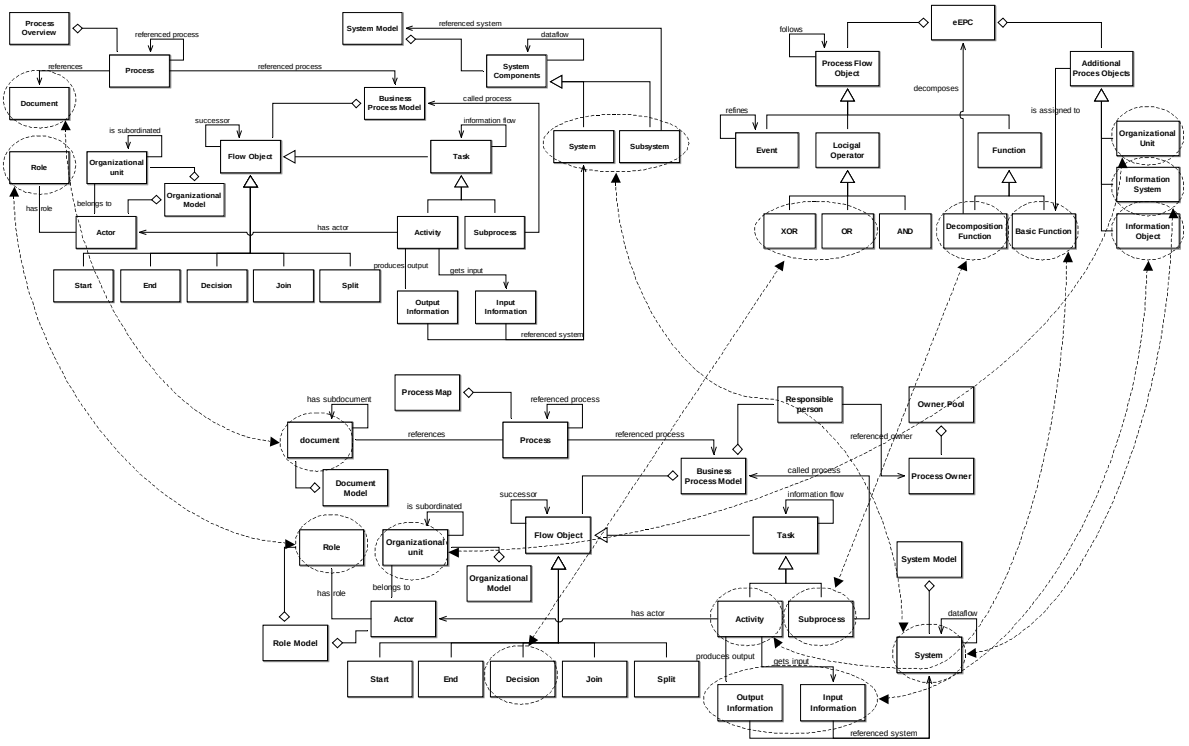

Figure 4: Integration of Metamodels

To physically transform the models to fit to the new integrated metamodel the interoperability problems concerning the access service domain must be solved. In our case the models of three source metamodels are described with the same format, the ADONIS XML format. The additional models of the fixed metamodel environment are also described in XML but in a different structure. So first of all the models which are described differently must be converted into the structure of ADONIS XML format. This transformation could be done by the means of an XSLT script which does this conversion. After that all model files exist in the same structure. Now a transformation tool is needed to automatically transform the models to

fit to the new integrated metamodel. For this purpose the BOC Model Transformer (BMT) [10, 15] has been used. It is a tool that supports the transformation of models between different modelling languages within ADONIS. Different kinds of navigations, rules, functions, conditions and definitions make it possible to specify a rule file which contains the semantical mappings for the transformation of the models between each source metamodel and the new integrated metamodel. Furthermore the BMT supports the creation of graphical information for new objects, for example the start and end classes. Also the interdependencies within and between the models are preserved and new references could be created automatically. After the transformation of all models they fit to the integrated metamodel and could be imported into the new metamodel which has been configured in ADONIS.

5. Related Work

Due to the fact that this work provides an overview of existing problems regarding interoperability issues in metamodelling platforms, the following related work concentrates on technology and approaches in the metamodelling domain.

ADONIS is a meta business process management tool [1]. It offers a three-step modelling hierarchy with a rich meta^2 model. Meta models can be customized as instances of the meta^2 model. Mechanisms such as "simulation" or "analysis" are defined on the meta^2 level and can be redefined on the metamodel level. The scripting language AdoScript provides mechanisms to define specific behaviour and functionalities.

MetaEdit+ offers also a three-step modelling hierarchy [13]. The meta^2 model forms the "GOPRR" model, offering the basic concepts "Graph", "Object", "Property", "Relationship" and "Role". A diagram editor, object and graph browsers and property dialogs support the definition of a new modelling language without hand coding. Furthermore MetaEdit+ includes XML import and export, an API for data and control access and a generic code generator.

The OMG´s Meta Object Facility (MOF) [18], the open source Eclipse Modelling Framework (EMF) [26] and the Graphical Editor Framework (GEF) [25] are no metaCASE tools themselves. With the MOF the OMG created a meta^2 model standard, which provides a basis for defining modelling frameworks. UML [22] and the Common Warehouse Metamodel (CWM) [21] are examples of instantiated meta models of the MOF. Interoperability issues concerning the meta model and the model domain are addressed by the ongoing standardisation of MOF Query/Views/Transformations (QVT) [23] which should provide mechanisms for mappings between models and meta models. The EMF which was influenced by the MOF is a shared code base for public use. Together with the GEF it provides a possibility to create a new modelling tool.

The main difference between meta tools such as MetaEdit+ or ADONIS and the MOF or the EMF is that meta tools provide the user a graphical environment to create new meta models, whereas in EMF and GEF everything must be coded.

It is easier and faster to build new meta models within meta tools, but due to the implemented meta^2 model the degree of freedom to create the specific meta model desired is lower than in modelling frameworks like EMF [7].

In [4] E-MORF - a XSLT-based transformation tool - is introduced. E-MORF supports the conversion between MOF and EMF. The transformation is executed by applying the XSLT to XMI which is supported by MOF and EMF. Beside the mapping concept where all fragments of both meta^2 models are opposed also the mapping problems for example "non-corresponding fragments" and "name mangling" are described.

The XMF (eXecutable Metamodelling Facility) [3] created by Xactium is a metamodelling facility that fully supports language definition. The heart of the XMF is the XCore which represents its metamodel which is comparable to the MOF model. To support mappings

between models two further language are defined. XMap which is a unidirectional pattern based mapping language and XSync which is a bidirectional synchronisation language.

6. Conclusion

Metamodelling platforms are getting more and more a kind of base technology [6]. Additionally, domain specific languages, model transformation approaches, and lifecycle management within large model bases are active research issues. The interoperability of metamodelling platforms becomes a crucial aspect in managing corporations' knowledge assets. This paper presented an overview of interoperability issues according to conceptual domains in metamodelling platform architectures. Some of these aspects were illustrated by a case study from the insurance sector. The issues overview can serve as a starting point to stimulate further research on interoperability problems in the metamodelling platform domain. Additionally to interoperability, we see three important trends in the area of metamodelling platforms in the near future:

- *Metamodelling gets commodity*: metamodelling provides suitable concepts for flexible modelling platform solutions. Furthermore, metamodelling concepts diffused more and more in other domains such as MOF, into language design such as the language definition of UML 2.0 or in Software Engineering e.g. in product family-based approaches. We expect, metamodelling will also attract more attention in domains such as Workflow Management, IT Architecture Management and Knowledge Management. Expecting this, challenging interoperability issues will have to be solved.
- *Integration of business-oriented and IT-oriented methodologies*: we see strong demands integrating approaches such as Strategy Management, Process Management and IT Management into single, integrated methods. A promising approach is MOF and MDA. Nevertheless, their focus is currently focused on systems development. Upper-level models such as business specifications and computation independent models (CIM) are not well represented until now. Here, research need in areas such as low-loss transformations and treatment of heterogeneous semantics is expected.
- *Method Integration and Knowledge Management*: our society is seen as a "knowledge society". Methods represent experts knowledge, they represent the know how to do and process things in a certain way. As a future research domain we see the investigation of interdependencies of Knowledge Management and Method Engineering and corresponding issues in integrating both.

References

1. ADONIS Homepage. www.boc-eu.com, access 30 November 2004.
2. X. Blanc, M. Gervais, P. Sriplakich: *Model Bus: Towards the interoperability of modelling tools*, Proceedings of Model-Driven Architecture: Foundations and Applications 2004, Linköping, Sweden.
3. T. Clark; A. Evans; P. Sammut; J. Willans: *Applied Metamodelling – A Foundation for Language Driven Development*, Version 0.1, Xactium, August 2004.
4. K. Duddy; A. Gerber; K. Raymond: *Eclipse Modeling Framework (EMF) import/export from MOF / JMI*, DSTC Technical Report, May 2003.
5. Gruber, T. R.: Toward principles for the design of ontologies used for knowledge sharing. In: Guarino, N.; Poli, R. (Eds.): Proceedings of the International Workshop of Formal Ontology, Padova, Italien, August 1993.
6. D. Karagiannis; H. Kühn: *Metamodelling Platforms*. Invited paper in: Bauknecht, K.; Min Tjoa, A.; Quirchmayer, G. (eds.): Proceedings of the Third International Conference EC-Web 2002 - Dexa 2002, Aix-en-Provence, France, September 2-6, 2002, LNCS 2455, Springer-Verlag, Berlin, Heidelberg, p. 182.

7. S. Kelly: *Comparison of Eclipse EMF/GEF and MetaEdit+ for DSM*, http://www.softmetaware.com/oopsla2004/kelly.pdf, access 20 November 2004
8. H. Kühn: *Methodenintegration im Business Engineering.* PhD Thesis, University of Vienna, April 2004.
9. H. Kühn; F. Bayer; S. Junginger; D. Karagiannis: *Enterprise Model Integration*. In: Bauknecht, K.; Tjoa, A M.; Quirchmayr, G. (Hrsg.): Proceedings of the 4th International Conference EC-Web 2003 - Dexa 2003, Prague, Czech Republic, September 2003, LNCS 2738, Springer-Verlag, pp. 379-392.
10. H. Kühn; M. Murzek; F. Bayer: *Horizontal Business Process Model Interoperability using Model Transformation.* In: INTEREST'2004 Workshop at ECOOP 2004, Oslo, Norway, June 2004.
11. Ledeczi A., Maroti M., Bakay A., Karsai G., Garrett J., Thomason IV C., Nordstrom G., Sprinkle J., Volgyesi P. *The Generic Modeling Environment*, Workshop on Intelligent Signal Processing at WISP'2001, Budapest, Hungary, May 17, 2001.
12. Linthicum, D. S.: Enterprise Application Integration. Addison-Wesley, 2000.
13. MetaEdit+ Homepage. www.metacase.com, access 30 November 2004.
14. METIS Homepage. www.computas.com, access 30 November 2004.
15. M. Murzek: *Methodenübergreifende Modelltransformationen am Beispiel von ADONIS*. Diploma Thesis, University of Vienna, April 2004.
16. NoE INTEROP, Interoperability Research for Networked Enterprises Applications and Software, IST Network of Excellence, www.interop-noe.org, 2004.
17. Object Management Group: *OMG XML Metadata Interchange (XMI) Specification*, Version 1.2, January 2002. http://www.omg.org/cgi-bin/doc?formal/02-01-01.pdf, access 10 November 2004.
18. Object Management Group: *Meta Object Facility (MOF) Specification*, Version 1.4, April 2002. http://www.omg.org/cgi-bin/doc?formal/02-04-03.pdf, access 30 November 2004.
19. Object Management Group: *MDA Guide, Version 1.0.1*, 12. June 2003. http://www.omg.org/cgi-bin/apps/doc?omg/03-06-01.pdf, access 30 November 2004.
20. Object Management Group: *Human-Usable Textual Notation (HUTN) Specification*, Final Adopted Specification December 2002, http://www.omg.org/docs/ptc/02-12-01.pdf, access 30 November 2004.
21. Object Management Group: *Common Warehouse Metamodel (CWM)*, OMG Specification/203-03-02, March 2003, http://www.omg.org/docs/formal/03-03-02.pdf, access 30 November 2004.
22. Object Management Group: *Unified Modeling Language Specification*, OMG Specification/2003-03-01, March 2003, http://www.omg.org/docs/formal/03-03-01.pdf, access 30 November 2004.
23. Object Management Group: *Revised submission for MOF 2.0 Query/Views/ Transformations RFP*, OMG Specification/2003-08-18, August 2003, http://www.omg.org/docs/ad/03-08-08.pdf, access 30 November 2004.
24. A. M. Ouskel; A. Sheth: Semantic Interoperability in Global Information Systems. A brief Introduction to the Research Area and the Special Section, SIGMOD Record Vol. 28, No. 1, March 1999.
25. The Graphical Editing Framework (GMF), http://www.eclipse.org/gef/
26. The Eclipse Modeling Framework (EMF), http://www.eclipse.org/emf/
27. UEML – Unified Enterprise Modelling Language, http://www.ueml.org, access 30 November 2004.