

Enterprise Model Integration

Harald Kühn¹, Franz Bayer¹, Stefan Junginger², and Dimitris Karagiannis³

¹ BOC Information Systems GmbH, Rabensteig 2, 1010 Vienna, Austria
{harald.kuehn, franz.bayer}@boc-eu.com

² BOC Information Technologies Consulting GmbH, Voßstr. 22, 10117 Berlin, Germany
stefan.junginger@boc-de.com

³ University of Vienna, Department of Knowledge Engineering, 1210 Vienna, Austria
dk@dke.univie.ac.at

Abstract. Due to rapid changing business requirements the complexity in developing enterprise-spanning applications is continually growing. A vital field of delivering technical concepts and technologies for integrating heterogeneous applications and components to support inter-organisational business processes is the area of Enterprise Application Integration (EAI). A common characteristic of all EAI approaches is their focus on technical and runtime aspects of integration. From our project experiences in developing large B2B applications, it is necessary to integrate applications on the business and conceptual level as well. Because of the diversity of models and modelling languages for developing enterprise applications, we propose the *Enterprise Model Integration (EMI)* approach. In this paper we describe basic concepts of EMI, a pattern system for metamodel integration, and a case study applying EMI for developing B2B applications. The EMI approach is compatible with the MDA infrastructure and implemented within the meta model management tool ADONIS.

1 Introduction

Due to rapid changing business requirements such as faster time to market, shorter product lifecycles, increased interdependencies between business partners, and tighter integration of their underlying information systems, the complexity in developing enterprise-spanning applications is continually growing. Amongst others, a vital field of delivering technical concepts and technologies for integrating heterogeneous applications and components to support inter-organisational business processes is the area of *Enterprise Application Integration (EAI)*. The main idea of EAI is to provide technical solutions to integrate workflows and heterogeneous parts of enterprise applications in a continuous business application [5, 19, 22]. For technical integration different approaches have been helpful, e.g. *data-oriented-, application interface-oriented-, method-oriented-, portal-oriented- and process-oriented application integration approach* [8]. A common characteristic of all EAI approaches is their *focus on technical and runtime aspects of integration*, i.e. on the level of target systems and execution environments (see fig. 1).

Our project experiences in developing large B2B applications indicated that it is necessary to *integrate applications on the business and conceptual level* as well as

using models such as business models, business process models, product models, interaction models, (abstract) interface models etc. [14, 15]. Today, target systems such as workflow management systems (WMS), ERP systems, J2EE systems, groupware and messaging systems or company-specific software systems, mainly use proprietary model representations to describe aspects such as business logic, business rules, control and information flow, security aspects etc.

Standardisation organisations such as the World Wide Web Consortium (W3C), the Object Management Group (OMG) or the Workflow Management Coalition (WfMC) have reached to establish *standardisations on an execution level* such as W3C's HTTP-protocol or the XML data meta language, OMG's Common Object Request Broker Architecture (CORBA) or WfMC's interface 4 for process interoperability [18, 23, 24]. *Standardisations on modelling level* still can rarely be found, almost the only, but prominent, example is the UML for modelling object-oriented systems [17]. In addition, the OMG community started to establish a relatively new vision with the Meta Object Facility (MOF) and the Model Driven Architecture (MDA) to improve productivity in software development, applying object-orientation, metalevel concepts and modeling [2, 16].

Because of the diversity of models and modelling languages for developing enterprise applications, we propose the *Enterprise Model Integration (EMI) approach*. This approach is based on object-oriented metamodelling concepts to describe context-specific, integrated modelling languages. These modelling languages are used in enterprise application development projects integrating heterogeneous target systems. The EMI approach is compatible to the MDA infrastructure and is implemented within the meta model management tool ADONIS [9].

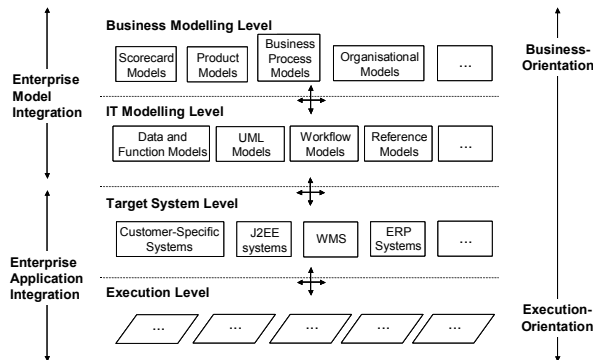


Fig. 1: From EMI to EAI

The remainder of the paper is organised as follows. Chapter 2 gives an introduction in model integration concepts. In chapter 3 a pattern system for metamodel integration patterns is presented and three concrete metamodel integration patterns are described. Chapter 4 describes a real life case study applying EMI developing a B2B application for direct sales companies based on heterogeneous and web-based technologies. Chapter 5 gives a summary and outlook to future developments and research directions.

2 Model Integration Concepts

Under *Enterprise Model Integration (EMI)* we subsume all tasks and concepts necessary for the integration of metamodels of modelling languages describing different aspects of a company. The following sections describe some basic concepts of EMI.

2.1 Object-Oriented Metamodelling

According to Falkenberg et al. [6], we define a model as *"a purposely abstracted, clear, precise and unambiguous conception. A model denotation is a precise and unambiguous representation of a model, in some appropriate formal or semi-formal language"*. The language for representing a model is described by its metamodel, i.e. the metamodel is a model of its corresponding modelling language [3, 12]. Applying language theory for levelling languages, the result is a hierarchy of models, meta-models etc. [21]. This paper focuses on the metamodel level. In the following the terms "modelling language" and "metamodel" will be used synonymously.

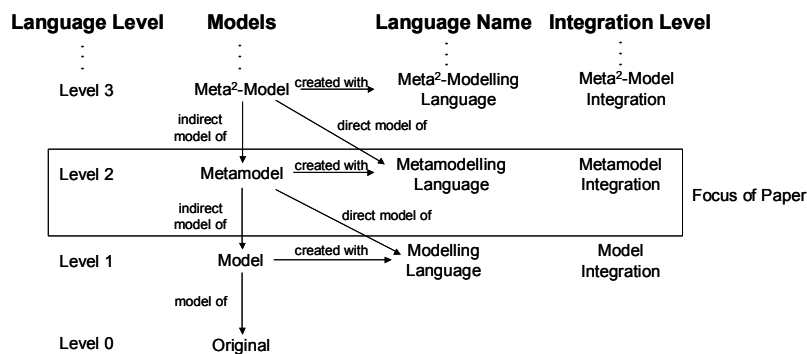


Fig. 2: Model Integration Levels

To enable the integration of models describing an enterprise, a necessary prerequisite is the integration of their underlying metamodels [13, 20]. For describing metamodels, we use an object-oriented metamodelling approach. The concepts of a modelling language are described by elements such as classes, relationships, attributes, and behaviour. These elements form a generalisation/specialisation hierarchy in which subordinated classes inherit common structure and behaviour from higher classes. Using view mechanisms, subsets of classes, relationships and attributes from the class hierarchy can be assigned to model types. A model type represents a concrete modelling language, e.g. a business process modelling language, a modelling language describing organisational structures or an information system modelling language. Linking and relating the model types, they form a set of interrelated modelling languages to describe a certain domain under consideration. For a detailed description of our object-oriented metamodelling approach see e.g. [9, 11].

For the integration of metamodels, we distinguish three major approaches: vertical, horizontal, and hybrid metamodel integration. These approaches will be briefly explained in the following sections using the levels described in fig. 1, which represent different views in metamodel integration.

2.2 Vertical Integration

Vertical integration of metamodels is a typical top-down or bottom-up integration approach (inter-level integration). Taking business goals and business strategies as starting points for application development, the top-down approach of EMI showed to be helpful. Metamodels from the business modelling level are integrated and refined with metamodels from the IT modelling level, e.g. business process metamodels are integrated with metamodels for data and function modelling. The metamodels of the IT modelling level themselves will be integrated and refined with metamodels of the target system level, e.g. metamodels for logical data modelling will be integrated with metamodels describing business objects of a concrete ERP system.

In re-engineering oriented modelling approaches (bottom-up), the existing target systems and IT modelling metamodels serve as starting points, which will be integrated with metamodels from the business modelling level, e.g. to bring models of legacy applications in relation to models of the business strategy.

2.3 Horizontal Integration

In the *horizontal integration approach*, metamodels of same level of detail are integrated (intra-level integration). This is the typical case when business partners, which form a supply or service chain, integrate their business processes, their applications, and their IT systems. Therefore, the metamodels of the modelling languages used have to be aligned and integrated on each modelling level. E.g. if one business partner uses a business process modelling language focusing on business events and the other partner uses a business process modelling language focusing on information flows, the concepts of event and information flow have to be integrated. A possible integration is that events can produce and consume information objects to represent the information flow.

2.3 Hybrid Integration

The *hybrid integration approach* is adequate if the horizontal and vertical integration directions are mixed and/or the modelling levels of the metamodels to be integrated differ noticeably. Hybrid integration can be useful for integrating metamodels of business partners which model their processes, systems etc. on different abstraction levels. Another example is the integration of metamodels of Balanced Scorecard approaches [10] with metamodels for data modelling to provide a starting point to gain quantitative data from runtime and data warehouse systems in the context of measuring mid-term and long-term business goals.

3 Metamodel Integration Patterns

To facilitate metamodel integration and to reuse experiences from former metamodel integration projects, we propose to use *metamodel integration patterns*. Some important patterns will be described in the following sections.

3.1 Patterns: Templates and Systems

The reuse of expert knowledge and experiences is of high interest in all areas of human work. Major advantages are savings in time and cost and improved quality of problem solutions. A possibility to make expert experiences explicit and reusable are so called "*patterns*". The basic idea of patterns comes from the architect Christopher Alexander, which captured and reused design experiences in architecture and civil engineering by using a pattern language [1]. The community of object-oriented software development adapted this idea for capturing software design experience in so called "design patterns" [4, 7].

We adapt the pattern idea of the before-mentioned authors to capture experiences in the area of metamodel integration. To describe a pattern, we use a *pattern template* consisting of five elements:

- *Name*: the name of the metamodel integration pattern.
- *Context*: a description of the situation an integration problem occurs.
- *Problem*: a description of the integration problem which has to be solved.
- *Solution*: a set of guidelines, instructions and rules to solve the problem.
- *Example*: a description of a typical application of the integration pattern.

A set of patterns describing a family of problem solutions for a given domain is called a *pattern system*. A pattern system consists of pattern descriptions and relationships between the patterns, describing their interdependencies. The following section presents our proposal of a pattern system for metamodel integration. Pattern structures will be described by UML class diagrams.

3.2 A Pattern System of Metamodel Integration Patterns

Our pattern system proposal for metamodel integration patterns consists of six patterns. These patterns are classified into loose integration patterns, intermediate integration patterns, and strong integration patterns. All patterns are applicable within the integration approaches explained in chapter 2. *Loose integration patterns* are used for metamodels which are largely complementary. Each of the metamodels can exist without depending on the existence of the other metamodel. *Intermediate integration patterns* are used to reuse parts of a source metamodel to build a new metamodel or to aggregate parts into an existing one. Some parts of the new metamodel can exist without depending on the source metamodel, other parts can not exist independently. *Strong integration patterns* are used to build a new metamodel completely depending on a source metamodel. The new metamodel can not exist independently of the source metamodel.

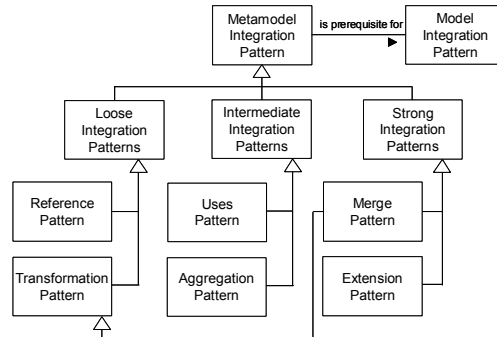


Fig. 3: Pattern System for Metamodel Integration

The *reference pattern* can be compared to a hyperlink which references parts of another metamodel. Applying the reference pattern results in navigation paths from one metamodel to another independent metamodel. The *transformation pattern* provides rules to transform parts of source metamodels to concepts provided by a target metamodel. This pattern can be used e.g. for describing transformation rules between Platform Independent Models (PIM) and Platform Specific Models (PSM) in the MDA infrastructure [16]. The *uses pattern* enables the usage of a part of a source metamodel in a target metamodel without redefining the part in the target metamodel. The *aggregation pattern* aggregates one or more parts from one or more source metamodels in a new concept in a target metamodel. The *merge pattern* takes one or more parts from one or more source metamodels and merges them into a new concept in a target metamodel. The *extension pattern* enlarges a part of a metamodel with new concepts to broaden the expressiveness of the metamodel.

The following sections will describe the reference, transformation, and merge pattern in more detail.

3.3 Reference Pattern

Name: Reference Pattern

Context: Two or more metamodels are complementary. The integration should support loose coupling and navigation between the complementary parts.

Problem: The integration of two or more metamodels should not change the original metamodels. Each metamodel should be independent from each other to enable the further development and improvement of each metamodel without influencing the other metamodels.

Solution: A reference link connects exactly one part of the source metamodel with one part of the target metamodel. A reference link has at least one reference domain which describes the possible source and target parts to connect.

Example: Integration of a business process metamodel with a use case metamodel to refine activities of business processes to use cases.

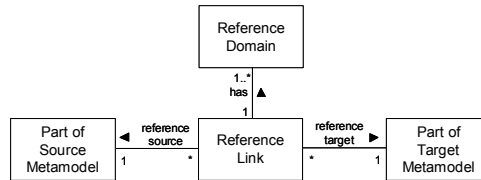


Fig. 4: Reference Pattern

3.4 Transformation Pattern

Name: Transformation Pattern

Context: Two or more metamodels are complementary. The integration should support loose coupling, but parts of the target metamodel are generated from parts of the source metamodels.

Problem: The integration of two or more source metamodels should not change these metamodels. Each source metamodel should be independent from other source metamodels and from the target metamodel, but the target metamodel depends on the source metamodels. Each source metamodel should be developed and improved independently, but the target metamodel should be changed accordingly, if necessary.

Solution: A transformation rule generates exactly one part of the target metamodel from one or more parts of the source metamodels. The transformation rule consists of at least one transformation action and none, one or more transformation constraints. The transformation constraints determine the behaviour of the transformation actions.

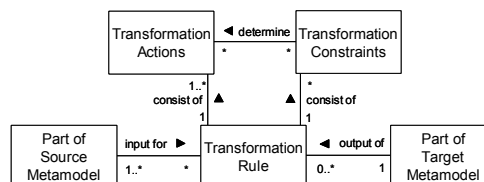


Fig. 5: Transformation Pattern

Example: Activities in business processes are executed by resources such as human or technology-based actors. The business process metamodel contains an assignment rule to relate the relevant parts of the resource metamodel with the business process metamodel. The assignment rule is automatically updated by a transformation rule after changing the resource metamodel.

3.5 Merge Pattern

Name: Merge Pattern

Context: Two or more metamodels are used concurrently. They describe modelling languages ranging from nearly equivalent up to orthogonal expressiveness. To reduce

the effort in maintaining the metamodels or to use their expressiveness within an integrated modelling language, the relevant parts should be merged into a single metamodel.

Problem: By integrating different parts of source metamodels into one part of the target metamodel, no syntactic, semantic or notational expressiveness should be lost or misinterpreted. The source metamodels should not be changed.

Solution: The merge pattern is a specialisation of the transformation pattern. A merge rule generates exactly one part of the target metamodel from one or more parts of the source metamodels. The merge rule consists of at least one merge action and none, one or more merge constraints. Three types of merge constraints can be distinguished, which determine the behaviour of the merging: syntactic, semantic, and notational constraints.

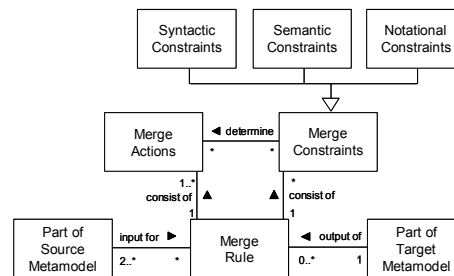


Fig. 6: Merge Pattern

Example: A data modelling language and a business process modelling language should be merged to enable data flow modelling. The resulting language should still have the former properties of data and process modelling.

4 Case Study: Development of a Direct Sales B2B Application

Case Study Description

Companies in the direct sales business use Internet technologies either to reduce their costs of administration or to establish new strategies for their world wide sales network. Usually, regional sales partners of world wide acting sales companies established different business processes and information systems over the years. The main reasons are legal requirements, varying regulations in different markets, diversified product bundles, and logistical constraints. If such a company decides to develop a new standardised sales application based on ERP, a large set of notations and application scenarios have to be integrated in a common metamodel.

The following case study presents the realisation of a *B2B application of a direct sales company* developed on an Internet technology based ERP system. Modelling the business details of the considered sales processes in different countries requires a modelling language designed for the use of business experts. The language for modelling the customer-specific requirements of the processes to be implemented from the IT point of view is the proprietary modelling language of the ERP system. This modelling language is EPC (event driven process chains), in which the reference

modelling language is EPC (event driven process chains), in which the reference processes of the ERP system are described. Based on a first compliance check if the functionality of the ERP system will be sufficient to implement all requirements, it was decided to enhance the functionalities with customer-specific application components. UML use case and class diagrams were integrated into the metamodel to be used to analyse and design the particular domain object model and the additional application components. Fig. 7 shows an overview of the different metamodels which have to be integrated and merged. The following sections describe the integration in more detail using the levels of fig. 1.

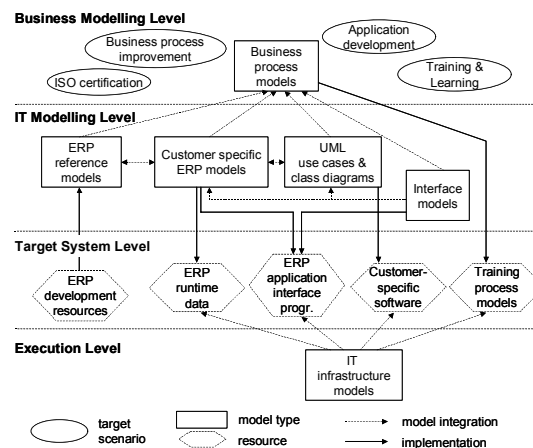


Fig. 7: EMI architecture of case study

Applying EMI on Business Level

In the past, the subsidiaries of the direct sales company modelled business processes in each country with different modelling languages and for different target scenarios such as ISO certification, business process improvement, and legacy systems documentation. The result of applying the EMI approach with *horizontal integration* and using *transformation and merge patterns* was a new metamodel on the business level which integrated all modelling languages and requirements of the before-mentioned application scenarios. The new metamodel supports requirements definition, modelling concepts for quality management, business process-based training&learning, and mechanisms for managing quantitative aspects of the business processes such as cycle times, workloads etc.

Applying EMI on IT level

After deciding to use ERP technology for implementing the new direct sales application, ERP experts developed a compliance check to match business requirements with system functionality. The compliance check is based on business process details and the ERP reference models. For this reason the underlying modelling languages had to be *integrated vertically*. By applying the *reference pattern* using references from business activities to EPC functions and their application context, the business process metamodel and the EPC metamodel were integrated. Each function of an

EPC represents a set of calls and transactions in the ERP system. The extended metamodel supports detailed compliance checks, documentation of customising know-how, and the definition of the ERP system requirements. Applying the compliance checks, i.e. comparing the as-is functionality and the to-be requirements, it was identified that the ERP system lacks components needed for the direct sales specific commission system. The company decided to develop these components with its own development staff using an object-oriented development environment. Therefore, UML use case and class diagrams were integrated in the metamodel by using *reference patterns*. Additionally, a new model type was designed for modelling the interfaces between the ERP application components, new commission components, and existing legacy systems. The *horizontal integration* of this model type on the IT level implied interdependencies to EPCs and use cases. The main advantages of this integration are the comprehensive requirements definition, the documentation of business process details, and the corresponding conversion in system functionalities.

Applying EMI on Execution Level

At the early beginning of the project the IT strategy and the corresponding IT infrastructure were set up. Administrative processes specify the way releases are uploaded in the production environment. These processes and the responsible actors are linked with the corresponding environment in the IT infrastructure model. Additionally, the infrastructure model is used to manage the complexity of IT operations. As shown in fig. 7 the infrastructure model was *integrated vertically* by using the *reference pattern*.

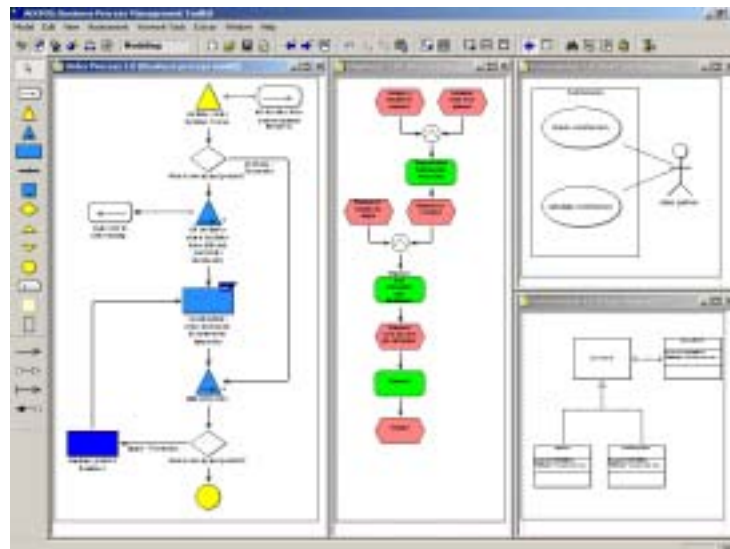


Fig. 8: Integrated Modelling Languages of the Case Study

Realisation within the meta model management tool ADONIS

The target metamodel was realised within the meta model management tool ADONIS. In fig. 8 instances of the model types "business process model", "EPC",

"use case diagram" and "class diagram" are modelled and linked according to the integrated metamodel.

5 Summary & Outlook

Enterprise Model Integration (EMI) has shown to be a valuable aid in B2B application development projects. The diversity of modelling languages, programming languages, target systems, development methodologies, and standardisations, led us to a flexible metamodel integration approach using object-oriented metamodeling concepts and technologies. The major advantages from our experiences using EMI are considerable savings in time and costs in application development, increased quality of delivered solutions, and enhanced acceptance because of directly mapping the domain under consideration. The usage of metamodel integration patterns helped us to standardise the procedures of metamodel integration. This was especially helpful in handling situations where the steps to follow integrating heterogeneous metamodels were not always obvious for the metamodeler. E.g. integrating similar metamodels and how to select the necessary parts of each metamodel to have a minimal, but complete, metamodel.

We are currently working on concepts and mechanisms to integrate on model level (level 1 in fig. 2). There, we expect increased usage of best practices and reference models to reduce modelling effort and to improve optimisation and benchmarking.

Another trend in enterprise modelling we expect is the combination of different modelling paradigms such as prescriptive, descriptive and decision-oriented approaches to an integrated modelling approach. An example is the integration of Balanced Scorecard languages with IT oriented languages.

Modelling always means effort in time and costs. Nevertheless, to handle complexity in business and application engineering, we see models moving from some kind of "luxury article" to an "everyday necessity".

References

1. Alexander, C.; Ishikawa, S.; Silverstein, M.: A Pattern Language: Towns, Buildings, Construction. Oxford University Press, 1977.
2. Atkinson, C.; Kühne, T.: The Role of Metamodeling in MDA. In: Bezivin, J.; France, R. (Eds.): Proceedings of the International Workshop in Software Model Engineering (in conjunction with the UML'2002), Dresden, Germany, October 2002.
3. Brinkkemper, S.; Lyytinen, K.; Welke, R. J. (Eds.): Method Engineering. Principles of method construction and tool support. Proceedings of the IFIP TC8, WG8.1/8.2 Working Conference on Method Engineering, Atlanta, USA, August 1996. Chapman & Hall, 1996.
4. Buschmann, F.; Meunier, R.; Rohnert, H.; Sommerlad, P.; Stal, M.: Pattern-Oriented Software Architecture – A System of Patterns. John Wiley & Sons, 1996.
5. ebXML: ebXML Homepage. <http://www.ebxml.org>, access 2003-03-08.
6. Falkenberg, E. D.; Hesse, W.; Lindgreen, P.; Nilsson, B. E.; Oei, J. L. H.; Rolland, C.; Stamper, R. K.; van Assche, F. J. M.; Verrijn-Stuart, A. A.; Voss, K.: The FRISCO Report:

In: Bauknecht, K.; Tjoa, A. M.; Quirchmayr, G. (Eds.): Proceedings of the 4th International Conference EC-Web 2003 – DEXA 2003, Prague, Czech Republic, September 2003, LNCS 2738, pp. 379-392.

- A Framework of Information Systems Concepts. International Federation of Information Processing, WG 8.1 Task Group FRISCO, 1996.
7. Gamma, E.; Helm, R.; Johnson, G.; Vlissides, J.: Design Patterns – Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994.
 8. Johannesson, P.; Wangler, B.; Jayaweera, P.: Application and Process Integration – Concepts, Issues, and Research Directions. In: Brinkkemper, S.; Lindencrona, E.; Solvberg, A. (Eds.): Information Systems Engineering Symposium CAiSE 2000, Springer-Verlag, 2000.
 9. Junginger, S.; Kühn, H.; Strobl, R.; Karagiannis, D.: Ein Geschäftsprozessmanagement-Werkzeug der nächsten Generation – ADONIS: Konzeption und Anwendungen. In: WIRTSCHAFTSINFORMATIK, Vol. 42, Nr. 5, Vieweg-Verlag, 2000, pp. 392-401.
 10. Kaplan, R. S.; Norton, D. P.: The Balanced Scorecard: Translating Strategy into Action. Harvard Business School Pr., 1996.
 11. Karagiannis, D.; Kühn, H.: Metamodelling Platforms. Invited Paper. In: Bauknecht, K.; Min Tjoa, A.; Quirchmayer, G. (Eds.): Proceedings of the 3rd International Conference EC-Web 2002 – Dexa 2002, Aix-en-Provence, France, September 2002, LNCS 2455, Springer-Verlag, p. 182, (full version: <http://www.dke.univie.ac.at/mmp>).
 12. Kelly, S.; Lyytinen, K.; Rossi, M.: MetaEdit+ - A Fully Configurable Multi-User and Multi-Tool CASE and CAME Environment. In: Constantopoulos, P.; Mylopoulos, J.; Vassiliou, Y. (Hrsg.): Advanced Information System Engineering. Proceedings of 8th International Conference, CAiSE'96, Heraklion, Crete, Greece, May 1996, LNCS 1080, Springer-Verlag, 1996, pp. 1-21.
 13. Kronlöf, K. (Ed.): Method Integration – Concepts and Case Studies. John Wiley & Sons, 1993.
 14. Kühn, H.; Junginger, S.; Bayer, F.: How Business Models Influence the Development of E-Business Applications. In: Stanford-Smith, B.; Kidd, P. T. (Eds.): Proceedings of the eBusiness and eWork 2000, Madrid, Spain, October 2000, IOS Press, pp. 1024-1030.
 15. Kühn, H.; Junginger, S.; Bayer, F.; Petzmann, A.: Managing Complexity in E-Business. In: Baake, U. F.; Herbst, J.; Schwarz, S. (Eds.): Proceedings of the 8th European Concurrent Engineering Conference 2001 (ECEC'2001), Valencia, Spain, April 2001, SCS, pp. 6-11.
 16. OMG Object Management Group: Model Driven Architecture (MDA). Document number ormsc/2001-07-01. <http://www.omg.org/cgi-bin/doc?ormsc/01-07-01.pdf>, access 2003-03-08.
 17. OMG Object Management Group: OMG Unified Modeling Language Specification, Version 1.4. September 2001. <http://www.omg.org/cgi-bin/doc?formal/01-09-67.pdf>, access 2003-03-08.
 18. OMG Object Management Group: Meta Object Facility (MOF) Specification, Version 1.4. April 2002. <http://www.omg.org/cgi-bin/?formal/02-04-03.pdf>, access 2003-03-08.
 19. RosettaNet: RosettaNet Homepage. <http://www.rosettanet>, access 2003-03-08.
 20. Sprinkle, J. M.; Karsai, G.; Ledeczi, A.; Nordstrom, G.: The New Metamodeling Generation. In: Proceedings of the 8th Annual IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, Washington, D. C., April 2001.
 21. Strahinger, S.: Metamodellierung als Instrument des Methodenvergleichs. Eine Evaluierung am Beispiel objektorientierter Analysemethoden. Shaker-Verlag, 1996.
 22. Vernadat, F.: Enterprise Modeling and Integration – Principles and Applications. Kluwer Academic Publishers, 1996.
 23. WfMC Workflow Management Coalition: Workflow Process Definition Interface – XML Process Definition Language. WfMC-TC-1025. <http://www.wfmc.org>, access 2003-03-08.
 24. W3C World Wide Web Consortium: W3C Homepage. <http://www.w3c.org>, access 2003-03-08.