UNIVERSITÄT
KOBLENZ · LANDAU

Institut für
Wirtschaftsinformatik

Fachbereich Informatik
Universität Koblenz-Landau

ULRICH FRANK

MEMO: VISUAL LANGUAGES FOR ENTERPRISE MODELLING

Juni 1999

Arbeitsberichte des Instituts für Wirtschaftsinformatik Nr. 18

UNIVERSITÄT
KOBLENZ · LANDAU

Institut für
Wirtschaftsinformatik

Fachbereich Informatik
Universität Koblenz-Landau

ULRICH FRANK

MEMO: VISUAL LANGUAGES FOR ENTERPRISE MODELLING

Juni 1999

Arbeitsberichte des Instituts für Wirtschaftsinformatik Nr. 18

Die Arbeitsberichte des Instituts für Wirtschaftsinformatik dienen der Darstellung vorläufiger Ergebnisse, die i.d.R. noch für spätere Veröffentlichungen überarbeitet werden. Die Autoren sind deshalb für kritische Hinweise dankbar.

The "Arbeitsberichte des Instituts für Wirtschaftsinformatik" comprise preliminary results which will usually be revised for subsequent publications. Critical comments would be appreciated by the authors.

Alle Rechte vorbehalten. Insbesondere die der Übersetzung, des Nachdruckes, des Vortrags, der Entnahme von Abbildungen und Tabellen - auch bei nur auszugsweiser Verwertung.

All rights reserved. No part of this report may be reproduced by any means, or translated.

Anschrift des Verfassers/ Address of the author:

Prof. Dr. Ulrich Frank
Institut für Wirtschaftsinformatik
Universität Koblenz-Landau
Rheinau 1
D-56075 Koblenz

Arbeitsberichte des Instituts für Wirtschaftsinformatik Herausgegeben von / Edited by:

Prof. Dr. Ulrich Frank
Prof. Dr. J. Felix Hampe

©IWI 1999

Bezugsquelle / Source of Supply:

Institut für Wirtschaftsinformatik
Universität Koblenz-Landau
Rheinau 1
56075 Koblenz

Tel.: 0261-287-2520
Fax: 0261-287-2521
Email: iwi@uni-koblenz.de
WWW: http://www.uni-koblenz.de/~iwi

Abstract

Enterprise models provide various abstractions that help with the design of corporate information systems which are in line with a company's organisation and its long term strategy. At the same time an enterprise model can be instantiated into a corporate knowledge base. Different from other methods for enterprise modelling, MEMO puts special emphasis on modelling languages. The visual languages provide intuitive abstractions for various observers. Against the background of the requirements imposed by enterprise modelling, the paper presents an extensible framework for specialised modelling languages and their reconstruction for an integrated design environment. The languages are defined in metamodels which in turn are instances of a common meta-metamodel. Similar to a technical language, they provide concepts that help with analysing and structuring a domain with respect to a specific task. The languages share common concepts which allow for a tight integration of the various parts of an enterprise model. To give an impression of the language definitions within MEMO, one particular language, the MEMO Organisation Modelling language, is described in more detail.

1. Introduction

Planning, designing, introducing, and maintaining corporate information systems is a complex endeavour. More than demanding a deep understanding of a company's current situation, it has to be taken into account that introducing advanced information technology allows for or may require new ways to target and organize the business - an aspect that has been stressed emphatically by numerous authors who recommend "business redesign" or "business process redesign" [Dav93].

Traditionally, there are various specialised professionals who deal with certain aspects of this complex task such as strategic planning, organisational analysis and design as well as software development. Like system design, analysing and redesigning a corporate strategy and a company's organisation respectively are complex tasks on their own. Management Science and organisational theory offer a wide range of dedicated approaches for analysing and shaping a firm's strategy as well as for organisational (re-) design. Often they are based on graphical models which are introduced to illustrate essential concepts and interrelations - and to communicate them to others who should be involved. Organisational models cover a wide range from rather prosaic to more formal representations. This is similar to models for strategic planning. They usually stress a more abstract view with highly aggregated data (for an overview see [Has92], [Sco86]). Strategic and organisational models are usually based on different concepts. Furthermore, they have, in general, nothing in common with conceptual models used in software engineering.

While there is certainly need for specialisation, such a separation of concerns can be seen as a major inhibitor of efficient information systems:

> "I see the artificial split between organizational and technical issues as dangerous and unnecessary, and the frequent cultural chasm between business people and information technology professionals as the one factor that can block the effective use of computers and communications." [Kee91]

The term "enterprise modelling" has been introduced ([Zac87], [Kat90]) in order to emphasize the need for a multi perspective approach. The basic idea is to model different views on a company and to allow for a seamless integration of the partial models. While there has been a substantial amount of work on enterprise modelling (for an overview see [Pet92], [Oll+91], [Fra97]), there is hardly a coherent state of the art. Usually the corresponding approaches remain on a rather abstract level. They mainly provide a set of views on the enterprise (like "data", "function", "people" etc., [SoZa92], [Zac87]) - usually without specifying modelling languages to represent the particular views. Other approaches are based on software development methods ([Jac+94], [Hen94]). They do not include specific concepts from organisational theory or management science. More elaborated approaches - like CIM/OSA ("Open System Architecture", [Gor92], [ESP93]) or ARIS [Sch94] - offer frameworks for information system architectures. However, except for a semi-formal language to model business processes that is part of ARIS, they lack specific modelling languages. Often they suggest to use entity relationship models.

MEMO ("Multi Perspective Enterprise Modelling") is a method for enterprise modelling that offers a set of specialised visual modelling languages together with a process model as well as techniques and heuristics to support problem specific analysis and design. The languages allow to model various interrelated aspects of an enterprise. They are integrated on a high semantic

level. MEMO models are to serve two goals. Firstly, they are an instrument to develop information systems that are well integrated with a company's strategy and its organisation. Secondly, they can be used as the foundation of an "enterprise schema". Its instantiation would allow for a permanent representation of all relevant aspects of an enterprise (strategy, business processes, organisational structure, business entities, business rules etc.), hence serving as an "organisational memory" [Ack94] or a corporate knowledge base. In this paper, we will focus on the modelling languages provided by MEMO.

2. Visual Languages for Enterprise Modelling: Requirements

A modelling language is an *instrument*, not an end in itself. That recommends to look at the requirements that are associated with the notion of enterprise modelling. The basic idea of enterprise modelling is to offer *different views* on an enterprise. The views should complement each other and thereby foster a better understanding of complex systems by systematic abstractions. The views should be *generic* in the sense that they can be applied to any enterprise. At the same time they should offer abstractions that help with designing information systems which are well integrated with a company's long term strategy and its organisation.

A model that is associated with a particular view should provide a medium for communication. It should also support analysis and (re-) design of the subject that is being modelled. Therefore a corresponding modelling language should provide *intuitive concepts* that are, at the same time, suited to structure the problem domain in a meaningful way. A concept can be regarded as intuitive if it corresponds directly to the observer's perception and conceptualisation. While those individual preferences are hard to identify - and may vary in a wide range, it is a good idea to (re-) use *existing concepts* that have already proved themselves. Those concepts can be found in specific terminologies that are common within a particular view. For instance: A language for information modelling should provide concepts software engineers are familiar with. Since the visualisation of a model may also contribute to a better understanding, the modelling language should provide a graphical notation that offers graphical symbols which are well known in the associated domain.

Enterprise modelling also aims at the *integration of the partial models* that represent particular views on an enterprise. In order to fulfil this requirement, the languages that are used to describe the partial models should share common concepts. The higher the level of semantics that is provided by those common concepts the tighter the integration. For instance: If two languages share a common notion of an integer, they are less integrated than two languages that share a common notion of an application level concept. Enterprise models tend to be very complex. That recommends the use of tools which support the development and management of models. Without tools it will be hardly possible to keep a model consistent over time. In addition to that, a tool supports search and navigation. Finally, a tool is mandatory with respect to simulation, model execution and code generation. In order to support the construction of adequate modelling tools, a language description should be *sufficiently formalized*. In other words: The language description should fulfil formal requirements such as completeness, simplicity, and correctness (see [SüEb97], pp. 2). Additionally, the language designers should take into account how a language description can be mapped to models that are used for the design of tools.

3. MEMO: Conceptual Foundation

The development of MEMO started in 1993 as an interdisciplinary research project at the German Research Center for Computer Science, GMD [Fra97]. It was motivated by the vision of generic enterprise models that can be (re-) used by many companies as a reference to build information systems which are effectively integrated with a company's organisation and its long term strategy. For this purpose, it was necessary to identify appropriate abstractions of an enterprise. While it is common sense that enterprise models should satisfy different views on an enterprise, it is not evident how many views are appropriate and how they should be conceptualized. ARIS offers four different views: an organisational view, a data view, a process view, and a control view ([Sch94]). The framework suggested within CIM-OSA (ESP92] consists of four views ("organisation", "resource", "information", "function"), each of which is differentiated in three levels of abstraction: "generic", "partial", and "particular".

MEMO differentiates three so called *perspectives* - strategy, organisation and information system - each of which is structured by four *aspects*: structure, process, resources, goals. A particular aspect within a perspective is called a *focus* - for instance "process" within "information system". One or more foci correspond to a particular model. For instance: an organisation model serves to represent an organisation structure, business processes as well as related resources and goals. The focus "structure" within the information system perspective is represented by an object model. A strategy model renders the structure (strategic business units) and the dynamic aspects (value chain) of a corporate strategy. Fig. 1 gives an overview of the relationships between selected partial models.

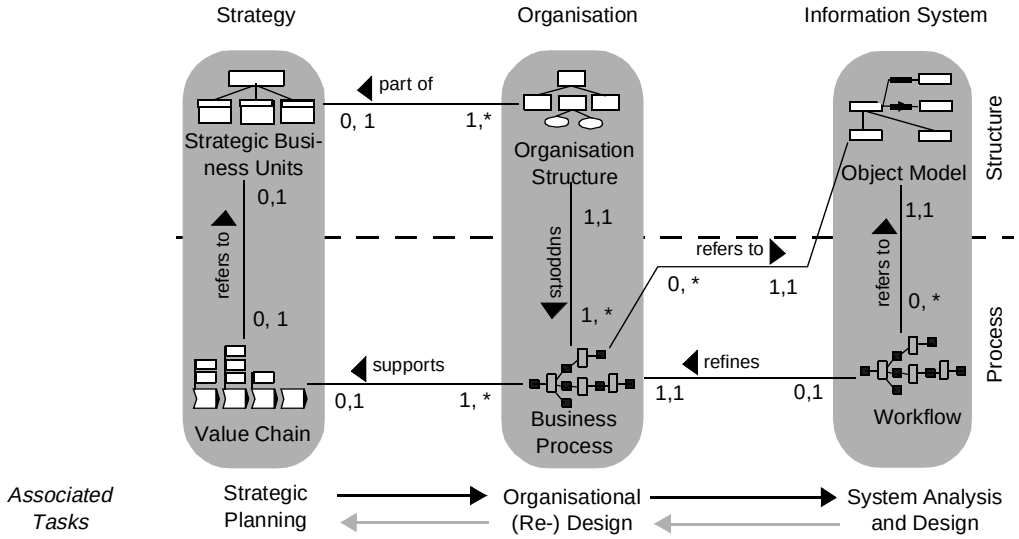


Fig. Fig. 1: Relationships between selected partial models

With respect to the integration of the partial models, it would be helpful to use one language only that would allow to describe any model. The entity relationship model (ERM) could be regarded as a potential candidate. However, a general formal or semi-formal language is not satisfactory for a number of reasons. Some candidates do not provide the expressive power that is required. The ERM, for instance, does not include any concepts to express temporal semantics or specific integrity constraints. This is different with object-oriented modelling languages

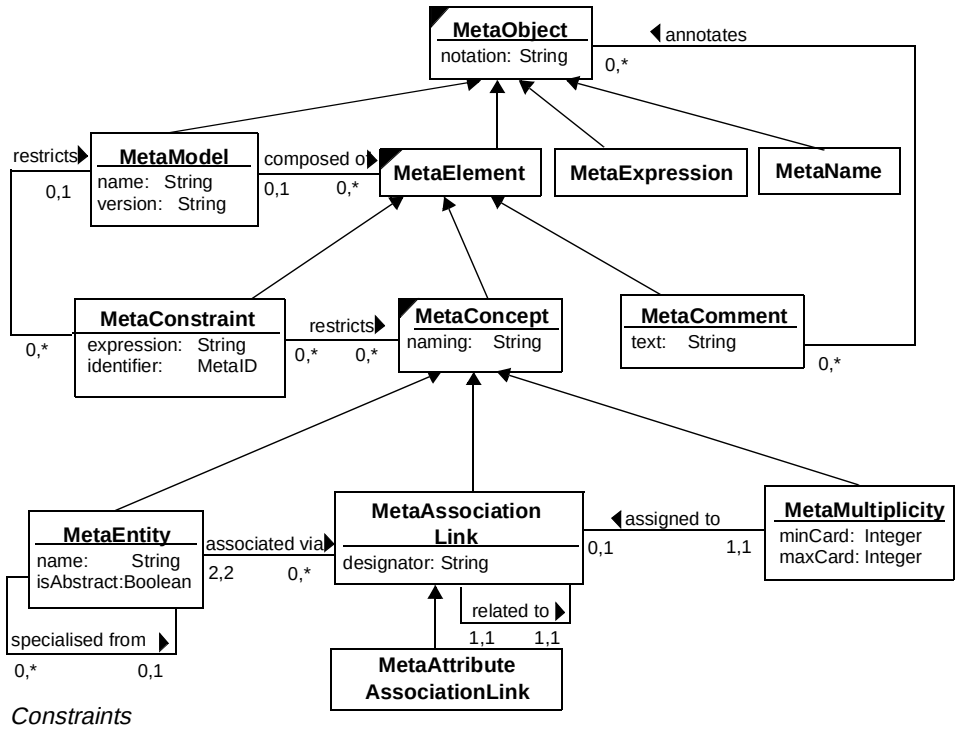
like UML [Rat97] or OML [FiHe96]. However, those languages are designed for the development of software systems. Therefore they do not provide graphical representations that are appropriate for all aspects of an enterprise model. For instance: the visualisation of a business plan should be different from the visualisation of an object model; a state transition diagram is not appropriate for the visualisation of a business process. There is, however, one even more relevant argument against a single language approach. More specific languages serve as an instrument to structure and analyse a domain of interest - very much like a technical language. Different from of a general purpose modelling language, they provide the modeller with concepts that have proved to be useful for certain tasks. For this reason the modeller does not have to reconstruct specific concepts from more generic ones. Hence, a specialised modelling language promotes the *reuse* of modelling artefacts. Therefore it fosters the economics of modelling. In addition to that, specialised modelling languages contribute to the integrity of models: Compared to general language concepts, specialised concepts have to be used in a more restricted way. That improves the chances to check a model's integrity on a syntactic level. To give a simple example: If you specify the concept "Organisational Unit" with an object-oriented modelling language, the language itself would not exclude that the association "responsible for <OrganisationalUnit>" must not be cyclic (instead, you would have to add an explicit constraint to the model). A language specialised for representing organisations could be enriched with the concept "Organisational Unit" in a way that would prevent an inconsistent use like in the above example. Despite this advantage of special purpose modelling languages, they are accompanied by a big challenge at the same time: The more specialised the concepts of a language, the less are the chances to use them in specific contexts ("over-specialisation"). That recommends to carefully refine domain level concepts before "freezing" a language.

The previous thoughts are reflected by two essential aspects of MEMO: a framework for the specification and integration of modelling languages and a corresponding research strategy. The framework is exensible in the sense that it allows for the specification of additional languages. It also takes into account the development of corresponding modelling tools. A language can be specified by a grammar or by a metamodel. Although grammars offer better means to check a model's syntax, we decided for graphical metamodels which are enhanced by textual constraints. This is mainly for two reasons. Using a grammar for the specification of a graphical modelling language would imply a paradigm shift between meta and object level. That would probably make it more difficult for language users to understand the language description. With respect to the development of modelling tools, meta models make sense, too: Typically, tools are designed with graphical models, like object models. Since there is no paradigm shift, mapping a metamodel to an object model can be done in a straightforward approach.

All languages within MEMO are specified with concepts defined in a common meta-metamodel (fig. 2). For a detailed description and a comparison with other meta-metamodels see [Fra98a]. The graphical representation of the meta-metamodel is supplemented by constraints which are defined in GRAL, a language that allows to specify constraints on so called TGraphs. A TGraph is a directed graph composed of vertices and edges. Both, vertices and edges, are typed and may have attributes. GRAL comes with a library of predicates typically needed to express properties of graphs (e.g.: "isAcyclic (G)", "isNeighbourOf (G,v,w)"). Additional predicates can be defined using the specification language Z. In order to specify first order predicates on TGraphs (and/or vertices and edges) it is required to navigate the graph on any path that might be of relevance for a particular constraint The GRAL expressions in fig. 2

and fig. 4 only serve to illustrate the use of the language. For a detailed specification see [Frz97]. We prefered GRAL over other candidates (like [Sch90], [Gog+93]) because it has been developed by a research group we are closely cooperating with.

.



Constraints

Specialisations must not be cyclic.

$$\forall me{:}V_{MetaEntity} \bullet \neg(me(\rightarrow_{specialisedfrom})^{+} cpt)$$

A minimum cardinality has to be less than or equal to the corresponding maximum cardinality.

$$\forall mmp{:}V_{MetaMultiplicity} \bullet (mmp.\min Card <= mmp.\max Card\ )$$

A MetaAttributeAssociationLink must not be associated with a MetaAttributeAssociationLink.

$$\forall mal{:}V_{MetaAttributeAssociationLink} \bullet (mal \rightarrow_{relatesto} \rightarrow mal) = \varnothing$$

The MetaMultiplicity assigned to the MetaAssociation a MetaAttributeAssociation is associated with, must have both its attributes, minCard and maxCard, set to 1.

$$\forall mmp{:}V_{MetaMultiplicity} \bullet (mmp \rightarrow_{assignedto \bullet MetaAssociationLink} \leftarrow_{relatedto \bullet MetaAttributeAssocLink}) \bullet mmp.\min Card = 1 \bullet mmp.\max Card = 1$$

Association

Generalisation

Fig. Fig. 2: MEMO meta-metamodel. The constraints are specified in Gral [Frz97]

In order to prepare for the development of a tool environment, every metamodel is reconstructed as an object model which is defined in MEMO-OML (Object Modelling Language, [Fra98b]). Although an object model representing a language is usually very similar to the corresponding metamodel, it is not always a trivial mapping. MEMO-OML offers much more

concepts (for instance: multiple inheritance vs. single inheritance, aggregation, delegation [Fra99], services ...) than the meta-metamodel. Therefore, it is sometimes useful to reconstruct a metamodel with more elegant or more appropriate concepts. Furthermore, an object model also includes additional specifications that are required for certain features of a tool - like user management or version control.Although we are aware of UML and the relevance of standards, we decided to develop MEMO-OML - mainly because we are not satisfied with UML (for an evaluation see [FrPr97]). The various object models are integrated into one common object model. *Conceptual* integration is accomplished by those concepts that are shared by different languages. For instance: The concept “BasicService” (within a class) is part of both, the MEMO-OML and the MEMO-OrgML (Organisation Modelling Language, see fig. 4).

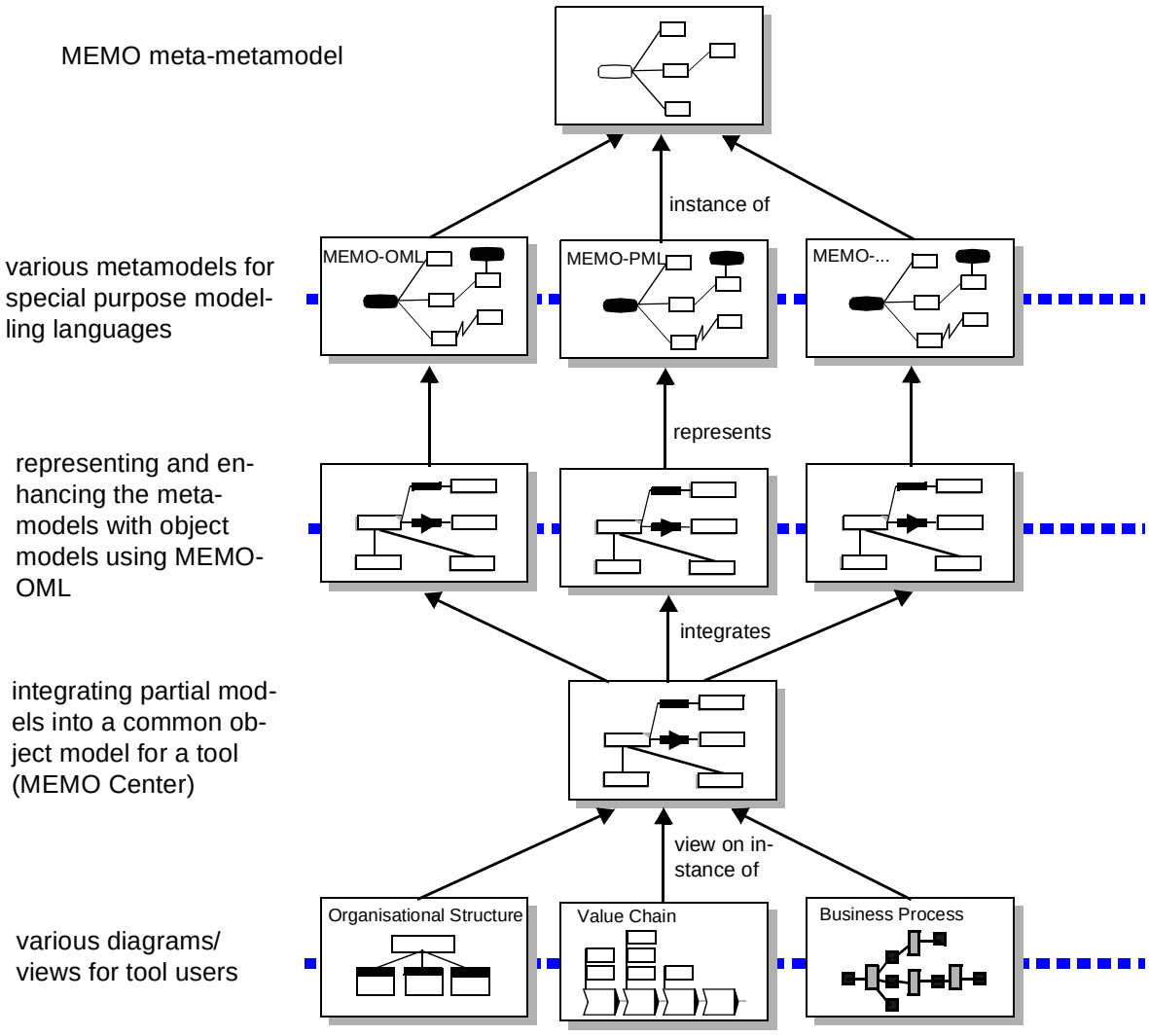


Fig. 3: Framework of MEMO Modelling Languages and their Implementation in a Tool Environmen

Instances of the common object model are managed in an integrated modelling environment (MEMO Center) that provides various views - each of which corresponds to a particular mod-

elling language - on an enterprise model. Besides conceptual integration, MEMO Center also allows for *annotational* integration: Whenever there is a relationship between particular parts of two different models which cannot be expressed in terms of (semi-) formal concepts, the user can connect them so that they annotate one another. For instance: Within a strategy model there may be the goal “customer orientation”. It corresponds to a business rule within the organisation model: “A business process should be managed by exactly one employee.” The rule would be expressed by assigning a corresponding multiplicity which could be associated with the goal in the strategy model. Different from conceptual integration, annotational integration does not include any formal semantics. It simply allows to indicate a relationship which requires human interpretation. Fig. 3 illustrates the different levels of the MEMO language framework.

From a methodological point of view, the specification of modelling languages is a delicate task. This is due to the fact that the requirements (see 2) cannot be specified in a comprehensive way: Some of them depend on subjective preferences which may vary from person to person and over time. Therefore it is impossible to optimise a modelling language straight off. Instead a language has to be evaluated by prospective users against the purpose it should serve. Since the preferences (as well as the modelling purpose) of users may vary over time (the more familiar they become with a language), there may be several refinements of the original specification. For this reason, we decided for the following research strategy. First, a modelling language within MEMO is specified in a semi-formal way applying the concepts provided by the MEMO meta-metamodel and additional natural language constraints. Then the language will be tested by various users. Depending on their feedback, the language will be refined step by step. Only when a language seems to have reached a mature state, we would formalize it. Formalisation is based on the formalisation of the meta-metamodel and the formalisation of the natural language constraints within a particular metamodel. Currently, only MEMO-OML has been completely formalised [Zic99] with GRAL.

The same considerations that apply to the definition of the abstract syntax and semantics of a language are valid for the graphical notation (concrete syntax), too. While there are some general requirements, a graphical notation should fulfil (for instance: [Rum96]), the evaluation of a particular notation depends mainly on individual preferences which are hard to identify. At present time, the notations used within the MEMO languages follow wide spread graphical representations in the respective domains. For instance: An organisational structure is visualised as an organisational chart, a corporate strategy is shown as a value chain similar to Porter’s suggestion [Por85], the visualisation of an object model is similar to those of UML or OMT. To take into account varying user preferences, MEMO Center allows to modify the notation of a language by altering graphical symbols.

4. An Example: The MEMO-OrgML

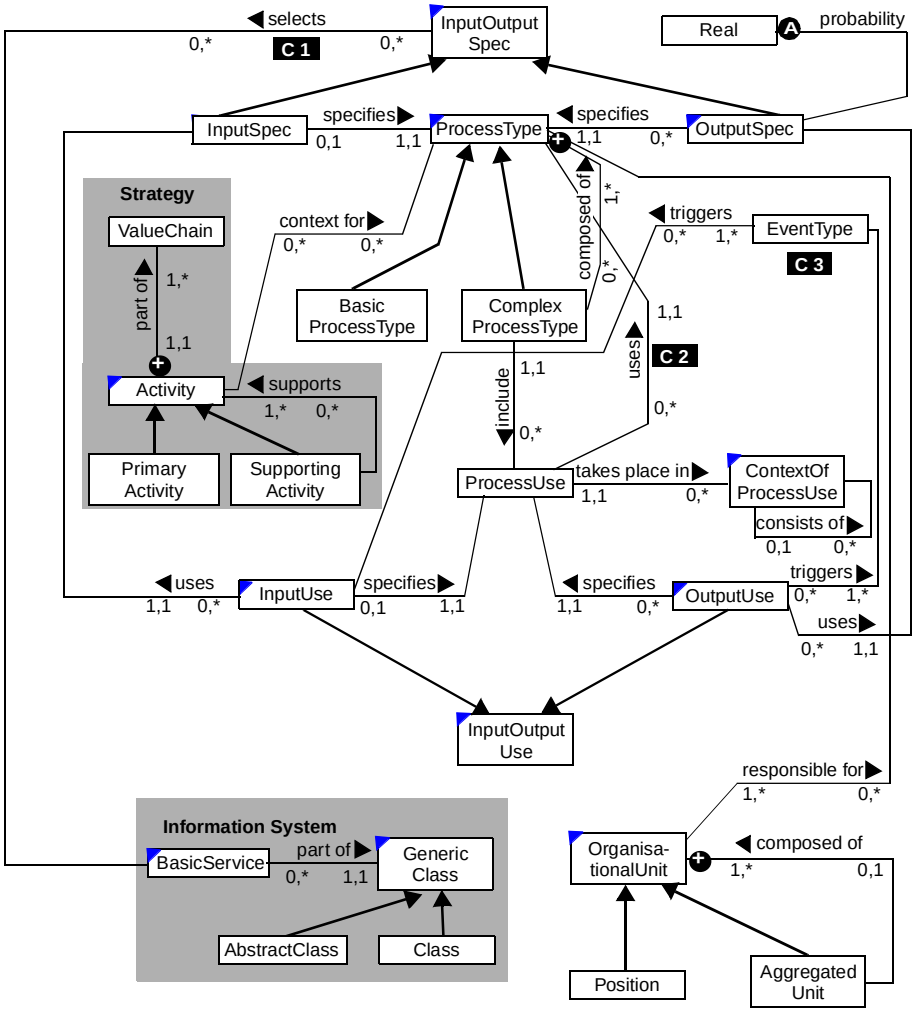
To give an impression of the language definitions within MEMO, we will look at one particular language in more detail. MEMO-OrgML serves to model a company’s organisation. For this purpose it provides concepts to describe the organisational structure, business processes and resources (such as machinery and personnel) that are required to perform the business processes. Modelling a company’s organisation before designing an information system is based on the assumption that often an organisation has to be redesigned in order to take full advantage of the potential offered by information technology. In order to contribute to a common under-

standing of the business, an organisational model should serve as a medium to foster the communication with domain experts. In addition to that, an organisational model should prepare for the design of an information system. The basic idea of organisational analysis and design is to focus on business processes first. It isbased on the assumption that business processes are the key concept to analyse and improve a company's performance - in terms of customer orientation and competitive edge.

MEMO-OrgML provides concepts to model a business process in a detailed way. This is different from the use case approach [Jac+94] that is mainly based on natural language descriptions and therefore offers only little help with structuring a process. Formal languages for process modelling - like Petri nets or process algebra - on the other hand lack expressive, domain level concepts. The key concepts offered by MEMO-OrgML are ProcessType, ProcessUse, ContextOfProcessUse, InputSpec, OutputSpec and Event. ProcessType is an abstract concept that is specialised into two concrete concepts: ComplexProcessType and BasicProcessType - with ComplexProcessType being composed of n ProcessType. In order to differentiate between many occurrences of the same ProcessType within a ComplexProcessType, we introduced the concept ProcessUse. A ComplexProcessType may be composed of many ProcessUse, each of which is assigned exactly one ProcessType. In case the decomposition hierarchy of a ComplexProcessType contains more than one occurrence of a particular ComplexProcessType, there is need to differentiate between the associated ProcessUse. For instance: A business process is composed of n ProcessType "Write User Documentation" which is aggregated from - among other things - the ProcessType "Create Figures". The different occurrences of "Create Figures" within "Write User Documentation" could be differentiated by their associated ProcessUse. To differentiate between identical ProcessUse within different occurrences of "Write User Documentation", every corresponding ProcessUse would be assigned to exactly one ContextOfProcessUse. A ProcessType may require an InputSpec and may produce one or more OutputSpec. Both are associated with events and serve as containers for information resources - like documents, files or objects which are specified in an associated resource or object model respectively. For instance: a subprocess may produce the two outcomes (as instances of OutputSpec) "order accepted" and "order refused". To prepare for simulations, every OutputSpec can be assigned a probability.

Events serve to define the flow of control within a process. They are created by certain states of a process. There are three basic constructs to specify the effect of a control event: processes can be executed in *sequential* order, *simultaneously* and *alternatively*. The fourth construct, the loop, results from combining sequential and alternative execution. Any process can be assigned organisational units and additional resources, such as roles, applications or communication devices. Those concepts are also part of MEMO-OrgML. MEMO-OrgML is defined by a metamodel with additional constraints specified in GRAL [Wen97] (see fig. 4).

Sometimes a process type can be regarded as a specialisation of another process type. For instance: The process of selling a home contents insurance policy may be similar to selling a fire insurance policy. So both may be regarded as a specialisation of a common (abstract) super concept. Although this seems to be an intuitive approach, we did not succeed in defining the semantics of process specialisation in a satisfactory way. Therefore process specialisation is simply defined as an association between two process types. While a specialisation must not be cyclic, there is no other constraint that would restrict the redefinition of inherited properties.



C 1 Only InputSpecs that are assigned to BasicProcessType(s) can select BasicService(s).

$$\forall cpt : V_{ComplexProcessType} \bullet (cpt \rightarrow_{specifiedby \bullet InputSpec} \rightarrow_{selects}) = \varnothing$$

C 2 No cyclic decomposition of ProcessType(s)

$$\forall cpt : V_{ComplexProcessType} \bullet \neg (cpt (\rightarrow_{consistsof} \rightarrow_{uses})^{+} cpt)$$

C 3 An Event must not be isolated.

$$\forall ev : V_{Event} \bullet \left(degree(et, \rightarrow_{triggers}) > 0 \wedge degree(et, \leftarrow_{triggers}) > 0 \right)$$

specialised from
interaction
aggregation
attribute
class
abstract class
C n constraint

Fig. 4: Excerpt from the MEMO-OrgML metamodel and illustration of its integration with the MEMO-OML (Object Modelling Language) and the MEMO-SML (Strategy Modelling Language, [Rie96])

A model created with MEMO-OrgML supports various methods for organisational analysis. Media clashes can be detected because the information described within InputSpec or OutputSpec is assigned the medium it is stored on. It is also possible to detect bottlenecks: Every process can be assigned a minimum and maximum execution time. This is also the case for some kinds of deadlocks. Notice, however, that a model of a business process is an abstraction. It will usually not represent every aspect of a real process in a comprehensive way. This is especially the case for intellectual tasks performed by humans. Sometimes it can be helpful to run a simulation of a process. A simulation, however, requires instances of a process type. It also requires instances of resources, roles, organisational units etc. To support the definition of process instances that can be used for simulation, MEMO-OrgML allows to specify so called *prototypical instances*. A prototypical instance is associated with a concept - like Employee. However, it does not have to be conceptualized in the same way. For instance: Within a simulation you may want to take into account the *average* number of days per year a clerk is absent through illness.

The graphical notation (concrete syntax) of MEMO-OrgML tries to follow common representations. The symbols used to render organisational structures are similar to those used in organisational charts. The visualisation of processes is also similar to common abstractions. To foster an intuitive understanding, there are a number of mnemonic symbols to visualize certain types of processes/tasks and resources. There are three different diagrams to render processes. A *decomposition diagram* shows the composition/decomposition hierarchy of a number of processes. A process *generalisation diagram* serves to render generalisation/specialisation relationships between processes. Finally, the *process diagram* (see fig. 5) allows to represent a process in more detail. For some observers too much detail may be irritating. Therefore MEMO-OrgML also offers a “light” version of the graphical notation.

An organisation model also supports the design of a corresponding information system. The description of a business process includes the specification of the information that is required or produced. If this information should reside in a computerized information system, it can be specified through a service of a particular class. For instance: If the age of a customer is required within a business process, this information would be specified as a reference to the corresponding service of the class Customer within the associated object model. According to our experience, a process model is more intuitive for many domain experts than an object model. Therefore the development of a process model also serves as a heuristics to find and refine objects. The model of a business process may also be used to specify a *workflow*. We regard a workflow as an abstraction of a business process: Only those parts are taken into account that are relevant for the management of the process by appropriate software. That includes, among other things, events that correspond to state changes of objects, the instantiation and release of objects (or applications) as well as the use of particular object services. A workflow model can be generated by eliminating all other aspects of a business process model. Such a workflow model could then be transformed into a workflow definition (which may require to neglect further details) that could be executed by a workflow management system - using, for instance, the WPDL (Workflow Process Definition Language) specified by the Workflow Management Coalition [WFM96].

In addition to that, a process model provides information that can be used for the generation of prototypical user interfaces. A process can be decomposed into subprocesses in a way that every subprocess represents a specific context of work for the person that is in charge of the subprocess. As already described above, the information that has to be accessed within the infor-

mation system is specified through services which in turn are defined in a corresponding object model. In order to prepare for the generation of user-interfaces, MEMO-OML allows to assign a *view* to every class. A view may be a basic view (for instance: a text view that is assigned to the class String) or a composed view (for instance: a view for the presentation of an address) that is aggregated from basic and/or composed views. Therefore it is possible to assign a view to every class that is used within a service's signature. A user-interface for a particular working context can then be composed of the set of views assigned to the services used by the corresponding subprocess. For a detailed description of the concepts used to generate prototypical user-interfaces see [Geu98].

An organisation model is supposed to be developed together with other partial models of an enterprise model within one common environment. MEMO Center has been implemented in Smalltalk. Among other things, it controls the referential integrity of an enterprise model and provides for navigation and retrieval. It also allows to generate code from an object model and corresponding workflow models. Within the organisational model, MEMO Center allows to detect media clashes. Furthermore it is possible to run simulations based on the instantiation and initialisation of prototypical instances. Fig. 5 gives an impression of the structure that is used to describe a business process in detail. It is based on a previous version of MEMO-Center. Fig. 6 shows the new version of the business process editor.

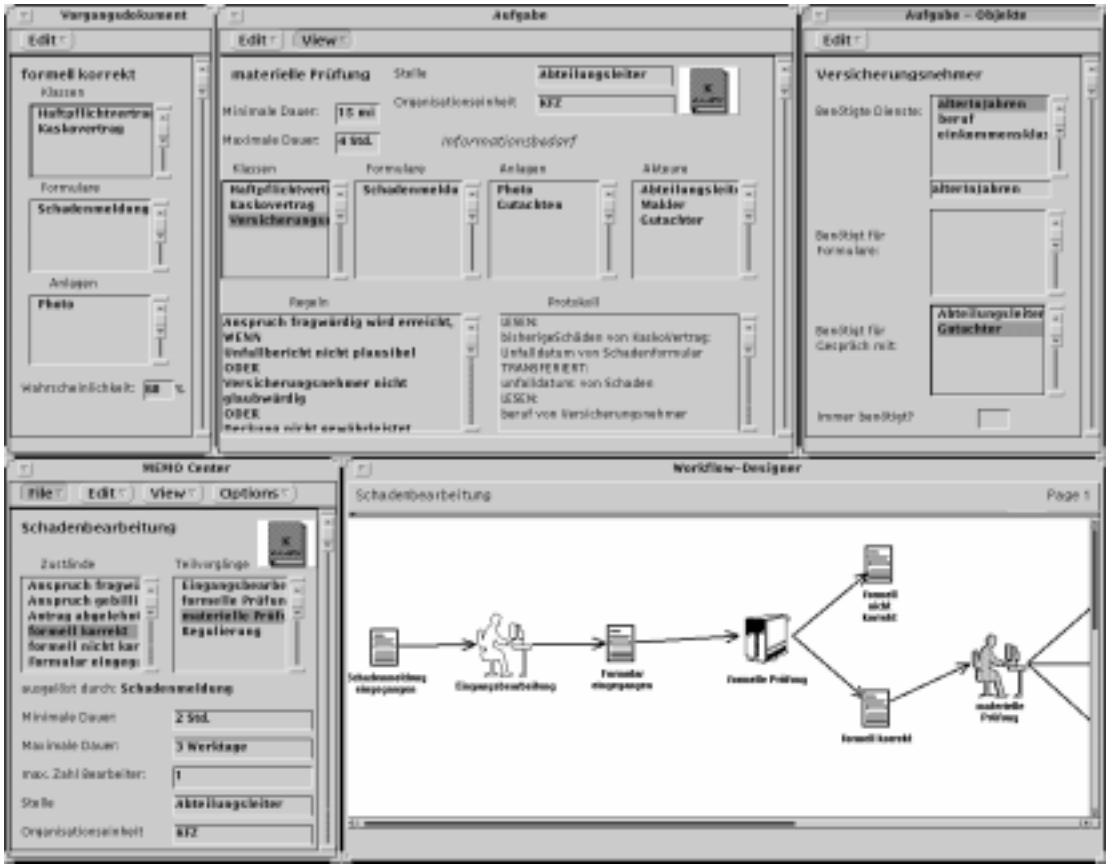
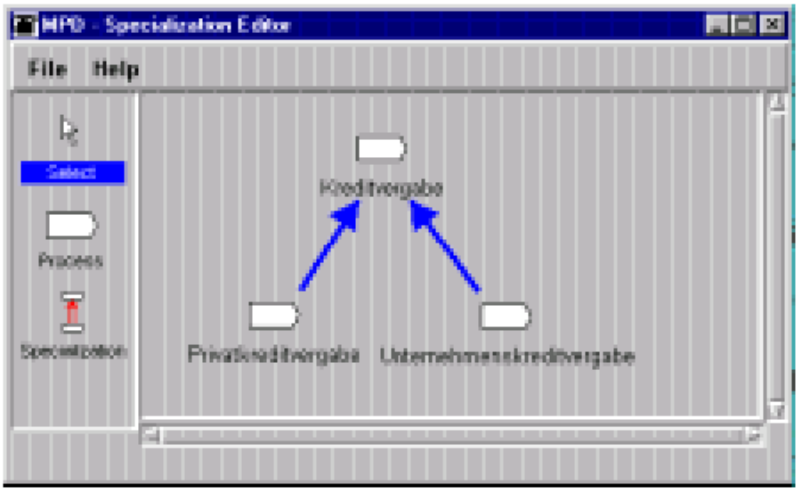
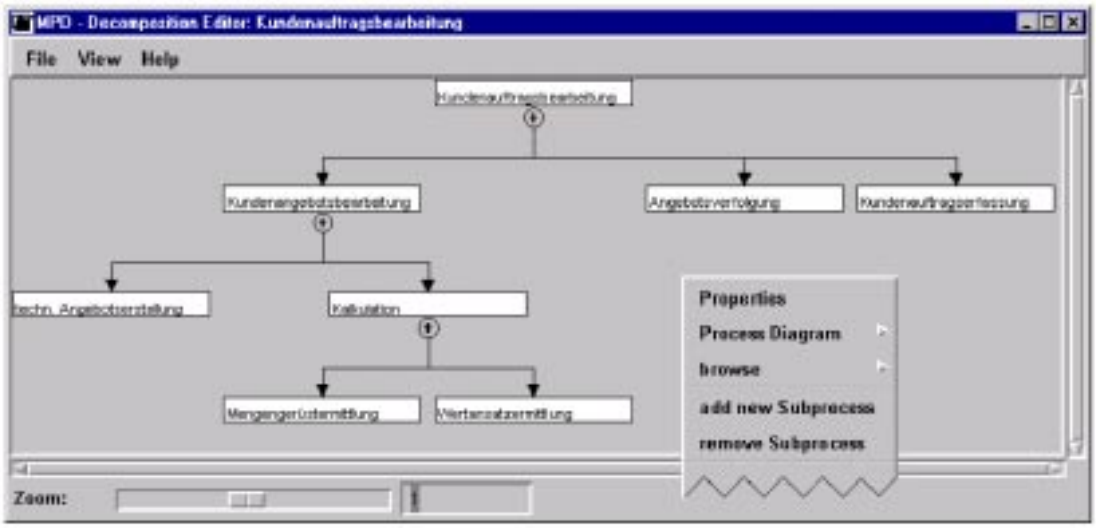


Fig. 5: Screenshot of MEMO-Center. The graphical Visualisation of a Business Process is supplemented by various textual Editors that allow to describe Details of a Process and the Ressources/Information it uses.

1



2



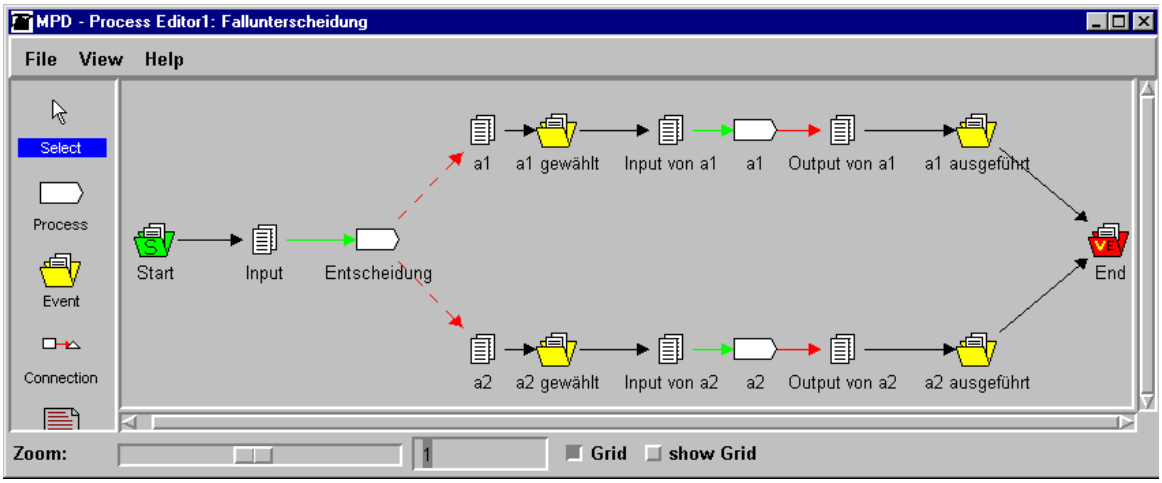


Fig. 5: Different Levels of Abstraction to visualize Business Processes and Relationships between different Process Types (1: Generalisation, 2: Aggregation), ([Wen97], pp. 152)

5. Concluding Remarks

Different from traditional conceptual modelling, enterprise modelling does not only focus on the development of software. Instead, it is based on the assumption that the design of efficient information systems requires to model a company's organisation and its long term strategy as well. That recommends the cooperation of people with different professional backgrounds, like domain experts, organisation analysts, top executives and software engineers. In order to take into account those different professional perspectives, MEMO offers specific modelling languages. They provide problem specific concepts together with syntactic and semantic constraints to use them. Thereby they help to structure and understand a particular domain - similar to a technical language. All languages within MEMO are integrated through common concepts. Therefore, enterprise models designed with MEMO offer abstractions that are appropriate for different observers. At the same time they provide a medium to foster communication and to avoid redundant work. The language framework of MEMO can be enhanced with modelling languages for further perspectives - like production planning and logistics. Within a current project, we are working on an enhancement of MEMO-OrgML that includes specific concepts for project management.

The different perspectives covered by MEMO proved to be a helpful framework for teaching. Not only that they emphasize the need for a multi-perspective approach. In addition to that they motivate the students to learn and use the corresponding concepts. According to our experience, languages for enterprise modelling can serve another purpose as well: Both, the development and use of those languages is suited to foster cross-disciplinary cooperation of various engineering disciplines (especially: computer science) and management science or business and administration respectively.

Besides the specification of further modelling languages and refinements of existing ones, our long term research is directed on higher level artefacts. We are working on design patterns for the development and documentation of specific models. Organisation theory in particular and management science in general offer a wide range of guidelines and heuristics which can be used for the creation of specific design patterns. In the long run we plan to use the MEMO languages for the design of generic enterprise models that serve as a reference for certain types of enterprises. Such reference models would not only provide blueprints for the organisation of the business and the architecture of corresponding information systems. They would also contribute to a common terminology which is a prerequisite for the establishment of a market for high level components ("business objects"). In addition to that, they could also be used as a conceptual foundation for the implementation and documentation of (software) frameworks (like, for instance [IBM98]) for corporate information systems.

References

[Ack94] Ackerman, M.S.: Answergarden and the Organisation of Expertise. Working Paper, MIT, Cambridge, Mass. 1994

[Dav93] Davenport, T.H.: Process Innovation. Reengineering Work through Information Technology. Boston/Mass.: Harvard Business School 1993

[ESP93] ESPRIT Consortium AMICE (Eds.): CIMOSA: Open System Architecture for CIM. Berlin et al.: Springer 1993

[FiHe96] Firesmith, D.; Henderson-Sellers, B.; Graham, I.; Page-Jones, M.: OPEN Modeling Language (OML). Reference Manual. Version 1.0. 8 December 1996 http://www.csse.swin.edu.au/OPEN/comn.html

[Fra97] Frank, U.: Enriching Object-Oriented Methods with Domain Specific Knowledge: Outline of a Method for Enterprise Modelling. Arbeitsberichte des Instituts für Wirtschaftsinformatik, Nr. 4, Universität Koblenz-Landau, 1997

[FrPr97] Frank, U.; Prasse, M.: Ein Bezugsrahmen zur Beurteilung objektorientierter Modellierungssprachen - veranschaulicht am Beispiel vom OML und UML. Arbeitsberichte des Instituts für Wirtschaftsinformatik, Nr. 6, September 1997

[Fra98a] Frank, U.: The MEMO Meta-Metamodel. Arbeitsberichte des Instituts für Wirtschaftsinformatk, Nr. 9, Koblenz 1998

[Fra98b] Frank, U.: The MEMO Object Modelling Language (MEMO-OML). Arbeitsberichte des Instituts für Wirtschaftsinformatk, Nr. 10, Koblenz 1998

[Fra99] Frank, U.: Delegation: An Important Concept for the Appropriate Design of Object Models. In: Journal of Object-Oriented Programming, forthcoming 1999

[Frz97] Franzke, A.: GRAL 2.0: A Reference Manual. Fachberichte Informatik, Universität Koblenz-Landau, 1997

[Geu98] Geuß, S.: Konzepte für die Generierung graphischer Benutzeroberflächen auf Basis objektorientierter Workflowbeschreibungen. Diplomarbeit, Universität Koblenz-Landau, Koblenz 1998

[Gog+93] Gogolla, M.; Conrad, S.; Herzig, R.: Sketching Concepts and Computational Model of TROLL light. In: Miola, A. (ed.): Proc. of the 3rd Int. Conf. Design and Implementation of Symbolic Computation Systems (DISCO'93), Berlin, Heidelberg etc.: Springer 1993, pp. 17-32

[Gor92] Goranson, H.T.: The CIMOSA Approach as an Enterprise Integration Strategy. In: Petrie, C.J. (Ed.): Proceedings of the First International Conference on Enterprise Integration Modeling. Cambridge, Mass.: MIT Press 1992, pp. 167-178

[Hen94] Henderson-Sellers, E.: Book two of Object-Oriented Knowledge: The Working Object. Prentice-Hall 1994

[IBM98] IBM: San Francisco - Concepts and Facilities. (http://www.ibm.com/Java/Sanfrancisco/concepts/ibmsf.sf.SFConceptsAndFacilitiesDevelopingApplications.html) 1998

[Jac+94] Jacobson, I. et al.: The Object Advantage. Business Process Reengineering with Object Technology. Wokingham 1994

[Oll+91] Olle, T.W., Hagelstein, J., MacDonald, I.G. et al.: Information Systems Methodologies. A Framework for Understanding. Reading, Ma.: Addison-Wesley 1991

[Petr92] Petrie, C.J. (Ed.): Enterprise Integration Modeling. Proceedings of the First International Conference. Cambridge, Ma.: MIT Press 1992

[Por85] Porter, M.E.: Competitive Advantage. New York 1985

[Rat97] Rational: UML-Semantics. Version 1.1. 09/01/1997 (http://www.rational.com)

[Rie96] Riemenschnitter, R.: Rechnerunterstützte strategische Planung: Rekonstruktion und objektorientierte Modellierung strategischer Konzepte ausgehend vom Ansatz Porters. Diplomarbeit, Universität Koblenz-Landau 1996

[Rum96] Rumbaugh, J.: Notation notes: Principles for choosing notation In: Journal of Object-Oriented Programming, Vol. 8, No. 10, May 1996, pp. 11-14

[Sch94] Scheer, A.-W.: Business Process Engineering. 2. ed., Berlin, New York, et al.: Springer 1994

[Sch90] Schürr, A.: PROGRES: A VHL-language based on graph grammars. In: Ehrig, H. et al. (eds.): Graph Grammars and Their Application To Computer Science. LNCS 532, Berlin, Heidelberg etc.: Springer 1990, pp. 641-659

[SoZa92] Sowa, J. F.; Zachman, J. A.: Extending and formalizing the framework for information systems architecture. In: IBM Systems Journal, Vol. 31, No. 3, pp. 590-616, 1992

[Wen97] Wenzel, J.: Entwurf einer Modellierungssprache zur Beschreibung von Geschäftsprozessen im Rahmen der Unternehmensmodellierung. Diplomarbeit, Universität Koblenz-Landau 1997

[WFM96] WfMC (Workgroup 1): Interface 1: Process Definition Interchange WfMC TC-1016, Version 1.0 Beta, May 29, 1996. Obtained via http://www.aiai.ed.ac.uk/WfMC/ 1996

[Zac87] Zachman, J. A.: A framework for information systems architecture. In: IBM System Journal, Vol. 26, No. 3, pp. 276-292, 1987

[Zic99] Zickhardt, J.: Formalisierung von MEMO-OML. Diplomarbeit, Universität Koblenz-Landau, Koblenz 1999 (forthcoming)