Multi-Perspective Enterprise Modeling (MEMO) - Conceptual Framework and Modeling Languages

Ulrich Frank

Universität Koblenz-Landau
Rheinau 1, D-56075 Koblenz, Germany
E-mail: ulrich.frank@uni-koblenz.de

Abstract

For many companies, the strategic as well as the organizational fit of their information systems is a pivotal factor for staying competitive. At the same time, there is an increasing demand for integrating business processes and informations systems with those of customers and suppliers. The resulting need for organizational changes and the introduction of corresponding information systems is a challenging task. The complexity of the task requires a separation of concerns. At the same time it causes language barriers between various stakeholders, especially between business people and information technology professionals. Enterprise models provide various abstractions that help with the design of corporate information systems which are in line with a company's organization and its long term strategy. They also promise to provide a common conceptual foundation to foster the communication between people with different professional backgrounds. In this paper we introduce a model for enterprise modelling that is based on an extendable set of special purpose modeling languages, e.g. for describing corporate strategies, business processes, resources or information. The visual languages provide intuitive abstractions for various observers. The languages are defined in metamodels which in turn are specified through a common meta-metamodel. Similar to a specialized technical language, they provide concepts that help with structuring and analyzing a domain according to specific objectives. Since the languages are specified in a semi-formal way, the models allow for the generation of software prototypes. The languages share common concepts which allow for a tight integration of the various parts of an enterprise model. In addition to offering specialized modeling languages, the modeling method also includes examples, case studies and reference models - to promote the re-use of concepts and artefacts. The use of the method is illustrated by an example, where two different partial models are being integrated.

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". With the increasing importance of electronic commerce, more and more companies have to rethink the way they do business, including the redesign of both internal and cross-organizational business processes. In this situation, the main challenge results from the complexity and diversity of the tasks to be performed. While system design, analyzing and redesigning a corporate strategy and a company's organization respectively are complex tasks on their own, their coordination is required in order to provide for information systems that are consistent with strategic and organizational guidelines. To meet this challenge, enterprise models have been introduced on various levels of abstraction. Vendors of large ERP systems supplemented their software with models of business processes and conceptual data models to help system analysts and users with understanding a system (for instance: [22]). While they are useful for the deployment of particular systems, they are usually of limited use only for other systems. To promote the establishment of common e-commerce infrastructure, a number of consortia provide high level enterprise models that emphasize roles and responsibilities of partners within certain types of business transactions. However, they remain on a superficial level - see, for instance, the so called "OBI Architecture", or RosettaNet's "E-Business Model". In principle, general purpose modeling languages such as UML [21] or OML [3], allow for modeling a wide range of domains. They also offer an adequate foundation for software development. However, those languages are designed for the development of software systems. Therefore they do not provide concepts and graphical representations that are appropriate for all aspects of an enterprise model. For this reason, approaches to apply general purpose modelling languages for enterprise modelling (like [15]) suffer from semantic shortcomings of these languages. More specific languages serve as an instrument to structure and analyze a domain - 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.

Management science and organizational theory offer a wide range of dedicated approaches for analyzing and shaping a firm's strategy as well as for organizational (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. Organizational 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 [11]). Strategic and organizational models are usually based on different concepts. Furthermore, they have, in general, nothing in common with conceptual models used in software engineering.

These different abstractions of an enterprise are in part necessary, because there is need for a separation of concerns. At the same time they reflect the well known cultural chasm between business people and information technology professionals. The term "enterprise modelling" has been introduced ([24], [14]) in order to emphasize both the need for high levels of abstraction and the importance of a multi view 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 [18], [17], [7]), 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., [23], [24]) - usually without specifying modelling languages to represent the particular views. Other approaches are based on software development methods ([13], [12]). They do not include specific concepts from organizational theory or management science. More elaborated approaches - like CIM/OSA ("Open System Architecture", [2]) or ARIS [22] - 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 ([7]) that offers a set of specialized 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 organization. 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, organizational structure, business entities, business rules etc.), hence serving as an "organizational memory" [1] or a corporate knowledge base. In this paper, we will focus on the modelling languages provided by MEMO.

2. Enterprise Modelling: Vision and 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 emphasizing appropriate 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 organization.

A model that is associated with a particular view should provide a medium for communication between people with different professional backgrounds. 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 conceptualization. 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 visualization 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.

Often, communication between people that belong to different professional communities will not require a high level of detail. Instead it is sufficient, and helpful, to seek a common understanding of the "big picture". On the other hand, there are also specific tasks, like the re-design of a business process or the design of an object model that require the use of detailed concepts. For this reason, a method for enterprise modeling should allow for *various levels of abstraction*.

Enterprise modelling also aims at the *integration of the partial models* that represent particular views on an enterprise. Integration means that semantic relationships between partial models can be expressed. This is a prerequisite for referential integrity between different models.

Designing and maintaining an enterprise model is an expensive and challenging tasks. Therefore, a method for enterprise modelling should provide *concepts that can be re-used* and adapted in a convenient way. Re-use can take place on different levels. The concepts provided by a specialized modelling language are one example. Other examples include design patterns and generic reference models for certain types of industry.

Enterprise models tend to be very complex. That recommends 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. Additionally, the language designers should take into account how a language description can be mapped to models that are suited for the design of tools, such as object models.

3. MEMO: Basic Abstractions

The construction of an enterprise models requires a common understanding of essential abstractions. Such a "big picture" does not only serve as a common reference, but also as a framework to integrate special purpose models.

3.1 Generic Framework

As an introduction to more detailed abstractions, and as a medium to foster cross-disciplinary discourses, MEMO offers a generic conceptual framework that corresponds to common abstractions of business firms. It differentiates three so called *perspectives* - strategy, organization and information system - each of which is structured by four *aspects*: structure, process, resources, goals. A participant in a discussion can literally point to the *focus* of his concern by selecting a particular aspect within a perspective. In fig. 1 the meaning of the various foci is illustrated by exemplary terms.

The special purpose models provided by MEMO allow to "zoom" into the framework. One or more foci correspond to a particular model. For instance: an organization model serves to represent an organization 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.

3.2 Models and their Interrelationships

Usually the construction of an enterprise model will start with modeling the corporate strategy. The next step would be to analyze and (re-) design core business processes. Based on the information need specified in models of business processes, an information model, e.g. an object model can be developed. MEMO offers three specialized languages that support the construction of these kinds of

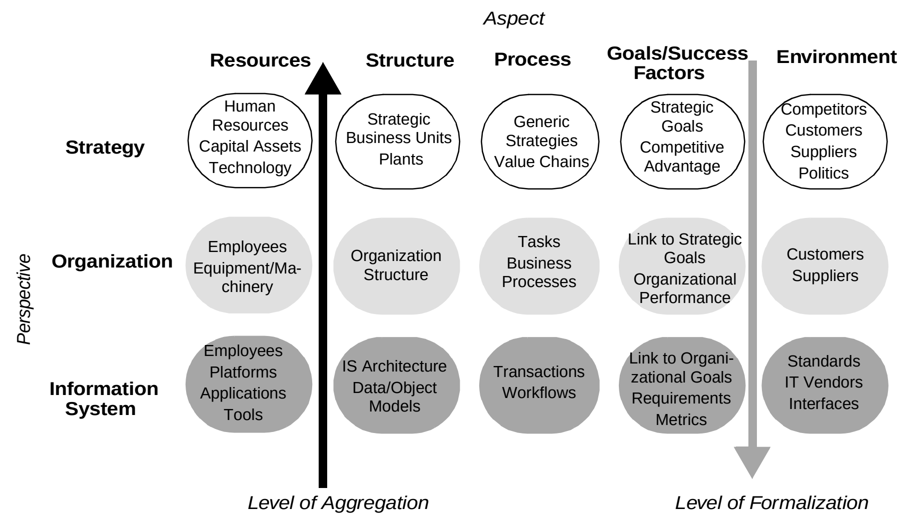

Aspect
Resources
Structure
Process
Goals/Success Factors
Environment
Strategy
Human Resources Capital Assets Technology
Strategic Business Units Plants
Generic Strategies Value Chains
Strategic Goals Competitive Advantage
Competitors Customers Suppliers Politics
Organization
Employees Equipment/Machinery
Organization Structure
Tasks Business Processes
Link to Strategic Goals Organizational Performance
Customers Suppliers
Information System
Employees Platforms Applications Tools
IS Architecture Data/Object Models
Transactions Workflows
Link to Organizational Goals Requirements Metrics
Standards IT Vendors Interfaces
Perspective
Level of Aggregation
Level of Formalization

Fig. 1: Aspects, Perspectives and related Subjects

models. The strategy modeling language (MEMO-SML) includes known well known concepts from strategic planning, such as value chains [19], portfolio analysis etc. The organization modeling language (MEMO-OrgML) serves to model a firm's organization, including business processes and resources. To allow for the specification of information as a foundation of database design or software development in general, MEMO also includes an object-oriented modeling language (MEMO-OML).

Fig. 3 shows excerpts of models designed with these three languages. Notice that each model refers to concepts within one or more other models. The excerpt of a strategy model includes the representation of a value chain with one activity group being decomposed into further activities. An activity within a strategy model can refer to a set of strategic resources as well as to types of business processes. To foster the analysis of a company's competitive position, both MEMO-SML and MEMO-OrgML include concepts that guide the user with the evaluation of resources. The concepts shown in fig. 3 can be specified in much more detail. For this purpose, corresponding forms and templates can be used.

3.3 Tool Environment and Reference Models

Enterprise modelling recommends the use of appropriate tools. The various interrelated models require support for browsing and searching. Furthermore it is hardly possible to manually maintain an enterprise model without jeopardizing its integrity. Finally it is impossible to do without tools when simulation (for instance: of business processes) or code generation is an issue. MEMO is accompanied by a development environment, called MEMO Center. It controls a model's integrity and provides means to navigate through the views of an enterprise model on various levels of detail. Furthermore it allows for generating Smalltalk code from object models. All components combine editors for graphical notations with textual editors and browsers. MEMO Center is implemented in Smalltalk and runs on any platform a Visualworks® virtual machine is available for. The components are integrated through a common object model. The integration with a corresponding object model is accomplished by referring to classes and their services respectively.

During the last years, we developed two major reference models using MEMO languages. One is targets the organization and information management of a research institute at a university. Based on this model a set of applications has been implemented in Smalltalk. The second model serves as a reference for building e-commerce platforms. It includes about 20 types of business processed and almost two hundred classes and versatile abstractions to model almost any kind of product. In order to foster its reuse, we deployed Java and a relational DBMS for its implementation.

4. MEMO Modeling Languages: Architecture, Specification and Integration

So far, we illustrated the use of the MEMO modeling languages. We also gave an idea of their integration. However, in order to promote consistent models and to specify a conceptual foundation for the development of corresponding modeling tools, there is need for precise, (semi-) formal language specifications. At the same time there should be support for enhancing and adapting modeling languages, as well as interdicting new languages, because it may turn out that the existing languages are not sufficient or appropriate for certain types of enterprises. Currently we are, for instance, working an enhancing MEMO-OML with concepts needed for modeling projects. We also think about defining a new language to model production lines and corresponding logistics processes.

4.1 Meta-Metamodel and Language Architecture

The previous thoughts are reflected by an architecture for the specification and integration of modelling languages. The architecture is extensible 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, metamodels make sense, too: Typically, designing tools requires conceptual 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 a meta language defined in a common meta-metamodel. For a detailed description and a comparison with other meta-metamodels see [4].

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, [5]). Although an object model representing a language is usually very similar to the corresponding metamodel, the mapping is not trivial. Firstly, MEMO-OML offers much more concepts (for instance: multiple inheritance vs. single inheritance, aggregation, delegation [6], services ...) than the meta-metamodel. Therefore, it is

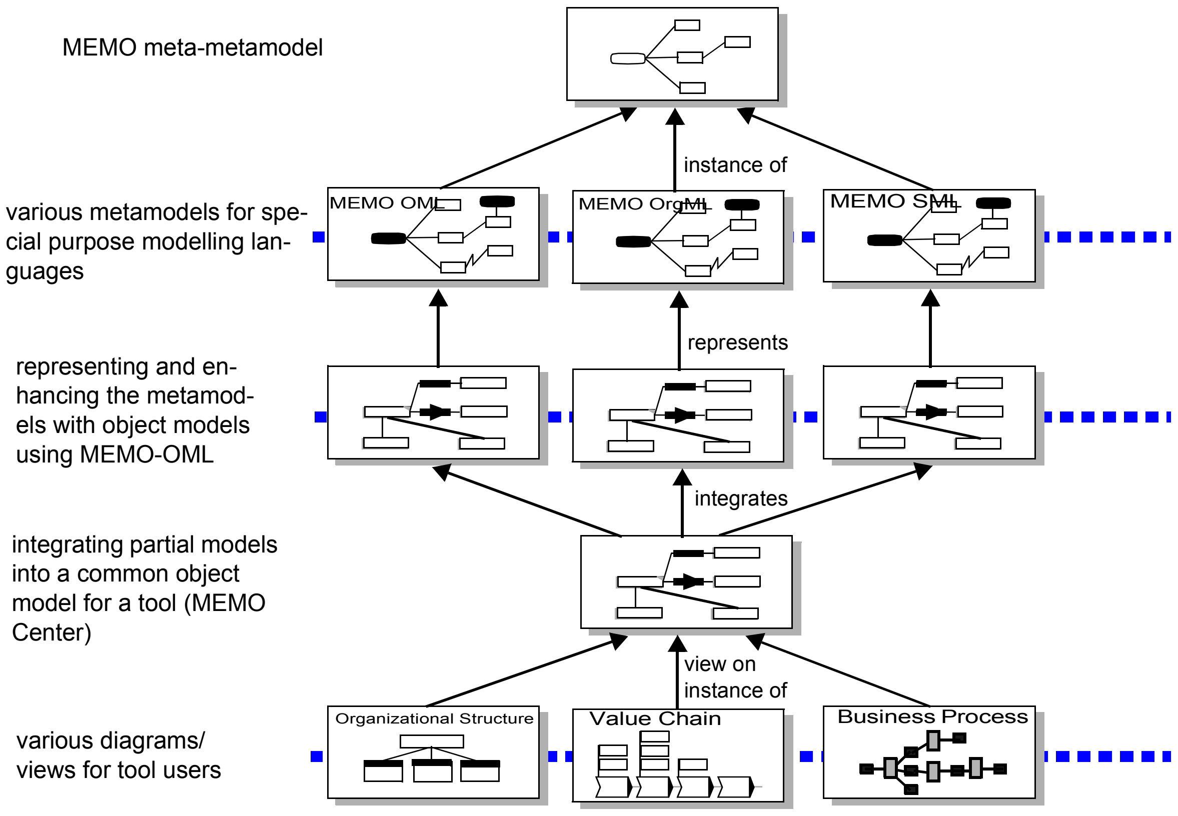

MEMO meta-metamodel
various metamodels for special purpose modelling languages
representing and enhancing the metamodels with object models using MEMO-OML
integrating partial models into a common object model for a tool (MEMO Center)
various diagrams/ views for tool users
instance of
MEMO OML
MEMO OrgML
MEMO SML
represents
integrates
view on instance of
Organizational Structure
Value Chain
Business Process

Fig. 2: Architecture of MEMO Modelling Languages

sometimes useful to reconstruct a metamodel with more elegant or more appropriate concepts. Secondly, modeling tools have specific features like user management or version control, which have to be represented in the corresponding object model. The various object models are integrated into one common object model that serves at the conceptual foundation of an integrated modeling environment. Fig. 2 gives an overview of the MEMO language architecture.

4.2 Language Specifications and their Integration

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 "Organizational Unit" with an object-oriented modelling language, the language itself would not exclude that the association "responsible for <OrganizationalUnit>" must not be cyclic (instead, you would have to add an explicit constraint to the model). A language specialised for representing organizations could be enriched with the concept "Organizational 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"). To give an impression of the language definitions within MEMO, we will give an overview of the concepts featured by MEMO-OrgML, MEMO-SML and MEMO-OML. The integration of the languages is illustrated by common concepts of the corresponding metamodels.

MEMO SML

The MEMO Strategy Modeling Language is intended to support the design of models, which describe a firm's goals and business strategy. It includes four groups of concepts. *Roles* serve to describe relevant actors, e.g. Customer, Supplier, Competitor. Different from the use of these concepts on the operational level, they do not serve to represent single instances. Instead, their instances represent abstractions of a corresponding sets of instances. This is the case, too, for *resources*, like HumanResource, Finan-

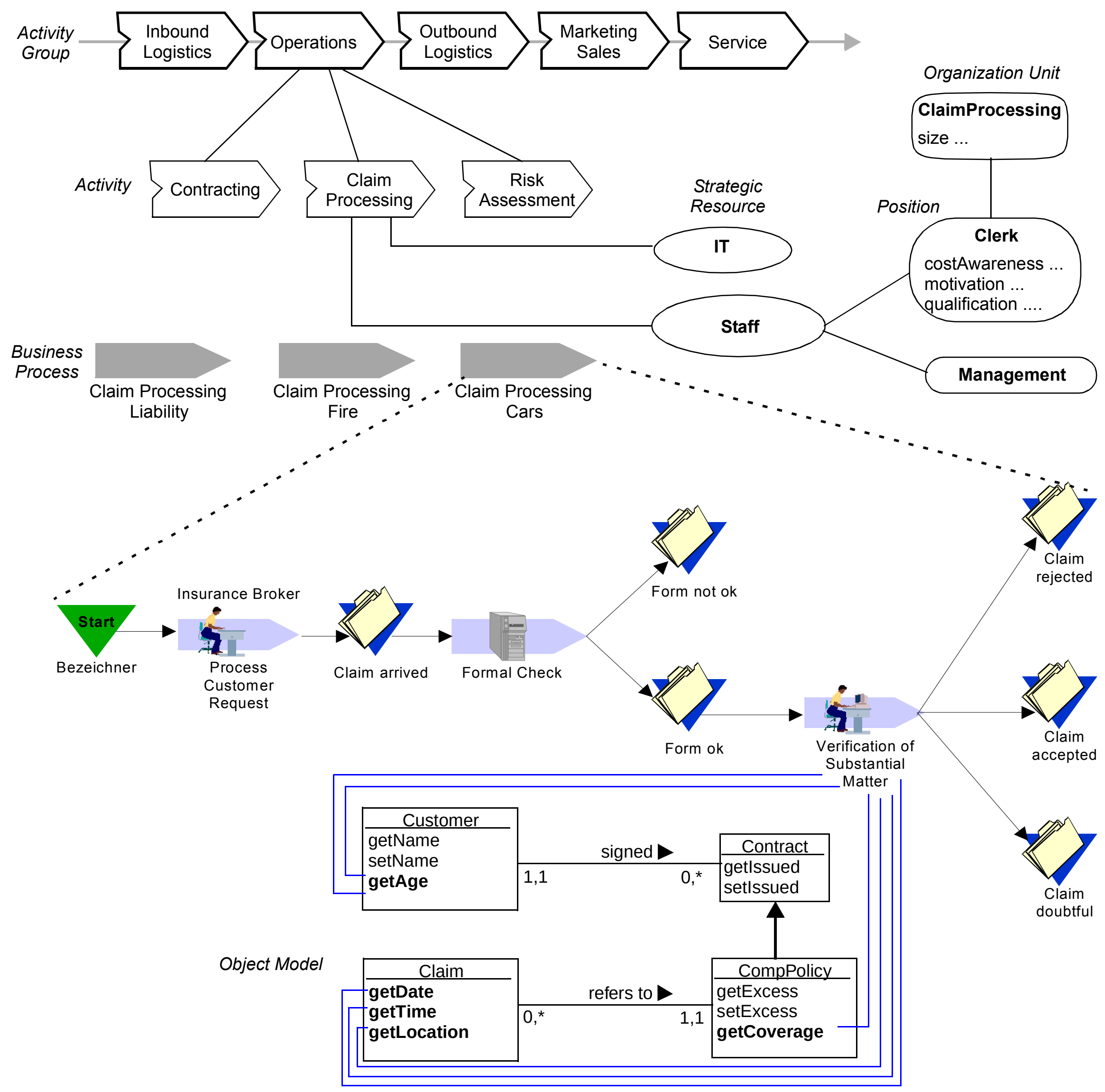


Fig. 3: Relationships between selected partial models. The semantics of links depends on the corresponding concepts.

cialResource, Technology etc. Their attributes allow for an aggregated evaluation of resource types. *Markets* are essential for strategic planning. Therefore MEMO SML offers a set of related concepts, such as Market, SubMarket, MarketShare, Segment etc. Last but not least, there are *analysis concepts* that aim at supporting the user with structuring, evaluating and refining a company's strategy, for instance: Activity, ActivityGroup, ValueChain, GenericStrategy, Asset etc.

Often, concepts used to describe a corporate strategy are aggregated from or associated with concepts used on operational levels. Therefore, many concepts of MEMO SML are associated with concepts of other languages. For instance, StrategicBusUnit and HumanResource are associated with OrganizationalUnit (MEMO OrgML), Technology with Platform (MEMO OML). Note that these associations contribute to a powerful conceptual foundation for Management Information Systems. Due to the nature of strategic planning, some of the concepts used in MEMO SML, for example CustomerOrientation, SuccessFactor, CompetitivePosition, balk at precise definitions. In these cases, the specification is restricted to associations to other concepts and to the use of attributes that allow for textual annotations, only. The graphical representation of strategy models is illustrated in fig. 3. However, since the specification of the languages and their notation are strictly separated, other graphical representations could be used as well.

MEMO OrgML

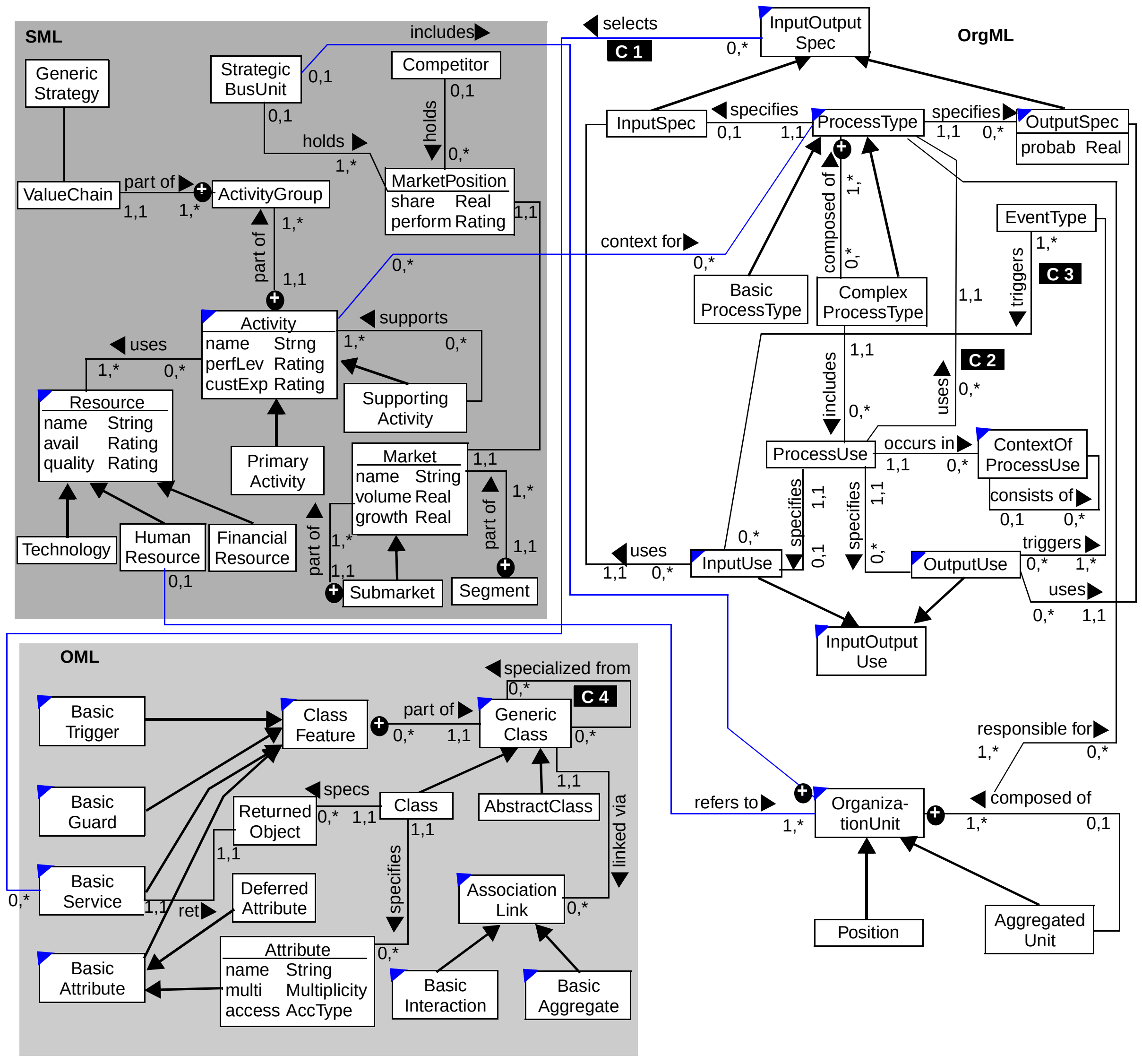
MEMO-OrgML serves to model a company's organization. It provides concepts to model a business process in a detailed way. This is different from the use case approach [13] that is mainly based on natural language descriptions and therefore offers only little help with structuring a process. Formal languages 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 organizational units and additional resources, such as roles, applications or communication devices.

A model created with MEMO-OrgML supports various methods for organizational analysis. Media clashes can be detected because the information described within InputSpec or OutputSpec (see fig. 4) is assigned the media 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, organizational 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, which is certainly not an adequate attribute for describing a particular employee. For a complete, updated specification of MEMO-OrgML see [9].

MEMO-OML

Different from UML, the MEMO Object Modeling Language is restricted to the design of static object models. It offers four categories of concepts. *Semantic* concepts serve to describe those parts of an object model which correspond to properties of real world entities or concepts, such as Class, Attribute, Service etc. *Organization* concepts serve to reduce the complexity of elaborated object models, either by aggregating concepts or by introducing additional abstractions. Examples are Cluster, AttributeCategory or DesignPattern. Resources are located on a different level of abstraction. They relate to artefacts that are required by an information system, such as Platform or Application. Finally, *management* concepts provide information that helps with the management of instantiated object models, like default values or a history flag for attributes (which serves to indicate that changes of the state of an attribute ought to be recorded). With respect to the design of static object models, MEMO OML is very similar to UML. Different from UML, it features *delegation*, a concept that allows for a more natural description of role relationship. It combines the benefits of inheritance with those of associations ([8]). While UML does not specify the semantics of inheritance, MEMO OML offers a more precise definition of inheritance, also taking into account the notorious "covariance" problem that results from inconsistencies caused by the redefinition of inherited features (see [Meye97], p. 621). For a comprehensive



Constraints

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

C 2 No cyclic decomposition of ProcessType(s)

C 3 An Event must not be isolated.

C 4 Specializations must not be cyclic.

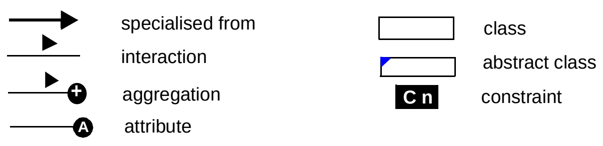


Fig. 4: Excerpts from MEMO metamodels and illustration of their integration

specification of MEMO OML and case studies that illustrate its use see [4], [5].

To allow for a more detailed description of a class, MEMO-OML offers additional features to specify attributes and services as well as two additional *semantic class features*: *guards and triggers*. Associations, attributes, and services can be defined as "deferred" which means that they have to be specified within a subclass. An association is regarded to be deferred if at least one of the two association links it belongs to is deferred. Only abstract classes may have deferred features. Also, the language includes various categories of object services, like *regular* services, *default access* services or *propagated* services. Default access services are services which provide default access (read, write, insert, delete) to an object's attributes (AttriAccessService) or its associated objects (AssocAccessService). Regular services can be specified by the attributes (of the same class) or services (of any class) they use. Propagated services mainly serve to provide an abstraction: They are propagated from the interface of an attribute of the particular class. For instance: A class may have an attribute "dateOfBirth" which provides a service "yearsOfAge". A propagated service allows to make that service available directly within the interface of the corresponding class (like "Person") without loosing track of the reference. In addition to that, a process model provides information that can be used for the generation of prototypical user interfaces.

By combining process models specified in MEMO OrgML and object models, MEMO allows for the generation of user-interfaces. For this purpose, a *view* can be assigned 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 [10].

Integrating languages requires 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 - because in the second case the chances for deviating interpretations are clearly lower. All MEMO modeling languages are specified in the same meta-language. Hence, from a formal point of view, it is easy to define common concepts for a set of modeling languages. Nevertheless, the identification of appropriate concepts that may serve as interfaces between different kinds of models requires to carefully coordinate the development of the corresponding languages. Fig. 4 illustrates the integration of MEMO OML, MEMO OrgML and MEMO SML. Concepts of two different languages that are associated, like Activity and ProcessType, are common to these languages. Note that fig. 4 renders small excerpts of the actual metamodels only. Also, details of the shown concepts, like attributes or additional constraints, are omitted except for a few examples.

5. Conclusions and Future Work

In addition to a set of modeling languages, MEMO also offers a process model as well as heuristics and techniques that guide with the design of enterprise models. Different from general purpose modeling languages like the UML, the MEMO languages not only allow for more intuitive representations of various perspectives on an enterprise. By providing specialized terminologies for various purposes (e.g. organizational analysis and design, strategic planning, information analysis, software development) they also help to structure a problem domain according to proven professional standards. In contrast to other methods for enterprise modelling, like ARIS [22] or CIM-OSA [2], the MEMO modeling languages are more mature in terms of completeness and precision. With respect to the crucial importance of standards and the wide spread use of UML, it may seem erroneous to develop yet another object modeling language. There are a few reasons why we still prefer our own language. Firstly, the first version of MEMO OML was specified prior to the first version of UML. Secondly, UML suffers from a number of shortcomings (for an evaluation see [20]). However, the most important reason is the fact that maintaining our own language rather than using UML helps to protect one of our core competencies. Nevertheless, we are not fanatic with MEMO UML. In some projects we are using UML for pragmatic reasons.

From a methodological point of view, the evaluation of modelling languages is a delicate task. This is due to the fact that the requirements 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 optimize a modelling language straight off. Instead a language has to be evaluated by prospective users against the purpose it should serve. Within several projects we had MEMO modeling languages evaluated by users with different backgrounds (students, business people, software engineers etc.). The results are ambiguous in the sense that the judgements certain concepts and their graphical representations received from people with similar backgrounds often varied substantially.

A multi-perspective enterprise model is a favorable subject of knowledge management. Therefore, we adapted the MEMO metamodels in order to specify a conceptual foundation of knowledge management systems ([8]). Our future work is focussing on applying MEMO languages to develop further reference models. Such reference models do not only provide blueprints for the organization of the business and the architecture of corresponding information systems. They also contribute to a common terminology or "ontology", 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 [16]) for corporate information systems. Within ECOMOD ("Electronic Commerce Modeling"), a project funded by the German National Research Foundation, we are developing enterprise models as a conceptual foundation for cross-organizational business processes and corresponding versatile platforms for electronic trading.

Last but not least, 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.

References

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

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

[3] 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

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

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

[6] Frank, U.: "Delegation: An Important Concept for the Appropriate Design of Object Models". In: *Journal of Object-Oriented Programming,* vol. 13, No. 3, June 2000, pp. 13-18

[7] Frank, U.: *Visual Languages for Enterprise Modelling*. Arbeitsberichte des Instituts für Wirtschaftsinformatik. Arbeitsberichte des Institut für Wirtschaftsinformatik der Universität Koblenz-Landau, No. 18, 1999

[8] Frank, U.: "Multi-Perspective Enterprise Models as a Conceptual Foundation of Knowledge Management Systems." In: *Proceedings of the Hawaii International Conference on System Sciences*, Los Alamitos, Ca. 2000

[9] Frank, U.; Jung, J.: *The MEMO Organisation Modelling Language (MEMO-OrgML)*. Arbeitsberichte des Institut für Wirtschaftsinformatik der Universität Koblenz-Landau, No. 26, 2001

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

[11] Grant, R. M.: *Contemporary strategy analysis: Concepts, techniques, applications. A guide for instructors*. 3. Ed., Blackwell Publishers 1998

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

[13] Jacobson, I. et al.: *The Object Advantage. Business Process Reengineering with Object Technology.* Wokingham 1994

[14] Katz, R.L.: "Business/enterprise modelling." In: *IBM Systems Journal*, Vol. 29, No. 4, pp. 509-525

[15] Marshall, C.: *Enterprise Modeling with UML: Designing Successful Software through Business Analysis*. Reading, Mass., et al.: Addison-Wesley 1999

[16] Monday, P.; Carey, J.; Dangler, M.: *San Francisco Component Framework: An Introduction*. Reading, Mass., et al.: Addison-Wesley 1999

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

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

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

[20] Prasse, M.: "Evaluation of Object-Oriented Modelling Languages: A Comparison between OML and UML." In: M. Schader, A. Korthaus (Eds.): *The Unified Modeling Language - Technical Aspects and Applications*, Heidelberg: Physica 1998, pp. 58-78

[21] Rumbaugh, J.; Jacobson, I.; Booch, G.: *The Unified Modeling Language Reference Manual*. Reading, Mass., et al.: Addison-Wesley 1999

[22] Scheer, A.: *ARIS - Business Process Modeling*. 2. compl. rev. and enlarged ed., Berlin, Heidelberg, New York, et al.: Springer 1999

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

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