Softw Syst Model (2014) 13:941–962
DOI 10.1007/s10270-012-0273-9

THEME SECTION PAPER

Multi-perspective enterprise modeling: foundational concepts, prospects and future research challenges

Ulrich Frank

Received: 12 November 2011 / Revised: 17 July 2012 / Accepted: 23 July 2012 / Published online: 31 August 2012

Abstract The paper presents a method for multi-perspective enterprise modeling (MEMO) and a corresponding (meta-) modeling environment. An extensive analysis of requirements for enterprise modeling serves to motivate and assess the method. The method is based on an elaborate conception of multi-perspective enterprise models and on an extensible language architecture. The language architecture is comprised of a meta modeling language and an extensible set of integrated domain-specific modeling languages (DSML). The DSML are supplemented with process models and with guidelines for their reflective use. The corresponding modeling environment integrates editors for various DSML into multi-language model editors. It includes a meta model editor which enables the convenient use, development and extension of the set of supported DSML and supports the generation of respective graphical model editors. Thus, it also serves as a foundation for method engineering. MEMO covers both software engineering as well as social, managerial and economic aspects of the firm. The presentation of MEMO is supplemented with a comparative overview of other approaches to enterprise modeling. The paper concludes bys summarizing fundamental technical, epistemological and political challenges for enterprise modeling research and discusses potential paths for future research.

Communicated by Dr. Tony Clark, Balbir Barn, Alan Brown, and Florian Matthes.

U. Frank (✉)
Information Systems and Enterprise Modelling Research Group, Institute for Computer Science and Business Information Systems (ICB), University of Duisburg-Essen, Essen, Germany
e-mail: ulrich.frank@uni-due.de

Keywords Enterprise modeling · Domain-specific modeling language (DSML) · Method engineering · Modeling tool · Reference model

1 Introduction

Information systems can be regarded as a key instrument for organizing and managing the firm: They enable new patterns of division of labor and coordination of work. At the same time, they are a pivotal instrument for initiating and performing business transactions. In addition to that, information systems need to be accounted for in strategic planning: Not only do they support the creation of new products and services, they may even enable entirely new business models that create sustainable competitive advantage. Despite their undisputed relevance for managing the enterprise and their potential for improving its competitiveness, the development, introduction, maintenance and management of information systems remain a substantial challenge for many firms. On the one hand, this challenge results from the peculiarities of information systems as information technology (IT) artifacts. On the other hand, the challenge is created by the growing awareness that focusing on IT alone is not sufficient. Instead, exploiting the potential of IT will often require changing the organizational action system. At the same time, IT is not only an enabler of organizational change. It may also inhibit it, if information systems lack flexibility. Therefore, it is widely undisputed that designing and managing information systems recommend accounting for both, the peculiarities of complex IT artifacts and of the organizational action system that needs to be supported. As a consequence, designing and managing information systems require people with different professional skills to collaborate effectively, which is not trivial to accomplish, since "the frequent cultural chasm between

business people and information technology professionals" that Keen [32, preface] complained about 20 years ago, has still not been overcome.

Various developments of the last two decades respond to these challenges. In business practice, the need for "business IT alignment" has been recognized as a key success factor. In many companies, the head of IT—usually a person with a clear technical background—has been replaced by IT managers that are expected to regard IT as a service to support the business. Both in academia and business practice, the increasing awareness of business processes as a foundation for the joint analysis and design of a firm's organization and its information system has arguably contributed to dismount the walls between IT and business. Computer Science, especially the fields of requirements analysis and software engineering, has produced various approaches to conceptual modeling as an instrument for systems analysis and design that help with reducing complexity and foster the involvement of prospective users. However, the focus of these approaches is clearly on designing software systems and not on co-designing them with the organization of a firm. This is different in Information Systems where it is constitutive to not regard an information system as an end in itself, but always as an instrument to improve organizational performance and competitiveness. Business process models provide an important abstraction to serve this purpose. Respective languages do not only support the representation and analysis of business process models, some of them—such as BPMN—also allow for the specification of workflow schemata. However, business process models cover a narrow focus only. The development of information systems that are mutually adjusted with the business and that account for strategic options requires including more aspects of an enterprise. The term "enterprise model" that was created in the late 1980s of last century addresses this demand. Early approaches focused in particular on the design of computer-supported manufacturing systems [2,67] or more general on the design of business information systems. The latter combined business process modeling with data modeling [50] or with object-oriented approaches [9,11,13]. Since the early days, the focus of enterprise modeling has been extended to also address the configuration and management of existing IT infrastructures. Although the relevance of modeling the enterprise is widely undisputed in Information Systems, the field is not in a coherent state yet. There is no unified conception of enterprise models and corresponding methods. Research is fragmented into different fields such as Information Systems, Computer Science and Artificial Intelligence. Also, ideas concerning use scenarios for enterprise modeling and related benefits vary to some extent.

Against this background, the paper pursues two interrelated objectives. On the one hand, it aims at contributing to a coherent conception of enterprise modeling by analyzing characteristic requirements and objectives, by suggesting a foundational technical terminology and by identifying prospects and challenges of future research. On the other hand, a particular method for multi-perspective enterprise modeling (MEMO) is presented in detail. It is to substantiate the general considerations by suggesting a comprehensive structure for an enterprise modeling method and a corresponding tool architecture. Also, MEMO serves to illustrate the range of actual and prospective use scenarios. Finally, the method shows how to supplement an engineering approach to enterprise modeling with concepts that account for peculiarities of action systems. The paper starts with developing a common foundation for enterprise modeling. For this purpose, high-level objectives and related requirements are considered and stepwise refined by looking at enterprises as action systems. The structural elements of MEMO are then introduced with respect to the previously analyzed requirements. Subsequently, the prospects and challenges of using enterprise models at run-time are illustrated by a conception of future enterprise software systems. The presentation of MEMO is complemented with a brief overview of related research. Finally, key elements of a future research agenda are suggested that reflect the prospects of enterprise modeling, but also fundamental technical, epistemological and political challenges.

2 Objectives, terminology and core requirements

Early contributions to enterprise modeling had three initial assumptions in common:

- The realisation of efficient business information systems recommends the *joint* analysis and design of the software system and the corresponding action system.
- The complexity of both, software system and action system, recommends developing appropriate *abstractions*.
- Co-designing information system and action system requires involving people with different professional backgrounds and different agendas. As a consequence, there is need for abstractions, i.e., models, that represent particular views. Furthermore, there is need for overcoming communication barriers between the various stakeholders.

These assumptions lead to a first definition of the term "enterprise model":

> An *enterprise model* comprises conceptual models of software systems, e.g., object or component models, that are integrated with conceptual models of the surrounding action systems, e.g., business process models or strategy models. Action system and information system are

not limited by the boundaries of a particular organization. Instead, an enterprise model may represent inter-organizational aspects as well.

Note that in recent years the term "enterprise architecture" has been introduced with a similar intention. While it is sometimes used with a special focus on high-level abstractions to address upper management, various definitions of the term (e.g., [35,62]) correspond with the above conception of enterprise models. The remainder of this section serves to develop a more elaborate conception of enterprise models and enterprise modeling. For this purpose, we will take a closer look at the subject—enterprises—and analyze requirements for an enterprise modeling method.

2.1 Basic requirements

An enterprise model is not an end in itself. On a high level of abstraction, it serves three interrelated purposes: to promote *communication and collaboration, control* and *change*. All three purposes are related to the information system in conjunction with the action system. Refining these generic purposes into high-level requirements provides a clearer idea of how to achieve them. The complexity of information systems and action systems stresses the need for thorough analysis and design. Analysis and design require concepts that help to structure the targeted problem domain appropriately. They also demand for concerted and purposeful guidelines for applying these concepts.

Requirement HR1 Enterprise models should include concepts that are suited to support the *conjoint* analysis and design of information system and action system. They should be supplemented by corresponding *methods*.

Complexity also demands for specialization and separation of concerns, which lead to the well-known problem of communication barriers that are caused by diverse agendas and technical terminologies. The following requirement reflects both aspects:

Requirement HR2 Enterprise models should provide abstractions and representations that correspond to the professional backgrounds of prospective users. To foster communication, enterprise models should also offer concepts that serve as *common reference* to diverse groups of stakeholders.

The demand for supporting analysis and design of information systems leads to a further high-level requirement:

Requirement HR3 Enterprise models should include concepts that can be mapped to implementation-level concepts according to clear transformation rules.

An enterprise model needs to represent the relevant aspects and features of a particular firm. These relevant features may vary to a large extent from firm to firm. Therefore, an enterprise model needs to be built for or adapted to the specific requirements of a certain enterprise, which results in the following requirement:

Requirement HR4 An approach to enterprise modeling should support the convenient and safe design of particular enterprise models. Convenient refers to the effort it takes to realize a particular enterprise model. Safe refers to support for model integrity.

Developing an enterprise model is a demanding project that requires a major investment. Therefore, it is important to account for the economics of enterprise modeling. This includes the effort it takes to build and use an enterprise model as well as its prospective benefits. Promoting productivity and reuse are proven approaches to reduce costs, which directs the focus on tools:

Requirement HR5 An approach to enterprise modeling should be supplemented by corresponding modeling tools. Tools should not be restricted to developing enterprise models, but should also support their purposeful use by supporting specific analysis and design tasks.

Note that tools are also suited to promote safe and convenient specification and adaptation of enterprise models (requirement HR4). However, even with sophisticated modeling tools, the effort to design an enterprise model from scratch may impose an effort too high for many firms. To cope with this challenge, providing reusable—and adaptable—artifacts can be an attractive option:

Requirement HR6 To reduce the costs of developing a particular enterprise model, an approach to enterprise modeling should include *reference models*.

Reference models come both with a descriptive and a prescriptive claim. On the one hand, they should account for actual features of the represented domain. On the other hand, they are supposed to serve as a blueprint for especially effective designs. To justify an investment into enterprise modeling, its prospective and actual benefits need to be accounted for. Benefits (and costs as well) depend on a range of factors that vary and cannot be controlled entirely—such as the skills and attitudes of users. Therefore, it seems unrealistic to demand for an approach to determine costs and benefits. Nevertheless, an approach to enterprise modeling should provide meaningful guidance.

Requirement HR7 An approach to enterprise modeling should contribute to the assessment of its economics

by clear descriptions of the intended rationale and by providing criteria to foster the transparency of ex-ante and ex-post assessment criteria.

2.2 Peculiarities of action systems

To a large extent, the previous high-level requirements reflect common assumptions about analyzing and designing business information systems. However, these assumptions remain on a superficial level as far as specific characteristics of action systems are concerned. An action system is a system of interrelated actions that reflect the corresponding actors' intentions and abilities, organizational goals and guidelines, contextual threats and opportunities, as well as mutual expectations. A major research stream in Organizational Studies was aimed at analyzing peculiarities of organizational action systems to gain a deeper understanding of how they work and how they can be managed and changed. This research has produced a plethora of organizational theories (e.g., [33,41,48]), interpretative schemes (e.g., [42,63]), and methods for guiding change (e.g., [3,28]). With respect to creation and use of enterprise models, the following aspects are of particular relevance:

Lack of transparency While the idealized conception of an enterprise assumes clear goals, objectivity and rational action, research in Organizational Psychology has produced overwhelming evidence that factual enterprises often lack this kind of transparency and coherence. Weick regards the lack of an explicit and sound goal system as more characteristic than its existence. According to his analysis, action systems are often "saturated with subjectivity, abstraction, guesses, ... and arbitrariness" [63, p. 5]. For our course of investigation, this lack of transparency leads to the question whether this is a characteristic feature of action systems that we better account for—or an insufficiency that could be overcome.

Contingent subject Modeling a system recommends abstracting on aspects that are widely invariant. Organizational action systems depend on individual action and on the respective environment. Both are not only hard to detect—relevant parts may not be visible—but they may also be characterized by substantial variety, the causes of which are hard to dissolve. On the level of individual actions this thought refers to unknown or varying intentions and related personal, sometimes deliberately hidden agendas. As a consequence, organizational action systems are characterized by multiple contingencies, which may be reciprocally intensified [38, pp. 148ff]. The so-called "contingency approach" in Organization Studies [36,48] was aimed at dissolving the factors that cause observed contingency but except for a few correlations on the macro level, it did not produce convincing results. With respect to enterprise modeling these sobering findings have two important implications. On the one hand, they raise the question whether there are commonalities beyond the dissolvable diversity (requirement HR6). On the other hand, they show a principal limitation of enterprise models. While they abstract—for good reasons—from individual actors, accounting for individual agendas may be nevertheless relevant for developing an appropriate assessment of a firm's potential. This is even more problematic as some actors will hide their agendas on purpose.

Pivotal relevance of language An action system is based on communication and cooperation which in turn imply the existence of a common language. At the same time, actions enrich utterances with meaning and reproduce certain patterns of reducing complexity. In other words: They constitute and reproduce *sense*. "Action, perception, and sense-making exist in a circular, tightly coupled relationship ..." [63, p. 159]. As a consequence, action systems will usually bulk against a formal specification. The concepts they are based on are often characterized by intentional semantics: The intentions that they reflect, make sense only through references to the corresponding actors' "Lebenswelt" (literally: "life world") [54]—an aspect that the late Wittgenstein illustrates with a hypothetic construction: "If a lion could talk, we could not understand him" [64, p. 358]. Hence, describing action systems solely with formal languages goes along with the risk of dysfunctional simplifications.

Cognitive perspectives Developing an elaborate appreciation of individual action and a framework for analyzing enablers and inhibitors of organizational collaboration recommends accounting for individual perception and conceptualization. This is also the case for developing models that fit individual cognitive styles—and account for inter-subjective differences. The term *perspectivity* ("Perspektivität" in German), which has a long tradition in Philosophy, Psychology and Sociology, serves to express that the way an individual perceives and understands the world, his "Weltanschauung", is characterized by a specific perspective, i.e., a cognitive disposition that is shaped by socialization, experiences, language games, etc. Hence, a perspective as a *psychological construct* constitutes a conception of reality, comparable to a particular viewpoint in spatial perception [23, p. 159], which helps to reduce complexity by constituting sense [37, p. 182]. If perspectives are shared among individuals, they foster communication, otherwise they impede communication.

Resistance to change Often, the analysis of action systems is aimed at change, e.g., to improve efficiency, to decrease costs, etc. However, action systems—and their linguistic foundations— constitute and reproduce *sense*, which is essential for understanding a complex environment and for reducing uncertainty and risk. As a consequence, action system will often show a remarkable persistence; many actors

will be extremely reluctant to accept or even support change [42, p. 233ff], [46].

Relevance of symbolic context The success of action systems depends on individual intentions, motivations and commitment. Therefore, it is often not sufficient to focus on organizational guidelines or governance only. Instead, there is need to account for corporate value systems, rituals, legends, common beliefs, i.e., to organizational culture [53]. Among other things, culture is represented and mediated through "symbolic actions" that are aimed at fostering motivation and commitment [47, p. 5]. They are regarded as a key element of managerial competence and a core prerequisite of promoting organizational change by creating sense. At the same time, they are directly related to characteristics of a human actor such as charisma, persuasive power, empathy, etc. In other words: They constitute a key success factor of organizational change that goes clearly beyond the mere application of enterprise models and corresponding tools.

2.3 Refinement of requirements

What is the consequence of our brief consideration of action systems? While some may regard the peculiarities of organizational action systems as insufficiencies that need to be overcome, others may see them as typical characteristics of social systems that need to be accepted as such. The interpretation we prefer is in between these extreme positions. First, it is reflected by the assumption that promoting rationality will often help to make action systems more effective—and more attractive, too. Promoting rationality includes various aspects that relate to enterprise models. It recommends fostering *transparency* by (re-) constructing clear structures. These do not only include a concise design of business processes and organization structure, but also a specification of a coherent goal system. A transparent goal system does not have to be sound. However, it would make conflicts explicit. Hence, clear (linguistic) structures contribute to decreasing contingency. Note that introducing such structures depends on the dynamics of the environment, which may require to challenge existing structures from time to time—or to introduce less restrictive rules to better cope with uncertainty. Furthermore, rationality demands for *justifying* decisions. An enterprise model that makes underlying assumptions explicit can effectively contribute to comprehensible justifications. Second, the peculiarities of action systems show clear limitations of formalization and of a mere engineering approach. It is not feasible to reconstruct the meaning of all utterances relevant for analyzing and understanding action systems adequately in an enterprise model. Also, changing an action system is not just a matter of "engineering" a better solution.

Against this background, the basic requirements presented above need to be partially revised. First, there is need to account for different cognitive perspectives. However, we cannot recognize the cognitive perspectives of others with certainty. Also, there is no need for representing particular individual perspectives. Instead, an enterprise model should account for perspectives that are characteristic for relevant groups of stakeholders. To develop representations of principal perspectives, it is important to focus on the corresponding technical language, e.g., the language of strategy analysts, the language of sales personnel, etc., since it will reflect characteristic goals, common practices and preferred levels of abstraction. Concepts of these languages could be reconstructed with a general purpose modeling language (GPML) such as the UML. Hence, one could use the UML as a language for enterprise modeling (see, e.g., [39]). However, such an approach comes with two major drawbacks that can be avoided by the use of a DSML: First, a GPML would compromise productivity because it would require reconstructing domain-level concepts from scratch—using primitive concepts such as "class", "attribute", etc. Second, it would jeopardize the integrity of enterprise models because the constraints that are built into the DSML would have to be defined by model users, too.

With respect to requirement HR4 (support for safe and convenient design), concepts of a certain technical language should be provided by a dedicated modeling language:

> **Requirement RR1** An approach to enterprise modeling should provide modeling languages that offer reconstructions of language concepts characteristic for cognitive perspectives of relevant stakeholder groups.

The notion of a modeling language allows for defining the term "modeling method", which we will use from now on as a replacement of the more generic term "modeling approach". It is based on a generic conception of method:

> A *method* is aimed at solving a class of problems. It consists of a terminology, accepted assumptions about successful action and a corresponding process model that guides the course of problem solving steps.
> A *modeling method* is a specific kind of method. It consists of at least one modeling language and at least one corresponding process model which guides the construction and analysis of models, i.e., it guides the meaningful use of language concepts.

Note that often the term "modeling methodology" is used instead. This is, however, misleading: A methodology is a study of methods. With respect to the contingency of action systems and to the variety of cognitive styles, it cannot be expected that a set of modeling methods provided with a method to enterprise modeling would be sufficient.

> **Requirement RR2** There should be support for tailoring methods to the needs of a particular enterprise and

the cognitive styles of prospective users. This includes the adaptation of modeling languages and corresponding process models.

Note that this requirement also relates to tool support: Only if modifying a modeling language is supported by tools, it can be accomplished efficiently. To avoid the naïve application of a method for enterprise modeling, it is mandatory to account for the peculiarities of action systems.

Requirement RR3 A method for enterprise modeling should stress the limitations of formalisation and a mere engineering approach. Instead, it should supplement a rational approach by emphasizing the reflective appreciation of action systems. For this purpose, it should stress the need for comprehension and empathy (e.g., [66, p. 20]) as well as for sense making [47,63].

This requirement shows some similarities to the assumptions underlying so-called "agile approaches" to software development. However, there is a clear difference: It demands for supplementing a model-centric approach, not for overcoming it (which is arguably the case for some proponents of agile approaches).

2.4 Requirements for modeling environments

To effectively and efficiently use modeling methods, corresponding tool support is required. This is for various reasons: Without a modeling tool, it will be hardly possible to verify (and guarantee) the correctness of a model's syntax. Furthermore, a tool promotes the protection of semantic constraints and referential integrity between integrated models. In addition to that, a tool may allow for transforming a model into further representations, e.g., implementation-level documents. Last but not least, a tool provides support for analyzing models and for adapting graphical representations to specific views or styles. Apart from these well-known benefits offered by modeling tools, there are further specific requirements. Since the set of DSML provided with a method cannot be regarded as static, there is need for supporting the modification of existing DSML and the creation of new ones.

Requirement TR1 A tool environment for enterprise modeling should include a meta model editor for specifying and modifying meta models.

To actually use a new DSML, it is not sufficient to specify its abstract syntax and semantics. Instead, a corresponding model editor is required that also features a graphical notation. Only, if the process of creating model editors for new or modified DSML is not too costly, it is a realistic option to extend an existing language base.

Requirement TR2 A meta model editor should efficiently support the creation of a model editor from a meta model. This includes the implementation of the abstract syntax and semantics as well as the additional definition of the concrete syntax.

It is a key characteristic of enterprise models that they integrate models of various perspectives on an enterprise. To illustrate the relationship between different models, it is useful to create diagrams that include representations of these models:

Requirement TR3 A tool environment for enterprise modeling should allow for creating *multi-language diagrams*, i.e., diagrams that integrate diagrams of models that were created with different DSML.

3 Multi-perspective enterprise modeling

The above considerations concerning requirements and limitations of enterprise models have inspired the development and evolution of "MEMO", a method to guide the creation and application of enterprise models. It is based on a conception of "multi-perspective enterprise models":

A *multi-perspective enterprise model* is an enterprise model that emphasizes accounting for *perspectives*.

In this definition, the term 'multi-perspective' is purposefully overloaded. The first conception of the term refers to a *cognitive perspective*. It represents a specific professional background that corresponds to cognitive dispositions, technical languages, specific goals and capabilities of prospective users. Hence, it is not an implicit feature of an enterprise model, but characterizes its intended purpose—to satisfy prospective users' perspectives. The second conception refers to the *representation* of cognitive perspectives within an enterprise model. The third conception stresses the need for additional perspectives that go beyond the construction and use of enterprise models. It recommends reflecting upon the limitations of an exclusively engineering approach by accounting for peculiarities of action systems.

MEMO is comprised of four key elements: A high-level framework, domain-specific modeling languages, accompanying methods and tools. The high-level conceptual framework represents a holistic perspective on an enterprise that is composed of certain *generic perspectives* (e.g., strategy, organization, information system) each of which can be further detailed into various *aspects* (e.g., resource, structure, process, goal). Both, perspectives and aspects can be adapted to the needs of a particular application domain. Each aspect of a generic perspective represents a *particular perspective*. Particular perspectives can be combined to further specific perspectives. In the exemplary framework in Fig. 1, each

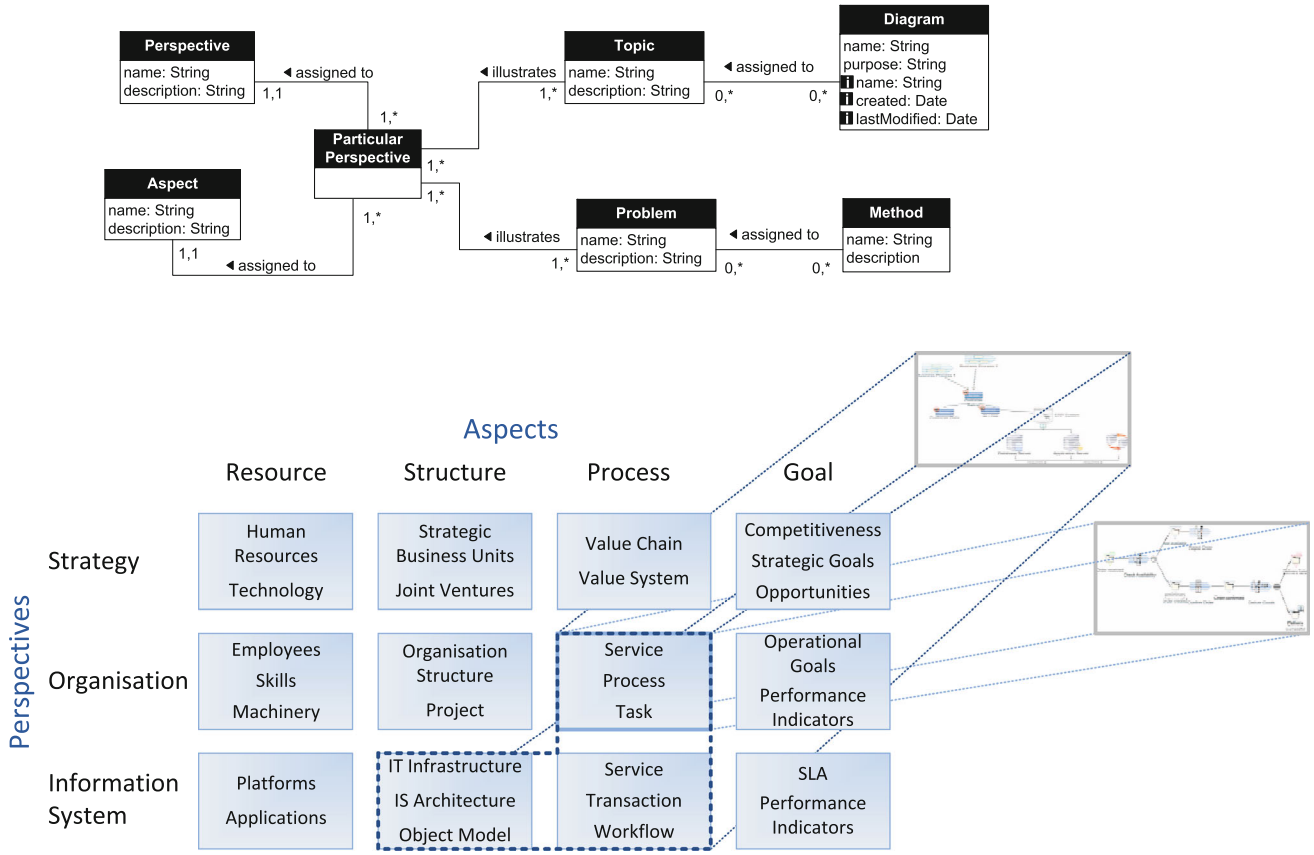

Fig. 1 High-level framework for structuring the enterprise and associated diagram types

particular perspective is illustrated by characteristic topics of interest for analysis and design. The framework serves as a starting point for identifying perspectives that require further attention. To allow for more elaborate analyses, each selected perspective is associated with a set of diagram types. Each diagram type is associated with a set of DSML. A diagram type in turn can be assigned to a collection of perspectives. Perspectives can also be supplemented and characterized by typical problem classes, which can be associated with corresponding modeling methods. Figure 1 shows a typical configuration of the framework and the corresponding meta model.

3.1 Language architecture and meta meta model

To provide concepts that support specific analysis and design tasks (requirement HR1) and to foster the representation of cognitive perspectives (requirement RR1), MEMO provides a set of DSML. The DSML also promote productivity and integrity of designing enterprise models (requirement HR4). A *language architecture* serves to satisfy the demands for adaptability, extensibility and integration (requirements HR2, RR2). It consists of a meta meta model that specifies the abstract syntax and semantics of a meta modeling language—the MEMO MML—and an extensible set of meta models which serve the specification of DSML and that are specified with the MEMO MML (see Fig. 2). The language architecture also reflects the demand for fostering reuse: While the range of reuse a specific enterprise model (M_1) allows for will usually be low, the range of reuse of corresponding DSML (M_2) will be clearly higher—but is still restricted to certain domains, e.g., organizational goals or business processes. The meta meta model (M_3) can be reused independent from particular domains.

The decision for developing a meta modeling language instead of using an existing one was based on two reasons. The first is a historical one. When we started with specifying DSML, there was no adequate meta modeling language available. The second reason relates to the experience we have gained with the specification and utilization of DSML. From time to time, it resulted in new, partially challenging requirements for a meta modeling language. As a consequence, maintaining our own meta modeling language became an important part of our research and helped us shaping our conception of DSML. At the same time, alternative options that emerged during the last years turned out to be not satisfactory for our purpose (for a comparative evaluation of UML MOF [44], Ecore [59] and the MEMO MML see [16]). Core

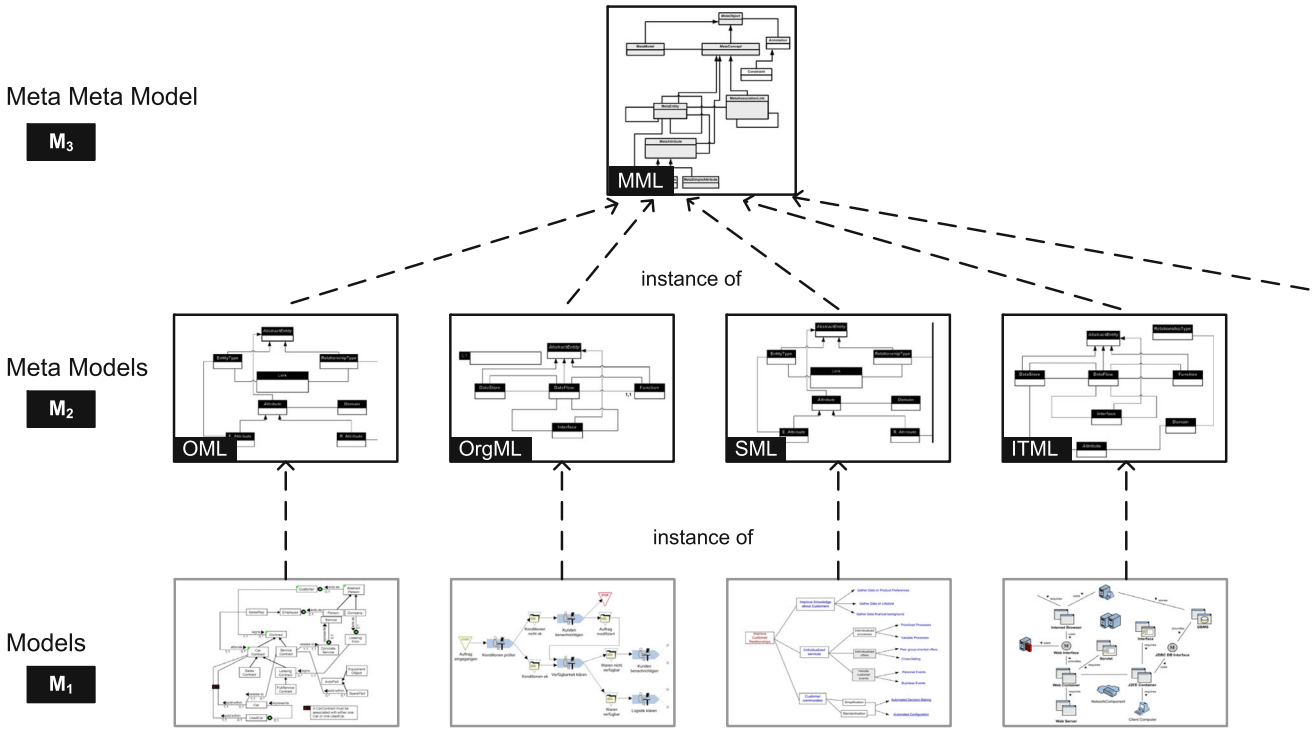

Fig. 2 MEMO language architecture

concepts of the MEMO MML, such as the specification of specialisation, correspond to the semantics of object-oriented programming languages to support a clear mapping to implementation-level concepts (requirement HR3). Among others, specific features of the MEMO MML include support for defining language concepts to specify *intrinsic features*, language *concepts to model instances* and concepts to support the integration with operational information systems. Figure 3 shows the meta meta model that serves the specification of the abstract syntax and semantics of the MEMO MML. The different background shades—grey and white—of the rectangles representing language concepts indicate different levels of abstraction. For instance: Different from "MetaEntity", "Comment" is not a language concept, but is instantiated only once as a supplement to a model.

While concepts such as "MetaEntity" and "MetaAttribute" correspond directly to respective instances in a meta model, "MetaAssociationLink" is introduced to allow for the elaborate specification of associations. Each instance of "MetaAssociationLink" is linked to exactly one instance of "MetaEntity"—and to a further instance of "MetaAssocationLink". Hence, multiplicities and role names are defined for each instance of "MetaAssociationLink". Also, an optional designator can be assigned to both instances of "MetaAssociationLink", allowing for defining one designator for each reading direction.

Specifying a meta model—i.e., reconstructing the technical terminology of the targeted domain—requires reflecting upon the ontological essence of a term. At the same time, it recommends taking into account that instances of a meta concept are types. Sometimes, this results in the problem that the essence of a term includes features that do not apply directly to the type level (i.e., the M_1-level in Fig. 2). Instead, they apply to the instances represented by a type. For example: The meta type "Business Process" can be instantiated into the type "Order Management". Although our idea of an order management process includes the assurance that every particular instance has a start time and an end time, it is not possible to express this knowledge with the specification of the type. The problem is known in object-oriented modeling and software development. The UML offers a concept called "powertype" [45, p. 57], which, however, seems artificial and therefore hinders its intuitive use (for a more detailed evaluation see [16, pp. 17ff]. Atkinson and Kühne [4] suggest a concept which they call "clabject": A clabject can be specified using "fields" that either represent a meta type attribute—which is supposed to be instantiated and initialized on the type level—or a feature of instances of the type. These two meanings of a field are differentiated through so-called "potencies". For example, the field "startTime" within the meta type "Business Process" could be assigned the potency 2 to express that it must not be initialized on the type level (potency 1), but only on the instance level. The concept of "intrinsic" features or types that is part of the MEMO MML is similar to the concept of a "clabject", however, it was designed for easier use. A (meta) type

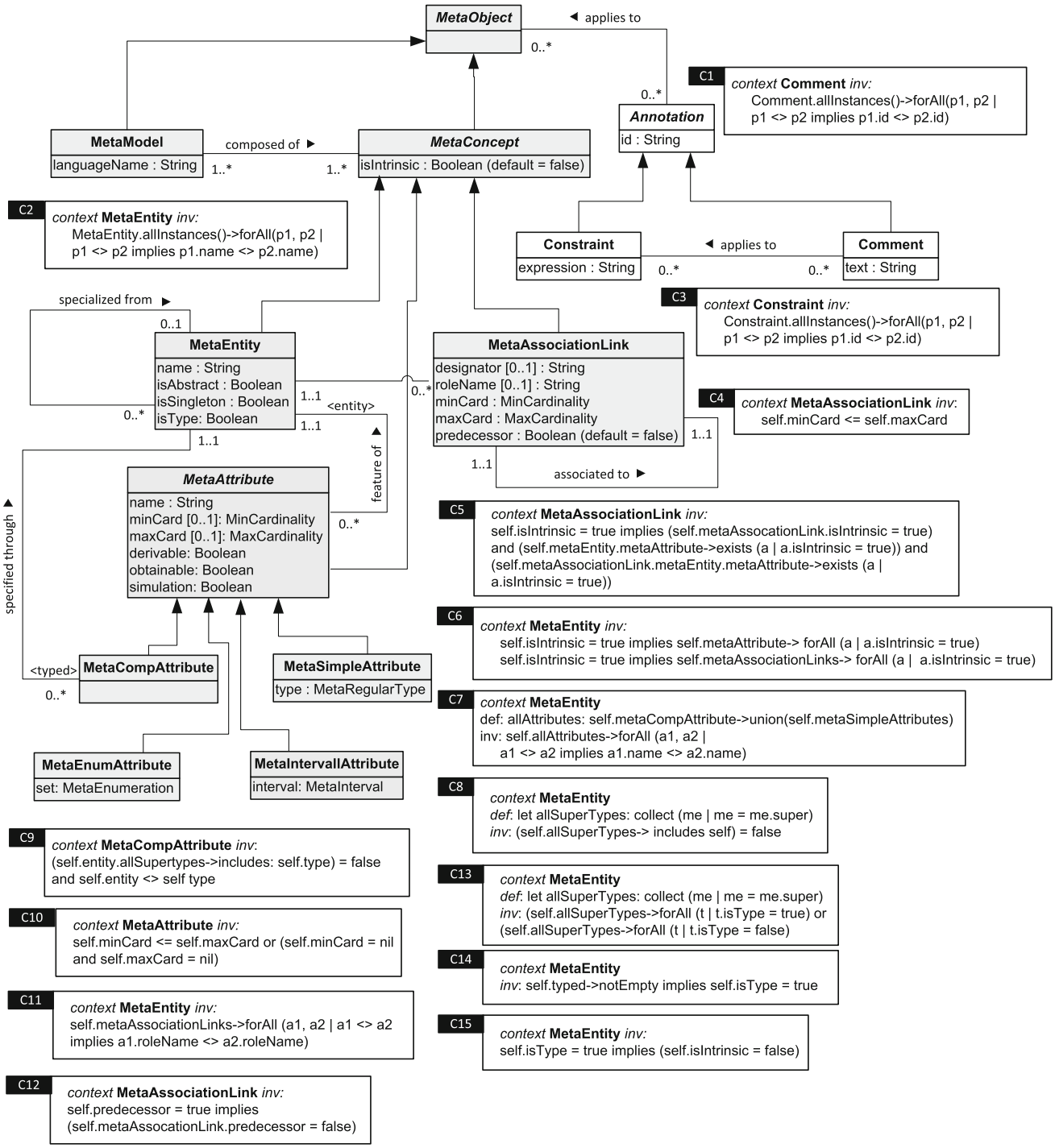

Fig. 3 MEMO meta meta model

may have (regular) attributes that apply to its instances or "intrinsic attributes" which can be instantiated only with the instances of its instances. Hence, intrinsic attributes correspond to fields with a potency value of 2 in Atkinson and Kühne's terminology. Furthermore, our concept covers associations, too: An association that gets effective only with the instances of the entity types it connects is called an "intrinsic association". An entity type that must not be instantiated directly, but only on the level below the one it is presented on, is called an "intrinsic type". Note that all attributes of an intrinsic type are intrinsic by default for the entire lifecycle of that type. Also, all associations an intrinsic type is involved in must be intrinsic as well. The concept is specified in the meta meta model through the attribute "isIntrinsic" in

"MetaConcept" and the constraints C5, C6 and C15. "Obtainable feature" is a further concept provided by the MEMO MML (attribute "obtainable" in "MetaAttribute"). It serves to mark attributes which may be initialized with data obtained from an external system, e.g., an ERP system. To foster the distinction of meta models and models on other levels of abstraction, the MEMO MML features a characteristic graphical notation (see Fig. 4).

Conceptual models are aimed at abstraction. Hence, they should not represent particular instances, the state and even the existence of which may change over time. However, sometimes it can make sense to include representations of instances into an enterprise model. Possible examples of instances that could be included into models are cities, countries, or organizations (e.g., a particular company). To enable the specification of modeling languages that allow for representing instances, a meta modeling language needs to offer concepts that can be instantiated into types (instead of meta types). The MEMO MML [16] provides a corresponding concept—realized through the attribute "isType" of "MetaEntity"—however, at the price of overloading the meta meta model, which is accounted for with the constraints C13, C14 and C15.

Currently the language architecture includes a language for object-oriented modeling, OML [12]; a language for modeling organizations, both organizational structures and business processes, OrgML [17,18]; a language for representing strategic aspects such as goal systems or value chains, SML [21]; and a language for modeling IT resources on various levels of detail, ITML [34]. Further languages target the modeling of resources [30], the design of performance indicator systems [60] or various aspects of corporate knowledge management [49]. Distinguishing these languages is mainly motivated by the need for reducing complexity: While it is conceivable to define one multi-purpose language that allows for creating all intended diagram types, such an approach would result in a level of complexity that could hardly be maintained anymore. The modeling languages are integrated through common concepts. For example, both, the OrgML and the ITML utilize the common language concept "Business Process". In turn, corresponding models are integrated through common instances of these common concepts. For example: A model of an IT infrastructure may refer to an order management process type that is used in a corresponding organization model as well. Figure 4 illustrates the integration of modeling languages in MEMO through common language concepts. Note that multiplicities are omitted to foster readability. Concepts of the meta model are assigned to perspectives. Also, it is indicated in which diagram types they are intended to be used. The screenshot at the bottom of Fig. 7 in Sect. 3.4 shows a corresponding multi-language diagram (i.e., an instance of a diagram type).

3.2 Support for method engineering

Enterprise models provide the conceptual foundation for supporting a wide range of analyses and design tasks related to the interplay of information system and action system. Originally, the main emphasis of MEMO was on methods to support the design of information systems that are conjointly developed with the action system. A typical example of this strand of research has focused on generating workflow applications, i.e., a workflow schema and additional code, from a workflow model that refers to a corresponding object model [29] or on the model-based development of customized electronic commerce systems [21]. In recent years, the focus shifted from support for software development to other scenarios. They include the design of performance indicator systems that are integrated with business processes and business objects [60], a modeling method to support IT management [20], a modeling method for IT audit risk assessment [61] and a modeling method for supporting mergers [57]. Due to the immense diversity of problems that may be addressed with enterprise modeling, a predefined set of modeling methods is not sufficient. To cover a wider range of problem classes it is required to support the tailoring of modeling methods. Specialization of existing methods is hardly satisfactory, because it would be restricted to adding further features and would not allow for defining entirely new process models.

3.2.1 Conceptual foundation

To promote greater flexibility, MEMO supports method engineering. For this purpose, a meta model of modeling methods can be instantiated into particular methods or projects respectively. Since a modeling method consists of a set modeling languages and at least one corresponding process model, the meta model includes concepts to define a process model on the macro and the micro level. Figure 5 shows an excerpt of the meta model and an excerpt of an exemplary instantiation into a particular project or method type. Note that the process model should not be mistaken as a waterfall model. Instead, it represents an idealized process that may require returning to previous phases. Each phase can be further detailed on a micro-level. For this purpose the meta model offers concepts such as "Goal", "Role", "Action", "Position", "Diagram" or "Resource". Among other things the structure includes the required input, objectives, participants, required diagram types, required actions and intended results. Participants are specified through references to positions within a corresponding organizational chart and/or to roles of a corresponding role model—both specified with the MEMO OrgML.

Among other things, each phase can be assigned states of diagram types which provide views on one or more models, each of which is specified by one of the MEMO DSML. With

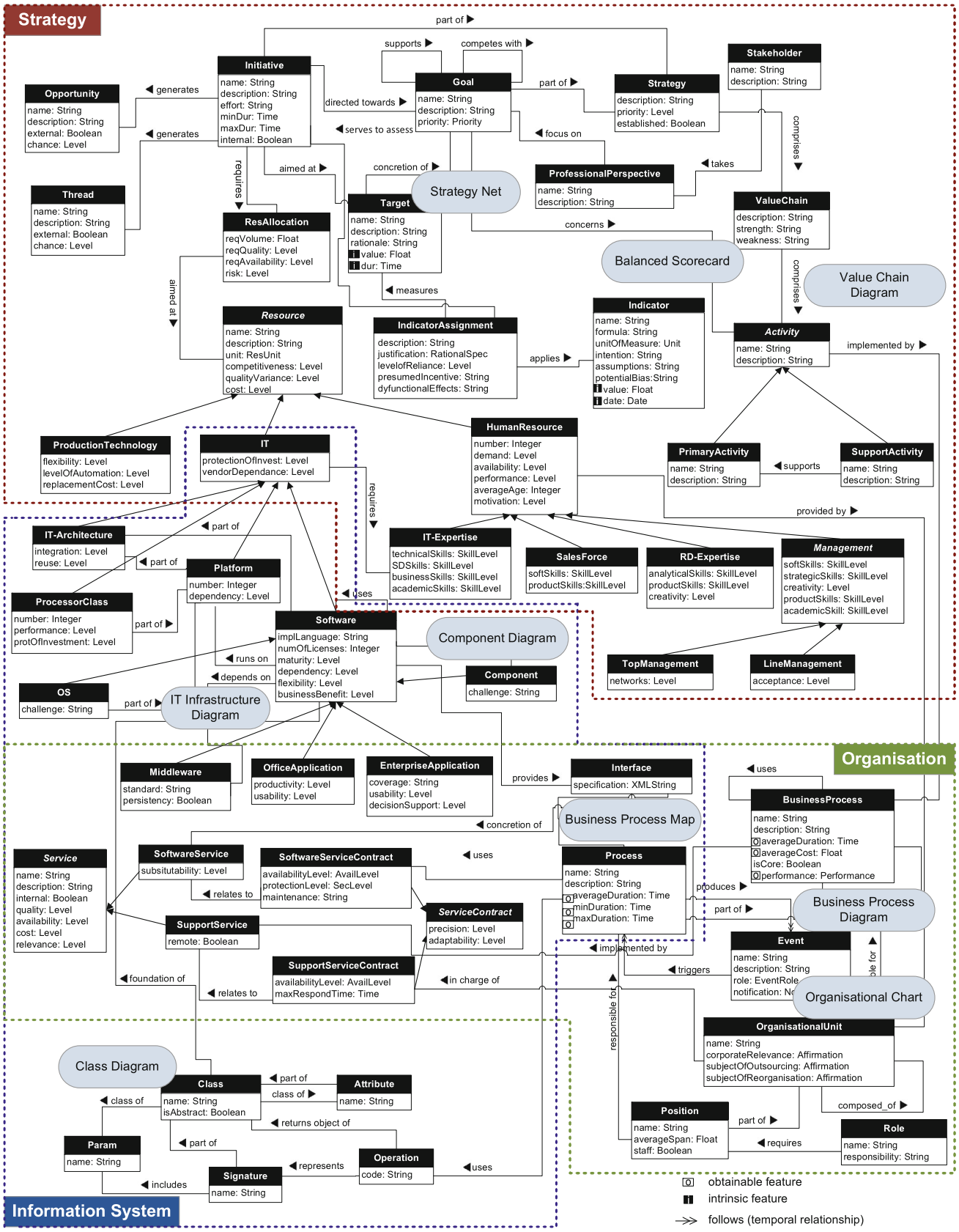

Fig. 4 Excerpt of integrated meta models

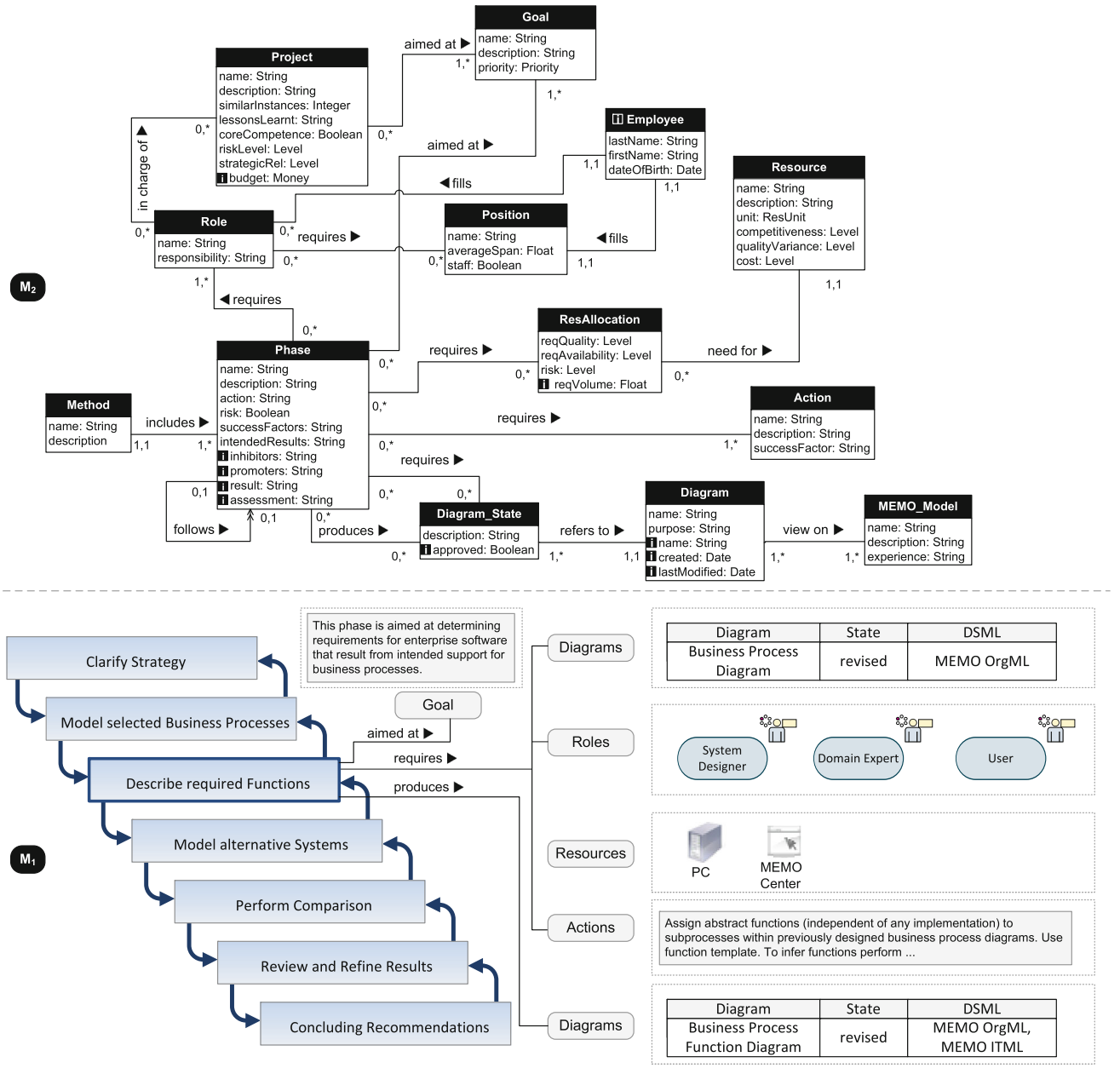

Fig. 5 Excerpt of meta model for method engineering and exemplary instantiation

respect to the expressiveness and flexibility of the approach, it is important that the concepts shown in the meta model are associated with concepts in meta models of the DSML used in a method. For instance: "Position" is part of the OrgML and could be associated with organizational units, skill profiles, etc. "Goal" can be associated to other goals up to those represented in a strategy model. The meta model includes various intrinsic features that can be initialized on the M_0 level only. Take, for instance, the specification of the meta type "Diagram". The regular attribute "name" serves to specify the name of a diagram type, e.g., "Business Process Diagram". The intrinsic attribute "name" serves to define the name of a particular business process diagram, e.g., "Order Management". The intrinsic attributes "created" and "lastModified" refer to a particular diagram. The intrinsic type "Employee" can be instantiated only on the M_0 level (which is not shown in Fig. 5). This allows for assigning particular employees to a specific project—which would be instantiated from a project type. The meta type "MEMO_Model" serves to define types of models, hence, modeling languages.

The range of possible methods can be further increased, if one takes advantage of the extensible set of DSML: Modifying or adding DSML will allow for use scenarios not covered by the given set of languages. This will, however,

demand for substantially higher expertise and effort. With respect to requirement RR3 it is important to note that the term "method *engineering*" may trigger inappropriate expectations. The suggested approach to guide the configuration of a method or the organization of a project respectively focuses on one aspect of methods only: their analytic or rational structure. While we trust the assumption that an elaborate (not: a bureaucratic) structure supports purposeful reduction of complexity and thus supports analytical reasoning, we do not think it is sufficient. In addition to the analytic part of a method there is need for supplementary measures that are aimed at creating sense, build trust, getting people involved, etc.

3.2.2 *Exemplary application: a method for selecting ERP systems*

The acquisition and introduction of large enterprise software systems such as enterprise resource planning (ERP) systems requires a major investment with a long-term impact. Therefore, selecting a system demands for thorough analysis and assessment. Since large enterprise software systems tend to penetrate the entire firm, they have a clear impact on the action system. At the same time, the organization of the action system affects the performance of the software. Hence, a specific enterprise model promises to serve as a useful conceptual foundation. A corresponding modeling method can be customized using the above approach to method engineering. It could comprise three DSML: the strategy modeling language (SML), the organization modeling language (OrgML) and the language to model IT infrastructures (ITML). The exemplary instantiation in Fig. 5 shows the macro-level process model and the refined specification of a selected process phase according to concepts defined in the meta model. The core of the method is aimed at the analysis of requirements for the targeted range of ERP systems. For this purpose, the MEMO ITML includes the concept "RequiredFunction". An instance of this concept—i.e., a type of a required function—can be assigned to a subprocess within a business process and then specified according to the predefined structure.

Figure 6 shows an excerpt of a corresponding business process diagram (MEMO OrgML) that is enhanced with representations of required functions. After required functions have been specified for a number of business process types, there may be need to refine and harmonize their specifications. Subsequently, the requirements are checked against the functions provided by the candidate enterprise software systems—resulting, e.g., in tables that serve as input for the selection decision. While using a tool is not mandatory for constructing and applying the method, it would certainly contribute to efficiency and integrity. It could, for instance, enforce that all required entries are made, check them for plausibility and transform models into other representations used for decision making—such as tables or drawings.

3.3 Additional support

The construction and application of customized methods is supported by further instruments. They include reference models (requirements HR4 and HR6), a method for designing DSML (requirement RR2), a method to guide the assessment of economic aspects (requirement HR7) and specific approaches to account for peculiarities of action systems (requirement RR3).

In the last 15 years, the application of MEMO has resulted in a number of reference models for different domains. Various reference strategy models and more than 80 corresponding reference process models for electronic commerce have been published in detail [21].[1] The design and adaptation of a DSML raises a particular challenge. Since DSML represent an artifact most prospective users are not familiar with, it will usually be no option to directly ask them for requirements. Instead, there is need to give users an idea of what they could expect from a DSML (and from a corresponding method). Our experience has shown that *use scenarios* provide a suitable starting point: Against the background of a use scenario prospective users are familiar with, they are presented with a (mock) diagram in a preliminary notation specifically conceived for encouraging discussions with prospective language users. To account for cognitive perspectives (requirement RR1), the concepts represented in the diagram are reconstructed from technical terms and from prevalent analysis approaches prospective users can be expected to know about, such as a balanced scorecard, a widely used conceptual tool in management practice. The diagram then serves to jointly develop analysis and design tasks that it could support to address.

This process will gradually result in requirements for refining the concepts and/or adding further ones. The corresponding method for designing DSML is outlined in [15]. It also provides support for typical design decisions such as the differentiation of modeling language and model (for a more detailed analysis see [19]). An approach that allows to model essential aspects of potential modeling scenarios supports evaluating the economics of enterprise modeling by increasing transparency [65].

Four dedicated approaches account for peculiarities of action systems. At first, it is an essential characteristic of MEMO to stress a critical stance on a naïve view of action systems. Second, language specifications include concepts

[1] For a comprehensive documentation of the reference models see http://www.wi-inf.uni-duisburg-essen.de/FGFrank/ecomod/.

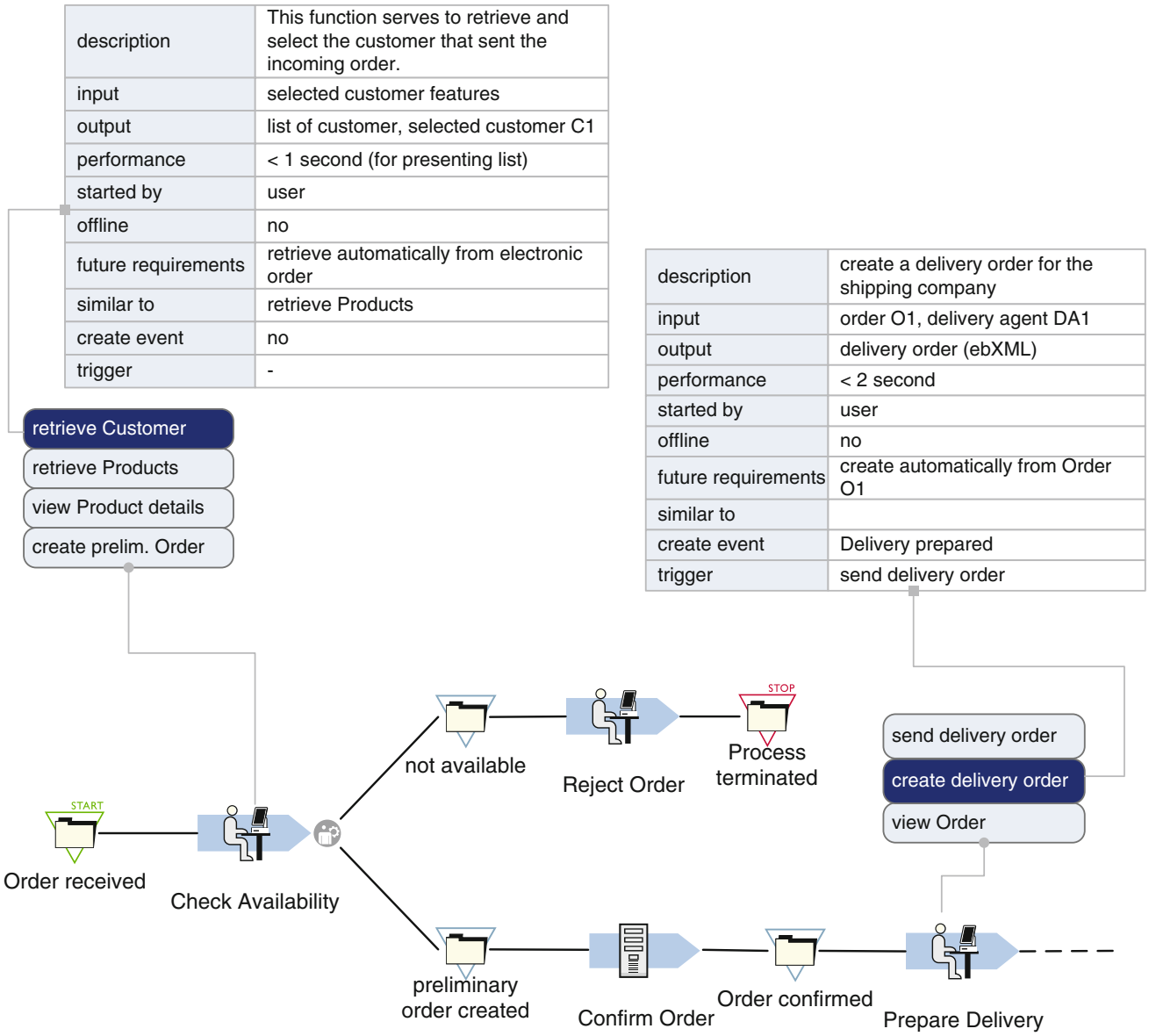

Fig. 6 Example of diagram to support requirements analysis for ERP systems

to draw attention to limitations of formalization and possible dysfunctional effects (see, for instance, the specifications of the meta types "Indicator" and "Indicator Assignment" in Fig. 4). Third, there are language concepts that demand for an explicit justification of modeling decisions. For example: If a certain indicator type for measuring the performance of a business process is specified with a respective DSML, a corresponding modeling concepts provides a structure ("RationaleSpec") for describing the rationale of related modeling decisions [60]. Fourth, process models can be supplemented with references to dedicated instruments, e.g., workshops, discursive assessments, etc., and additional documents, e.g., transcripts, "rich pictures" [6], etc.

3.4 Tool environment

The tool environment, MEMO Center, combines a set of model editors that allow for creating multi-language diagrams (requirements HR5 and TR3) with a meta modeling component that supports the development of new model editors (requirements TR1 and TR2). Originally implemented in Smalltalk, it was later re-implemented in Java on the Eclipse platform—primarily to take advantage of the Eclipse Modeling Framework (EMF) and Graphical Modeling Framework (GMF) [24,59].

The meta model editor allows to design meta models with the MEMO MML. The implementation is based on a

reconstruction of the meta meta model as an Ecore instance [16, p. 35]—representing an abstraction on the M_3 level. As soon as a meta-model is finalized, it can be transformed into a corresponding Ecore instance—this time representing an abstraction on the M_2 level.

Through the transformation, the original meta model is finally represented as an object model that includes additional features which go beyond the mere language specification, e.g., create and delete operations or time stamps. Subsequently, further specifications, mainly assigning the symbols of the concrete syntax, have to be added. Finally, the new model editor needs to be integrated with the existing modeling environment. While this still requires remarkable expertise and effort, the MEMO meta modeling editor and the GMF, it is part of, facilitate the implementation of additional model editors to a great extent. Currently, MEMO Center lacks specific support for method engineering. Figure 7 shows the relationship between the language architecture and the tool environment. It also illustrates the process of generating/implementing and integrating particular model editors.

4 Related work

Based on the conception of enterprise models suggested in Sect. 2, there are various approaches that focus on enterprise modeling. They can be differentiated with respect to their primary purpose, the institutional context (academia, commercial), the DSML and the methods they provide. The selection is based on the assessment of two aspects: the contribution to the evolution of the field and the correspondence to the conception of enterprise modeling proposed in this paper. The "Unified Enterprise Modeling Language" (UEML), which resulted from an EU project and an ISO standard related to enterprise modeling (ISO 14258:1998) are not accounted for. Work on the UEML finished at a premature level. The ISO standard has a focus different from the conception of enterprise modeling proposed in this paper. "Business Engineering" [43] is also aimed at fostering a more efficient alignment of business and IT. It also accounts for cultural and political aspects and puts emphasis on change management. However, while it makes use of conceptual models for this purpose, it lacks a conception of enterprise model. (Meta) modeling environments such as MetaCase,[2] the "Next Generation Modeling Framework" [31] or Cubetto[3] are not part of the comparison either: They may be used for creating enterprise modeling tools, however, that is not their primary or only purpose.

[2] http://www.metacase.com/.

[3] http://www.semture.de/de/cubetto-toolset.

4.1 Selected approaches

One of the first approaches to enterprise modeling was presented by Zachman [67]. He used the term "information systems architecture" in a broader sense. According to his terminology, an enterprise model is part of an information systems architecture. Zachman's approach was motivated by his experience as systems engineer and sales representative with a major software vendor. He realized that there is need to communicate the functions and organizational effects of complex software systems, especially of Computer Supported Manufacturing Systems to prospective customers. His approach was inspired by an architect's paradigm. Similar to a building plan, he wanted to present customers and other stakeholders with comprehensible representations of an information systems architecture. At its core, the approach consists of a high-level framework that differentiates roles, e.g., "planner","owner", "designer", and topics such as "data", "function", "time", "people". The framework was intended to guide descriptions of an information system and the corresponding enterprise that satisfied the perspectives of the proposed roles. Although Zachman gives a few examples for modeling particular perspectives, e.g., the ERM or DFD, his framework is not accompanied by specific modeling languages. A later revision was aimed at filling this gap. Together with the logician Sowa [56] he presented a logic-based graphical language that was, however, not further developed. The approach does not include specific methods that guide the design and use of particular "information systems architectures".

The "Architecture of Integrated Information Systems" (ARIS) was originally aimed at supporting the design of information systems in industrial enterprises. To achieve this purpose, Scheer [51] suggests a high-level framework, referred to as "House of Business Engineering", that differentiates various views on a firm ("organization", "data", "function", "process", etc.). Each view is further differentiated into domain-level concepts, IT concepts and implementation. ARIS includes basically one DSML—which gained outstanding relevance: the event-driven process chains for modeling business processes. ARIS does not include an elaborate meta modeling language. For assessing the support with further DSML and methods provided by ARIS it is recommended to distinguish between the academic version that is subject of various publications and the commercial version that has evolved around a modeling environment, the "ARIS toolset". In his publications, Scheer [52] provides meta models of the various views and also process models as part of corresponding methods. However, both remain on a superficial level and seem to serve as illustrations of generic concepts only. The "ARIS toolset" includes a wider range of further modeling languages. It is also accompanied by a method handbook for commercial use.

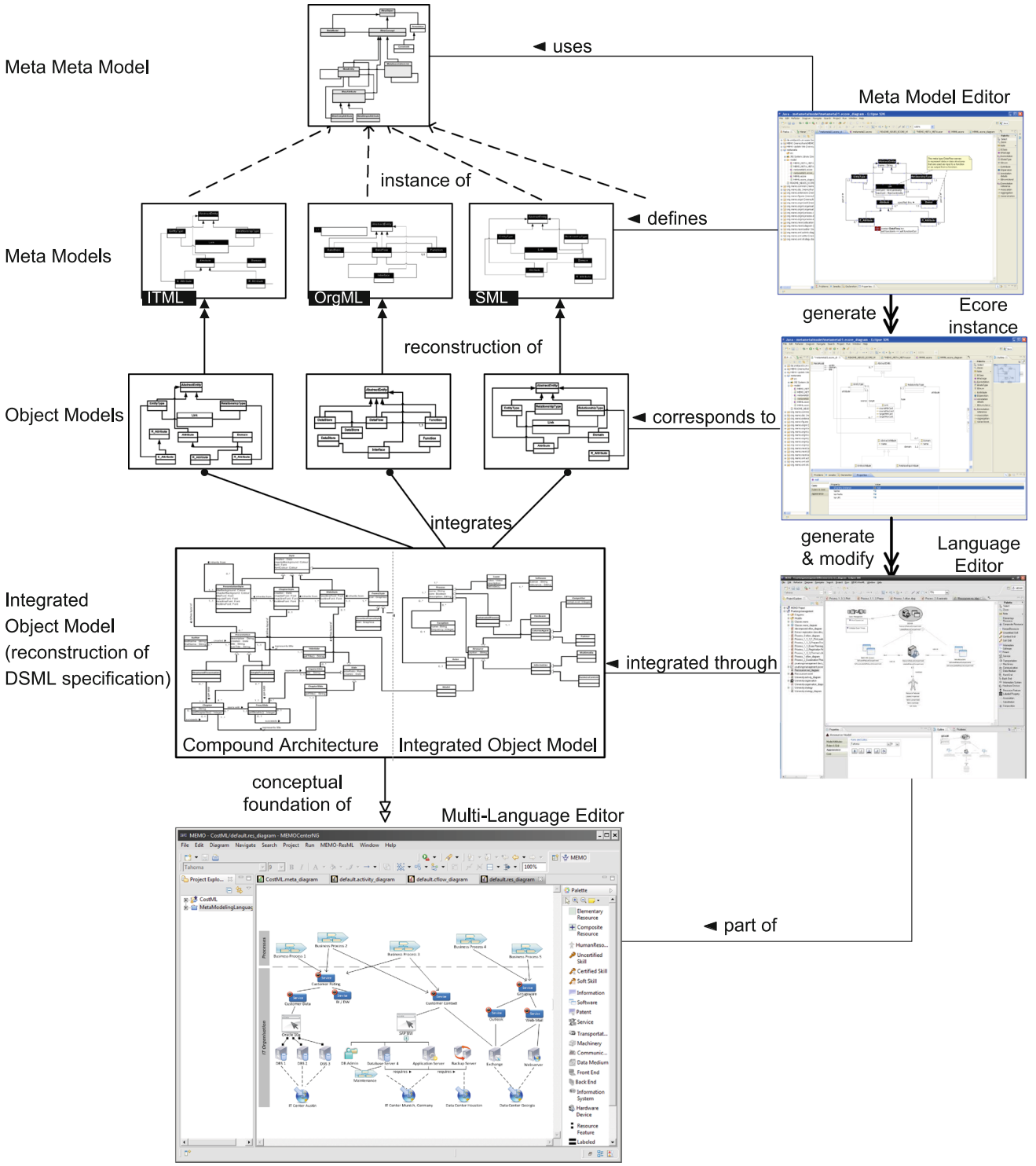

Fig. 7 Language architecture and corresponding elements of tool environment

Similar to ARIS, CIM-OSA (computer integrated manufacturing, open systems architecture) was aimed at developing information systems for manufacturing enterprises—with special emphasis on reuse [2]. It resulted from a project conducted by an industrial consortium funded by the European Community. The approach includes a high-level framework that features three dimensions: Views ("function", "resource", "information", "organization") serve to distinguish relevant perspectives on the firm. Different levels of abstraction stress the claim for reuse:

"Generic"' models are suited for a wide range of enterprises. "Partial" models are suited for a restricted range, e.g., a certain industry, and "particular" models are specific to a particular enterprise. Evolutionary phases represent the process of a high-level method for developing information systems from enterprise models. CIM-OSA suggests templates to represent views, but it does not include DSML. Similar to Zachman, CIM-OSA is not further developed anymore.

SOM ("Semantic Object Model") by Ferstl and Sinz [10] has its primary focus on supporting the development of information systems and corresponding action systems. For this purpose, the authors follow a paradigm that is chiefly influenced by systems theory. They also emphasize the use of economic concepts, such as "business transaction". SOM includes a set of rudimentary DSML—for process modeling and for object-oriented modeling. Its main contribution, however, is an elaborate analysis and development method that follows a systems theory approach.

Different from the other approaches, DEMO ("Dynamic Essential Modeling of Organizations") does not primarily take a top-down approach to analyzing enterprises. Instead, its main focus is on individual action, patterns of interaction and the role of language. DEMO [7] builds on various theoretical approaches in the Social Sciences and in Philosophy—including the Speech Act Theory [55], Organizational Semiotics [58], the Theory of Communicative Action [26] and Bunge' work on philosophical Ontology [5]. Based on these theories, DEMO provides elaborate concepts to analyze communication. Dietz pursues both a critical and a constructive goal. On the one hand, he uses his specific terminology to show deficiencies of prevalent (data) modeling languages. On the other hand, he guides the design of systems with a method by starting with basic transactions. DEMO is supplemented by some high-level modeling concepts—referred to as "meta ontology" or "World Ontology Specification Language" [8]. However, it seems not to include elaborate DSML. Also, it seems not to account for software-engineering aspects. The main contribution of DEMO is the specific perspective on action and language. While MEMO also emphasizes the relevance of language and the need to account for peculiarities of action systems, the concepts suggested by DEMO allow for a more differentiated analysis of communication acts.

TOGAF[4] is "a framework ... for developing enterprise architectures". It is promoted by an industrial consortium, the "Open Group", which comprises some large international software vendors and user organizations. The framework is documented in an extensive report [25]. While the term "enterprise architecture" is not clearly defined by the Open Group, it seems to be aimed at abstractions of both, the IT and the business in order "to achieve the right balance between IT efficiency and business innovation" [25, p. 6]. TOGAF comprises a method for the development of enterprise architectures, the "Architecture Development Method" (ADM). It includes eight main phases which are subdivided into further steps. The method includes references to diagram types, e.g., "Application Communication diagram", "Application and User Location diagram", "Enterprise Manageability diagram", "Application Migration diagram" [25, p. 133]. However, TOGAF lacks corresponding language specifications. It only offers a "content metamodel" [25, pp. 375ff] that remains on a rudimentary level and is not suited as a specification of elaborate modeling languages. In addition to ADM, the framework includes a somewhat eclectic collection of—among other things—references to "architecture principles" and "architecture patterns", guidelines for developing business scenarios and high-level reference models. Different from the original claim to account for business issues, the reference enterprise architectures represent technical aspects only [25, pp. 578ff].

Archimate[5] is also supported by the Open Group. It supplements TOGAF with a language for modeling enterprise architectures. Similar to the MEMO language architecture, it differentiates between domain-specific concepts (which correspond to the MEMO DSML) and generic concepts such as "Object", "Relation", etc. Different from MEMO, these layers are part of one language. There seems to be no distinction between a meta modeling language and a set of modeling languages. With respect to enterprise modeling, it is important that the language is differentiated into three so-called "layers": The "business layer", the "application layer" and the "technology layer". These layers are defined in specific parts of the overall meta model. The language specification remains on a coarse grained level. The Archimate language can be extended by adding attributes to meta types or by specialising meta types.

4.2 Comparative assessment

The overview of related work exhibits clear commonalities such as high-level structures to distinguish principle views on the enterprise, or the use of modeling languages and methods. A closer look, however, shows that there are clear differences between these artifacts: not only that DSML and corresponding methods are specific for each approach, there is also no common meta modeling language that would support the integration of DSML. MEMO is characterized by elaborate language specifications—and a corresponding tool environment as well as support for method engineering. ARIS is an exception insofar as it includes an elaborate DSML for

[4] While "TOGAF" looks like an acronym, it seems not to be dissolved in the official documents.

[5] http://www.opengroup.org/archimate/doc/ts_archimate/.

business process modeling and a comprehensive commercial tool environment that covers further languages. While the extension of the TOGAF documentation is impressive at first, it is far from a coherent method and lacks DSML—which are provided by Archimate, but still on a level that requires refinement. SOM deserves special attention for its foundation in systems theory, which is suited to provide an inspiring additional perspective on the enterprise. DEMO may also enrich prevalent approaches with its specific language/action perspective. CIM-OSA still deserves attention for its approach to promote reuse on various levels of abstraction. Table 1 summarizes the comparative overview.

5 Concluding remarks and outlook

Enterprise models are versatile abstractions of complex enterprises that support communication, but also dissemination and reuse of knowledge. If they are based on respectively designed DSML, they do not only help bridging the linguistic/cognitive gaps between various stakeholders in and outside a company, they also support the transformation of models into corresponding implementation level representations. Hence, enterprise models contribute to reuse and integration that comprises both, the information system and the action system. They also provide abstractions of the enterprise which can be used in a wide range of modeling methods that address complex analysis and decision scenarios. Furthermore, using them at run-time allows for evolving enterprise software systems to versatile management tools that integrates conceptual perspectives with instance-level representations. While the prospects of enterprise models will probably be regarded as attractive by many, they are accompanied by peculiar challenges.

5.1 Peculiar challenges

A main focus of research on enterprise modeling is on the specification of DSML. This objective is related to three substantial challenges. First, the development of a DSML requires domain-specific knowledge that may not be available in Information Systems or Computer Science. Thus, there is need for collaboration with other disciplines, e.g., Business and Administration, and with representatives of the respective domains in practice. Limitations of current implementation languages recommend collaborating with researchers in the field of software engineering. Second, designing a DSML requires deciding for a certain language paradigm such as object oriented or logic based. Each paradigm has specific advantages and limitations. Therefore, future research can hardly avoid the question how different language paradigms can be effectively combined, e.g., for using parts of enterprise models as a foundation for deductive reasoning. Third, the development of DSML raises principle epistemological questions. Our ability to (re-) construct new linguistic concepts depends on the languages we know—both domain-specific language and modeling languages. At the same time, our language skills are subtle blinders that may block our ability to develop new, more appropriate concepts. In other words, we need to account for the pivotal role of language in this kind of research, both as an enabler and inhibitor of abstraction: "Language is my instrument—but simultaneously my problem, too." (translated from [40, pp. 90f])—and aim at moving the limitations of our current languages.

A further peculiar challenge is related to the organization of research. The development of DSML and of corresponding reference models requires an effort that goes beyond the capabilities of single research institutions. Only if we succeed in bundling resources effectively, we may cope with this challenge. Furthermore, developing a DSML or a reference model is not enough, since their benefit depends chiefly on dissemination and acceptance. Hence, there is need for pursuing common languages and models. Otherwise it is likely that the history of programming languages and general purpose modeling languages will be repeated: Major players outside the academic world will define standards and research is widely restricted to using or criticizing them. However, while commitments to common standards are a prerequisite for effectively bundling resources, they restrict the independence of the participating researchers at the same time. Similar to that, emphasizing the need for agreements can be seen as a threat to competition, which is arguably an important driver of scientific research.

Finally, research on enterprise modeling encounters considerable methodological challenges. Any contribution to a scientific body of knowledge requires comprehensible justification. Hence, the claim that a new linguistic construction is superior with respect to a certain purpose needs to be supported by some kind of evidence. Truth as the classical justification criterion is not sufficient. At the same time, an objective assessment of a language is hardly possible because it will be influenced by the reference language we use—and that we cannot transcend. Empirical investigations that aim at measuring acceptance, effects on productivity, etc., may appear as an appropriate approach to justify—or refute—a DSML. However, they are restricted by the contingency of the subject: Prospective users' language skills and preferences are a moving target that is influenced by a subtle mix of previous experiences, cognitive skills, learning processes, etc., which are all matter of possible change.

5.2 Elements of a research agenda

The depth and diversity of the peculiar challenges make it arguably impossible to outline a convincing and coherent

Table 1 Overview of related work

	Archimate	ARIS	CIM-OSA	DEMO	MEMO	SOM	TOGAF	Zachman
Purpose	Modeling of IT infrastructure in conjunction with business process	Original focus on designing joint development of IS and organizational context for manufacturing enterprises; in the meantime support for further application scenarios	Design of information systems for manufacturing enterprises; special emphasis on reuse on various levels of abstraction	Analysis and design of organizational action systems together with supporting information systems	Analysis and design of information systems and corresponding action systems; support for software development and various other analysis/design scenarios	Analysis and design of information systems and corresponding action systems	Support for developing "enterprise architectures" with special emphasis on information systems architectures	Selection and introduction of major enterprise software systems in manufacturing enterprises
Context	Part of industrial consortium	Part of an academic project and part of a commercial product and service package	Industrial consortium formed to build a joined, EU-funded project	Academic project	Academic project	Academic project	Developed and managed by industrial consortium	Individual initiative within major software vendor
DSML	Metamodels and graphical notation, on a rather generic level	Published meta models remain on a generic level; focus mainly on EPC	No	Only basic language concepts to illustrate approach	Extensible set of integrated DSML	Generic DSML for process and resource modeling	No—but supplemented by Archimate	No—only references to potential candidates
Meta modeling language	No specification of MML	Rudimentary meta meta model	No	No, however, "meta ontology"	Yes	No, only rudimentary meta meta model	No	No
Methods	No	High-level process models; commercial version supplemented by extensive method handbook	High-level process model	Yes	Yes	Yes	Yes	High-level guidelines
Method engineering	No	No	No	No	Yes	No	No	No
Tools	Various commercial implementations	Comprehensive commercial modeling environment	Rudimentary prototypes, no special emphasis on architecture	Prototypes, no special emphasis on architecture	Tool environment with specific architecture	Various prototypes	No—only guidelines for selecting tools	No—except for simple representations of the framework

Table 1 continued

	Archimate	ARIS	CIM-OSA	DEMO	MEMO	SOM	TOGAF	Zachman
Meta model editor	No	No, but options to extend languages offered by modeling tool	No	No	Yes, with support for developing model editors	No	No	No
Specific account for actions	No	No	No	Yes	Yes—emphasis on relevance, specific account in language concepts and methods	No specific concepts—restricted to business transactions	No	No
Comment	Supplement to TOGAF			Refers to language/action-paradigm		Based on systems theory	Marketing jargon sometimes preferred over clear descriptions	Later version extended with logical modeling language—but was not further developed
Subject of ongoing research	Yes	Yes, centred on commercial development	No	Yes	Yes	Yes	Yes	No—maybe refinements carried out by respective consulting firm

research strategy. The elements of a research agenda presented below only serve to point at research foci that seem suited to promote the field.

Focus on use scenarios The development of a DSML requires a clear understanding of the purpose it should serve. According to our experience, it is a promising approach to start with developing use scenarios. A use scenario is characterized by a problem situation and corresponding technical languages. Based on a use scenario preliminary diagrams are introduced to develop an idea of the concepts needed to address and represent certain problem aspects. Use scenarios are inspired by existing problem situations. They may, however, also represent possible future worlds that are characterized by different institutional contexts, e.g., loosely coupled networks instead of particular organizations, and by the use of advanced concepts and tools. Hence, the creation of use scenarios implicates an act of abstraction and creativity. Elaborate use scenarios may also serve as a medium to foster collaboration with researchers from other fields and with domain experts in business practice. So far, research mainly focused on DSML to model business processes and information systems infrastructures. There is still demand to develop and refine DSML for strategic planning, for modeling inter-organizational systems, resources, products and markets.

Focus on language paradigms Like other approaches to enterprise modeling, the DSML provided with MEMO are specified with meta models the semantics of which is similar to object-oriented programming languages. Such a modeling language paradigm fosters the transformation of models to implementation-level documents. However, there are other paradigms that come with specific advantages, too. They include logic-based languages, which are used for specifying so-called enterprise ontologies and allow for machine reasoning on models. Languages used for creating simulation models would allow for supplementing enterprise models with simulation features. Petri Nets provide mature support for process analysis and automation. Also, modeling approaches used in Business and Administration or Operations Research should be considered as possible supplements to enterprise modeling. With respect to advanced modeling tools, the limitations of prevalent programming languages are a severe obstacle that might be overcome with more versatile meta-programming languages. In any case, the future development of the field recommends investigating the potential of combining different languages paradigms to advance the current state of dedicated modeling languages.

Focus on adaptation Reusing reference enterprise models will usually require adaptation, which demands for appropriate abstraction concepts provided by corresponding DSML. The latter aspect represents a specific challenge for adapting dynamic representations such as business process

models: They do not allow for a satisfactory specialization concept, because they do not permit monotonic extensions. As an alternative, one could use relaxed concepts of process specialization (e.g., [1]) or concepts to define process variants [27]. A further possibility would be to apply the idea of aspects to enterprise models to support adaptations, e.g., by assigning concerns such as "legal aspect x".

Focus on evolution For many firms, the development and introduction of a comprehensive enterprise model will be no option, because it implies too much effort and risk. To take advantage of enterprise modeling nevertheless, there is need for approaches that guide an evolutionary realization of enterprise models. Respective approaches need to account for relevant aspects of the enterprise, such as size, dynamics, available budget, etc., and for priorities with respect to diagram types and the level of detail required for certain models. In addition to that support is required for merging partial models and for model evolution, i.e., versioning of models and corresponding modeling languages.

Focus on collaboration To cope with the extraordinary need for resources, new forms of organizing research on enterprise modeling need to be investigated. On the one hand, they should enable a larger group of researchers from various fields as well as practitioners to bundle their resources. On the other hand, they should account for the conflict between the need for common standards and the quest for scientific progress. For a similar subject, some "Open Software Systems" initiatives have produced impressive results. Therefore, they could serve as an orientation for building respective "Open Model" initiatives [22]. To allow for both, common standards and scientific competition, they could combine a common stream of research that is based on a set of common languages and tools with additional more specific projects that aim at developing and testing new language features and corresponding tools.

Focus on research methods Research on enterprise modeling cannot ignore specific methodological problems. They mainly relate to the justification of research results and the comparative assessment of artifacts. Usually it will not be sufficient to focus on one particular justification criterion only—such as a certain concept of truth and corresponding justification procedures. Therefore, respective research will probably require the purposeful, eclectic use of various justification criteria and procedures (for a corresponding proposal to guide the configuration of research methods see [14]).

References

1. van der Aalst, W., Basten, T.: Inheritance of workflows: An approach to tackling problems related to change. Theor. Comput. Sci. **270**(1–2), 125–203 (2002)
2. Amice, E.C.: Open System Architecture for CIM. Springer, Berlin (1989)
3. Argyris, C.: Overcoming Organizational Defenses. Prentice Hall, Boston (1990)
4. Atkinson, C., Küehne, T.: Reducing accidental complexity in domain models. Softw. Syst. Model. **7**(3), 345–359 (2008)
5. Bunge, M.: Treatise on basic philosophy. In: Ontology II: A World of Systems. D. Reidel, Dordrecht (1979)
6. Checkland, P., Scholes, J.: Soft Systems Methodology in Action. Wiley, New York (1999)
7. Dietz, J.L.: Transaction-based enterprise modeling. In: Gan, R. (ed.) Information Technology for Business Management—16th World Computer Congress 2000, pp. 105–114. Publishing House of Electronics Industry, Beijng (2000)
8. Dietz, J.L.: A world ontology specification language. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM Workshops, pp. 688–699. Springer, Berlin (2005)
9. Ferstl, O.K., Sinz, E.J.: SOM Modeling of Business Systems, pp. 339–358. Springer, Berlin (1998)
10. Ferstl, O.K., Sinz, E.J.: Modeling of Business Systems Using SOM, 2nd edn. Springer, Berlin (2005)
11. Frank, U.: Multiperspektivische Unternehmensmodellierung: Theoretischer Hintergrund und Entwurf einer objektorientierten Entwicklungsumgebung. Oldenbourg, Muenchen (1994)
12. Frank, U.: The memo object modelling language (MEMO-OML). Technical Report 10, University Koblenz-Landau, Koblenz (1998)
13. Frank, U.: Multi-perspective enterprise modeling (MEMO)—conceptual framework and modeling languages. In: Proceedings of the 35th Annual Hawaii International Conference on Systems Sciences (HICSS), pp. 72–81. Computer Society Press, Los Alamitos (2002)
14. Frank, U.: Towards a pluralistic conception of research methods in information systems research. Technical Report 7, ICB, University Duisburg-Essen, Essen (2006)
15. Frank, U.: Outline of a method for designing domain-specific modelling languages. Technical Report 42, ICB, University Duisburg-Essen, Essen (2010)
16. Frank, U.: The MEMO meta modelling language (MML) and language architecture, 2nd edn. Technical Report 43, ICB, University Duisburg-Essen, Essen (2011)
17. Frank, U.: MEMO organisation modelling language: focus on business processes. Technical Report 49, ICB, University Duisburg-Essen, Essen (2011)
18. Frank, U.: MEMO organisation modelling language: focus on organisational structure. Technical Report 48, ICB, University Duisburg-Essen, Essen (2011)
19. Frank, U.: Some guidelines for the conception of domain-specific modelling languages. In: Nuettgens, M., Thomas, O., Weber, B. (eds.) Proceedings of the Conference Enterprise Modelling and Information Systems Architectures (EMISA 2011), pp. 93–106. GI, Bonn (2011)
20. Frank, U., Heise, D., Kattenstroth, H., Ferguson, D., Hadar, E., Waschke, M.: A domain-specific modeling language for supporting business driven IT management. In: Tolvanen, J.P., Rossi, M., Gray, J., Sprinkle, J. (eds.) Proceedings of the 9th OOPSLA Workshop on Domain-Specific Modeling (DSM). Helsinki Business School, Helsinki (2009)
21. Frank, U., Lange, C.: E-MEMO: a method to support the development of customized electronic commerce systems. Inf. Syst. E Bus. Manag. **5**(2), 93–116 (2007)
22. Frank, U., Strecker, S.: Open reference models—community-driven collaboration to promote development and dissemination of reference models. Enterp. Model. Inf. Syst. Archit. **2**(2), 32–41 (2007)

23. Graumann, C.F.: Perspektivitaet in kognition und sprache. SPIEL **12**(2), 156–172 (1993)
24. Gronback, R.C.: Eclipse Modeling Project: A Domain-Specific Language (DSL) Toolkit. Addison-Wesley Longman, Amsterdam (2009)
25. Group, T.O.: The open group architecture framework (TOGAF)—version 9. Technical report (2009)
26. Habermas, J.: Theorie des kommunikativen Handelns. In: Handlungsrationalitaet und gesellschaftliche Rationalisierung, vol. 2. Suhrkamp, Frankfurt/M. (1981)
27. Hallerbach, A., Bauer, T., Reichert, M.: Configuration and Management of Process Variants, pp. 237–255. Springer, Berlin (2010)
28. Hammer, M., Champy, J.A.: Reengineering the Corporation: A Manifesto for Business Revolution. Harper Business Books, New York (1993)
29. Jung, J.: Mapping of business process models to workflow schemata—an example using MEMO-OrgML and XPDL. Technical Report 47, University Koblenz-Landau, Koblenz (2004)
30. Jung, J.: Entwurf einer Sprache fuer die Modellierung von Ressourcen im Kontext der Geschaeftsprozessmodellierung. Logos, Berlin (2007)
31. Karagiannis, D., Visic, N.: Next Generation of Modelling Platforms. Springer, Berlin (2011)
32. Keen, P.G.: Shaping the Future: Business Design Through Information Technology. Harvard Business School Press, Boston (1991)
33. Kieser, A.: Organisationstheorien. Kohlhammer, Stuttgart (2006)
34. Kirchner, L.: Eine Methode zur Unterstuetzung des IT-Managements im Rahmen der Unternehmensmodellierung. Logos, Berlin (2008)
35. Lankhorst, M.: Enterprise Architecture at Work. Modelling, Communiation and Analysis. Springer, Berlin (2005)
36. Lawrence, P.R., Lorsch, J.W.: Organization and Environment: Managing Differentiation and Integration. Harvard University, Boston (1967)
37. Luhmann, N.: Zweckbegriff und Systemrationalitaet. Suhrkamp, Frankfurt/M. (1977)
38. Luhmann, N.: Soziale Systeme. Grundriss einer allgemeinen Theorie. Suhrkamp, Frankfurt/M. (1984)
39. Marshall, C.: Enterprise Modeling with UML: Designing Successful Software through Business Analysis. Addison-Wesley, Reading (2000)
40. Maturana, H.: Kognition. In: Schmidt, S.J. (ed.) Der Diskurs des Radikalen Konstruktivismus, p. 118. Suhrkamp, Frankfurt/M. (1987)
41. Mintzberg, H.: The Structuring of Organizations. Prentice-Hall, Englewood Cliffs (1979)
42. Morgan, G.: Images of Organization. Sage, Thousands Oaks (1986)
43. Oesterle, H., Winter, R.: Business Engineering-Auf dem Weg zum Unternehmen des Informationszeitalters, 2nd edn. Springer, Berlin (2003)
44. OMG: Meta object facility (MOF) core specification, version 2.0. Technical report (2006)
45. OMG: Unified modeling language. Superstructure, version 2.1.2. Technical report (2007)
46. O'Toole, J.: Leading Change. Jossey-Bass, San Francisco (1995)
47. Pfeffer, J.: Management as symbolic action. In: Cummings, L.L., Staw, B. (eds.) The Creation and Maintenance of Organizational Paradigms. Vol. 3, pp. 1–52, JAI Press, Greenwich, Conn. (1981)
48. Pugh, D., Hickson, D.: Organizational Structure in Its Context: The Aston Programme I. Gower, Westmead-Lexington (1976)
49. Schauer, H.: Unternehmensmodellierung fuer das Wissensmanagement. Eine multi-perspektivische Methode zur ganzheitlichen Analyse und Planung. VDM, Saarbruecken (2009)
50. Scheer, A.W.: Architecture of Integrated Information Systems: Foundations of Enterprise Modelling. Springer, Berlin (1992)
51. Scheer, A.W.: ARIS-Business Process Frameworks, 3rd edn. Springer, Berlin (1999)
52. Scheer, A.W.: ARIS-Business Process Modeling, 3rd edn. Springer, Berlin (2000)
53. Schein, E.: Organizational Culture and Leadership. Jossey-Bass, San Francisco (2004)
54. Schuetz, A.: Der sinnhafte Aufbau der sozialen Welt, vol. 2, 2nd edn. Suhrkamp, Frankfurt/M. (1981)
55. Searle, J.: Speech Acts. Cambridge University Press, Boston (1969)
56. Sowa, J.F., Zachman, J.A.: Extending and formalizing the framework for information systems architecture. IBM Syst. J. **31**(3), 590–616 (1992)
57. Sperling, S.: Konzeption einer Methode zum Integrationsmanagement bei Unternehmenszusammenschluessen auf der Basis von multiperspektivischen Unternehmensmodellen. Logos, Berlin (2007)
58. Stamper, R.: Social Norms in Requirements Analysis an outline of MEASUR. Academic Press, New York (1993)
59. Steinberg, D., Budinsky, F., Paternostro, M., Merks, E.: EMF: Eclipse Modeling Framework, 2nd edn. Addison Wesley, Reading (2009)
60. Strecker, S., Frank, U., Heise, D., Kattenstroth, H.: MetricM: a modeling method in support of the reflective design and use of performance measurement systems. Inf. Syst. E Bus. Manag. **10**(2), 241–276 (2011)
61. Strecker, S., Heise, D., Frank, U.: Prolegomena of a modelling method in support of audit risk assessment: outline of a domain-specific modelling language for internal controls and internal control systems. Enterp. Model. Inf. Syst. Archit. **6**(3), 5–24 (2011)
62. Urbaczewski, L., Mrdalj, S.: A comparison of enterprise architecture frameworks. Inf. Syst. J. **VII**(2), 18–23 (2006)
63. Weick, K.E.: The Social Psychology of Organizing, 2nd edn. McGraw-Hill, New York (1979)
64. Wittgenstein, L.: Philosophische Untersuchungen, 2nd edn. Suhrkamp, Frankfurt/M. (1980)
65. Wolff, F., Frank, U.: A multiperspective framework for evaluating conceptual models in organisational change. In: Bartmann, D., Rrajola, F., Kallinikos, J., Avison, D. (eds.) ECIS 2005–13th European Conference on Information Systems, pp. 1–12. Association for Information Systems (2005)
66. Wright, G.H.v.: Explanation and Understanding. Cornell University, New York (1971)
67. Zachman, J.A.: A framework for information systems architecture. IBM Syst. J. **26**(3), 277–293 (1987)

Author Biography

Ulrich Frank is Professor of Information Systems and Enterprise Modeling at the Institute for Computer Science and Business Information Systems, University of Duisburg-Essen in Germany where he heads the Enterprise Modeling research group. His work focuses on the design and evaluation of languages and methods for multi-perspective enterprise modeling. In addition, his research topics include software engineering, IS management, and Philosophy of Science. He can be reached at ulrich.frank@uni-due.de.