Ulrich Frank

Enterprise Modelling: The Next Steps

Enterprise modelling is at the core of Information Systems and has been a subject of intensive research for about two decades. While the current state of the art shows signs of modest maturity, research is still facing substantial challenges. On the one hand, they relate to shortcomings of our current knowledge. On the other hand, they are related to opportunities of enterprise modelling that have not been sufficiently addressed so far. This paper presents a personal view of future research on enterprise modelling. It includes requests for solving underestimated problems and proposes additional topics that promise to promote enterprise models as more versatile tools for collaborative problem solving. In addition to that, the paper presents requests for (re-) organising research on enterprise modelling in order to increase the impact of the field.

1 Introduction

It has been a wide-spread conviction for long that the complexity of large information systems recommends the use of models. Information systems are aimed at representing domains through data that is accessible by prospective users. Representing a–factual or aspired–domain cannot be accomplished by modelling it directly. Instead, it comprises a twofold abstraction: We perceive a domain primarily through language, which in turn reflects an abstraction over "objective" features of a domain. At the same time, using an information system requires an interface that corresponds to the language spoken in the targeted domain. Therefore, the construction of information systems recommends the design of *conceptual models*. They do not only promise to reduce complexity by abstracting from ever changing peculiarities of technical infrastructures; they also allow for getting prospective users involved in the analysis and design process. Exploiting the potential of information systems will often require reorganising existing patterns of action—sometimes in a radical way. Therefore, analysis and design of information systems should usually be done conjointly with analysing and designing the organisational action system. The conception of an enterprise model was developed to address this demand. An enterprise model integrates at least one conceptual model of an organisational action system with at least one conceptual model of a corresponding information system. Usually, but not necessarily, the various models that constitute an enterprise model are created with domain-specific modelling languages (DSML). To emphasise that enterprise models are intended to provide a medium both for fostering analysis and design tasks and for communication across traditional professional barriers, the term "multi-perspective enterprise model" has been introduced (Frank 1994). A multi-perspective enterprise model is an enterprise model that emphasises accounting for multiple perspectives. A perspective represents a specific professional background that corresponds to cognitive dispositions, technical languages, specific goals and capabilities of prospective users (Frank 2013b).

In recent years, the term "enterprise architecture" has gained remarkable attention (Buckl et al. 2010; Land et al. 2009; Lankhorst 2005). The differences between enterprise model and enterprise architecture are mainly related to the intended audience. Enterprise modelling is aimed at various groups of stakeholders that are involved in planning, implementing, using and maintaining information systems. Therefore, enterprise models are supposed to offer a variety of corresponding abstractions. These include models

that serve as a foundation of software development. Therefore, the development of respective DSML is a particular characteristic of enterprise modelling. Different from that, enterprise architecture mainly targets IT management. Therefore, it puts less emphasis on the specification of DSML. Nevertheless, there is no fundamental difference between both approaches. Instead, the abstractions used in enterprise architectures can be seen as an integral part of more comprehensive enterprise models. In Information Systems, enterprise modelling has been a pivotal field of research that has evolved over a period of more than 20 years (CIMOSA: Open system architecture for CIM 1993; Ferstl and Sinz 1998; Group 2009; Scheer 1992; Zachman 1987). It has produced various modelling frameworks, DSML as well as corresponding tools. The field has achieved a remarkable degree of maturity which is indicated by the fact that enterprise modelling is part of many IS curricula—even though to different extent. Nevertheless, there is still need for further research to exploit the potential of enterprise modelling. In this paper I will point at relevant shortcomings of the current state of the art in order identify core elements of a future research agenda.

2 The Need for More Context

At the beginning, approaches to enterprise modelling were mainly focussed on developing high-level frameworks to provide a common structure or architecture of an enterprise and its information system (CIMOSA: Open system architecture for CIM 1993; Scheer 1992; Zachman 1987). Apart from using general purpose modelling languages (GPML) like the ERM and the UML, the development of DSML was mainly aimed at business process modelling. Later, DSML were created for modelling strategies, organisational structures or generic resources. In recent years, some approaches have evolved that are aimed at DSML for modelling IT infrastructures and IT architectures. This focus is remarkable for two reasons. At first, it represents a renunciation of the original approach to enterprise modelling, which was aimed at developing information systems from scratch. With respect to the complexity of today's IT infrastructures and the fact that most organisations will not develop substantial parts of their information system on their own anymore, this additional focus is certainly reasonable. Secondly, it is not only aimed at supporting the design of IT infrastructures that are in line with the corporate action system, but also at providing an instrument for IT management. Figure 1 illustrates the representation of an enterprise model through a set of interrelated diagrams that correspond to the current state of the art. To ensure integration, the partial models that are represented by the diagrams should be created with modelling languages that were specified with the same meta modelling language and that share common concepts (Frank 2011).

While the abstractions covered by today's enterprise modelling methods arguably represent relevant perspectives on an enterprise, a comprehensive representation of all aspects that may be relevant for analysing, designing and managing a company together with its information system requires accounting for more context. While such a demand may look like an exaggeration to some, it is actually the consequent continuation of current practice: All professional activities in a company are characterised by the use of conceptual abstractions, i.e., by a specific technical terminology and corresponding *language games*. Reconstructing these terminologies through additional DSML would not only enable further use scenarios, it would also enrich existing models with additional context. Context does not only refer to the topics that are represented in an enterprise model. It also refers to the context in which the development and use of models occur. On the one hand, this kind of context includes specific methods for enterprise modelling. On the other hand, it refers to organisational and managerial arrangements to foster an adequate handling of enterprise models.

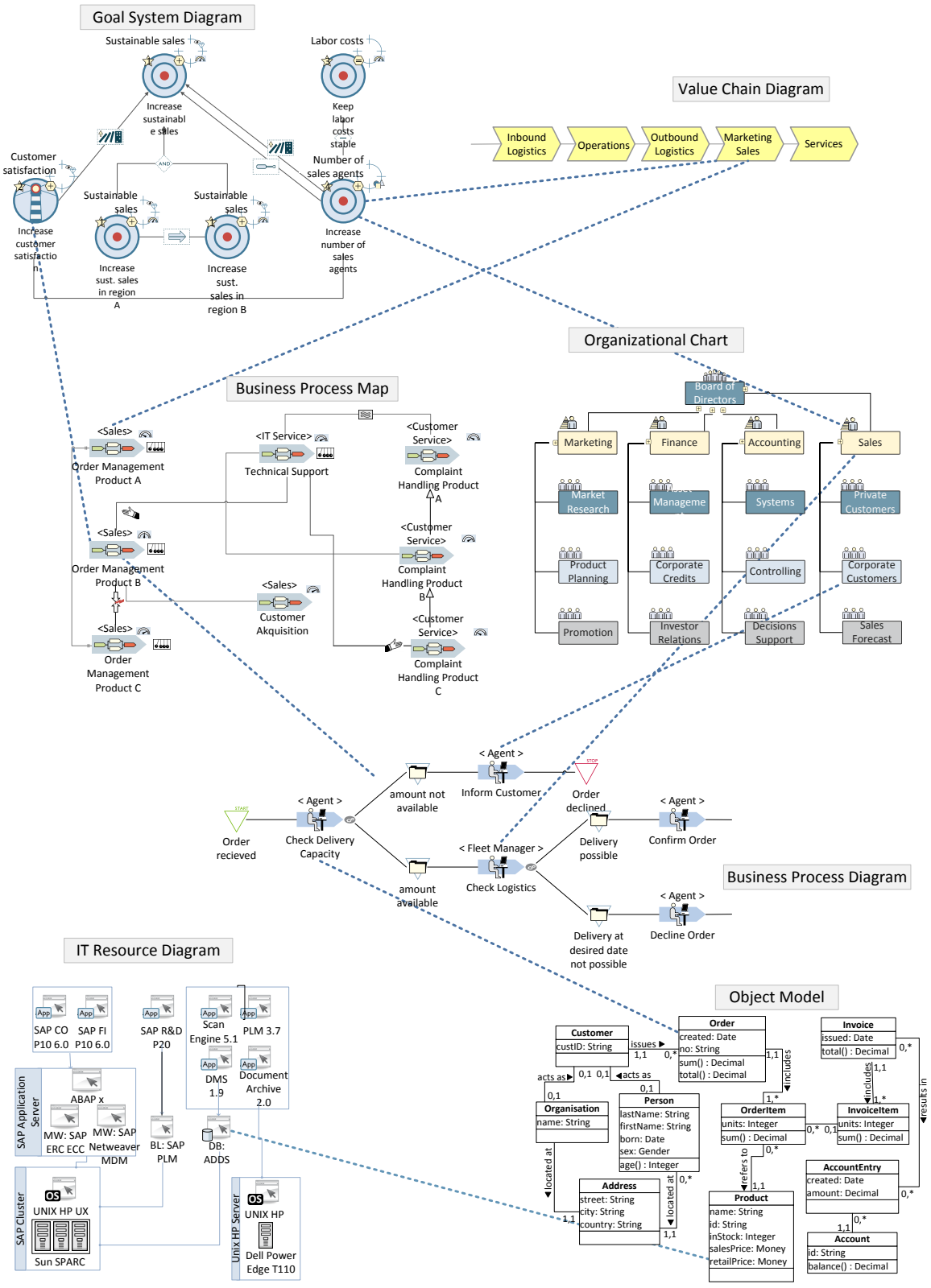

Figure 1: Diagrams Representing Example Enterprise Model

2.1 Further Modelling Topics

The variety of topics that are handled in enterprises is enormous. Among those that have not been sufficiently addressed in enterprise modelling are products, production processes, projects, markets and logistic. While product modelling is an issue on its own, integrating product models with enterprise models makes sense for various reasons. Products can be very complex and may demand for quick adaptations. At the same time, developing, producing and handling products relates to various aspects of an enterprise that are usually part of an enterprise model: goals, business processes, organisational units or software systems. More and more, products comprise software or are constituted by software. In addition to that products are often bundled—with services and/or other products. Therefore, integrating product models with enterprise models would enable additional analysis scenarios such as checking the effect of changing a product on required skills and on business processes. There are numerous approaches to model production processes. They aim at developing algorithms and approximation procedures for production planning, process scheduling and process control. Integrating respective models with enterprise models will often be not trivial, because they are based on different modelling paradigms. At the same time, including elaborate models of production processes in enterprise models promises various advantages, such as supporting the conjoint analysis of production processes and related business processes or generating software for controlling production processes from respective models. In an increasing number of organisations, projects play a key role. Integrating project models with enterprise models would support project management by providing meaningful links to organisational resources. Also, project modelling could benefit from existing approaches to business process modelling and would allow to take advantage of similarities between projects. Markets have not been part of enterprise models for an apparent reason: They are outside of an enterprise and are usually not subject of design processes. Nevertheless, markets are of crucial importance for successful action in an enterprise. Furthermore, markets are getting more complex and contingent: Often, they expand on an international scale and may be very dynamic in the sense that products are displaced by innovations or that customer preferences change quickly. Therefore, integrating models of markets with enterprise models promises to gain a more differentiated understanding of relevant market forces and to develop a better foundation for decision making. Similar to production processes, logistic networks have been subject of optimisation efforts for long. The respective models, often designed to satisfy the requirements of Operations Research methods, are mainly focussed on optimisation with respect to certain goals. Integrating the respective modelling concepts with languages for enterprise modelling would enable to enrich both enterprise models and logistic models with more relevant context.

In addition to traditional topics, organisations are confronted with new phenomena that may demand for appropriate action. They include social networks, virtual enterprises, nomad employees and many more. Extending enterprise models with models of these phenomena would foster analysing and handling them. This would, however, require new modelling concepts. Finally, enterprise models can be supplemented with concepts that are related to important further aspects of managerial decision making. These include accounting concepts, e.g., specialised cost and benefit concepts, concepts to design and analyse indicator systems (Strecker et al. 2012) as well as concepts for modelling organisational knowledge. Adding these concepts would make enterprise models and corresponding tools a versatile instrument for management—both on the operational and strategic level. At the same time, it could serve as a valuable extension of enterprise software systems (see Sect. 4.1).

Request: To make enterprise models a versatile tool for supporting professional action in organ-

isations, research needs to widen the scope of modelling by adding further topics that also comprise concepts to support managerial decision making.

2.2 Method Construction

Modelling languages are an important foundation of enterprise modelling, since they provide a purposeful structuring of a domain. However, they are not sufficient for designing and using enterprise models. In addition to languages, there is need for processes that guide the purposeful development, interpretation and use of respective models. In other words: There is need for *modelling methods*. Due to the diversity of projects that can benefit from conceptual models, it is evident that a given set of modelling methods cannot not fit all demands—except for the price of oversimplification. This insight shifted the focus on approaches that guide the development of customised methods. The only chance to provide support for the conceptualisation of a range of methods is to increase the level of abstraction by searching for essential characteristics shared by all modelling methods. Against this background, the emergence of method engineering as a new field of research is a reasonable consequence in two respects: First, it is aimed at rich abstractions that cover a wide range of modelling projects. Second, it makes use of the same paradigm that it suggests for the field of conceptual modelling, too: The construction of particular methods should follow an engineering approach, which—among other things—recommends accounting for linguistic rigour, consistency and coherence as well as for the development of supportive tools. During the last 15 years, a plethora of method engineering approaches—originating mostly in Requirements Engineering and Software Engineering—has evolved (Brinkkemper 1996; Ralyté et al. 2005, 2007). For an overview see Henderson-Sellers and Ralyté (2010). Some emphasise the construction of methods from reusable elements, others focus on the instantiation of methods from metamodels, while further approaches are based on a combination of composition and instantiation. It seems that the field has reached a stage of moderate maturity, which is also indicated by the specification of a respective ISO standard (ISO/IEC 2007).

Nevertheless, there are some aspects that have been widely neglected so far. At first, current approaches to method engineering are mostly generic in the sense that they are not restricted to particular domains, nor do they account for the peculiarities of enterprise models. That leaves prospective developers and users of enterprise models with the demanding task of adapting generic concepts to the idiosyncrasies of particular organisations. Second, current approaches to method engineering focus on the design of process models and take the modelling language as given. However, the diversity of topics that can be reasonable subjects of enterprise models may also require to adapt or even create modelling languages. While a number of tools support the specification of DSML and the realisation of corresponding model editors, prospective users can expect only little guidance with designing a language that fits its purpose. At the same time the design of DSML is especially demanding. Often, prospective users will not have an idea of what they might expect from a DSML. As a consequence, requirements analysis is a remarkable challenge. In addition to that, design conflicts need to be handled. Also, the creation of the concrete syntax requires a specific competence that many language designers do not have. Therefore, there is need for substantiated guidance to reduce the risk of poorly designed modelling languages. Currently, there are only few approaches that offer respective support (Frank 2013a; Moody 2009). Finally, method engineering is often based on two assumptions: First, a method is an artefact that is created through an engineering act. Second, applying the method appropriately is pivotal for successful action. However, with respect to successfully using a method, it is not

sufficient to restrict it to its explicit definition, i.e., to take a mere technical perspective. This is for two interrelated reasons. First, a method will usually not be based on a pure formal specification. Instead, its conceptual and theoretical foundation as well as the process description require interpretations that produce some degree of shared understanding. Second, for a method to work, it has to become an accepted orientation for individual and collaborative action. To summarise both aspects: A method needs to make sense. From this point of view, a method can be regarded as a social construction that reflects established patterns of professional action, ideas of professional values and aesthetics, organisational culture, common beliefs as well as individual interests. Against this background, we can distinguish between a method as a linguistic artefact, stressing a technical view, and a method as an actual practice, stressing a more pragmatic or organisational view. Therefore I intentionally avoid the term "method engineering" and speak of "method construction" instead. This is to express that a method is also constructed by those who use it, because it is shaped by actual interpretations and actions. A method as an artefact could be regarded as input or stimulus to trigger such a process. While for analytical reasons it may be useful to focus on methods mainly as linguistic artefacts, such a restricted view is certainly not sufficient with respect to a pragmatic objective such as improving efficiency and quality of collaborative problem solving in organisations. The benefit of methods for enterprise modelling will not only depend on the qualification of the involved stakeholders, but also on certain aspects of the respective corporate culture. It makes a clear difference, whether conceptual models are regarded as corporate assets or as cost drivers with dubious outcome.

Request: To promote the beneficial development and use of enterprise models it is required to support the construction of respective modelling methods that account for both, the conceptual foundation of designing/customising methods as linguistic artefacts and additional organisational/managerial measures that promote the appropriate use of methods in practice.

3 The Need for More (Re) Use

The remarkable effort that is required to build elaborate enterprise models makes reuse of models and modelling concepts a pivotal issue for achieving higher productivity. At the same time, reuse can also contribute to model quality, if reusable artefacts are designed and evaluated with specific care. In addition to that reusable concepts can serve to foster integration of those components that share them. Approaches to promote reuse have been on the research agenda for long. The idea of reference enterprise models seems to be especially attractive. However, so far, reuse of enterprise modelling artefacts remained on a modest level (Fettke and Loos 2007). There are various reasons that contributed to this unsatisfactory situation. Two especially important reasons are related to modelling languages. On the one hand, current languages for enterprise modelling lack concepts that enable reuse. On the other hand, the design of reusable DSML is facing a substantial challenge.

3.1 The Lack of Abstraction in Process Modelling

Taken the fact that business process modelling has been a research subject for long, it seems surprising that respective modelling languages are rather primitive in the sense that they do not allow for powerful abstractions. As a consequence, reuse of business process models remains on a dramatically poor level. Since business process models play a pivotal role within enterprise models, this is a serious shortcoming. The following scenario illustrates the problem. A company comprises a few tens of business process types including a core order management process type. A process type includes activity types. Various process types share similar activity types. Now two more specific order management process types need to be implemented. For this purpose,

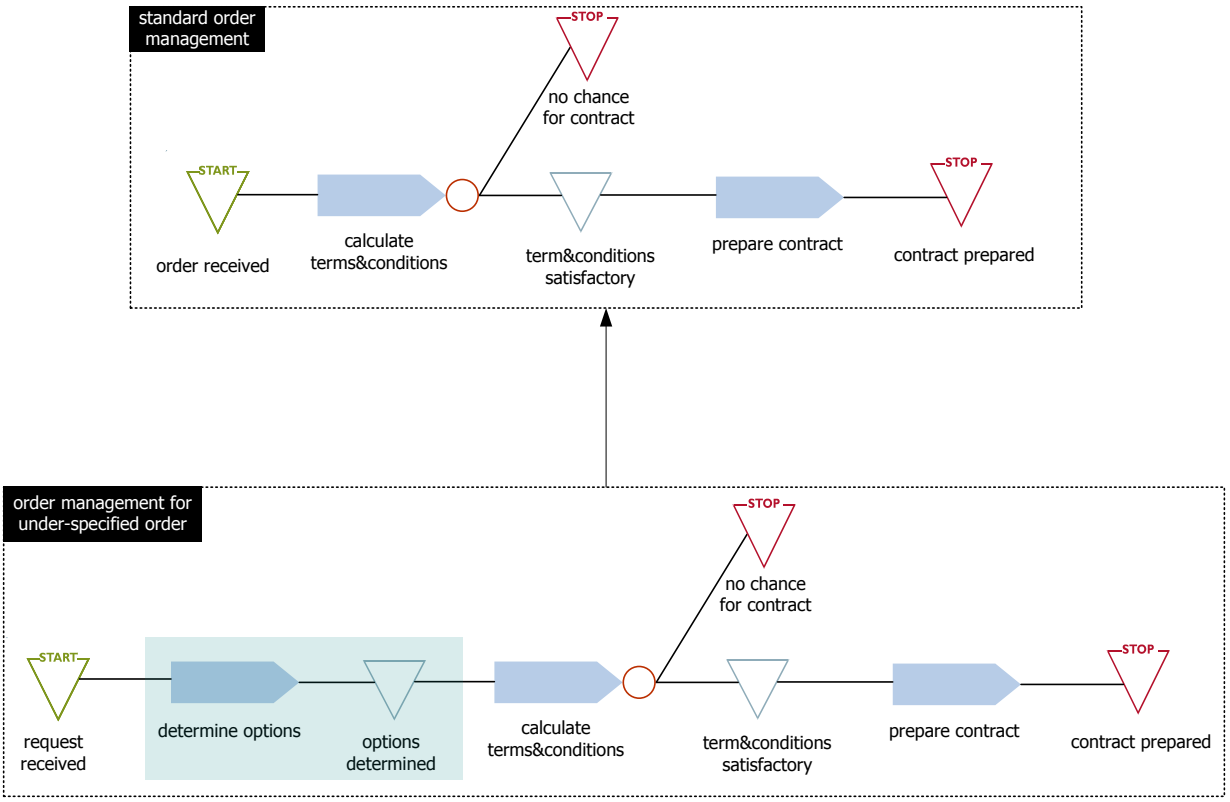

Figure 2: Example of extending a business process type

it would be most helpful to specialise the existing order management process type.

This would not only allow reusing the respective model and corresponding software implementations, it would also promote safe and convenient maintenance: Future changes of the core process type would be immediately effective in the specialised types, too. To satisfy the demand for integrity, a respective concept of process specialisation would have to satisfy the substitutability constraint: Every instance of a process type can act as an instance of the corresponding super process type without causing harm (Liskov and Wing 1994). The substitutability constraint is satisfied, if the extensions defined for specialised concepts are monotonic. This can be accomplished fairly easy for static abstractions. However, for dynamic abstractions such as business process models adding further activity or event types cannot be monotonic, because it will always effect the original control flow (Frank 2012). The example in Fig. 2 illustrates this problem.

There are a few approaches in Software Engineering and process modelling which are aimed at a relaxed conception of specialisation of behaviour (Schrefl and Stumptner 2002) or of "workflow inheritance" (Aalst and Basten 2002). Other approaches focus on analysing structural similarities of control flows to promote reuse through process variants (Koschmider and Oberweis 2007). However, the restrictions these approaches imply remain unsatisfactory (Frank 2012). At the same time, the still growing relevance of efficiently creating and maintaining business process models demands for abstractions that allow taking advantage of similarities.

Request: Future research should aim at concepts of relaxed process specialisation—which may be combined with instantiation—that promote reuse without unacceptable restrictions.

While the lack of a sound concept of process specialisation creates a serious problem, the current state of business process modelling is even more dreary. The above scenario would suggest to reuse an activity type that was defined already for a certain business process type in a new process type. However, this is not possible: Every business process type has to be designed from scratch using the basic concepts provided by today's process modelling languages. Hence, an activity such as "prepare contract" cannot be specified as a reusable type. Instead it is yet another instance of a basic (meta) type like "activity" or "automated activity" that is distinguished from other primarily through its label. There are approaches that focus on analysing labels in order to detect similar activity types (Dijkman et al. 2011). However, their contribution to reuse is limited: Instead of removing the mess, they try coping with it.

Request: There is need for extending business process modelling languages and tools with the possibility to define and reuse activity and event types.

This request is not easy to satisfy. An activity type is not only defined by its internal structure and behaviour, but also by its context such as the event that triggers it or the events it produces. Reuse will be possible only, if the context can be adapted to some extent. Therefore, the required concept of an activity type—and of an event type respectively—must abstract from the context without compromising reusability too much.

3.2 The Essential Conflict of Designing DSML

DSML are characterised by convincing advantages (Kelly and Tolvanen 2008; Kleppe 2009; Völter 2013). By providing domain-specific concepts, they promote modelling (and programming) productivity: Modellers are not forced anymore to reconstruct domain-level concepts from generic concepts such as "entity" or "attribute". At the same time, DSML foster model integrity, because they prevent the creation of inconsistent models to a certain extent. By featuring a domain-specific concrete syntax, they also promote model comprehensibility. Against this background, it does not come as a surprise that DSML are regarded by many as the silver bullet of conceptual modelling and model-driven software development. However, their construction is facing a dilemma. The more a DSML is tuned to a specific domain, the better is its contribution to productivity and integrity. However, the more specific a DSML is, the more unlikely it can be used in a wide range of particular domains. Figure 3 illustrates the conflicting effects of semantics on range of reuse and productivity.

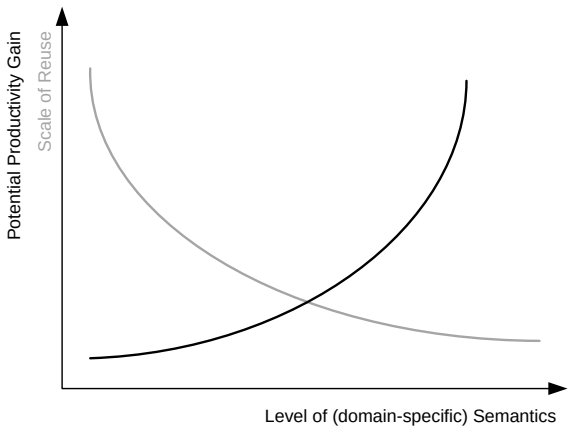

Figure 3: DSML: Illustration of Essential Design Conflict

Some authors suggest to design DSML to the needs of particular organisations or even projects only (Kelly and Tolvanen 2008; Völter 2013). This recommendation is based on two assumptions. First, the variety of organisations would not allow for powerful DSML that fit all individual requirements. Second, there is no need for further reuse, because creating and using a DSML in one particular project will usually pay off already. Even though both assumptions may be valid to a certain extent, they are hardly convincing. There may be remarkable differences between organisational actions systems and corresponding terminologies. However, it would be a sign of epistemological defeatism to deny the

chance of finding substantial commonalities. Furthermore, it can be assumed that actual variety is also a result of in part arbitrary processes of organisational evolution, i.e., it is not a reflection of inevitable differences. Also, there is evidence that technical languages work in a wide range of organisations of a certain kind: The terminology used in textbooks will often fit an entire industry in the sense that it provides a respected linguistic structure and serves as as common reference for professionals. Nevertheless, there are organisation-specific adaptations of textbook terminology. They include extensions, refinements and modifications, some of which may be questionable. The argument that a DSML will already pay off in single use scenarios is fine, but it could still be much more profitable, since a wider range of reuse would allow for much better economies of scale. Therefore, it would be beneficial to create hierarchies of DSML, where more specific ones are extensions and/or instantiations of more general DSML.

Request: Research on DSML should aim at hierarchies of languages to enable both a wide range of reuse and customised languages for narrow domains.

Figure 4 illustrates the idea of providing modelling languages on different classification layers. The highest level ("universal DSML") corresponds to textbook terminology. The concepts on this level should be applicable to a wide range of organisations, hence, promote economies of scale. The universal DSML should be designed by experts that possess deep knowledge about the general domain as well as rich experience with designing DSML. "Local" DSML represent more specific technical languages for organisation modelling that apply to a few organisations or to one only. They are designed by organisation analysts that are familiar with the respective domain. These local DSML that feature a graphical notation much like the universal DSML can be used by authorised managers to specify particular organisational settings. The example also shows that there are cases where it makes sense to create models that include concepts on the M0 level.

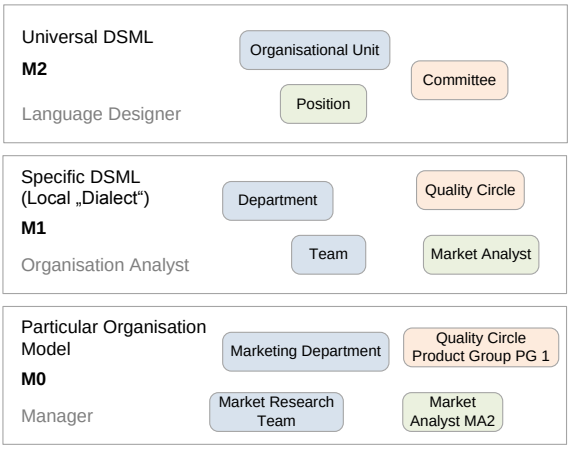

Figure 4: Illustration of multi-level modelling languages

Designing such language systems and corresponding tools is far from trivial. It requires giving up prevalent architectures of modelling languages that feature a given set of classification layers (for respective approaches see Atkinson et al. (2009); Clark et al. (2008); Simonyi et al. (2006). Instead, recursive language models such as the "golden braid" architecture are more promising—and more demanding at the same time, because they are not supported by most of today's development environments. Apart from that, designing languages for enterprise modelling should account for a further issue. Current DSML are usually specified with metamodels. This is for a good reason: On the one hand, this kind of specification corresponds to a paradigm the modelling community is familiar with. On the other hand, it fosters the construction of corresponding tools, because a metamodel can be used as a conceptual foundation of a respective modelling tool. The semantics of DSML, e.g., the semantics of specialisation concepts, is usually based on the semantics of prevalent programming languages to facilitate the transformation of models into code. However, there are other language paradigms and specification styles that might enrich enterprise models. For instance, models designed with logic-based languages allow for deduction could

enable more sophisticated approaches to analysing enterprise models. Since the semantics of respective languages, which are typically found in Artificial Intelligence, is different from that of DSML used for enterprise modelling today, integrating them into enterprise modelling environments is a demanding task.

4 The Need for Run-Time Use

Originally, enterprise models like most other conceptual models were intended for supporting the creation of information systems only. However, it is obvious that they should be beneficial during the entire life cycle of an information system. On a more generic level, this issue is addressed by research on "models at runtime" (Blair et al. 2009). Multi-perspective enterprise models provide abstractions of the enterprise that support decision making and other managerial tasks. Also, they can help people in organisations to develop a deeper understanding of the action system, i.e., how their work is integrated into a bigger picture. In addition to that, enterprise models can enable users to develop a better understanding of the information system and its interplay with organisational patterns of action.

4.1 Integrating Enterprise Models with Enterprise Systems

In an ideal case, an enterprise modelling environment would be integrated and synchronised with a corresponding enterprise (software) system. On the one hand, this would enrich enterprise systems not only with their conceptual foundation, but also with a representation of the context they are supposed to operate in. On the other hand, enterprise models would be supplemented with corresponding instance populations.

This would enable users to navigate from concepts on various classification levels to instances—et vice versa. The following scenario illustrates the benefit of drilling down from an enterprise model to instances. A department manager who is new to a firm wants to get a better understanding of the way business is done. For this purpose, he could browse a graphical representation of the corporate business process map, which shows all business process types, their interrelationships and key performance indicators at a glance. He could then select a business process type he is interested in, study the model that describes its execution and demand for further aggregate data that characterises it, such as the number of instances per month, average revenues etc. Also, he could select specific analysis views, e.g., a view that associates a selected business process type with the IT resources it requires. If he was interested in one particular business process type, he could view the corresponding model. Then, he could leave the conceptual level and ask for the list of currently active business processes of this type and inspect the state of the instances he is interested in. In addition to that, advanced users could modify the enterprise system by changing the enterprise model. The DSML, an enterprise model is created with would help preventing arbitrary modifications and hence contribute to system integrity. An outline of a respective system, referred to as "self-referential enterprise system" is presented in (Frank and Strecker 2009). Figure 5 illustrates the idea of integrating enterprise systems with enterprise modelling environments.

Such a system would allow realising the vision of interactive models propagated by Krogstie (2007, p. 306): "The use of interactive models is about discovering, externalising, capturing, expressing, representing, sharing and managing enterprise knowledge." In other words: It would be a contribution to empowering people who work in and interact with organisations. The realisation of self-referential enterprise systems does not only require developing further DSML, but also redesigning enterprise software systems.

Request: To further exploit the potential of both enterprise software systems and enterprise modelling environments, research should aim at developing the foundations for integrating both kind of systems into a versatile tool for managing and adapting an organisation and its information system.

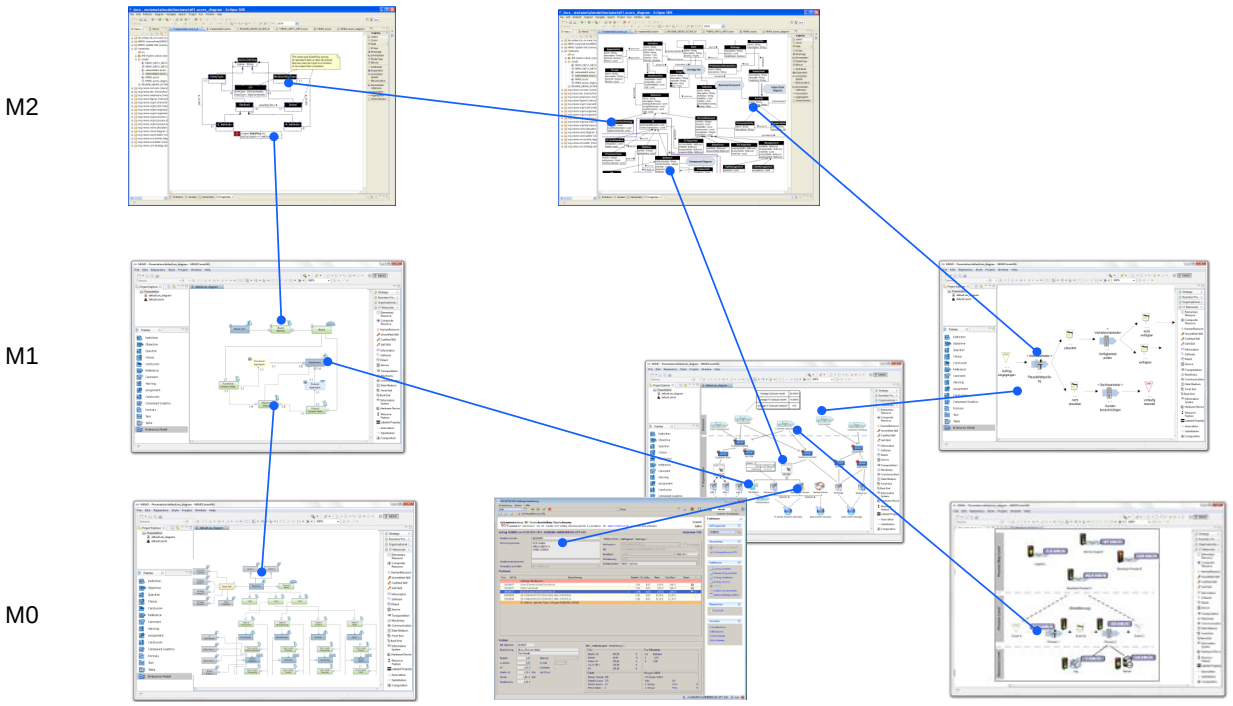

Figure 5: Navigating an enterprise model and corresponding instances

4.2 Deficiencies of Prevalent Programming Languages

The integration of enterprise modelling environments and enterprise systems does not only require research on enterprise models and their representation. It also demands for system architectures that cannot be satisfactorily accomplished with prevalent programming languages. Integration implies common representations of shared concepts. In today's modelling environments, conceptual models are usually represented by objects on the M0 level—even though they belong to the M1 or even a higher level. Overloading the M0 level happens for a good reason: Prevalent programming languages are restricted to the dichotomy of objects and classes. Hence, there are no meta classes that were required to specify classes—and that would allow treating classes as objects, too. Therefore, a common representation of classes in both systems is not possible. Instead, the only way to associate a modelling environment with a corresponding enterprise system would be to generate code, i.e., classes in the enterprise system, from objects in the enterprise modelling environment. As a consequence, one would have to deal with the notorious problem of synchronising models and code. Figure 6 illustrates how the M0 layer of modelling tools is overloaded and that concepts in modelling tools are located on a classification layer that is different from that of corresponding concepts in an associated enterprise information system.

Recent developments in research on programming language has produced (meta) programming languages that were designed for creating domain-specific programming languages. Languages like XMF (Clark and Willans 2012; Clark et al. 2008) are especially promising, since they allow for an arbitrary number of classification levels, which enables a common representation of models and respective code. Hence, modifying an enterprise model implies modifying the respective part of the enterprise software simultaneously.

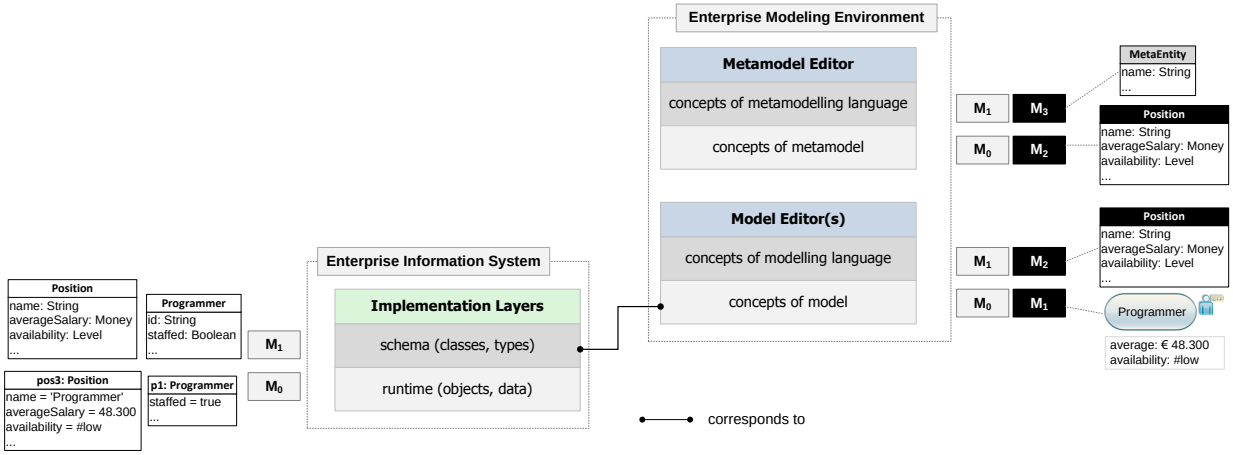

Figure 6: Mismatch of Classification Levels

Request: To advance the state of current modelling environments, research needs to focus on tools that overcome the limitations of current programming languages.

5 The Need for Collaboration

Extending the scope of enterprise models and developing them to an omnipresent representation of organisations requires an amount of research and development that cannot be carried out by the current enterprise modelling community alone. To advance the field, there is need for cross-disciplinary collaboration and for accumulating resources.

5.1 The Importance of Bundling Resources

The development of comprehensive enterprise models will overburden most organisations. This is the case, too, for respective DSML. Reference enterprise models and wide-spread DSML are suited to effectively address this problem. Furthermore they would provide a foundation for cross-organisational integration of action systems and information systems, which could enable a tremendous boost of productivity. At the same time, the development of reference artefacts that combine a descriptive and a prescriptive approach constitute an attractive research goal: It is characteristic for research to abstract from single cases and aim at constructions that work for an entire class of cases. Furthermore, applied research is motivated by improving existing practice with respect to certain goals. Unfortunately, the development of reference enterprise models and corresponding languages and tools requires resources that are not available to a single research institute. Furthermore, establishing and disseminating them in practice depends on economic and political aspects that are beyond the abilities and intentions of academics. Against the background, it is obvious that there is need to bundle resources of research institutions. At the same time, it is necessary to get vendors of enterprise software and prospective users involved. On the one hand, they need to be involved to support requirements analysis. On the other hand, using reference artefacts in practice is the only way to promote their dissemination. Unfortunately, there are serious obstacles that impede both bundling of research resources and involvement of companies. Research is based on competition and the idea of scientific progress. Collaboration of research institutes implies to give up competition to a large extent. At the same time, for reference artefacts to be beneficial they need to be consolidated—which may jeopardise scientific progress. While there are probably many vendors and client organisations that would be

happy to use reference models, most of them will likely be reluctant to participate in respective development projects, since the return on such an investment would be hard to determine. Nevertheless, to promote the benefit of enterprise models, reference artefacts that enable attractive economies of scale are of pivotal relevance.

Request: There is need for initiatives to collaboratively develop and disseminate reference artefacts. They need to provide convincing incentives both for academics and practitioners.

One of the prime examples of community-driven collaboration is free and open source software (FOSS). Respective initiatives have successfully promoted collaboration of developers and users. Also, they led to software systems of surprising quality, and, in some cases, to an impressive dissemination. Inspired by the apparent success of some FOSS projects, corresponding "Open Models" initiatives have been proposed (Frank and Strecker 2007) and inspired the creation of open model repositories (France et al. 2007), (www.openmodels.org, www.openmodels.at). While these repositories have triggered remarkable attention, there is still need for more active participation.

5.2 Enterprise Models as Object and Promoter of Cross-Disciplinary Collaboration

Enterprise models are aimed at providing a medium to foster communication between stakeholders with different professional background. On the one hand that requires reconstructing technical languages and professional patterns of problem solving. On the other hand it recommends analysing how prospective users react upon the models they are presented with. That includes concepts as well as their designation and (graphical) representation. For using enterprise models effectively, software tools are mandatory. Developing and integrating them with other enterprise software systems creates substantial challenges for Software Engineering or—in other words—interesting research questions.

Enterprise modelling is not an end itself. Instead, it is supposed to have a positive impact on a company's economics and competitiveness. However, assessing the costs of creating and maintaining enterprise models, which may include the development of languages and tools, is not a trivial task—and it is similarly challenging to determine the benefits that can be contributed to the deployment of enterprise models. Apart from economic effects, the extensive use of enterprise models within an organisation may have an impact on how people perceive not only their tasks but also the entire organisation. The increase in transparency may have an effect on established patterns of organisational power and may require new approaches to managing organisations. Against this background it is obvious that enterprise modelling involves a wide range of demanding research questions that concern various disciplines. Business and Administration in general is aimed at developing and improving terminologies and methods that are suited to structure and guide purposeful action in enterprises. Various subfields, such as Financial Management, Accounting, Industrial Management, Logistics could contribute to extend and deepen enterprise models. In Psychology, the interaction between cognitive models and external representations is a core research topic. Applied to enterprise modelling this would include the question how conceptual models effect individual and collective decision making. That includes analysing the impact of graphical notations on people's ability to understand complex matters—and the development of guidelines for designing notations that fit certain cognitive styles. Both, from a psychological and a sociological point of view, it would be interesting to analyse how enterprise models effect the social construction of reality, i.e., to what extent people perceive the model as the enterprise and what that means for the way they (inter) act. Assuming that enterprise models may have a substantial effect on an organisation's performance implies challenging research questions for economic studies that are not restricted to enterprise models, but comprise the economics

of models and methods in general. Combining research results from various disciplines would not only contribute to advance our knowledge about enterprise models and our ability to use them effectively, it is also suited to enrich the state of the art in the participating disciplines, since it would integrate it with contributions from other fields. Therefore the following request could have a better chance to succeed than yet another call for inter-disciplinary research.

Request: Advancing the field of enterprise modelling recommends to establish inter-disciplinary research collaboration.

6 Conclusion

In the past, enterprise modelling, though arguably pointing at a core topic of Information Systems, has been subject of a rather small, specialised research community. In Business and Administration it is regarded as too much focussed on technical aspects by some, while some traditionalist colleagues in Computer Science suspect it of lacking formal rigour. However, enterprise modelling is more than analysing and designing information systems—and it is certainly much more than drawing "'bubbles and arrows". Enterprise modelling is about conceptualising an important part of the world—as it actually is and as it might be. Hence, it requires knowledge about how people (inter) act in organisations, how information systems infrastructures are built—and the creativity to develop substantial images of attractive future worlds that comprise the purposeful construction and use of information systems. It is about how we perceive the world we live and work in and how we think about it and might change it—alone and together with others. In addition to supporting collaboration between stakeholders with different professional backgrounds in organisations, enterprise models may also serve as a medium and object of inter-disciplinary research. At the same time, they are suited to foster the exchange between practice and academia, because they allow to integrate more abstract representations of enterprises with more specific ones. Last but not least, enterprise models provide a laboratory for learning, because they convey a solid conceptual foundation of information systems and surrounding action systems—and enable students to navigate through an enterprise on different levels of detail and abstraction. With respect to such a wide and deep conception of enterprise modelling it is important not only to identify relevant steps of future research, but also to spread the word and encourage others to participate in joint projects. Further developing the field also requires to put more emphasis on assessing model artefacts. On the one hand that comprises the development of pragmatic criteria to evaluate models and modelling languages with respect to an intended practical use. On the other hand, it relates to assessing the epistemological quality of model artefacts as research results. Developing and applying respective criteria is an important prerequisite of scientific competition and progress.

In this paper I gave a personal account of the topics we should address in the next years to advance our field. It is needless to say that other relevant topics exist, too. I would hope that the requests presented in this paper contribute to a discourse on our future research agenda.

References

der Aalst W., Basten T. (2002) Inheritance of Workflows – An Approach to Tackling Problems Related to Change. In: Theoretical Computer Science 270, p. 2002

Atkinson C., Gutheil M., Kennel B. (2009) A Flexible Infrastructure for Multilevel Language Engineering. In: IEEE Transactions on Software Engineering 35(6), pp. 742–755

Blair G., Bencomo N., France R. B. (2009) Models@ run.time: Computer. In: Computer 42(10), pp. 22–27

Brinkkemper S. (1996) Method Engineering: Engineering of Information Systems Development Methods and Tools. In: Information and Software Technology 38(4), pp. 275–280

Buckl S., Matthes F., Roth S., Schulz C., Schweda C. (2010) A Conceptual Framework for Enterprise Architecture Design. In: Proper E., Lankhorst M. M., Schönherr M., Barjis J., Overbeek S. (eds.) Trends in Enterprise Architecture Research. Lecture Notes in Business Information Processing Vol. 70. Springer, Berlin, Heidelberg and New York, pp. 44–56

CIMOSA: Open system architecture for CIM. Springer, Berlin, Heidelberg and New York

Clark T., Willans J. (2012) Software Language Engineering with XMF and XModeler. In: Mernik M. (ed.) Formal and Practical Aspects of Domain-Specific Languages. Information Science Reference, pp. 311–340

Clark T., Sammut P., Willans J. (2008) Superlanguages: Developing Languages and Applications with XMF. Ceteva

Dijkman R., Dumas M., van Dongen B., Käärik R., Mendling J. (2011) Similarity of business process models: Metrics and evaluation. In: Inf. Syst. 36(2), pp. 498–516

Ferstl O. K., Sinz E. J. (1998) Modeling of business systems using the Semantic Object Model (SOM): A methodological framework. In: Bernus P., Mertins K., Schmidt G. (eds.) Handbook on Architectures of Information Systems. International Handbooks on Information Systems Vol. 1. Springer, Berlin, Heidelberg and New York, pp. 339–358

Fettke P., Loos P. (eds.) Reference Modeling for Business Systems Analysis. Idea Group, Hershey

France R., Bieman J., Cheng B. (2007) Repository for Model Driven Development (ReMoDD). In: Kühne T. (ed.) Models in Software Engineering. Lecture Notes in Computer Science Vol. 4364. Springer Berlin Heidelberg, pp. 311–317

Frank U. (1994) Multiperspektivische Unternehmensmodellierung: Theoretischer Hintergrund und Entwurf einer objektorientierten Entwicklungsumgebung. Oldenbourg, München

Frank U. (2011) The MEMO Meta Modelling Language (MML) and Language Architecture. 2nd Edition. ICB-Research Report 43

Frank U. (2012) Specialisation in Business Process Modelling: Motivation, Approaches and Limitations. 51. University of Duisburg-Essen. Essen

Frank U. (2013a) Domain-Specific Modeling Languages - Requirements Analysis and Design Guidelines. In: Iris Reinhartz-Berger, Aron Sturm, Tony Clark, Yair Wand, Sholom Cohen, Jorn Bettin (eds.) Domain Engineering: Product Lines, Conceptual Models, and Languages. Springer, pp. 133–157

Frank U. (2013b) Multi-Perspective Enterprise Modeling: Foundational Concepts, Prospects and Future Research Challenges. In: Software and Systems Modeling (in print)

Frank U., Strecker S. (2007) Open Reference Models – Community-driven Collaboration to Promote Development and Dissemination of Reference Models. In: Enterprise Modelling and Information Systems Architectures 2(2), pp. 32–41

Frank U., Strecker S. (2009) Beyond ERP Systems: An Outline of Self-Referential Enterprise Systems: Requirements, Conceptual Foundation and Design Options

Group T. O. (2009) TOGAF Version 9. Van Haren, Zaltbommel

Henderson-Sellers B., Ralyté J. (2010) Situational Method Engineering: State-of-the-Art Review. In: Journal of Universal Computer Science 16(3), pp. 424–478

Kelly S., Tolvanen J.-P. (2008) Domain-specific modeling: Enabling full code generation. Wiley-Interscience and IEEE Computer Society, Hoboken and N.J

Kleppe A. G. (2009) Software language engineering: Creating domain-specific languages using metamodels. Addison-Wesley, Upper Saddle River and NJ

Koschmider A., Oberweis A. (2007) How to detect semantic business process model variants? In: Proceedings of the 2007 ACM symposium on Applied computing. SAC '07. ACM, New York, NY and USA, pp. 1263–1264

Krogstie J. (2007) Modelling of the People, by the

People, for the People. In: Krogstie J., Opdahl A., Brinkkemper S. (eds.) Conceptual Modelling in Information Systems Engineering. Springer Berlin Heidelberg, pp. 305–318

Land M. O., Waage M., Proper E., Cloo J., Steghuis C. (2009) Enterprise architecture: Creating value by informed governance. Springer, Berlin

Lankhorst M. M. (2005) Enterprise architecture at work: Modelling, communication, and analysis. Springer, Berlin, Heidelberg and New York

Liskov B. H., Wing J. M. (1994) A Behavioral Notion of Subtyping. In: ACM Transactions on Programming Languages and Systems 16, pp. 1811–1841

Moody D. L. (2009) The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. In: IEEE Transactions on Software Engineering 35(6), pp. 756–779

Ralyté J., Agerfalk P. J., Kraiem N. (2005) Methods, Techniques and Tools to Support Situation-Specific Requirements Engineering Processes: SREP'05. University of Limerick, Limerick

Ralyté J., Brinkkemper S., Henderson-Sellers B. (2007) Situational method engineering: Fundamentals and experiences. Proceedings of the IFIP WG 8.1 Working Conference, 12-14 September 2007, Geneva, Switzerland. IFIP - the International Federation for Information Processing Vol. 244. Springer, New York

Scheer A.-W. (1992) Architecture of Integrated Information Systems: Foundations of Enterprise Modelling. Springer, Berlin and New York

Schrefl M., Stumptner M. (2002) Behavior-consistent specialization of object life cycles. In: ACM Transactions on Software Engineering Methodologies 11(1), pp. 92–148

Simonyi C., Christerson M., Clifford S. (2006) Intentional software. In: Proceedings of the 21th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2006). ACM, pp. 451–464

Strecker S., Frank U., Heise D., Kattenstroth H. (2012) MetricM: a modeling method in support of the reflective design and use of performance measurement systems. In: Information Systems and e-Business Management 10(2), pp. 241–276

Völter M. (2013) DSL Engineering: Designing, Implementing and Using Domain-Specific Languages. dslbooks.org

Zachman J. A. (1987) A framework for information systems architecture. In: IBM Systems Journal 26(3), pp. 276–292

Ulrich Frank

Chair of Information Systems and Enterprise Modelling
Institute for Computer Science and Business Information Systems
University of Duisburg–Essen
Universitätsstraße 9
45141 Essen
Germany
ulrich.frank@uni-due.de