Power-Modelling
Toward a More Versatile Approach to Creating and Using Conceptual Models

Ulrich Frank
University of Duisburg-Essen, Essen, Germany
ulrich.frank@uni-due.de

Keywords: DSML, Interactive Modelling, Multi-level Modelling, Empowerment.

Abstract: The prospects of conceptual modelling are widely undisputed. Nevertheless the current practice of conceptual modelling remains unsatisfactory. Usually, modelling languages offer primitive concepts only—with respective effects on productivity and model quality. The creation of models is restricted to early phases of system life-cycle. Hence, the benefits of models in later phases are ignored. Furthermore, the creation and use of conceptual models is still restricted to experts only. In this paper, the outline of a new modelling paradigm, referred to as power-modelling, is presented. It builds on the potential of domain-specific modelling languages (DSML), application frameworks and reference models. It regards models as the primary medium to perceive, interact with and change systems and the environment they are supposed to operate in during the entire system life-cycle. For this purpose, power-modelling is built on an extensible set of multi-level DSML that fit the conceptual perspectives of a wide range of prospective users and a common representation of models and code, which allows overcoming the notorious problem of synchronizing models and code.

1 INTRODUCTION

If maturity comes with age, a few decades of research and application should have resulted in a widely perfected state of conceptual modelling. There are indeed various signs of maturity. Both in Information Systems and Computer Science it seems undisputed that conceptual models are a prerequisite to manage the complexity of large software systems. Furthermore, it is acknowledged that conceptual models are much better suited than code to involve prospective users and other stakeholders in the process of developing software. The benefits of conceptual models for designing software systems do not come as a surprise. After all, software systems are *linguistic artefacts*. On the one hand, they are realized with some kind of implementation language. On the other hand, as non-physical artefacts they can be perceived—and used—by humans only through some kind of linguistic representation. At best, this linguistic representation corresponds to the language used in the targeted domain. Conceptual models are aimed at reconstructing domain-specific languages in a way that prospective users perceive them as familiar, while at the same time they allow for transformations into implementation-level languages. By structuring and eventually automating the transformation of models into code, as it is pursued by approaches to model-driven software development (Atkinson and Kühne, 2003; France and Rumpe, 2007), conceptual modelling is promising to substantially improve the productivity of developing software systems.

But conceptual modelling is not restricted to modelling software systems. Exploiting the potential of information systems often requires re-organizing respective action systems. Consequently, corresponding modelling approaches, such as business process modelling, turned out to be a good choice for supporting people with analysing and (re-) structuring action systems. The insight that corporate action systems do not only comprise business processes, but also other subjects, such as goals, resources or organizational structure, contributed to the emergence of approaches to enterprise modelling (Scheer, 1992; Ferstl and Sinz, 2006; Frank, 2013). By integrating conceptual models of software systems with conceptual models of action systems, enterprise models promise to provide a foundation for jointly analysing and designing information systems and the relevant organizational context. Since more and more organizations have given up developing software on their own, the original idea of using conceptual models for designing software systems does not fit the demands of many organizations anymore. At the same

Frank U.
Power-Modelling Toward a More Versatile Approach to Creating and Using Conceptual Models.
DOI: 10.5220/0005423800090019
In *Proceedings of the Fourth International Symposium on Business Modeling and Software Design (BMSD 2014)*, pages 9-19
ISBN: 978-989-758-032-1

time, the ever growing complexity of IT infrastructures created additional challenges. Again, conceptual models of IT infrastructures, including various representations of their high-level structure or architecture, have evolved as a remedy. While respective models are often part of enterprise modelling methods, they are also featured by approaches to enterprise architecture, which in general aim mainly at managers and therefore emphasize a higher level of abstraction (Lankhorst, 2005; Proper et al., 2010; Buckl et al., 2010). Therefore, conceptual modelling supports a wide range of activities related to analysing, designing and managing IT infrastructures and corresponding action systems. A relatively large research community, both in Information Systems and Computer Science, may serve as further evidence for the maturity of the field.

However, our brief assessment of the field's contributions may be deceptive—and maturity may have its downsides, too. As we shall see, there are serious reasons for not being satisfied with the state of the art in conceptual modelling. There are even reasons to challenge the current paradigm of conceptual modelling. In any case, it does not seem appropriate to further follow existing paths of research by focussing solely on problems within this paradigm without leaning back once in a while to reflect upon other ways to conceptualize, develop, maintain and use models. While I had my doubts for some time whether we are doing the right thing, I started to intentionally abandon some characteristics of conceptual modelling which I had not only taken for granted, but which I used to preach enthusiastically. I am still in the—sometimes painful, sometimes exciting—process of realigning my perspective and my research agenda. However, a few ideas have emerged that I regard as promising. One of them is subject of this paper. I dared calling it "power-modelling", not only to express that it might be suited to substantially promote the power of modelling as a tool to create and modify systems, but also to empower users by supporting them with sophisticated, but convenient instruments to use and modify the systems they work with.

The paper starts with a critical review of the current state of the art. Subsequently, an overview of approaches that address certain shortcomings of conceptual modelling is given. Against this background, I will describe the idea of power-modelling by outlining the foundational concepts and by illustrating how it could be implemented. Since the respective research is in a very early stage and faces serious challenges, the conclusions will especially focus on future research.

2 CONCEPTUAL MODELLING: A CRITICAL REVIEW

There are various reasons, why the current state of conceptual modelling might not be regarded as satisfactory (Frank, 2014a). At first, there is the sobering fact that the dissemination of conceptual modelling in practice is still rather modest. Many software developers still regard it as dispensable. Managers perceive it as a cost-driver that does not deliver measurable benefit. Finally, prospective users are often not keen to look into conceptual models, nor are they capable of designing them on their own. The unsatisfactory adoption in practice may be contributed to a lack of respective professional education among today's workforce. However, I am afraid, we would take the easy way out, if we were satisfied with this explanation. There are other, more essential reasons for questioning the power of current approaches to conceptual modelling, which also may, in part, hinder its acceptance and dissemination in practice. They relate to the economics of modelling and the psychological assumptions underlying the construction and perception of conceptual models.

2.1 Economics of Modelling

With respect to economics of modelling, three aspects are of particular relevance. At first, there is the productivity of creating, analysing and modifying conceptual models, i.e. the time these activities take for a certain outcome. While modelling productivity depends on modellers' skills and experiences, it is also affected by the available modelling languages and tools. A modelling language can contribute to productivity by providing reusable artefacts and by allowing for abstractions that foster reuse and adaptation of models. Currently, most modelling languages are restricted to a few semantic primitives that remind of basic ontologies like the one suggested by (Bunge, 1977) or (Grossmann, 1983). While generic concepts such as "entity type", "class", "attribute", etc. can be applied to any domain—which is the purpose of a general ontology—they require modellers to reconstruct all concepts of a model from scratch. Hence, they promote a wide range of reuse, which should improve economies of scale e.g. of modelling tools, but in a particular case, their contribution to productivity is very poor. Imagine, you would have to describe a domain such as accounting and you were restricted to a language that consisted of generic concepts like the ones provided by the ERM and the UML! Furthermore, modelling languages provide only a few abstraction concepts such as classification, generalization or encapsulation. As a consequence, similarities between a range of models can often not be accounted for by abstracting on a set of common properties. The lack of abstraction is especially painful in business process modelling, where reuse is widely restricted to copy&paste (Frank, 2012). Second, there is little protection of investment. The use of conceptual models is widely restricted to the build time phase. In later phases, models are used for documentation only, if they are used at all. Even in those cases, where models were used to generate code, they will usually get devalued over time, because during maintenance changes are directly applied to code and synchronizing models and code, beside representing a serious challenge, does not happen. Third, the benefit of models depends on their quality. However, judging the quality of conceptual models is a demanding task and requires experts. Current modelling languages and tools hardly contribute to model quality, since their generic concepts allow for almost any kind of absurd models as long as they are syntactically correct. From a managerial perspective there is the additional problem that the economics of modelling is hard to judge. While it is often not trivial to determine modelling costs in advance, quantifying the benefits of models ex ante is almost impossible. As long as managers are not convinced that modelling is suited to generate substantial benefit, the lack of legitimate quantification methods creates a serious obstacle to conceptual modelling in practice.

2.2 Cognitive Capabilities

Conceptual modelling is based on basic assumptions that most members of the modelling community—including myself—are convinced of. First, there is the assumption that the analysis and design of complex systems recommends a rational approach, i.e. an approach that is characterized by the differentiated consideration of pros and cons and that puts emphasis on justifying decisions. Second, related to the first assumption, following a Kantian tradition, a rational perspective demands for focussing on *concepts*. Conceptual modelling is typically aimed at the *reconstruction* of terms used in the domain of interest. Referring to existing terminology is supposed to make conceptual models accessible by people working in the domain. At the same time, a reconstruction is required in order to create concepts that fit the specific modelling purpose and that facilitate mapping to implementation languages. As a consequence, it it common to specify concepts according to the definitions of types or classes in implementation languages. While these basic ideas make perfect sense, they are based on an idealization that is seriously challenged by research in cognitive psychology. There is a plethora of work that shows the rather limited ability of most humans to precise and consistent thinking, for an overview see (Kahneman et al., 1982). Furthermore, the way people acquire and use concepts is often in clear contrast with the definitions we use in conceptual modelling (Lakoff, 1990). Among the most relevant insights are the following: Concepts are often not associated with (extensional) definitions, but rather with a few typical examples (prototypes"). Conceptual categories are often perceived as having no clear boundaries (membership gradience", (Lakoff, 1990), p. 12). The last example is of especial relevance for our investigation: Most cognitive models are embodied with respect to use." ((Lakoff, 1990), p. 12) As a consequence, we cannot expect prospective users to be much interested in and capable of thinking in concepts as we use them for modelling purposes.

2.3 Preliminary Conclusion

Our brief analysis of the current state of conceptual modelling resulted in a number of problems. That does not mean, however, to question the idea of conceptual modelling in general. In order to cope with complexity and change there is no alternative for us other than to somehow develop models. Without abstraction not only our understanding of systems and of the world in general will remain poor, but so would be our ability to design and implement systems. Nevertheless, those problems should make us think. In particular, there are three areas that demand for more attention. There is need to promote modelling productivity. For this purpose, reuse has to be fostered by providing modelling constructs that incorporate domain-specific semantics. To promote model quality, the reusable artefacts offered to users should be thoroughly developed and tested. At the same time, economies of scale are crucial. For this purpose, an artefact needs to be reused in a wide range of cases, which may create a conflict to modelling productivity. To improve the involvement of prospective users, models, new forms of presenting concepts and models to users are required. Since there is evidence that people imagine or learn concepts by associating them with contexts of use, it might be advisable to develop modelling environments that focus on the use of concepts. The use of models during later phases of the system life-cycle demands for developing scenarios that illustrate the respective benefit. In addition, technical challenges, such as solving the notorious problem of synchronizing code and model have to be addressed.

3 SELECTED APPROACHES

There have been various approaches that address some of the problems elucidated above. Some of them are aimed at improving modelling productivity and model quality, while others focus on implementation issues or on user involvement.

3.1 Focus on Productivity and Quality

Reference models are based on a convincing idea. By developing models that serve as a reference for certain domains, the costs for realizing large models in respective domains could be reduced substantially. At the same time, reference models should be developed with outstanding care and expertise. Therefore, they should effectively improve model quality. Furthermore, reference models do not only stress a descriptive intention, i.e. representing a domain as it is, but also a prescriptive intention, i.e. representing conceptualisations that seem especially favourable for the future development of organizations in the targeted domain. Despite their convincing foundation, the dissemination of reference models in practice remained extremely modest. Domain-specific modelling languages (DSMLs) are build on a similar idea. Different from general-purpose modelling languages (GPMLs), they comprise domain-specific concepts, thus freeing modellers from the need to specify these concepts from scratch. At the same time, DSMLs feature usually, but not necessarily, a concrete syntax that is adapted to representations known in the domain they cover. In addition to promoting productivity through improved reuse and ergonomics, DSML also contribute to model quality, since their syntax and semantics facilitate preventing models that are all too strange. Modelling is more convenient, and the range of possible models is clearly restricted compared to a GPML.

While DSMLs are certainly suited to address the productivity challenge, they are not the silver bullet of conceptual modelling, since they face a fundamental conflict of system design. On the one hand, increasing productivity demands for concepts that are specific to a particular domain, i.e. that incorporate a high degree of domain-specific semantics. On the other hand, economies of scale demand for DSMLs that cover a wider domain, i.e. that include a smaller degree of domain-specific semantics. This fundamental conflict of designing DSMLs is illustrated in fig. 1

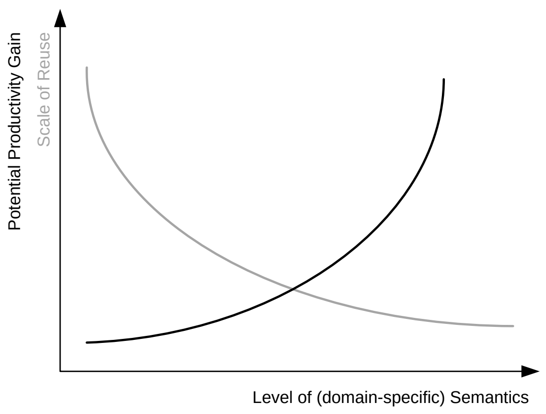

Figure 1: The fundamental conflict of designing DSMLs.

3.2 Focus on Implementation

If one regards a model mainly as an instrument for developing software, the transformation of models into code is a major aspect of increasing the value of models. Some time ago, Wiederhold et al. proposed an approach, called “megaprogramming” to substantially promote the productivity of *programming* large application systems (Wiederhold et al., 1992). Even though they do not explicitly speak of modelling, there are clear relations to conceptual modelling in their work. They suggest to compose large software systems from “megamodules”. Megamodules can be thought of as domain-specific abstraction: megamodules “capture the functionality of services provided by large organizations like banks, airline reservation systems, and city transportation systems. ... The concepts, terminology, and interpretation paradigm of a megamodule is called its ontology.” (Wiederhold et al., 1992), p. 89. However, the representation of megamodules as well as their composition happens on the code level only. Therefore, they are hardly suited for being used by non-experts. It is, however, conceivable to combine the idea of megamodules with a representation that corresponds more clearly to a terminology users are familiar with. Even though megamodules, like DSML, promote domain-specific artefacts, their use is based on a bottom-up approach in the sense that a system is created from some kind of building block without providing a blueprint for the overall design. *Application frameworks* stress a top-down approach. An application framework represents an architecture and the partial implementation of a class of application systems. “Black box” frameworks can be adapted only through interfaces, while “white box” frameworks allow for modifying their code. Frameworks can substantially boost the productivity of application development. However, while the adaptability of black box frameworks is limited, using white box frameworks efficiently requires considerable effort. Similar to megamodules the representation of frameworks is usually restricted to code. Approaches to *model-driven software development* take advantage of conceptual models for designing systems. At the same time, they promote implementation productivity by aiming at generating software systems from models (France and Rumpe, 2007), (Stahl and Völter, 2006). To cope with a multitude of platforms and programming languages, specific effort has been put on generating platform- and language-independent representations that allow for a straightforward transformation into particular implementations (Mellor, 2004), (Pastor and Molina, 2007). Even though model-driven software development is suited to improve development productivity and software quality, it remains unsatisfactory with respect to the evolution of software systems: Usually it is not possible to generate an entire software system from models, i.e. there is need for manual extensions. As a consequence, the evolution of code and models has to be synchronized. While there are a few approaches that address the challenge of synchronizing models and code (e.g. (Balz et al., 2010), (Agrawal, 2003)), they are not satisfactory, because they cannot always ensure consistency.

3.3 Focus on User Involvement

To some extent, the development of DSML and corresponding editors is aimed at making modelling more convenient for users who are not trained in conceptual modelling. The idea of using models at run time (Blair et al., 2009), while also contributing to the protection of investments, may be suited to motivate more people to use models: as a conceptual representation of the systems they interact with and may want to modify. However, recent research in this area is mainly focused on software engineering aspects such as synchronisation of models and systems, e.g. (Song et al., 2011), or self-adapting systems, e.g. (Amoui et al., 2012), (Morin et al., 2009). Krogstie proposes using models during the entire life-cycle of a system to emphasize user *empowerment* (Krogstie, 2007). For this purpose, modelling should not longer be restricted to system development. Instead, for modelling to have a "larger effect", he proposes "to enable all knowledge workers to be active modelers." (p. 305). Enterprise software systems should be presented to their users as "interactive models", the use of which is "about discovering, externalizing, capturing, expressing, representing, sharing and managing enterprise knowledge." (p. 306). In addition to known approaches to enterprise modelling, Krogstie stresses the need for more intuitive representations of models, such as "visual scenes for pro-action learning" and descriptions of the relevant context that focus on actions (p. 308). Using models as objects and objectivation of organizational knowledge work and of individual learning is appealing. However, Krogstie remains vague about the realization of his vision. He suggests a "model-generated workplace (MGWP)" that "is a working environment for the business users involved in running the business operations" (p. 312), but he does not provide details of how to accomplish "integrated modelling and execution platforms" (p. 308).

4 OUTLINE OF POWER-MODELLING

The idea presented in this section was inspired by various streams of work, some of which are described above. It is also a result of our long-standing-work on enterprise modelling that confronted us with some serious problem we were not able to solve as long as we were still bound to the traditional principles of conceptual modelling and of implementing modelling tools. Hence, the outline of power-modelling is also an attempt to suggest a new paradigm of creating and using conceptual models.

4.1 Objectives and Challenges

Against the background of our previous discussion, the following objectives mark desirable features of a future conception and realization of modelling.

Objective O1: A powerful approach to conceptual modelling should enable the use of models during the entire life-cycle of a system. *Rationale*: The complexity of enterprise systems demand for a representation that users are able to understand, not only to obtain a better comprehension of a software system, but also of the the context, since an increasing part of an enterprise is represented by its software systems. At the same time, people need (cognitive) models anyway to make sense of software systems and of organizations. Therefore, explicit models that fit the cognitive capabilities of users should be suited to increase organizational transparency. *Objective O2*: Conceptual models that represent software systems and the context they operate in should be *interactive*. *Rationale*: In order to serve as a universal interface to enterprise systems that fosters user empowerment, models need to allow for navigation, for searching and for modifying their states—not only during build time, but during run time. *Objective O3*: Conceptual models should provide access to their conceptual foundation, i.e. to the modelling language they are defined with, i.e. respective modelling tools should be *self-reflective*. *Rationale*: Users may want to get a better understanding of the concepts the models they interact with are based on. Furthermore, advanced users may even want to change those concepts. *Objective O4*: Conceptual models should be constructed from concepts that are used in professional discourse in the relevant domain. *Rationale*: Domain-specific concepts promote modelling productivity and make the use of modelling concepts more intuitive. *Objective O5*: Conceptual models of an enterprise should cover multiple perspectives and foster their integration. *Rationale*: The complexity of many organizations goes along with specialization which in turn results in different professional perspectives and languages. In order to satisfy the demand for providing users with representations they are familiar with, models of an enterprise need to account for different perspectives. On the one hand, that relates to providing modelling concepts which correspond to the technical language that is characteristic for a certain perspective. On the other hand, it should also be possible to present a model using different notations, both graphical and textual, in order to satisfy different cognitive styles. *Objective O6*: To increase the economics of modelling economies of scale have to be increased by promoting reuse. *Rationale*: Today, the effort it takes to develop elaborate models is often still prohibitively high. The above objectives create considerable challenges. Among those, three are hard to overcome within the current paradigm. Aiming at both, modelling concepts that reflect particular, domain-specific technical languages, and that promote economies of scale faces an obvious conflict: A language can either be more domain-oriented or built for serving more general purpose. Using models at run time that allow interacting with and eventually changing a software system demands a tight integration of models and code in order to keep their changes in synch. However, this is almost impossible due to limitations of prevalent programming languages. The elements of a model on M_1 are represented as objects on M_0, even though they are conceptually located on M_1. This is the case, too, for the representation of metaclasses in meta model editors. As a consequence, there is need to generate code (objects on M_0 cannot be further instantiated), which creates the notorious problem of synchronizing models and code. The lack of abstraction concepts in process modelling languages creates a serious obstacle for reuse. While generalisation/specialisation may be regarded as a suitable approach, it cannot be applied to processes in a straightforward way: To satisfy the substitutability constraint (Liskov and Wing, 1994), specialisation has to be monotonic, which is impossible to achieve for process models (Frank, 2012)

4.2 Cornerstones of Power-Modelling

For the vision of power-modelling to become real, the current paradigm is not sufficient. There is need to change our perspective on conceptual modelling and to aim at a different linguistic foundation—both of modelling and implementation languages. The following aspects mark cornerstones of a conception of power-modelling.

Emphasis on DSML: DSMLs that are reconstructed from existing technical terminologies (objective O4) promise clear advantages with respect to productivity, model quality and comprehensibility. For this purpose, we can build on an existing integrated set of DSMLs for enterprise modelling (Frank, 2013).

Multi-Level Modelling: In order to overcome the fundamental conflict of designing DSMLs (objectives O4 and O6), a *multi-level language* architecture is proposes. It is inspired by the definition and refinement of technical languages in practice. On a more generic, textbook" level, concepts are introduced that are supposed to fit a wide range of more specific domains. For that purpose, they remain intentionally abstract and underspecified. On more specific levels that could cover, for example, particular industries, concepts are refined and added. This process of stepwise refinement may in the end lead to numerous language levels, from more generic, over regional" to local" DSMLs. Multi-level modelling describes a language architecture that allows for an arbitrary number of classification levels. This allows to achieve both, a wide range of reuse, i.e. beneficial economies of scale, on higher levels, and a high productivity that is enabled by more specific DSML that reuse concepts of more generic DSML. Since all DSMLs are integrated in one language architectures, users can navigate all classification levels they are interested in (objective O3). Fig. 2 illustrates a multi-level language architecture and corresponding editors. For a detailed description of multi-level modelling see (Frank, 2014b).

Common representation of models and code: Using models during the entire life-cycle of a systems create the challenge of synchronizing models and code, which is caused by the fact that current implementation languages allow for one or two classification levels only. There are, however, a few languages, which are based on the recursive golden braid" architecture that facilitate systems with more classification levels. Among these languages, XMF (Clark et al.,

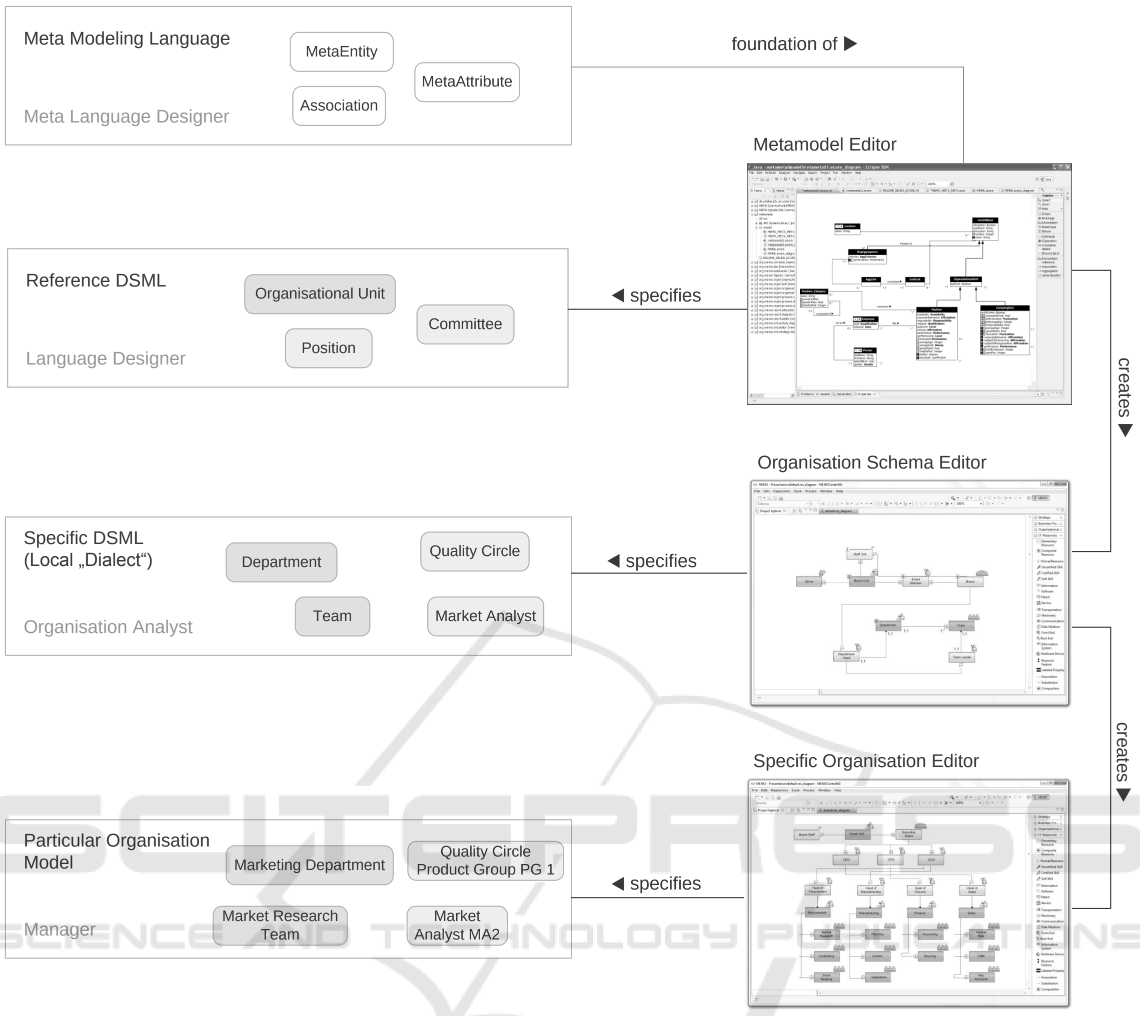

Figure 2: Illustration of multi-level language architecture.

2008b), (Clark et al., 2008a) is particularly suited for supporting power-modelling. First, it enables an arbitrary number of classification levels. Second, it is accompanied by a (meta) modelling environment, Xmodeler, that features a common representation of models and code, hence, it eliminates the need for synchronizing models and code and facilitates the use of models during the entire life-cycle of a system. In order to develop a suitable foundation of multi-level modelling and multi-level enterprise systems, we modified the metamodel of XMF (Frank, 2014b). By overcoming the separation of model and code, every system can be seen as a collection of models which can be used interactively (objective O2) with multiple representations that can be exchanged based on an implementation of the model-view-controller pattern (for details see (Clark et al., 2008a)).

Wider conception of conceptual modelling: In the existing paradigm, a conceptual model is created from the concepts of a modelling language, typically arranged in diagrams that represent some kind of a graph. However, modelling can also be thought of as an act of configuration that makes use of existing, more generic models and of DSML. In that case, modelling is more interactive, where a respective modelling system suggests alternative modelling options and requests user preferences. Since many users make sense out of action context rather than of singular concepts, interactive modelling should also go along with the representation of the context a model is supposed to address. A prototypical context could be provided by an enterprise model that integrates various perspectives on the enterprise (Frank, 2013) (objective O5). Furthermore, such a wider conception would include to not restrict modelling to drawing diagrams. Instead, models would be accessible through the elements of a graphical user interface, through tables or even through plain text. While it makes sense to re-

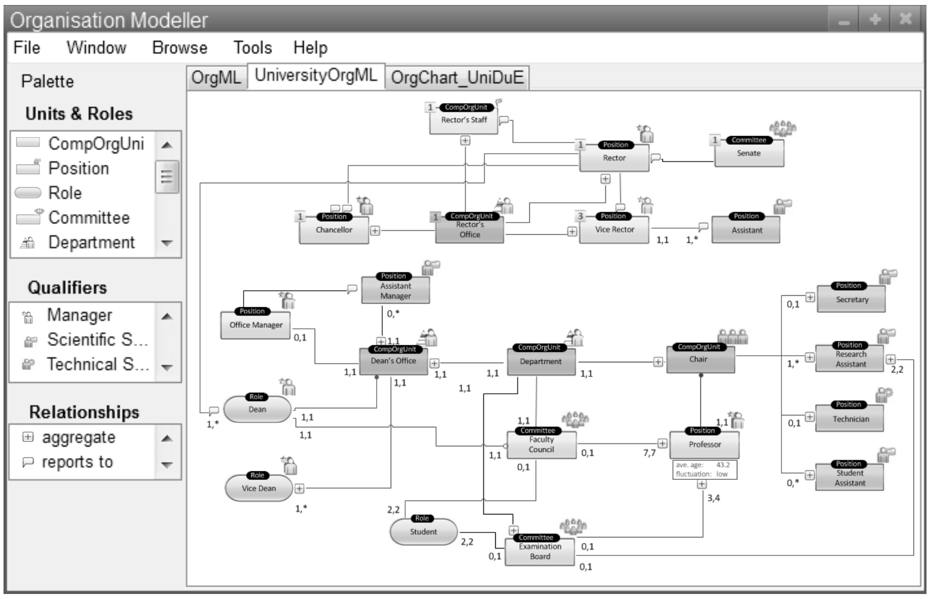

Figure 3: Definition of a DSML for modelling organizational structures of universities.

strict models to types or classes, a wider conception of modelling would also include models that comprise objects on M_0 using similar (graphical) representations to enable their interactive use. For example: The representation of a business process type could be supplemented by similar representations of business process instances (for an example in the context of so called "self-referential enterprise systems" see (Frank and Strecker, 2009), p. 14).

Ubiquitous use of (multi-perspective) models: Models are proposed as the primary medium to perceive, interact with and change systems—and the environment they are supposed to operate in during the entire system life-cycle. To serve this purpose, models have to be prepared for various perspectives to satisfy different skills and needs. Then they would no longer be a tool for system analysts and designers, but for everybody acting responsibly in an environment that is penetrated by information systems. As a consequence, power-modelling would contribute to empowering users by giving them (guided) access to a system's conceptual foundation and by supporting communication with other stakeholders.

4.3 Illustration

The following scenarios highlight selected aspects of power-modelling.

Refinement of DSML: Developing a DSML for modeling organizational structures may appear as a fairly trivial undertaking. However, this is not the case. First, a remarkable concept variance has to be accounted for. For example: A term like "department" may represent clearly different kinds if organizational units in different environments. In a university, a department consists of institutes and chairs. In some industrial enterprises, a department is part of a head department, in others not. Applying the idea of multi-level modelling to this domain would suggest developing a reference DSML for organization modelling. Such a reference DSML would include a descriptive graphical notation. A respective editor could be used by organization analysts to create an organizational schema for a certain domain, e.g. a particular organization or a range of organizations of a certain kind. Finally, this more specific DMSL would be used by local managers to build a particular organizational structure that corresponds to the previously defined schema. Fig. 3 illustrates the definition of a DSML for modelling organizational structures of universities by using a more general DSML that represents a textbook-level terminology.

The resulting DSML could then be used for creating an editor that facilitates modelling the organizational structure of a particular university. The resulting model is located on M_0. Even though it does not satisfy the abstraction usually demanded for by conceptual modelling, it is still beneficial to use a graphical notation that corresponds to that of the DSML defined on M_1. Such a model would enable interactive access to particular organizational units—and also allow for navigating to the specifications of respective concepts on higher classification levels.

Modelling as Interactive Configuration: Instead

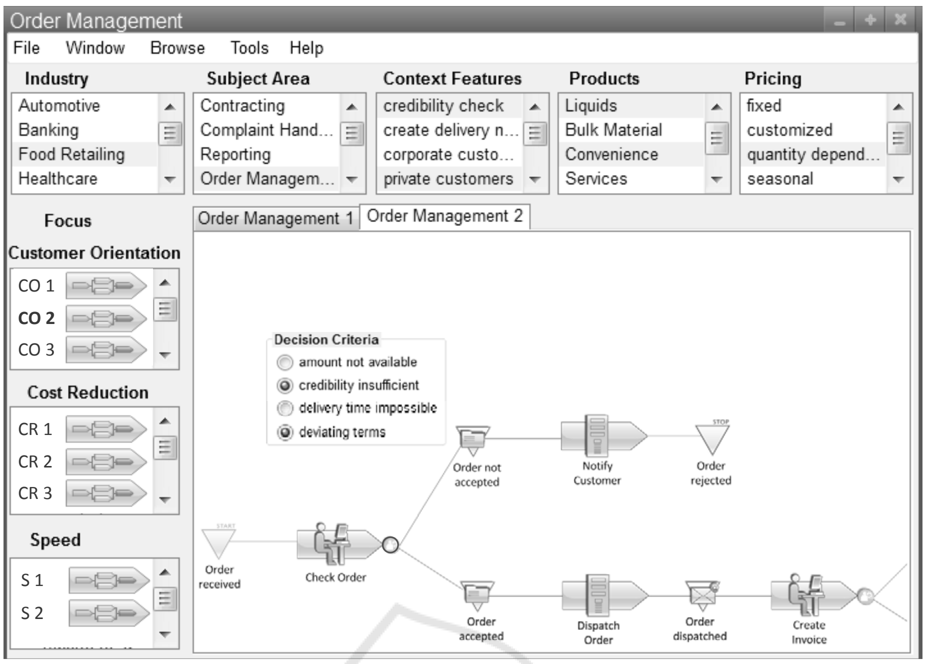

Figure 4: Configuration through stepwise selection of intended model properties.

of creating a model bottom-up from the concepts provided by a modelling language, the following scenario shows how a model can be created by combining reference models, application frameworks, multi-level modelling and a common representation of models and code to not only create models more conveniently but to realize a corresponding implementation at the same time. The scenario is based on the existence of a repository of various reference models that were created with a set of integrated DSMLs. It is aimed a modelling and thereby implementing an order management system for food retailing. Fig. 4 illustrates the use of a tool that facilitates the configuration of particular models from an existing set of reference models. At first, the user would specify the targeted domain by selecting options from lists presented by the modelling tool. Based on that selection, a set of possibly fitting business process types would be presented to the user. After selecting one business process type, the user could refine the process model by selecting from properties that are offered by the system. In case, the control flow requires further modification, the user could modify it using a respective DSML for business process modelling.

5 CONCLUSIONS AND FUTURE RESEARCH

The presented outline of a new approach to creating and using conceptual models has a twofold motivation. On the one hand, it stresses the need for leaning back once in a while and questioning assumptions that we tend to take for granted. On the other hand, it presents the cornerstones of a more versatile and powerful approach to conceptual modelling. The presentation of the vision is focussed on illustrating the idea in order to inspire a discussion about its benefits and about potential enhancements. Nevertheless, the required foundation, especially a multi-level language architecture and a common representation of models and code, is available. It has been developed during the recent years (Frank, 2014b), (Frank and Strecker, 2009) and builds on a mature meta-programming language (Clark et al., 2008b). While it required to give up certain assumptions that we had taken for granted, such as the rigid dichotomy of instantiation and specialization, it opens new ways of designing and implementing systems that are represented to users as models on different levels of abstraction. This paradigm shift requires not only giving up "standard" language architectures like MOF (Object Management Group, 2006), but also replacing existing implementation languages. Therefore, it may be regarded by some as not realistic. However, if we, at least in academia, do not give up the widespread fixation on mainstream technology, progress will hardly be possible. There is still a long way to go. Future research needs to address the development of reference models on different levels of abstraction that are specified with respective DSML. There is also need to develop advanced patterns of interaction that support the configuration and modification of models. Last but not least, it is required to overcome the limitations of current implementation languages by moving to more flexible languages architecture like the one XMF is based on.

REFERENCES

Agrawal, A. (2003). Metamodel based model transformation language to facilitate domain specific model driven architecture. In Crocker, R. and Steele, G. L. J., editors, *Companion of the 18th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2003)*, pages 118–119, New York. ACM.

Amoui, M., Derakhshanmanesh, M., Ebert, J., and Tahvildari, L. (2012). Achieving dynamic adaptation via management and interpretation of runtime models. *Journal of Systems and Software*, 85(12):2720–2737.

Atkinson, C. and Kühne, T. (2003). Model-driven development: a metamodeling foundation. *IEEE Software*, 20(5):36–41.

Balz, M., Striewe, M., and Goedicke, M. (2010). Continuous maintenance of multiple abstraction levels in program code. In *Proceedings of the 2nd International Workshop on Future Trends of Model-Driven Development (FTMDD 2010)*.

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

Buckl, S., Matthes, F., Roth, S., Schulz, C., and Schweda, C. (2010). A conceptual framework for enterprise architecture design. In Proper, E., Lankhorst, M. M., Schönherr, M., Barjis, J., and Overbeek, S., editors, *Trends in Enterprise Architecture Research*, volume 70 of *Lecture Notes in Business Information Processing*, pages 44–56. Springer, Berlin and Heidelberg and New York.

Bunge, M. (1977). *Treatise on Basic Philosophy: Volume 3: Ontology I: The Furniture of the World*. Reidel, Dordrecht.

Clark, T., Sammut, P., and Willans, J. (2008a). Applied metamodelling: a foundation for language driven development.

Clark, T., Sammut, P., and Willans, J. (2008b). *Superlanguages: developing languages and applications with XMF*. Ceteva.

Ferstl, O. K. and Sinz, E. J. (2006). Modeling of business systems using som. In Bernus, P., Mertins, K., and Schmidt, G., editors, *Handbook on Architectures of Information Systems*, International Handbooks on Information Systems, pages 347–367. Springer, Berlin and Heidelberg and New York.

France, R. B. and Rumpe, B. (2007). Model-driven development of complex software: A research roadmap. In Briand, L. C. and Wolf, A. L., editors, *Workshop on the Future of Software Engineering (FOSE '07)*, pages 37–54. IEEE CS Press.

Frank, U. (2012). Specialisation in business process modelling: Motivation, approaches and limitations. icb research report, no. 51, university of duisburg-essen.

Frank, U. (2013). Multi-perspective enterprise modeling: Foundational concepts, prospects and future research challenges. online first: http://link.springer.com/article/10.1007/s10270-012-0273-9. *Software and Systems Modeling*.

Frank, U. (2014a). Enterprise modelling: The next steps. *Enterprise Modelling and Information Systems Architectures*, 9(1):24–40.

Frank, U. (2014b). Multilevel modeling: Toward a new paradigm of conceptual modeling and information systems design. *Business and Information Systems Engineering*, 6:accepted for publication.

Frank, U. and Strecker, S. (2009). Beyond erp systems: An outline of self-referential enterprise systems: Requirements, conceptual foundation and design options. icb research report, no. 31, university of duisburg-essen.

Grossmann, R. (1983). *The Categorical Structure of the World*. Indiana University Press, Bloomington.

Kahneman, D., Slovic, P., and Tversky, A., editors (1982). *Judgment under uncertainty: Heuristics and biases*. Cambridge University Press, Cambridge and New York.

Krogstie, J. (2007). Modelling of the people, by the people, for the people. In Krogstie, J., Opdahl, A., and Brinkkemper, S., editors, *Conceptual Modelling in Information Systems Engineering*, pages 305–318. Springer Berlin Heidelberg.

Lakoff, G. (1990). *Women, fire and dangerous things: What categories reveal about the mind*. Univ. of Chicago Press, Chicago, 1 edition.

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

Liskov, B. H. and Wing, J. M. (1994). A behavioral notion of subtyping. *ACM Transactions on Programming Languages and Systems*, 16:1811–1841.

Mellor, S. J. (2004). *MDA distilled: Principles of model-driven architecture*. Addison-Wesley object technology series. Addison-Wesley, Boston.

Morin, B., Barais, O., Jézéquel, J.-M., Fleurey, F., and Solberg, A. (2009). Models@run.time to support dynamic adaptation. *IEEE Computer*, 42(10):46–53.

Object Management Group (2006). Meta object facility (mof) core specification: Version 2.0.

Pastor, O. and Molina, J. C. (2007). *Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling*. Springer, Berlin and New York.

Proper, E., Lankhorst, M. M., Schönherr, M., Barjis, J., and Overbeek, S., editors (2010). *Trends in Enterprise Architecture Research: 5th Workshop, TEAR 2010,* *Delft, The Netherlands, November 12, 2010, Proceedings*, volume 70 of *Lecture Notes in Business Information Processing*. Springer, Berlin and Heidelberg and New York.

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

Song, H., Huang, G., Chauvel, F., Xiong, Y., Hu, Z., Sun, Y., and Mei, H. (2011). Supporting runtime software architecture: A bidirectional-transformation-based approach. *Journal of Systems and Software*, 84(5):711–723.

Stahl, T. and Völter, M. (2006). *Model-driven software development: Technology, engineering, management*. John Wiley, Chichester and England and Hoboken and NJ.

Wiederhold, G., Wegner, P., and Ceri, S. (1992). Toward megaprogramming. *Commun. ACM*, 35(11):89–99.