2009

Applying lessons learned from counselling : On nurturing relations in design projects

Jörg Becker
University of Muenster, joerg.becker@ercis.uni-muenster.de

Milan Karow
University of Muenster, milan.karow@ercis.uni-muenster.de

Felix Müller-Wienbergen
University of Muenster, felix.mueller-wienbergen@ercis.uni-muenster.de

APPLYING THEORY-BUILDING TECHNIQUES TO THE DESIGN OF MODELING LANGUAGES

Becker, Jörg, University of Münster, Leonardo Campus 3, 4149 Münster, Germany, joerg.becker@ercis.uni-muenster.de

Karow, Milan, University of Münster, Leonardo Campus 3, 4149 Münster, Germany, milan.karow@ercis.uni-muenster.de

Müller-Wienbergen, Felix, University of Münster, Leonardo Campus 3, 4149 Münster, Germany, felix.mueller-wienbergen@ercis.uni-muenster.de

Pfeiffer, Daniel, University of Münster, Leonardo Campus 3, 4149 Münster, Germany, daniel.pfeiffer@ercis.uni-muenster.de

Seidel, Stefan, University of Liechtenstein, Fürst-Franz-Josef-Strasse, 9490 Vaduz, Principality of Liechtenstein, stefan.seidel@hochschule.li

Abstract

In their 2004 paper Hevner et. al proposed a set of guidelines for conducting design science research projects in the IS discipline. While useful, these guidelines have a relatively high level of abstraction. However, various IT artifacts such as models, methods, techniques and implementations require IS researchers to apply differing methods in order to construct and evaluate purposeful artifacts respectively. In this paper we discuss a particular class of IT artifacts: conceptual modeling languages. As constituent parts of software development methods, a multitude of such languages has been proposed and discussed. Yet, in the related literature on method design only little guidance is provided on how to derive appropriate conceptual modeling languages from empirical data. We believe that "good methods" need to be rigorously grounded in empirical findings. Taking a look at the related literature on inductive theory building reveals that at there are prominent similarities between the elements that constitute theories and those that constitute conceptual modeling languages: whereas theories comprise of constructs and relationships between these, conceptual modeling languages comprise of language constructs and relationships among these. We draw from the body of literature on grounded theory building and propose a new approach to designing conceptual modeling languages.

Keywords: Method Engineering, Conceptual Modeling Language, Theory Building, Grounded Theory Method

1 INTRODUCTION

Design Science Research has emerged as a popular research area in the IS discipline. Recently, there has been an increasing number of design science studies (cf. March & Storey, 2008; Winter, 2008) and also several discursive papers report on the usage of design science research in the IS discipline (e.g. Niehaves, 2007). Hevner et al. (2004) propose a set of guidelines for conducting design science research projects in the IS discipline. These guidelines are widely accepted as being feasible and providing guidance to an area of IS research that was often accused to not be rigorous and lack evaluation. While useful, the guidelines have a relatively high level of abstraction. However, particular IT artifacts such as constructs, models, methods, and instantiations require IS researchers to apply differing methods in order to construct and evaluate purposeful IT artifacts.

One such type of artifacts subsumes *conceptual modeling languages*, as being part of the "methods applied on the development and use of information systems" (Hevner et al. 2004, p. 82). The construction of conceptual modeling languages mainly originates from the field of information systems development (ISD). In the last two decades, a great number of modeling languages has been developed which left software engineers facing the major problem of method evaluation and selection (Harmsen, 1997). This problem has been addressed in a two-fold manner. First, unification efforts have been made in order to merge the mutual aspects of prevalent methods (Jacobson et al., 1999). Second, much effort has been devoted to the adaptation of development methodology by assembling specific aspects of different methods in order to meet project-specific requirements. This area of research became well-known in the ISD discipline under the notion of *(Situational) Method Engineering* (Brinkkemper, 1996; Harmsen, 1997; Ralyté & Rolland, 2001).

However, these approaches tend to not consider the actual *domains*, or *contexts*, methods are eventually applied to. The modeling languages used in this context mostly describe formal (software) systems, i.e. their constructs are anchored with formal semantics (such as programming language, c.f. Harmsen, 1997), thus having no denotation towards concepts of a material ("real-world") domain. Nevertheless these languages are increasingly used for describing material contexts, e.g. in business process modeling (Rosemann et al., 2008) or requirements engineering (Mylopoulos et al., 1999).

In this paper we advance the construction of conceptual modeling languages by introducing a new approach to ground the development of such methods in empirical data. The motivation for this approach rests in the awareness that existent literature to a great extent does not address inductive development of methods based on empirical data. Yet, the related literature on theory building reveals that there are prominent similarities between the elements that constitute theories and those that constitute conceptual modeling languages: whereas theories comprise of constructs and relationships between these, conceptual modeling languages comprise of language constructs and relationships among these. Thus, we draw from the body of literature on theory building and propose a new approach to designing conceptual modeling languages.

The paper is structured as follows. In the next section we introduce related work on method design, design science research and theory building. We then introduce a new approach to designing conceptual modeling languages by transferring existent knowledge on theory building to the domain of designing methods. We then discuss the proposed approach. The paper concludes with a brief discussion of contributions and limitations and provides an outlook onto our future research agenda.

2 BACKGROUND

2.1 Design Science Research

Design science research aims at solving practical and theoretical problems "by creating new and innovative artifacts." (Hevner et al., 2004 p. 75) The basic principles of design science research (DSR) can be traced back to engineering and Simon's (1996) sciences of the artificial. In contrast to

behavioral science, DSR does not seek to understand the world as it is and how it works. Rather, it strives to develop solutions to improve the current state of affairs. DSR intends to provide IT artifacts that are novel and useful. These IT artifacts must exceed the current state of the art and have to serve a human purpose.

There have been several attempts in the IS community to define the IT artifact (Orlikowski & Iacono, 2001; Weber, 2003; Benbasat & Zmud, 2003; Venable, 2006). March and Smith (1995) differentiate between four types of IT artifacts: *constructs* that provide language concepts in which problems are described, *methods* that explicate the process of how to solve a problem, *models* that utilize the constructs to represent an application domain and express the problem and solution space, and *instantiations* that constitute the technical realization of constructs, models, and methods. In the understanding of March and Smith (1995) design science research must ultimately lead to one of these artifacts. A design science research method seeks to systematically guide the development of an artifact.

Until now, no widely accepted research method for design science has been established in the IS community. Even more so, there exist concerns that a general design method cannot be defined (Hooker, 2004). It is argued that design is a creative process that cannot be fully formalized. Nonetheless, various procedures have been suggested to methodically support the design activities (Peffers et al., 2007; Takeda et al., 1990; Nunamaker et al., 1990; Walls et al., 1992; Cole et al., 2005). More specifically, various approaches for the design of conceptual modeling methods and languages have been proposed. the most prominent of which we discuss in the subsequent section.

2.2 Design of Conceptual Modeling Languages

As has been indicated, conceptual modeling languages are applied in order to represent the relevant knowledge of a domain (Wand et al., 1995). In this paper we focus on modeling languages that are designed in order to represent facts about *material domains,* meaning aspects of the *physical and social world* or – more precisely – perceptions thereof, for means of communication and understanding (Mylopoulos, 1992). Note that such methods are not ought to be used in order to specify formal systems.

Conceptual modeling languages comprise of fundamental modeling *constructs*, that is, language primitives which are called the *vocabulary*. For example, in the Business Process Modeling Notation (BPMN), constructs represent activities, events or sequence flows. In addition to that, conceptual modeling languages provide a collection of rules that describe how the constructs can be combined to create statements about the domain of discourse. Such rules specify what constructs *may be connected to* each other. For example, it can be defined that activities can be linked by a sequence flow. Usually, rules are specified in a language's *meta-model* and complementing contextual conditions (Earwig, 1999).

In the existent literature, two major approaches to the design of conceptual modeling languages can be found: language design in method engineering and ontology-based language development.

The first approach views conceptual modeling languages as artifacts that are created as part of a *method engineering process*. Thus, it is concerned with the selection, adaptation and design of (situation-specific) conceptual modeling languages as well as their corresponding modeling procedures. A *system development method* is assumed as to consist of a set of reusable fragments (Brinkkemper et al., 1998; Harmsen, 1997). Ralyté et al. (2004) describe different strategies for method engineering projects that differ regarding the degree of fragment *reuse*. However project specifics may, require the method engineer to derive *novel* language constructs from the problem domain at hand. While method engineering literature provides comprehensive directions on how to prepare method fragments for tool-supported integration and assembly, little guidance is given to the challenge of approaching a problem domain in order to derive feasible modeling language constructs. Furthermore, the concept of *domain* has primarily been perceived as the formal target systems in

earlier ISD modeling language development, such as programming languages or paradigms. Consequently, the resulting languages are mostly anchored to formal semantics.

The second approach aims to overcome the deficiencies of those languages to describe real-world-phenomena: to anchor modeling languages to material domains, it draws on the concept of *ontologies* as a theoretical foundation. Wand and Weber (1993) utilized a *top-level ontology* (Bunge, 1977) to evaluate existing modeling languages with regard to precision and completeness by matching the language elements with ontological concepts. Guizzardi (2005) developed an own ontological foundation for structural conceptual modeling languages and suggested an approach to derive language constructs based on this ontology. Although ontologies represent feasible *anchoring systems* for modeling language constructs (Harmsen, 1997), relying on this concept merely shifts the problem of how to identify useful constructs from language design to ontology design. Although one can find the notion of *ontology engineering* (Devedžić, 2002) and examples for the construction of particular domain ontologies (e.g. Fernández-López et al., 1999), the body of research work on this approach lacks generalized guidance on how to derive a conceptualization from empirical data.

Taking off from this discussion, in this paper we suggest an empirically-based approach to designing conceptual modeling languages. It is hoped that the inductive development of such languages based on empirical data can contribute to the languages' usability and adequateness.

3 ON THE APPLICABILITY OF THEORY BUILDING PROCEDURES TO DESIGN SCIENCE RESEARCH

In the IS discipline, there have been attempts to classify theories and develop a more narrow understanding of what theory is. Gregor (2006) distinguishes between theories for analysis, explanation, prediction, explanation and prediction and design and action. Theory for analysis "does not extend beyond analysis and description. No causal relationships among phenomena are specified and no predictions are made" (p. 620). Similarly, conceptual modeling languages provide an analysis and description of a problem domain. Thus, we argue that conceptual modeling languages can be compared to analytic theory: they provide clear definitions of constructs that are relevant in a certain problem domain and describe relations among these.

The most general term that describes the building blocks of a theory is called a concept (Strauss & Corbin, 1998). *Concepts* represent phenomena and can be grouped into more abstract concepts that are then referred to as *categories* (Strauss & Corbin, 1998). Categories have *properties*, which describe certain characteristics that objects of the same category share. For example, if one category was "actor" objects belonging to this category could share the property of "position". Thus, different actors could be placed on a dimensional range describing various positions, such as producer or director.

Generally, the scientific process comprises of the stages of observation, induction, and deduction (Eisenhardt, 1989; Handfield & Melnyk, 1998; Wallace, 1971). Thus, it can be argued that the scientific process starts with the inductive development of theory that is then deductively applied to incoming data and thus validated. This process of validation may lead to new or revised theory. There are various approaches of how to inductively develop theories, for example case study research (Eisenhardt, 1989), or Grounded Theory Method (GTM) (Glaser & Strauss, 1967; Strauss & Corbin, 1998). In this paper we particularly draw on the literature on GTM. As indicated, GTM aims at inductively develop theory based on empirical data. GTM is thought to ground the emergent theory in the data. It is not preconceived or forced upon the data but rather emerges from it (Glaser & Strauss, 1967).We argue that GTM offers the researcher a set of procedures that can be beneficial in order to inductively develop conceptual modeling languages. Our argument rests in the following observations:

- The process of building grounded theories is highly iterative. Theory and data are constantly compared (Glaser & Strauss, 1967). This process can be referred to as *comparative analysis*. Similarly, DSR processes are highly iterative and constantly compare the evolving artifact with its purpose (Hevner et al., 2004).

- Glaser and Strauss (1967) further introduce the term theoretical sampling as a process of ”data collection for generating theory whereby the analyst jointly collects, codes, and analyzes his data and decides what data to collect next and where to find them, in order to develop his theory as it emerges” (p. 45). When designing a modeling method, the method designer will start to identify what concepts may be relevant in a certain context. In order to advance the construction, he or she will further investigate the domain at hand by decisively choosing locations and respondents he or she talks to.
- Grounded theory studies typically start with a stage called open coding (Glaser & Strauss, 1967). In open coding the researcher identifies a set of themes or categories that appear to be relevant in order to describe and explain a phenomenon under investigation. Similarly, when designing a modeling method, the method designer has to identify those language constructs that are relevant and applicable to a particular domain.
- Grounded theory provides procedures that support the researcher in identifying relationships between concepts. For example, Strauss and Corbin (Strauss & Corbin, 1998) suggest to classify emergent categories by whether they represent (a) phenomena, (b) conditions, (c) actions/interactions, or (d) consequences. Thus, conditional structure is identified. Likewise, the method designer seeks to identify relationships among language constructs.
- Grounded theory relies on a technique called *memoing* (Miles & Huberman, 1994). Memos are used to document the researcher’s conceptual thoughts that eventually lead to the generation of theories. Memos are constantly written, re-written, and integrated (Strauss & Corbin, 1998). Thus, the process of memoing is conducive to the iterative nature of DSR projects, such as the development of methods. This concept is similar to what is referred to as a method rationale (Rossi et al., 2000).

In the following section we compare the basic elements of conceptual modeling languages to the basic elements of grounded theories. We then describe how the above outlined procedures of building theory can be applied to generate conceptual modeling languages.

4 APPLYING GTM PROCEDURES TO INDUCTIVELY DEVELOP CONCEPTUAL MODELING LANGUAGES

4.1 Concepts of Conceptual modeling languages and of GTM

Table 1 provides an overview of the comparison and matches the terminology of conceptual modeling language design and grounded theory method.

Table 1 Relationship between concepts from GTM and conceptual modeling language design

Modeling Language Design	**Description**	**Grounded Theory Method**	**Description**
Language Construct Candidate	Concepts of a domain can be translated to constructs of the domain-specific language	Code / Concept	building blocks of a theory, abstracts descriptions of real world phenomena
Language Construct	Categories indicate a core concept of the domain, thus a language for describing instances of that domain should provide a dedicated representation	Category	Derived from concepts, aggregated and structured, constituent part of a theory’s statement

Language Construct / Construct Property/	Properties will usually be translated to discrete language constructs, their existential dependency is codified in the language's syntax rules	Property	Derived from concepts, give concepts/categories further explanation
Language Rules	Constituent relationships will be represented in the language's syntax rules (meta model and context conditions)	Hypothesis / Propositions	Relationships between concepts may take the form of propositions or hypothesis.

Language construct candidates are early abstractions from phenomena that a method designer perceives as relevant. These elements are part of the individual *conceptualization* (Guizzardi, 2005) of the context at hand. In grounded theory development, *codes* represent first cognate incidents in the data, which a researcher assigns to more abstract terms or themes. In further iterations, these codes are assembled to *concepts* by further clarifying the context's structure and terminology.

The *language construct* is the central element of modeling language design. In a semiotical sense, language constructs are *types of particular signs* (Genova et al., 2005), that is, they have a *syntactic*, a *semantic* and a *pragmatic* dimension (Morris, 1970). The syntactic dimension of modeling languages can be split into abstract and concrete syntax (Earwig, 1999). While the former describes, what discriminate constructs are available and how they may be combined, the latter assigns a graphical representation to each construct, so as to create a *language primitive* (Guizzardi, 2005). Techniques utilized in Grounded Theory Method can contribute to outline the semantic component of language constructs. The *concepts* derived from codified data yield a promising starting point for the *material* backdrop of a domain-specific modeling language. The process of refinement and abstraction to develop early sketched concepts to structured and well-defined *categories* is analogous to the definition of a language's vocabulary.

Properties are a special class of constructs that denote existential dependency on other language constructs. We distinguish between different types of properties. So-called *intrinsic properties* are property types of language constructs that obtain a definite value when instantiated as model elements. For instance, if we defined a language construct "*task*" an intrinsic property could be "*duration*". *Mutual properties* describe property types that are shared by instances of language constructs, such as being in a relationship or being part of a composite concept (Shanks et al., 2008). In the process of theory building, properties emerge from concepts that give further explanation to particular categories and are therefore existentially dependent on them.

The *language rules* constrain the possible combinations of language constructs and are part of the abstract syntax of a modeling language (Guizzardi, 2005). As language constructs denote concepts grounded in the domain, these combinations denote meaningful statements that must also be grounded in the empirical world. One major component of theory building is the exposition of such basic statements by revealing the relevant relationships among the concepts.

Based on the identified analogies, we propose a process that guides the development of a special type of IT artifacts, namely domain-specific conceptual modeling languages. Modeling languages provide clear definitions of constructs and (potential) relationships between constructs. The process draws on techniques that stem from the literature on theory building and results in what can be referred to as analytic theory. The approach to building theory we consider generates substantive theory, that is, theory that is applicable to a certain domain. Similarly, any language developed according to the scheme we are presenting will depend on the context it was developed in.

4.2 Applying GTM procedures in order to inductively design conceptual modeling languages

In the following we describe how conceptual modeling language constructs and their relationships can be inductively developed based on empirical data. To illustrate the process, we have chosen examples

from a grounded theory study that was conducted based on data from the film industry in order to study business processes in creative environments (Seidel et al., 2008). Thus, the language to be designed would be a business process modeling language tailored to that specific material domain. The process comprises of the following steps (cf. Fig. 1): *Data Collection*, *Identification of concepts*, *further developing concepts*, *relating concepts, and concluding the design process*. As has been indicated, alike the generation of theory the design of conceptual modeling methods is a highly iterative and interwoven process, which becomes particularly evident through the use of constant comparative analysis and theoretical sampling.

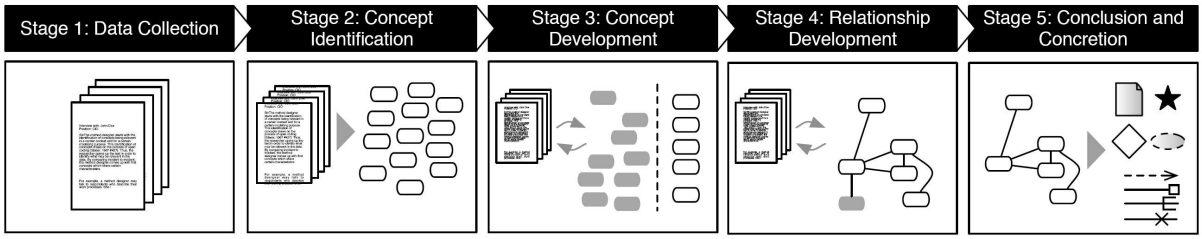

Figure 1: GTM-based Language Design Process

Stage 1: Data collection

At the outset, the modeling language designer must decide upon the data the language development is based on. Examples are the analysis of existent documentation, interviews, or observational data. Generally, a multitude of data sources can be considered, a process that is often referred to as triangulation across methods (Orlikowski, 1993). As triangulation across methods is typical for GTM studies (Glaser & Strauss, 1967), we suggest method designers to consider different data sources so as to allow for multiple vantage points for identifying what is relevant in a particular domain of interest. The result of this stage is a clear outline on what data sources will be used according to the intended scope of the modeling language to be designed.

Stage 2: Identifying concepts

The method designer starts with the identification of concepts being relevant in a certain context and for a certain modeling purpose. This identification of concepts draws on the process of open coding (Glaser & Strauss, 1967). Thus, the researcher opens up the text in order to identify what may be relevant in the data. By comparing incident to incident, the method designer comes up with first concepts which share certain characteristics and comes up with various concepts. Even though much of what will be needed may be found in the interview or observational data, the method designer may want to work with other techniques than simple comparisons. One such strategy that has also been proposed by Strauss & Corbin (1998) is that of making theoretical comparisons. Thus, the method designer enhances her "theoretical sensitivity" (Glaser, 1978), e. g. by evaluating existent modeling languages for reusable conceptualizations. The result of this stage is a quantity of domain-relevant concepts that are candidate language constructs.

Stage 3: Further developing concepts

It is not uncommon that the researcher ends up with generating a large number of concepts (Strauss & Corbin, 1998). To further integrate concepts, they are grouped under more abstract concepts called categories. By using categories the method designer reduces the number of items she works with. This process depends on the modeling purpose as well as of the individual perspective of the method designer. The result of this stage is a reduced list of categories, which comprises the elements of the first draft on the modeling language model (the meta model).

Stage 4: Relating Concepts

When starting to analyze the data, the researcher will recognize first relationships between concepts. Eventually, these relationships result in the formation of hypothesis or propositions. Similarly, when designing conceptual modeling languages, the method designer identifies potential relationships between language constructs. As has been indicated, depending on the type of language that is constructed it may be possible to distinguish different types of categories. By grouping categories accordingly, relationships emerge. Thus, the result of this stage is a first model of the domain-specific modeling language (the meta model) that comprises the quantity of language constructs and the typed relationships among them. These relationships represent allowed connections between instances of the involved constructs.

Stage 5: *Concluding the design process, development of the concrete syntax*

The iteration between analyzing data and generating language constructs and relationships can be concluded when additional data analysis does not provide any further insight. In GTM, this stage in the process is referred to as "theoretical saturation" (Glaser & Strauss, 1967). This highly iterative process rests in the application of procedures that were discussed earlier, namely the making of comparisons and theoretical sampling. During the conclusion of the theoretical development, the meta model will reach a state where no more substantially changes to the language core will be made. At this point, the language designer will assign a visual representation to each component the abstract language model (Guizzardi et al., 2002). Conclusively, the result of this stage is a domain-specific language prototype. This prototype can be used to further evaluate the underlying conceptualization, as well as to analyze the lucidity of the chosen representation (concrete syntax).

It is vital for the claim of traceability of language design and thus for the feasibility of language evaluation to rigorously document all decisions and their basis in the empirical data. As indicated, we propose to make extensive use of memoing (Miles & Huberman, 1994). There are different types of memos that can be used in order to provide a comprehensive method rationale: Codes notes, for example, accompany the process of conceptualizing based on constant comparison and theoretical sampling, whereas operational notes help to guide the researcher in deciding on what data to collect next, etc.(Strauss & Corbin, 1998).

4.3 Example Case

In the following, we illustrate the application of the proposed approach by developing exemplary language constructs based on data collected in a exploratory study on organizational creative processes. (c.f. Seidel, 2009).

Data collection: Data has been collected in three organizations with over 30 interviewees using semi-structured interview and process modeling techniques.

Concept identification: While studying the concept of CIP, the people conducting the tasks within these processes emerged as important context concepts. Example codes identified in the data were "visual effects artist", "editor" or "sound editor".

Further development of concepts: The roles identified within the CIP context were further investigated and mutual properties could be identified in the data. All these individuals share a certain *process expertise* that is necessary to carry out creative tasks, e.g. the ability to break down a creative problem in order to find a solution strategy to it. Furthermore they share the property of *creative skills*, i.e. the ability to generate novel artifacts and to judge solution on aesthetic aspects. As another important property with influence to CIP, the working *location* has been identified. The concepts have accordingly been generalized into the category *artist.*

The modeling language aims to provide for means to describe the processes within the domain. The category *artist* is codified as an element type within the language.

Relating concepts: Artists represent a specialized type of task owner in CIPs. Thus, they can be associated to creativity-intensive (sub-)processes. The property *location* has significant influence to

the collaboration with supervisors and clients, thus it will be modeled as an attribute of *artist* (c.f. Figure 2).

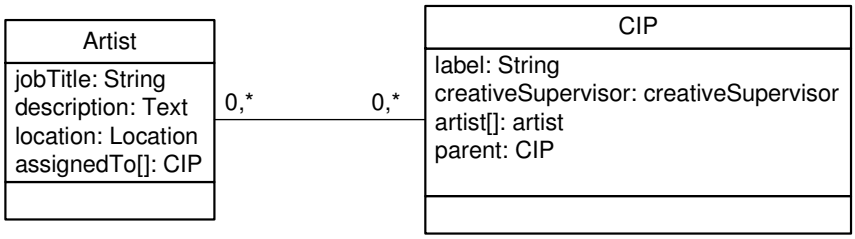

Figure 2: Meta model detail

Concluding design process: In this last step, a representation of the element type has to be developed. The CIP might be described in a form-based model where assigned artists can be added in a list and implemented into a modeling tool. To conclude the DSR process, the resulting language must be evaluated with appropriate measures (e.g. Recker, 2008).

5 DISCUSSION

In order to evaluate the approach we have suggested in this paper, we consider the guidelines proposed by Hevner (2004). Table 2 provides an overview.

Table 2: Evaluation with Guidelines as proposed by Hevner (2004)

Guidline	**Our Approach**
Guideline 1: Design as an artifact	The process we proposed in this paper aims at developing conceptual modeling languages. Thus, the process results in what is considered to be purposeful IT artifacts.
Guideline 2: Problem relevance	It lies in the responsibility of the method designer that the problem that is to be targeted by the conceptual modeling language is relevant.
Guidline 3: Design evaluation	Evaluation is an integral part of the suggested process. By applying the principles of constant comparative analysis and theoretical sampling, the researcher constantly compares concepts and relationships to incoming data.
Guideline 4: Research contributions	It is hoped that methods designed based on the approach described in this paper are both “clear and verifiable” as Hevner (2004) states. It is suggested to accomplish this by constantly iterating between inductively generating categories and relationships that are then deductively applied.
Guideline 5: Research rigor	The process we introduced aims at providing a set of procedures that can be applied in order to design conceptual modeling languages. It is hoped that by following and documenting these procedures, the method designer makes the process of method development transparent and traceable. It cannot be claimed that a particular method is complete or correct – however, it can be argued that the process of method development is plausible.
Guideline 6: Design as a search process	The development of conceptual modeling languages based on constant comparative analysis and theoretical sampling is a highly iterative search process that in every stage is highly dependent on the substantive area in which it is grounded.
Guideline 7: Communication of research	Applying rigorous procedures of documenting the research process can contribute meaningfully to successful and appropriate communication of results.

Note that the perspective we presented in this paper largely differs from that of those approaches to method engineering that stem from the discipline of software engineering. Whereas these streams of literature seek to technically sound developing conceptual modeling languages and focus on syntactic integrity, our approach focuses on the identification of relevant language constructs and relationships.

It is our belief that the appropriateness of modeling languages is determined by the context in which they are used. Of course, the development of software systems requires methods that enable to construct syntactically correct models. Thus, in many cases it will be necessary to combine formal procedures with empirically-based, inductive methods to identifying concepts and relationships that are relevant and applicable to a particular domain.

6 CONCLUSION

This research contributes to the IS body of knowledge by proposing a rigorous, empirically-based approach to inductively develop conceptual modeling languages methods based on well-established techniques known from the domain of theory building. It is our belief that there is need for detailed approaches supporting both researchers and practitioners in developing purposeful IT artifacts. Our argument rests in the observation that (a) there are similarities between basic elements of conceptual modeling languages and the elements that constitute theory, and (b) that the GTM offers researchers a set of procedures that can also be applicable to the development of such languages.

6.1 Limitations

Inductively developing theory or modeling methods limits the scope of the artifact to a so-called substantive area (Urquhart, 2001; Strauss & Corbin, 1998). Thus, they may be very practical but are also limited to a particular scope. Thus, the approach we have advanced in this paper is limited to developing modeling languages that are applicable to particular domains. The study so far limits the discussion on the development of the language elements as constituent parts of a modeling method. However, a functional method also has to provide for a modeling process that guides modelers on how to use the language constructed. Although we believe that an empirically grounded language will assist the efficient elicitation of information in its aspired domain, one can argue that the process of modeling is also dependent on the context of application.

Furthermore, it must be noted that the result of any method design process is highly dependent on both the method designer as well as the intended purpose of the language. A possible strategy to achieve a more independent view on the domain is triangulation (Eisenhardt, 1989). For instance a researcher might employ a colleague to develop an own conceptualization based on the same data.

Conclusively, any new guideline, theory, method, or approach must be tested in practice. Thus, we motivate researchers and method designers to applying those principles we presented in this paper.

Acknowledgements

This paper was written in the context of the research project ManKIP (Manangement of Creativity-Intensive Processes). The project is funded by the German Federal Ministry of Education and Research (BMBF) and by the European Social Fund of the European Union, promotional reference 01FM07061. We gratefully acknowledge the support of the Project Management Agency as part of the German Aerospace Center (PT-DLR).

References

Benbasat, I. and Zmud, R. W. (2003) The identity crisis within the is discipline: Defining and communicating the discipline's core properties. *MIS Quarterly* 27 (2), 183-194.

Brinkkemper, S. (1996) Method engineering: Engineering of information systems development methods and tools. *Information and Software Technology* 38 (4), 275-280.

Brinkkemper, S., Saeki, M. and Harmsen, F. (1998) Assembly techniques for method engineering. In *10th International Conference on Advanced Information Systems Engineering (CAiSE 1998)* (Pernici, B. and Thanos, C., Eds), pp 381-400, Springer, Pisa, Italy.

Bunge, M. (1977) *Ontology i: The furniture of the world.* D. Reidel Publishing Company, Dordrecht.

Cole, R., Purao, S., Rossi, M. and Sein, M. (2005) Being proactive: Where action research meets design research. In *26th International Conference on Information Systems (ICIS 2005)* (Avison, D. E. and Galletta, D. F., Eds), pp 1-12, Las Vegas, NV.

Devedžić, V. (2002) Understanding ontological engineering. *Communications of the ACM* 45 (4), 136-144.

Earwig, M. (1999) Visual graphs. *15th Symposium on Visual Languages*.

Eisenhardt, K. M. (1989) Building theories from case study research. *Academy of Management Review* 14 (4), 532-550.

Fernández-López, M., Gómez-Pérez, A., Pazzos-Sierra, A. and Pazzos-Sierra, J. (1999) Building a chemical ontology using methontology and the ontology design environment. *IEEE Intelligent Systems & their applications.*, 37-46.

Genova, G., Valiente, M. C. and Nubiola, J. (2005) A semiotic approach to uml models. In *1st Workshop on Philosophical Foundations of Information Systems Engineering (PHISE'05).*

Glaser, B. G. (1978) *Theoretical sensitivity: Advances in the methodology of grounded theory.* The Sociology Press, Mill Valley, CA.

Glaser, B. G. and Strauss, A. L. (1967) *The discovery of grounded theory: Strategies for qualitative research.* de Gruyter, Hawthorne.

Gregor, S. (2006) The nature of theory in information systems. *MIS Quarterly* 30 (3), 611-642.

Guizzardi, G. (2005) *Ontological foundations fo structural conceptual models.* Centre for Telematics and Information Technology, Telematica Istitute, Enschede.

Guizzardi, G., Pires, L. F. and Van Sinderen, M. J. (2002) On the role of domain ontologies in the design of domain-specific visual modeling languages. In *2nd Workshop on Domain-Specific Visual Languages at the 17th Annual ACM Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA 2002)* (Tolvanen, J.-P. and Gray, J. and Rossi, M., Eds), pp 1-14, Seattle, WA.

Handfield, R. and Melnyk, S. A. (1998) The scientific theory-building process: A primer using the case of tqm. *Journal of Operations Management* 16, 321-339.

Harmsen, A. F. (1997) Situational method engineering. University of Twente, Enschede, The Netherlands.

Hevner, A. R., March, S. T., Park, J. and Ram, S. (2004) Design science in information systems research. *MIS Quarterly* 28 (1), 75-105.

Hooker, J. N. (2004) Is design theory possible? *Journal of Information Technology Theory and Applications* 6 (2), 73-83.

Jacobson, I., Booch, G. and Rumbaugh, J. (1999) *The unified software development process.* Addison-Wesley, Reading, Mass.

March, S. T. and Smith, G. F. (1995) Design and natural science research on information technology. *Decision Support Systems* 15 (4), 251-266.

March, S. T. and Storey, V. C. (2008) Design science in the information systems discipline: An introduction to the special issue on deisgn science research. *MIS Quarterly* 32 (4), 725-730.

Miles, M. B. and Huberman, M. A. (1994) *Qualitative data analysis: An expanded sourcebook.* Sage, California.

Morris, C. W. (1970) *Foundations of the theory of signs.* University of Chicago Press, Chicago, IL.

Mylopoulos, J. (1992) Conceptual modelling and telos. In *Conceptual modelling, databases and case* (Loucopoulos, P. and Zicari, R., Eds), pp 49-68, Wiley.

Mylopoulos, J., Chung, L. and Yu, E. (1999) From object-oriented to goal-oriented requirements analysis. *Communications of the ACM* 42 (1), 31-37.

Niehaves, B. (2007) On epistemological diversity in design science: New vistas for a design-oriented is research? In *28th International Conference on Information Systems (ICIS 2007)*, pp 1-13, Montréal, Canada.

Nunamaker, J. F., Chen, M. and Purdin, T. D. M. (1990) Systems development in information systems research. *Journal of Management Information Systems* 7 (3), 89-106.
Orlikowski, W. and Iacono, C. (2001) Desperately seeking the 'it' in it research: A call to theorizing the it artifact. *Information Systems Research* 12 (2), 121-134.
Orlikowski, W. J. (1993) Case tools as organizational change: Investigating incremental and radical changes in systems development. *MIS Quarterly* 17 (3), 309-340.
Peffers, K., Tuunanen, T., Rothenberger, M. A. and Chatterjee, S. (2007) A design science research methodology for information systems research. *Journal of Management Information Systems* 24 (3), 45-77.
Ralyté, J. and Rolland, C. (2001) An assembly process model for method engineering. In *13th International Conference on Advanced Information Systems Engineering (CAiSE 2001)* (Dittrich, K. R. and Geppert, A. and Norrie, M. C., Eds), pp 267-283, Interlaken, Switzerland.
Ralyté, J., Rolland, C. and Deneckère, R. (2004) Towards a meta-tool for change-centric method engineering: A typology of generic operators. In *16th International Conference on Advanced Information Systems Engineering (CAiSE 2004)* (Persson, A. and Stirna, J., Eds), pp 202-218, Riga, Latvia.
Recker, J. (2008) Understanding process modelling grammar continuance: A study of the consequences of representational capabilities. *Faculty of Information Technology*, Queensland University of Technology, Brisbane, Australia.
Rosemann, M., Dreiling, A., Aalst, W. V. D. and Sadiq, W. (2008) From conceptual process models to running systems: A holistic approach for the configuration of enterprise system processes. . *Decision Support Systems* 45 (2), 189-207.
Rossi, M., Tolvanen, J.-P., Ramesh, B., Lyytinen, K. and Kaipala, J. (2000) Method rationale in method engineering. In *33rd Annual Hawaii International Conference on System Sciences (HICSS 2000)*, pp 1-10, Maui, HI.
Seidel, S. (2009) A theory of managing creativity-intensinve processes. *Doctoral Thesis, Chair for Information Systems and Information Management*, University of Muenster, Muenster, Germany.
Seidel, S., Rosemann, M. and Becker, J. (2008) How does creativity impact business processes? *16th European Conference on Information Systems 2008*, Galway, Ireland.
Shanks, G., Tansley, E., Nuredini, J., Tobin, D. and Weber, R. (2008) Representing part-whole relations in conceptual modelling: An empirical evaluation. *MIS Quarterly* 32 (3), 553-573.
Simon, H. A. (1996) *The sciences of the artificial.* The MIT Press, Cambridge, MA.
Strauss, A. L. and Corbin, J. (1998) *Basics of qualitative research. Techniques and procedures for developing grounded theory.* Sage, London.
Takeda, H., Veerkamp, P., Tomiyama, T. and Yoshikawa, H. (1990) Modeling design processes. *AI Magazine* 11 (4), 37-48.
Urquhart, C. (2001) An encounter with grounded theory: Tackling the practical and philosophical issues. In *Qualitative research in is: Issues and trends* (Publishing, I., Ed), Hershey, PA.
Venable, J. R. (2006) The role of theory and theorising in design science research. In *1st International Conference on Design Science Research in Information Systems and Technology (DESRIST 2006)* (Chatterjee, S. and Hevner, A. R., Eds), Caremont, CA.
Wallace, W. (1971) *The logic of science in sociology.* Aldine Atherton, Chicago, IL.
Walls, J., Widmeyer, G. and El Sawy, O. (1992) Building an information system design theory for vigilant eis. *Information Systems Research* 3 (1), 36-59.
Wand, Y., Monarchi, D. E., Parsons, J. and Woo, C. C. (1995) Theoretical foundations for conceptual modelling in information systems development. *Decision Support Systems* 15 (4), 285-304.
Wand, Y. and Weber, R. (1993) On the ontological expressiveness of information systems analysis and design grammars. *Journal of Information Systems* 3 (4), 217-237.
Weber, R. (2003) Still desperately seeking the it artifact. *MIS Quarterly* 27 (2), iii-xi.
Winter, R. (2008) Guest editorial: Design science research in europe. *European Journal of Information Systems* (17), 470-475.