2007

Context-based Modeling – Conceptualization of a Novel Modeling Approach and Application for the Design of Business Documents

Jörg Becker
European Research Center for Information Systems (ERCIS), becker@ercis.de

Christian Janiesch
European Research Center for Information Systems (ERCIS), janiesch@ercis.de

Daniel Pfeiffer
European Research Center for Information Systems (ERCIS), pfeiffer@ercis.de

143. Context-based Modeling – Conceptualization of a Novel Modeling Approach and Application for the Design of Business Documents

Jörg Becker
European Research Center
for Information Systems (ERCIS)
becker@ercis.de

Christian Janiesch
European Research Center
for Information Systems (ERCIS)
janiesch@ercis.de

Daniel Pfeiffer
European Research Center for Information Systems (ERCIS)
pfeiffer@ercis.de

Abstract

In this paper a novel reuse approach called context-based modeling is proposed and applied for the modeling of business documents. Business documents constitute mutual agreements, often legally binding between business partners. Already existing document standards reduce the efforts of implementing data exchange. However, the specific properties of an organization entail a need for adaptation. We show that available reuse approaches do not support this appropriately. Context-based modeling is proposed based on the reuse mechanisms aggregation, restriction, and specialization. Context-based modeling aims at both, minimal preparation of reuse combined with a high degree of guidance to create suitable models. The proposal is conceptually explored and practically applied to evaluate the feasibility and efficiency of the approach.

Keywords: Meta modeling, conceptual modeling, component-based development, reuse, business documents

Introduction

Business documents, which constitute mutual, often legally binding agreements, are exchanged between business partners (Glushko and McGrath 2005). Interfaces define the structure of these documents. Commonly, XML-based business document specifications are used (Janiesch and Thomas 2006). The definition of interfaces and exchanged documents is significantly dependent on the individual business processes of an organization. In general, it is not efficient to develop document specifications from scratch. It is rather sensible to reuse existing documents and interface definitions or parts thereof. For this purpose information models can be applied. Information models abstract from the actual implementation and thus provide potential for reuse (Wand and Weber 2002).

To support reuse of business documents, hierarchical modeling methods can be applied (Becker et al. 2007a; Janiesch 2007). The constructs of a modeling method should fit the application domain to be modeled (Guizzardi et al. 2002). Therefore, as XML documents have a hierarchical structure it is reasonable to use hierarchies also as constituent construct in the modeling method. Due to the structural similarities modeling methods that are based on hierarchies are a helpful tool to represent business documents.

Existing reuse approaches do not suffice for the specific needs of hierarchical modeling. Reuse approaches such as such as patterns, components, reference models, and views do not consider hierarchies as the main structural element of a model. They either require high preparation efforts in order to guide the reuse process or they are straightforward to apply but offer no or only little instructions on how to proceed. For the reuse of document

specifications an approach is necessary which can be employed easily and flexibly but simultaneously helps to find an appropriate solution based on existing knowledge.
The objective of this paper is to conceptualize and present context-based modeling as an approach to facilitate reuse of business documents. A context is commonly defined as the circumstances, conditions, and influence factors that act upon an object. As a reuse approach, a context connects the advantages of different existing mechanisms to allow for a more efficient modeling. Contexts aim at both, only marginal preparation combined with a high degree of guidance.

This paper proceeds as follows: In the next section different reuse approaches and their adaptation mechanism for conceptual modeling are discussed and compared. Their inherent limitations lead to the conceptualization of a new reuse approach called context-based modeling, which is presented in the following section. Subsequently, the application of contexts for the specification of business documents is described. The paper closes with a summary of the main results and an outlook to further research.

Reuse in Information Modeling

Overview

Reuse means to apply the experiences of other modelers or former projects to solve an actual problem (Wimmer and Wimmer 1992). It implies that an existing knowledge base is utilized to avoid starting from scratch.

For the reuse of knowledge within information models, different approaches have been developed: patterns (P) (Alexander 1977), components (CO) (Gupta and Prakash 2001; Szyperski et al. 2003), reference models (RM) (Becker et al. 2007b), and views (V) (Nuseibeh et al. 1996; Zachman 1987).

Reuse approaches apply different reuse mechanisms. The following mechanisms have been identified based on a literature analysis (Becker et al. 2006b; vom Brocke 2007): analogy construction (AC), aggregation (A), configuration (C), specialization (S), instantiation (I), and restriction (R).

In the following reuse approaches and reuse mechanism are explained in detail.

Reuse Approaches

Patterns (P): A pattern defines a template to solve a commonly occurring problem (Alexander 1977). It contains a problem description and the abstract structure of a possible solution. It is necessary to specialize the pattern and to fill the abstract solution with additional information in order to meet the specific conditions of the actual case (P/S). A pattern can be applied, when the issue at hand maps with the general problem specification in the pattern. Patterns are applied by analogy construction (P/AC). Examples are analysis patterns which contain the knowledge on how to appropriately represent a certain fact in systems analysis or requirements engineering (Fowler 1996). Patterns are also used to guide model-based design of software (Gamma et al. 2005).

Model components (CO): Components are items that can be aggregated to form a new model (Gupta and Prakash 2001; Szyperski et al. 2003) (CO/A). They have been derived from recurrently occurring elements within complex information models or they are formulated to reach compliance between the models and a certain standard. Model components provide a partial solution to a defined problem. Compared with patterns they are more concrete as they can be used without modification. They act as building blocks that can be assembled based on certain rules to achieve an intended solution. Process building blocks are an example for model components (Becker et al. 2006a).

Reference model (RM): A reference model is a robust yet flexible model which comprises universal information for a more than one situation (Becker et al. 2007b). Reference models contain information which is relevant for a class of modeling scenarios. The information within these models is applicable to several organizations in different domains. Reference models can be used as-is, but commonly, they are adapted to the specific conditions of a situation by applying reuse mechanisms. They represent common practices or best practices. They can also offer a normative suggestion to solve a certain problem. Compared with components, reference models do not just provide a small part of a solution but they are more comprehensive. Similar to patterns reference models normally also have to be adapted to meet the specific conditions of an organization. Examples for reference models are the Y-CIM model (Scheer 2002) or the Retail-H (Becker and Schütte 2004). Reference models are used by enterprise systems vendors to specify the functionality of their systems (Rosemann and van der Aalst 2007).

Views (V): Views act as a filter to restrict the solution of a problem based on a defined perspective (V/R). A view contains the knowledge of what information is relevant when a certain problem has to be solved (cf. e.g. Nuseibeh et al. 1996; Zachman 1987). Thus, a view does not provide any evidence about the solution or how to get closer to it but it structures a possible result. A view can define how detailed a solution is or what level of abstraction it refers to. A view does not reuse knowledge about the solution but rather defines what the relevant aspects for the solution are. Thus, views reduce the complexity of a solution based on existing experiences. A view implies that there is also an integrated perspective that consistently combines all the different views. We consider a view as part of a modeling method. Examples for this reuse approach are the data view, process view, function view, and organizational view in the Architecture of Integrated Information Systems (ARIS) (Scheer 2000).

These reuse approaches are rather complementary then competing. For example a reference model can be split up into components and later be aggregated. A reference model can also be part of a pattern. Reference models imply a top down approach. Components help to construct a solution bottom up. Patterns can, depending on their granularity, address both ways. Views structure the solution independently from the direction of reuse.

Reuse Mechanisms

Analogy Construction (AC): Conclusion by analogy implies the transfer of information from one subject to another. Patterns apply this mechanism, since they are also useful in domains they were not specifically constructed for. The application of a pattern requires a conclusion by analogy to establish a fit between the problem description in the pattern and the actual situation (P/AC). Also, reference models or their parts can be the basis of an analogy construction (RM/AC), e.g. as proposed by the ebXML initiative (Crawford 2003).

Aggregation (A): With the aggregation of model components a specific model is built by assembling independent parts (CO/A) (cf. e.g. Brinkkemper et al. 1999; Gupta and Prakash 2001). Interface descriptions of model components offer information on the possibility to combine or integrate the different components and their general compatibility (Leppänen 2007). There are also reference models that support aggregation (RM/A). These models are not available as monolithic blocks but rather as independent elements that can be assembled (vom Brocke 2007).

Configuration (C): Reference models can be designed as configurable artifacts (RM/C). They are provided with explicit configuration points, which specify model variants regarding purpose-specific characteristics (Becker et al. 2006b; Rosemann and van der Aalst 2007). Based on the specific values assigned to configuration parameters a reference model is projected into a company-specific model. Model elements are removed or modified depending on the parameters. The actual procedure of model projection can be automated based on the prior annotation of configuration parameters to the model.

Specialization (S): Through specialization a particular model is derived from a more general model by adopting, extending and/or partially modifying the more general model (Ralyté and Rolland 2001). Reference models which support specialization have a higher level of abstraction than their organization-specific models (RM/S). These more abstract reference models offer only a relatively low number of model elements. Patterns also use the specialization mechanism in order to transform the solution structure into a concrete solution (P/S). A high degree of specialization can also be regarded as free modification.

Instantiation (I): When the mechanism of instantiation is used, reference models must be annotated with placeholders (RM/I) (Becker et al. 2006b). The placeholders are added during the construction of the reference model. When a specific model is created, the placeholders are filled with valid occurrences with respect to the particular circumstances. Depending on the properties of the domain, numeric or alphanumeric attributes, distinct model elements, or even composed model clusters can be defined as placeholders.

Restriction (R): Restriction means to remove certain model element types (modeling method constructs) from a model and to rule out their use during the model construction. The restrictions apply based on a certain perspective of a modeler or a project. They are based on experiences of what model element types are relevant in a certain situation. Views use restrictions as part of a modeling method (V/R). Reference models can be modified based on restrictions (RM/R).

Table 1 maps the different reuse mechanisms to their corresponding reuse approaches.

Table 1: Mapping of Reuse Mechanisms to Reuse Approaches

	Pattern	Component	Reference Model	View
Analogy Construction	P/AC		RM/AC	
Aggregation		CO/A	RM/A	
Configuration			RM/C	
Specialization	P/S		RM/S	
Instantiation			RM/I	
Restriction			RM/R	V/R

Comparison of the Reuse Mechanisms

The comparison of the reuse mechanisms is based on a classification according to two dimensions: The first dimension is the degree of guidance; the second dimension is the degree of preparation necessary to apply the mechanisms.

The degree of preparation defines how much effort is necessary before a certain mechanism can be used. To be able to apply the mechanisms of configuration, rules must be defined and the model elements must be annotated according to the rules. This process is very time-consuming. The domains of valid values must be specified for an instantiation of each placeholder. The mechanism restriction requires the specification of model element types that are permitted or not allowed. An aggregation can define constraints which restrain the possible combinations of components but such rules are not obligatory. Specialization can rule out certain sorts of modification and allows the general adaptation of models. An analogy construction can always be applied and does not require any preparation.

The degree of guidance explains how much the modeler is instructed when a certain mechanism is used. The guidance of configuration is very high. When the parameters are filled with values the model can be automatically configured. Interaction with the user is only necessary to resolve possible conflicts. Restriction provides also a high guidance as model element can automatically be removed if their corresponding types do not belong to the view, which is applied to the model. Instantiation specifies the domain of possible values but gives no hints on what values to choose in a certain situation. The guidance of aggregations and specialization depends on whether any restrictions have been specified. It is to assume that an increased degree of guidance requires in the same increased amount of preparation so that no overall gain can be achieved. Analogy construction offers no instructions on how to proceed. In Figure 1 the different reuse mechanisms are arranged in a portfolio.

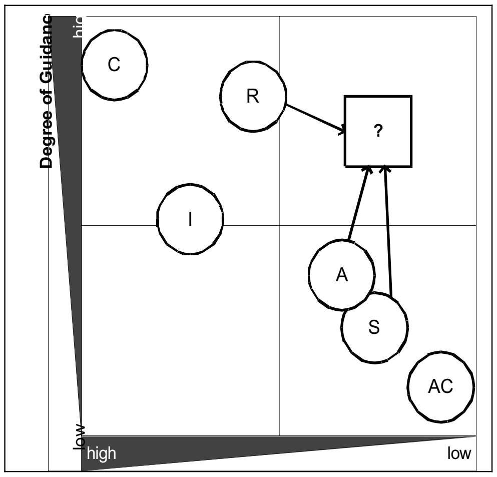


Figure 1: Portfolio of Reuse Mechanisms

The objective of this paper is to construct a reuse approach that can be applied to the domain of conceptual modeling without intensive preparation but that provides, nonetheless, a high degree of guidance. The mechanisms restriction, aggregation, and specialization are combined since they seem to be the most promising ones to be used with a varying degree of preparation. The result of this research is a novel combined approach that facilitates reuse in conceptual modeling.

Context-based Modeling

Conceptual Specification

In order to overcome the limitations of the isolated mechanisms to model reuse and to realize the benefits of a combined approach, we propose context-based modeling. A context of something is commonly perceived as the environment which surrounds it. The environment is a complex of circumstances, conditions, and influence factors. In the case of models, context describes or represents the environment in which the model or a component thereof has a certain meaning. Hence, we propose to disassemble modeling methods into contextualized components in order to view and construct models from different angles. We aim at utilizing a combination of the aforementioned mechanisms to achieve a plus in guidance with a minimum of adaptation preparation overhead. This overhead e.g. currently hinders a broad dissemination of more sophisticated configuration mechanisms such as those proposed by Becker et al. (2006b).

Practically any existing modeling method can be transformed into a context-based modeling method. Its feasibility, however, is strongly depending on the purpose of the modeling endeavor as well as the original design of the modeling method. As with all reuse mechanisms, certain types of modeling methods are more suitable for a range of problems than others.

A context-based modeling method comprises several contexts. A context acts as a filter as well as a container. In its role as filter, it applies the mechanism of restriction (*R*) to constrain the available constructs and their relations that can be used within an instance of this context. A context can also be related to (subordinate or sibling) contexts and, thus, applies the mechanism of aggregation (*A*). By the creation of these relations, model constructs from different contexts are linked. Specialization (*S*) can limit the display of the aggregated components. But more commonly it rather comes into place on a modeling level. If the aggregation and restriction of the model does not suffice individual needs, it is permissible to specialize the model by addition, deletion, and modification of the existing elements.

Figure 2 explains the assembly of a modeling method (*language definition*) based on contexts in ER notation. Each *language definition* consists of a number of *contexts*. Each *context* consists of several rules (*context rules*) which each describe the relation between two constructs (*object type definitions*). A construct can be for example a process function of the EPC (Scheer 2000) or a relationship type of the ERM (Chen 1976). We employ four mechanisms to describe the relations between constructs: *create definition (CD)*, *create definition explicit (CDe)*, *create occurrence (CO)*, *create occurrence explicit (COe)*. While *create definition* rules allow the modeler to create new objects, *create occurrence* rules enable the modeler to create a copy which references an existing object. *Create definition explicit* rules trigger that not only a new definition of an existing object is created but also the actual associations of the object's origin, i.e. the corresponding sub-model, are copied as definitions (and, thus, can be edited). *Create occurrence explicit* is essentially the same as *create occurrence* extended with the possibility to specialize the occurrence's sub model in a variable manner. Furthermore, a *context rule* contains the information from which context an object type definition may originate to restrict the aggregation.

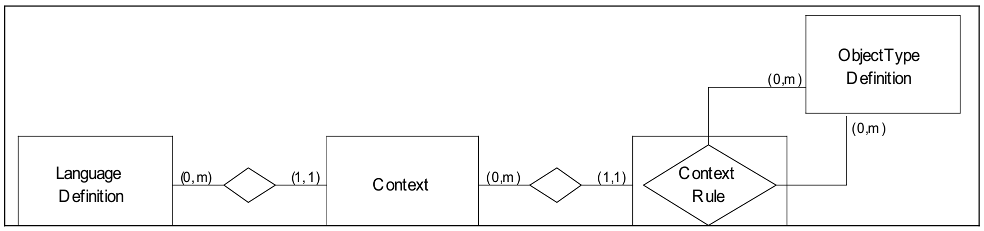


Figure 2: Associations of a *Context*

Basics of Context-based Modeling Method Design

Core of context-based modeling is to model in redundant as well as complementary contexts which each have distinct semantics. Views on a model, which create a consistent integrated perspective, are not in focus; contexts can, however, be employed in such a way.

Constructs from one context can be aggregated in other contexts. The semantics and syntactics that specify the relations of the different constructs to each other is unique to a context. For example, construct *A*, which is defined in context A', might also be defined in context B'. However, *C* might be a subordinate construct to *A* only in context A'. When linking constructs *A* of context A' with construct *D*, it is possible to restrict this to *A* originating only from A'. *A* from context B' cannot be reused in context D'. Cf. Figure 3 for an overview.

This leads to a distinction of different sorts of contexts. Again, cf. Figure 3 for the following discussion. While A' provides the basis for all other contexts and does not reuse constructs of other contexts, it acts as a *foundation context*. D' is a context with multiple relations, i.e. it reuses components from other contexts (A') as well as its components are reused (E'). It acts as a *link context*. Context E' in comparison is not intended for reuse but acts as an *aggregation context* which assembles components from other contexts as the highest node in a model. The relation of contexts D' and F' does not fit into this schema since they reuse each other. Thus, they are *recursive contexts*. Context B' is uncommon since it is not integrated with other contexts; it acts as a *single context*.

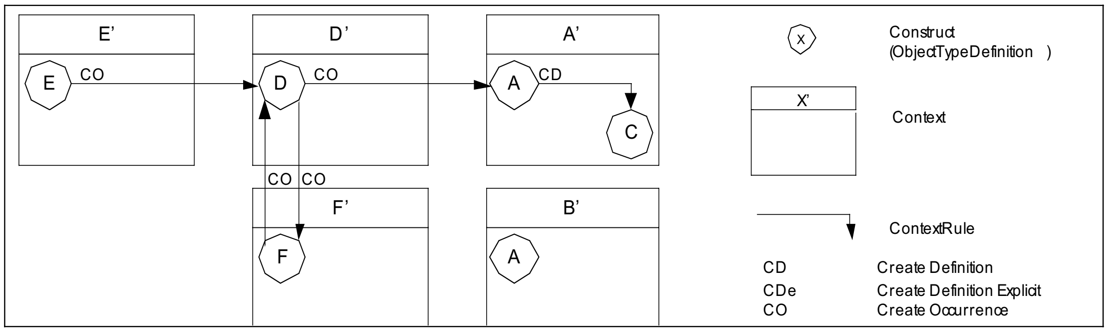


Figure 3: Exemplary Definition of *Context Rules* between *Object Type Definitions* in Different *Contexts*

Summarizing, the core of the context-based modeling approach is to integrate the mechanisms of restriction, aggregation and specialization. Aggregation is used in a very granular way since model components from within contexts are assembled and contexts are only linked implicitly. Restriction is employed to filter occurrences of objects from other contexts. Furthermore, the mechanism of specialization is incorporated since model component aggregation with explicit *context rules* allows for adaptations of the resulting model.

Software support is the basis for any comprehensive modeling project (Becker et al. 2007b). In an ongoing research effort, software, H2, has been developed that allows for the specification and management of context-based modeling methods. H2 contains both, method specifications and the actual models, to support careful adaptations of the meta models while modeling (Becker et al. 2007a). Consequently, the repository consists of two parts. The first part represents the functionality to create the method specifications (meta models) and the second part contains the actual models that are instances of a certain method (models). To limit the effort of the initial implementation, it was decided to constrain the tool to hierarchical modeling methods. A hierarchy is understood as a transitive, irreflexive, and asymmetric relationship of entities. It is represented as a connected directed acyclic graph with a designated initial (root-)node forming a tree structure. This entails that constructs within contexts are always subordinate or superordinate to each other. Furthermore, hierarchies represent an inherent concept of abstraction and have been found to be intuitively understandable for users.

To illustrate the application of the proposal, a comprehensive example will be given in the subsequent section. It is illustrated with models from the software.

Exemplary Application of Context-based Modeling for Business Documents

Scenario

One of the goals of modern business document standards is to avoid the confusion on how to treat data correctly in terms of structure, format, and meaning. Providing structure and format is currently achieved by utilizing XML as a markup language. Meta standards and code lists to homogenize meaning exist. However, they do not provide any concrete implementation, but only a way of standardizing business semantics. They provide components which can be (re-)used to structure documents. A prominent example is the Core Components Technical Specification (CCTS) (Crawford 2003). CCTS does not define any business documents. It is a framework on how to assemble documents and their data types as well as a methodology on how to create instantiations of the general business document. Several standards are proposed based on its conceptualization to prescribe a universal assembly for business documents. For an elaborate overview on business document standards (cf. e.g. Janiesch and Thomas 2006; Schroth et al. 2006).

In the following, a scenario is presented which motivates the need for the assembly of business documents. The concept of contexts is applied to design a method and to use it to create a suitable business document by reusing existing common components constituent to the design of business documents (Becker et al. 2006c; Janiesch 2007).

Consider a process including three parties: a customer, an agent, and a supplier. The customer uses the agent to find the best possible supplier and the agent places the customer order. The business document that is handed over from the customer to the agent and further to the supplier is called *order*. Both order documents do not have to look alike as the different parties involved have different information requirements. While the customers might specify only the item and the designated price in his order document, the agent might also include information on freight forwarding of the item or credit worthiness of the customer. The order is acknowledged with an *order response* document. Figure 4 sketches the scenario. In the following we focus on the *order* document.

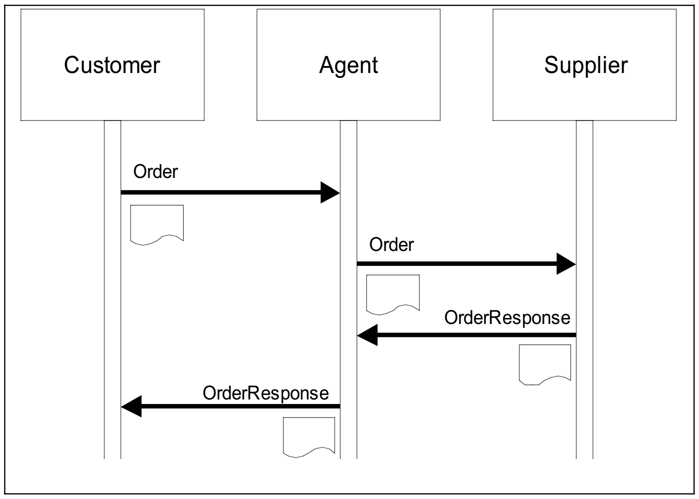


Figure 4: Purchasing Scenario with Three Parties

To exchange data, the business document *order* has to be specified. The Universal Business Language (UBL) (Bosak et al. 2006), which is based on the aforementioned Core Components Technical Specification, is chosen as an exchange format for the collaboration. Cf. Figure 5 for an overview of the CCTS meta model in the Unified Modeling Language (UML).

Core components are building blocks for semantically correct documents to exchange meaningful information documents. They can be detailed into core component types (*CCT*), basic core components (*BCC*), association core components (*ASCC*), and aggregate core components (*ACC*). *CCT* are of a data type and consist of a content component and one or more supplementary components (e.g. Amount, Identifier). Content components carry the actual value while supplementary components further define the content. *CCT* do not have business semantics. *BCC* have business semantics and provide a singular element of an *ACC* (e.g. StreetName in Address). An *ACC* is a collection of *BCCs* and can be part of other *ACCs* via an *ASCC* property (e.g. DeliveryAddress in Party). Since these associations in CCTS are of unique business semantics, an ASCC is needed. *ACCs* offer the highest level of aggregation. This definition of CCTS above is an excerpt only and does not contain all elements as specified within the specification.

UBL (Bosak et al. 2006) and the Open Applications Group Integration Specification (Open Applications Group Inc. 2006) are two major initiatives that aim at providing the uniform language for business documents exchange based on CCTS. SAP's Global Data Types (GDT) also use CCTS for harmonizing the business information of their Enterprise Systems (Stuhec 2004).

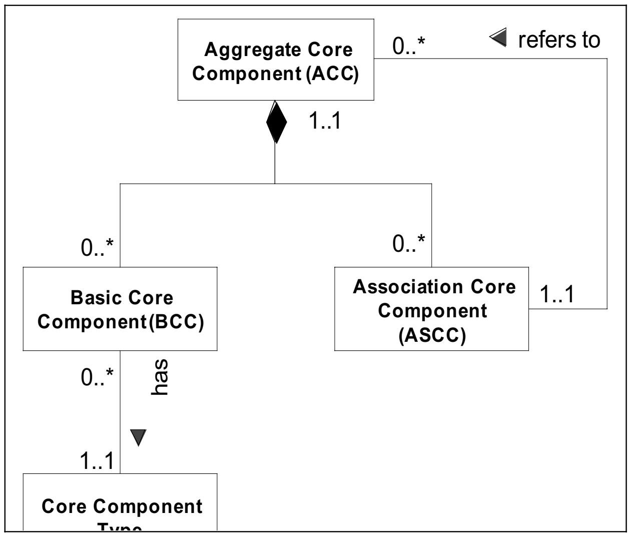

Aggregate Core Component (ACC)
0..*
refers to
1..1
0..*
0..*
Basic Core Component(BCC)
Association Core Component (ASCC)
1..1
0..*
has
1..1
Core Component

Figure 5: Excerpt of the CCTS Meta Model in UML Notation

Context-based Engineering of a Conceptual Modeling Method

Most major standards are currently offered as MS Word or MS Excel documents as well as XML schema (cf. e. g. Bosak et al. 2006; Open Applications Group Inc. 2006; Stuhec 2006). However, Word and Excel are only of descriptive nature and do not allow integrity constraints to be managed at a convenient level; XML schema is too technical oriented to conveniently work with on a higher level of abstraction. In order to specify and maintain document specifications, a proper modeling method is needed. UML is a possible candidate but does not provide the ease of use necessary to allow for fruitful discussions with decision makers and to manage an extensive amount of components. This leads to the need of a specialized, hierarchical modeling method that is based on common concepts such as abstraction and specialization, components as well as restriction on the creation of model elements (which might need governance processes). Thus, this is an adequate scenario for the proposed context-based modeling approach.

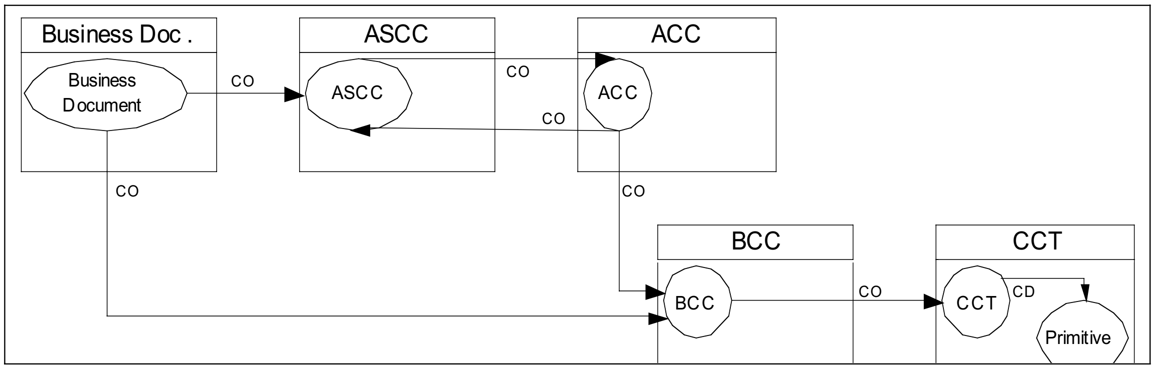
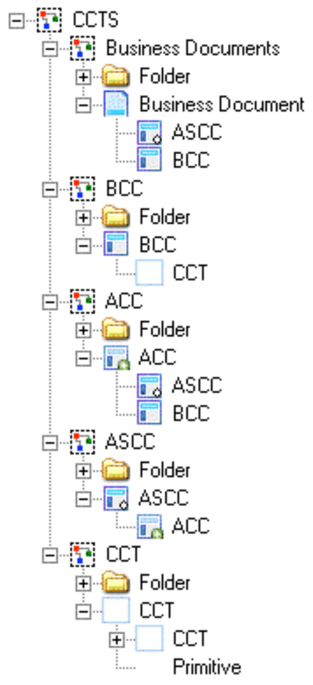


Figure 6: Assembly of the five CCTS contexts

The context-based meta model is based on the meta model of CCTS specified above in UML. The meta model consists of five contexts. Cf. Figure 6 for an additional overview of the relations between the constructs of the contexts.

CCT acts as the foundation context and provides all object types (CCT, Primitive) for itself. Primitives could have been created in an own context. Since they are not in focus here, they are modeled within *CCT*. Ultimately, this was a design decision. *BCC*, *ACC*, and *ASCC* are linking contexts, which link BCC to CCT, ACC to ASCC and BCC, and ASCC to ACC. In addition, the ASCC and ACC are *recursive contexts*. The *Business Documents* context acts as the aggregation context. It provides the highest level of abstraction and associates Business Documents with occurrences of ASCC and BBC. Syntactically, Business Documents are identical to ACC (Crawford 2003). However, to distinguish the semantics of a document from an aggregated component (which might e.g. exhibit different attributes), an own context was created. This separation of concerns resembles the restriction mechanism.

Cf. Figure 7 for an excerpt of the meta model and Figure 8 for a sample model assembly as it is documented with H2 (Becker et al. 2007a). On the left hand side of Figure 7 there is the graphical representation of an abridged CCTS meta model. On the right hand side, the context rules necessary to describe the relations are summarized in a table. Each context rule (*ID*) consists of two object types (*Object Type A + B*), the context (*Context*) and the rule type (*Type*). If *Object Type A* is empty (i.e. NULL), an object will be created as a root node in the respective context.



ID	Object Type A	Object Type B	Context	Type
0		Business Document	Business Documents	Create Definition
1	Business Document	BCC	Business Documents	Create Occurrence
2	Business Document	ASCC	Business Documents	Create Occurrence
3		BCC	BCC	Create Definition
4	BCC	CCT	BCC	Create Occurrence
5		ACC	ACC	Create Definition
6	ACC	ASCC	ACC	Create Occurrence
7	ACC	BCC	ACC	Create Occurrence
8		ASCC	ASCC	Create Definition
9	ASCC	ACC	ASCC	Create Occurrence
10		CCT	CCT	Create Definition
11	CCT	CCT	CCT	Create Occurrence
12	CCT	Primitive	CCT	Create Definition

Figure 7: Excerpt of the H2-based meta model for the CCTS approach

The specification of folders is not included in the table in Figure 7. Each folder is created by *create definition* rule and a context rule that associates the folder with the appropriate object type definition (e.g. CCT in context *CCT*).

Application of a Context-based Modeling Method

In this scenario, the *order* document of UBL 2.0 was regarded too large and too complicated to implement. Therefore, an own *custom order* was assembled reusing common components

(BCC, ASCC, ACC) of the UBL specification to provide a more simple document. The document specification procedure is depicted in Figure 8.

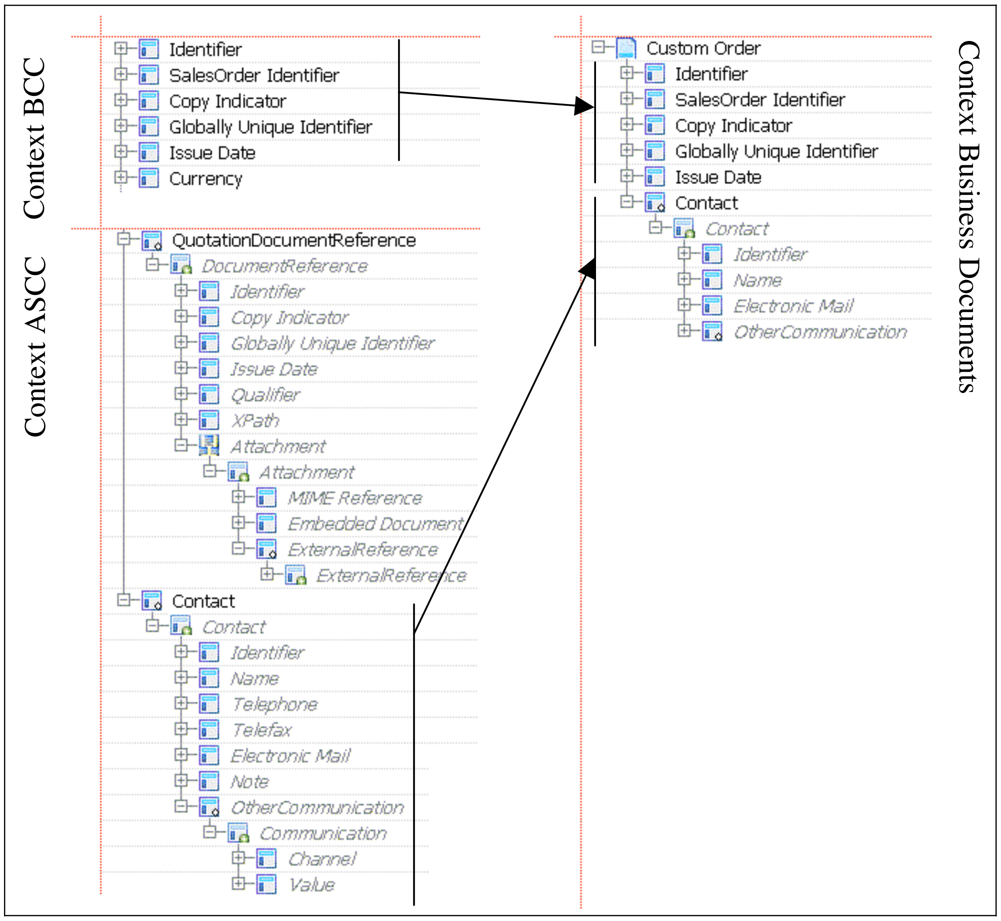


Figure 8: Assembly of sample business document *custom order* based on Figure 7 and UBL 2.0

The process is as follows: In the context *Business Documents* a new business document named *custom order* is created (*create definition*). Then the structure of the business document is assembled from components from other contexts (*ASCC* and *BCC*) since – as restricted in the method definition – only components defined outside the context can be used to assemble the document (*create occurrence*). Thus, several BCC are selected: *Identifier* to unambiguously identify the document, *Sales Order Identifier* to unambiguously identify the respective sales order, *Copy Indicator* to ensure that the document is the original and not a copy as well as two further BBC (*GUID* and *Issue Date*). In addition to that, an ASCC (*Contact*) is reused which itself reuses existing ACC (*Contact*) and BCC (e.g. *Identifier, Name, Telephone…*) and ASCC (*Other Communication*). Since this ASCC was also regarded as too complex it was further specialized to neglect unnecessary information such as *Telephone*, *Telefax*, and *Note*.

The representation of the document in a manageable tree view built from components eases the understanding of the structure of the document. Furthermore, it enables controlling options on all specifications to find obsolete components and identify major components. It ensures consistency that just cannot be provided by specification spanning multiple Excel and Word files. Note the shaded elements in Figure 8, they as well as their superordinate node, can only be changed in their respective contexts and are only linked via *create occurrence* to the root node *custom order*. They can, however, be omitted, specialized, or otherwise configured to a projection of a subset.

Since the approach is tool supported, an XML schema specification of the document can be exported and be used as input for concrete business document specifications.

Summary and Outlook

A variety of different document standards in combination with the specific requirements of a certain organization, call for reuse of existing design knowledge. In the course of this paper we showed that existing approaches do not support reuse in a way which is needed for the specification of business documents. Established reuse mechanisms are either lacking an appropriate guidance or are very time-consuming. To address this issue we proposed context-based modeling. This approach of reusing design knowledge has been exemplified within a case scenario.

Contexts integrate the mechanisms of aggregation, restriction, and specialization. They are offer adequate guidance on the reuse of models with a lesser amount of model preparation than current isolated adaptation approaches. The comprehensive design of business documents will become even more important in the future. Major enterprise system vendors have announced the introduction of solutions built upon the principles of a service-oriented architecture (SOA) (ORACLE Inc. 2006; SAP AG 2004). In this new world of SOA-based enterprise systems, processes are choreographed from a range of enterprise services (Pulier and Taylor 2005) which communicate with standardized business documents.

But contexts are not only applicable to design of business documents. They have proven to be useful for the description of management information systems (Becker et al. 2006d) as well as web information systems (Brelage 2006). Thus, we are confident that contexts can also be transferred to other areas and can act there as an innovative reuse mechanism. Subject of further research is to determine and evaluate other fields, which are relevant for contexts. Furthermore, it is to evaluate whether the existing four context rule types suffice for specifying more complex modeling methods. In addition to that, the development of the software needs to be extended. The working prototype produced promising results. With an extended interface the findings can be put on a firm footing and provide further evidence of the efficiency of the approach.

Acknowledgements

This research has partly been funded by the German Federal Ministry of Education and Research (BMBF) and the German Aerospace Center (DLR) with the research project "Internet Economy and Hybridity - Modelling of Hybrid Information Systems" as part of the EU research cluster "Internet Economy" (grant number 01 AK 704).

References

Alexander, C. A Pattern Language: Towns, Buildings, Constructions Oxford Univ. Press, New York, 1977.

Becker, J., Algermissen, L., Falk, T., Pfeiffer, D., and Fuchs, P. "Model Based Identification and Measurement of Reorganization Potential in Public Administrations – the PICTURE-Approach," 10th Pacific Asia Conference on Information Systems (PACIS), Kuala Lumpur, 2006a, pp. 860-875.

Becker, J., Delfmann, P., and Knackstedt, R. "Adaptive Reference Modeling: Integrating Configurative and Generic Adaptation Techniques for Information Models," Reference Modeling Conference (RefMod), Passau, 2006b.

Becker, J., Janiesch, C., and Dreiling, A. "A Framework for Interdependent Configuration of Enterprise Systems," Inaugural Workshop on Enterprise Systems Research in MIS. (Pre-)ICIS, Milwaukee, WI, 2006c.

Becker, J., Janiesch, C., Pfeiffer, D., and Seidel, S. "Evolutionary Method Engineering: Towards a Method for the Analysis and Conception of Management Information Systems," 12th Americas Conference on Information Systems (AMCIS), Acapulco, 2006d, pp. 3686-3697.

Becker, J., Janiesch, C., Seidel, S., and Brelage, C. "A Framework for Situational and Evolutionary Language Adaptation in Information Systems Development," in: Advances in Information System Development, G. Knapp, G. Wojtkowski, J. Zupancic and S. Wrycza (eds.), Springer, 2007a.

Becker, J., Kugeler, M., and Rosemann, M. Process Management: A Guide for the Design of Business Processes, (2nd ed.) Springer, Berlin, 2007b.

Becker, J., and Schütte, R. Handelsinformationssysteme, (2nd ed.) Redline Wirtschaft, Frankfurt am Main, 2004.

Bosak, J., McGrath, T., and Holman, G.K. (eds.) Universal Business Language v2.0: Committee Specification. OASIS, http://docs.oasis-open.org/ubl/prd3r1-UBL-2.0/UBL-2.0.html, 2006.

Brelage, C. "Web Information System Development: Conceptual Modelling of Navigation for Satisfying Information Needs," Münster, Berlin, 2006.

Brinkkemper, S., Saeki, M., and Harmsen, F. "Meta-modelling Based Assembly Techniques for Situational Method Engineering," Information Systems (24:3) 1999, pp. 209-228.

Chen, P.P.-S. "The Entity-Relationship Model. Toward a Unified View of Data," ACM Transactions on Database Systems (TODS) (1:1) 1976, pp. 9-36.

Crawford, C. (ed.) Core Components Technical Specification - Part 8 of the ebXML Framework. Version 2.01. UN/CEFACT, http://www.unece.org/cefact/ebxml/CCTS_V2-01_Final.pdf, 2003.

Fowler, M. Analysis Patterns: Reusable Object Models Addison-Wesley, Menlo Park, 1996.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. Design Patterns: Elements of Reusable Object-Oriented Software Addison-Wesley, Reading, 2005.

Glushko, R.J., and McGrath, T. Document Engineering: Analyzing and Designing Documents for Business Informatics and Web Services MIT Press, Cambridge, MA, 2005.

Guizzardi, G., Pires, L.F., and Sinderen, M.J.v. "On the role of Domain Ontologies in the design of Domain-Specific Visual Modeling Languages," 2nd Workshop on Domain-Specific Visual Languages, 17th ACM Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA 2002), Seattle, 2002.

Gupta, D., and Prakash, N. "Engineering Methods from Method Requirements Specifications," Requirements Engineering (6:3) 2001, pp. 135-160.

Janiesch, C. "Implementing Views on Business Semantics: Model-based Configuration of Business Documents," 15th European Conference on Information Systems (ECIS 2007), St. Gallen, 2007.

Janiesch, C., and Thomas, S.M. "Business Document Taxonomy: Comparison of the State-of-the-art and Recommendations for Future Applications," International Journal of Interoperability in Business Information Systems (IBIS) (2:2) 2006, pp. 59-78.

Leppänen, M. "Contextual Method Integration," in: Advances in Information System Development, G. Knapp, G. Wojtkowski, J. Zupancic and S. Wrycza (eds.), Springer, 2007.

Nuseibeh, B., Finkelstein, A., and Kramer, J. "Method Engineering for Multi-perspective Software Development," Information and Software Technology (38:4) 1996, pp. 267-274.

Open Applications Group Inc. (ed.) Open Applications Group Integration Specification (OAGIS) Release 9.0, http://www.openapplications.org/downloads/oagidownloads.htm, 2006.

ORACLE Inc. Oracle Fusion Strategy and Roadmap, http://www.oracle.com/applications/oracle-fusion-strategy-roadmap.html, 2006.

Pulier, E., and Taylor, H. Understanding Enterprise SOA Manning Publications, Greenwich, 2005.

Ralyté, J., and Rolland, C. "An Assembly Process Model for Method Engineering," 13th International Conference on Advanced Information Systems Engineering (CAiSE). Lecture Notes in Computer Science. Vol 2068, Interlaken, 2001, pp. 267-283.

Rosemann, M., and van der Aalst, W.M.P. "A Configurable Reference Modelling Language," Information Systems (32:1) 2007, pp. 1-23.

SAP AG Enterprise Services Architecture: An Introduction SAP Whitepaper, 2004.

Scheer, A.-W. ARIS: Business Process Modeling, (3rd ed.) Springer-Verlag, Berlin, 2000.

Scheer, A.-W. Business Process Engineering: Reference Models for Industrial Enterprises, (2nd ed.) Springer, Berlin et al., 2002.

Schroth, C., Janner, T., Schmidt, A., and Stuhec, G. "From EDI to UN/CEFACT: An Evolutionary Path Towards a Next Generation e-Business Framework," 5th International Conference on e-Business 2006 (NCEB), Bangkok, 2006.

Stuhec, G. "SAP GDTs Based on CCTS. https://www.sdn.sap.com/irj/servlet/prt/portal/prtroot/docs/library/uuid/b602d790-0201-0010-e3a8-9e4ddfc45d17," https://www.sdn.sap.com/irj/servlet/prt/portal/prtroot/docs/library/uuid/b602d790-0201-0010-e3a8-9e4ddfc45d17, Download: 2006-11-29.

Stuhec, G. "How to Solve the Business Standards Dilemma: CCTS Key Model Concepts. SAP Developer Network CCTS Article Series Vol. 2," https://www.sdn.sap.com/irj/servlet/prt/portal/prtroot/docs/library/uuid/1b873fc3-0901-0010-f7bc-9518e1aed0cf, Download: 2006-11-21.

Szyperski, C., Gruntz, D., and Murer, S. Component Software: Beyond Object-Oriented Programming, (2nd ed.) Addison-Wesley, London, 2003.

vom Brocke, J. "Design Principles for Reference Modelling – Reusing Information Models by Means of Aggregation, Specialisation, Instantiation, and Analogy," in: Reference Modeling for Business Systems Analysis, P. Fettke and P. Loos (eds.), Idea Group Publishing, Hershey, 2007, pp. 47-75.

Wand, Y., and Weber, R. "Information systems and conceptual modelling: A research agenda," Information Systems Research (13:4) 2002, pp. 363-378.

Wimmer, K., and Wimmer, N. "Conceptual modeling based on ontological principles," Knowledge Acquisition (4:4) 1992, pp. 387-406.

Zachman, J.A. "A Framework for Information Systems Architecture," IBM Systems Journal (26:3) 1987, pp. 277-293.