Association for Information Systems
AIS Electronic Library (AISeL)

ECIS 2007 Proceedings

European Conference on Information Systems (ECIS)

2007

Implementing Views on Business Semantics - Model-Based Configuration of Business Documents

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

IMPLEMENTING VIEWS ON BUSINESS SEMANTICS – MODEL-BASED CONFIGURATION OF BUSINESS DOCUMENTS

Janiesch, Christian, European Research Center for Information Systems, Leonardo-Campus 3, 48149 Münster, Germany, janiesch@ercis.de

Abstract

The efficacious exchange of business documents is vital to the successful collaboration of enterprises. Enterprise Systems (ES) played a vital part in standardizing these documents and the business processes involved. However, in a heterogeneous (service oriented) environment it is not necessarily the case that all partners share the same interfaces. Numerous standardization approaches have been proposed for different areas of application that intend to facilitate data exchange. Thus, the mapping of documents is still problematic. Recently, approaches to standardize document design evolved, the Core Component Technical Specification (CCTS) being the most current. However, it is not to assume that all standards and ES will integrate CCTS in the same manner and already deviations can be observed. Adaptations of the underlying meta model are inevitable. Furthermore, the instantiation of the actual documents as well as the management of the numerous components has to be supported by a proper method and tool. The aim of the paper is to present a concept and prototype which allows creating views on business documents to facilitate document design and variant management.

Keywords: Enterprise modeling and design, collaboration and interoperability, enterprise ontologies, configuration.

1 STATUS QUO OF BUSINESS DOCUMENT DESIGN

Enterprise Systems (ES) provide the technical basis for data exchange. They are typically understood as bundles of technology and business expertise that are used to support an organization's management and operations in a holistic way (Davenport, 1998, Ragowsky & Somers, 2002). ES are commercial off-the-shelf solutions by ES vendors and implemented and maintained in a considerable collaborative effort by vendor, customer, and consultants (Markus, 2004). Services are instruments to support informational and transactional business processes within ES. They commonly define their own interfaces for transaction processes. The unobstructed exchange of business documents with those processes is vital to the successful collaboration of two or more enterprises (Erl, 2005). Numerous standards have been proposed for different lines of business that intend to facilitate document exchange. Each has its own strengths and shortcomings depending on their specific purpose. In a service oriented architecture (SOA) environment it is desirable but not necessarily the case that all partners share the same interfaces. The mapping of these standards is problematic and takes at least several days to implement.

Approaches to structure business documents exist. One of the goals of modern exchange 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. A number of mostly industry-specific exchange standards exist, which have focused on static message definitions, e.g. RosettaNet, xCBL or CIDX. They do not exhibit a comprehensive degree of interoperability or flexibility with other open standards. In addition to that, application-specific exchange formats ex-

ist, which firstly address the needs of the application system. These proprietary formats are more or less inaccessible for the public. Only recently the adoption of open standards by ES can be observed.

EDIFACT is a mature standard for document exchange. It is widely used in Europe; its Northern American counterpart is ANSI ASC X12. Their structure is similar. As EDIFACT is not based on XML, it provides its own messaging structure as well as semantics. However, a structured composition of messages is not provided.

The ebXML Core Components Technical Specification (CCTS) (Crawford, 2003) provides the definition of data types for document exchange and, in addition to that, a concept on how to organize, i.e. aggregate and associate, data on a higher level to create business documents. Being a meta standard CCTS does not provide any concrete implementation but a way of standardizing business semantics, i.e. the document data's meaning. The Universal Business Language (UBL) (Bosak et al., 2006) and the Open Applications Group Integration Specification (OAGIS) are two approaches that aim at providing the uniform language for business document exchange based on CCTS. CCTS is the dominant approach underlying the design of most of the currently available open standards for business semantics as well as application-specific data models such as SAP's Global Data Types (Stuhec, 2006). For an elaborate overview on business document standards cf. e.g. (Janiesch & Thomas, 2006, Schroth et al., 2006).

It is not to assume that all standards and ES will integrate CCTS in the same manner. Deviations can already be observed (Stuhec, 2006). Thus, the instantiation of the actual document as well as the management of the considerable amount of components, both have to be supported by a proper method and tool. We propose that meta modeling (Marttiin et al., 1995, Rossi et al., 2004) provides the most efficient way of managing the diversity of models.

Thus, the research question for this paper is: How can the modeling of component-based documents be implemented and how can the management as well as the configuration of the documents be supported? The structure of the paper is as follows: Subsequent to this state-of-the-art of business document design, the underlying conceptualizations of the meta (meta) model are explored and a proposal is developed. An exemplary application concludes this research and provides a starting ground for further research.

2 IMPLEMENTING VIEWS ON BUSINESS SEMANTICS

2.1 Overview

The challenges of the conceptual specification of business documents are twofold on two layers. The first layer is the instance layer of the actual models. The second layer is the abstract layer of the corresponding modeling methods. Not only business documents have to be modeled and adapted but also their modeling method: i.e. the modeling of a generic purchase order as well as the modeling of the assembly method like CCTS-based UBL. The problem is twofold insofar as the management of the actual models as well as the management of the relations between the specified documents and their instantiations, both, has to be taken care of.

The H2 method is a tool-supported meta modeling method that explicitly provides assistance for the design of component-based, hierarchical methods (Becker et al., 2007b) as well as their configuration and management. As such, it assists the design and management of a CCTS-based modeling method (meta model) and subsequently their documents (modeling), such as those provided by UBL or OAGIS, as well as their respective adaptation (configuration). Cf. Figure 1 for an overview of the existing connections between the methods, which are explained in the following, and their relation to the actual instantiations. It is acknowledged that an abstract business document, i.e. a core component (CC), is not actually a method but an abstract model from which to instantiate the actual business information entities (BIE), i.e. contextualized business document specifications.

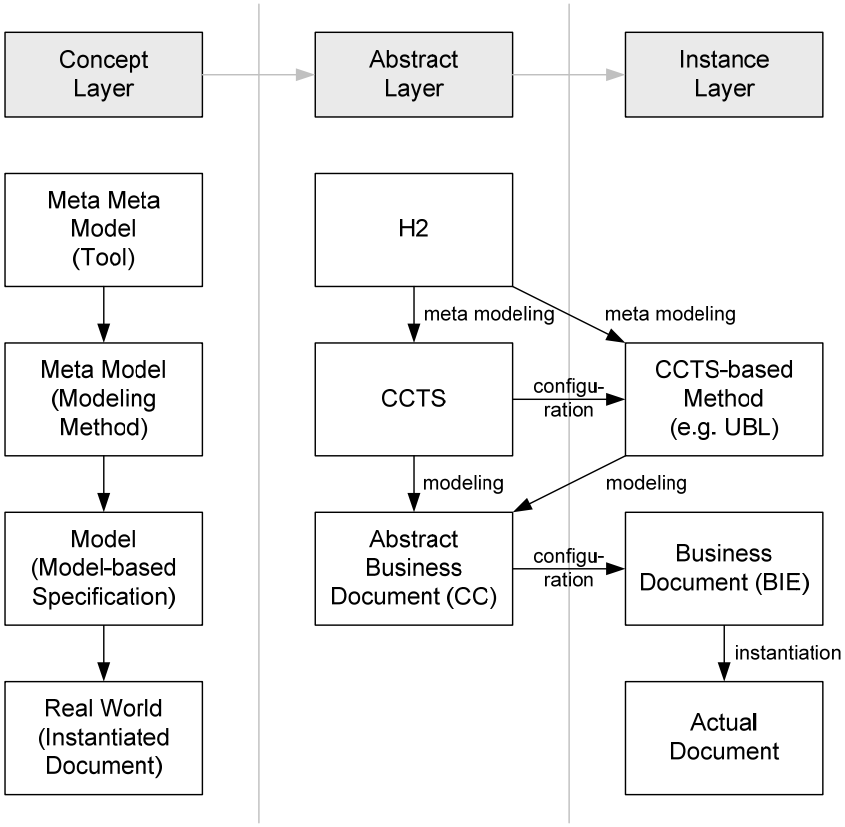

Figure 1. Layers of Conceptual Modeling for Business Documents.

Not only has a huge amount of models to be administered efficiently. Also the even bigger amount of projections of these models, which are configured to an actual business context, has to be maintained. The management of the relations between models and their actual instantiations is a challenging problem. While the first can be coped with utilizing generally accepted means of information retrieval (such as ordering, indexing, meta data search etc.), this research aims at providing a viable concept for the latter.

2.2 Specification of Business Semantics

CCTS does not define any business document. It is rather a framework on how to assemble documents and their data types as well as a methodology on how to create context-based instantiations of these general business documents. A number of external standards are prescribed as a basis for the design of the actual documents. The current version is 2.01 (Crawford, 2003), version 2.2 is a draft.

Core components are building blocks for semantically correct and meaningful business 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. Content components carry the actual value while supplementary components further define the content. According to the specification, 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 (e.g. Address) is a collection of BCC and can be part of other ACC via an ASCC property (e.g. DeliveryAddress in Party). Since these associations in CCTS are of unique business semantics, an ASCC is needed. ACC offer the highest level of aggregation; they are independent of a specific business context. In the context of this research the construct of Business Document is introduced to semantically distinguish minor ACC such as Address from major ACC such as PurchaseOrder, which are the root node of a document.

For a conceptual summary of CCTS (including Business Documents) in entity-relationship notation cf. Figure 2. For more detail cf. the original specification (Crawford, 2003). For an exemplary model cf. Figure 5 and Figure 6.

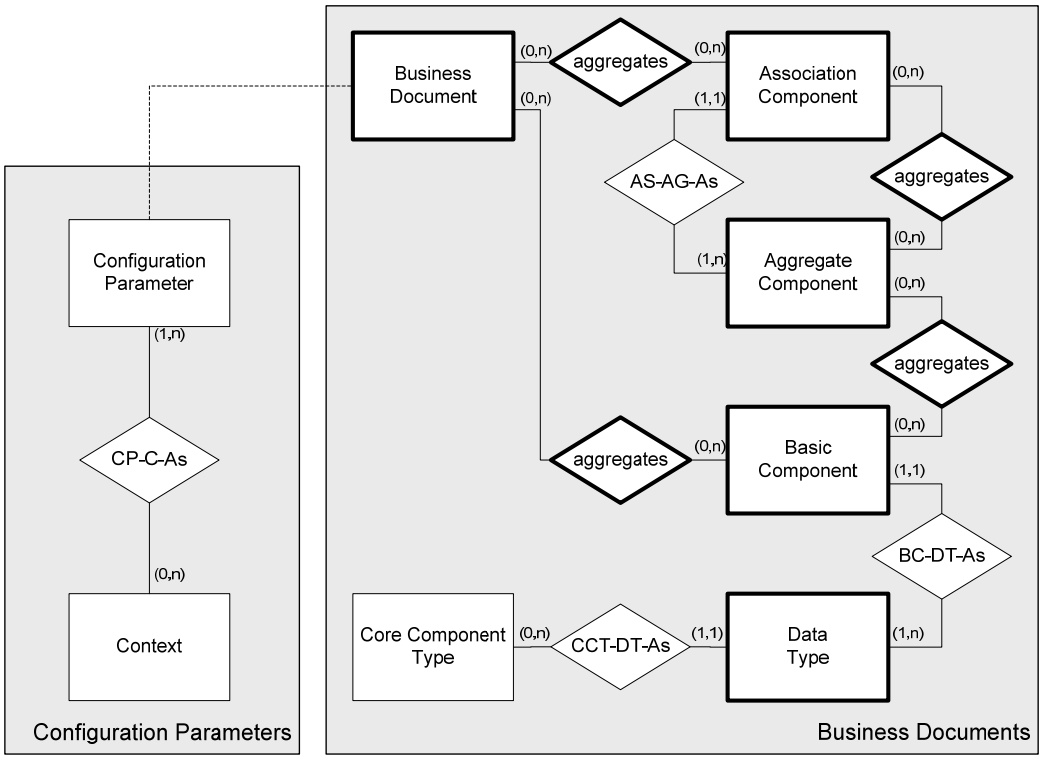

Figure 2. Meta Model of the CCTS Approach.

Based on the same data types, context-specific instantiation, so-called business information entities (BIE), can be created from abstract core components for deployment in the real world. Their relations to each other are similar to those of the business documents. According to a business context, a BIE is instantiated. A business context is specified by eight context drivers. However no procedure on how to apply the driver principle is proposed (Crawford, 2003). Current approaches define mainly BIE (cf. e.g. UBL, OAGIS, SAP GDT) and therefore neglect the fact that projections of the documents are needed to facilitate variant management.

We propose to configure business documents via *configuration parameters*. They may be based on *contexts* which comprise code lists for unambiguous identification or other standardized characteristics. Their connection to the business document meta model is visualized as a dashed line in Figure 2. This link is detailed in the following. All configurable element types are outlined with a bold border.

2.3 Configuration and Variant Handling

Numerous approaches to configuration exits (Becker et al., 2006, Dreiling et al., 2006, Frank & Lange, 2007, Janiesch et al., 2006a, Rosemann & van der Aalst, 2007, Soffer et al., 2005). Common modeling frameworks, however, only provide few options for variant handling (Delen et al., 2005, Scheer, 2000, Zachman, 1987). CCTS provides only indirect support for the configuration of its core components. Due to the distributed scenario of variant creation within the ebXML initiative, its approach is not applicable here. Rather than providing a structured means to project the variants of the models, it provides a constraint language to trace back the change that has been made to a variant; it is of inverse nature so to speak. This has certain drawbacks, the most striking ones being: No guidance on new projection is offered and each variant has to be stored redundantly (Crawford, 2003). Since the approach

proposed in this paper operates in the opposite direction, a method is needed that provides a means to derive business document variants from master models in a structured way.

Figure 3 details the dashed line of Figure 2 and its associated entity types, which are necessary for performing configurations. To provide more general evidence all entity types and relationship types from the business document meta model have been aggregated on a meta meta level as *model element*.

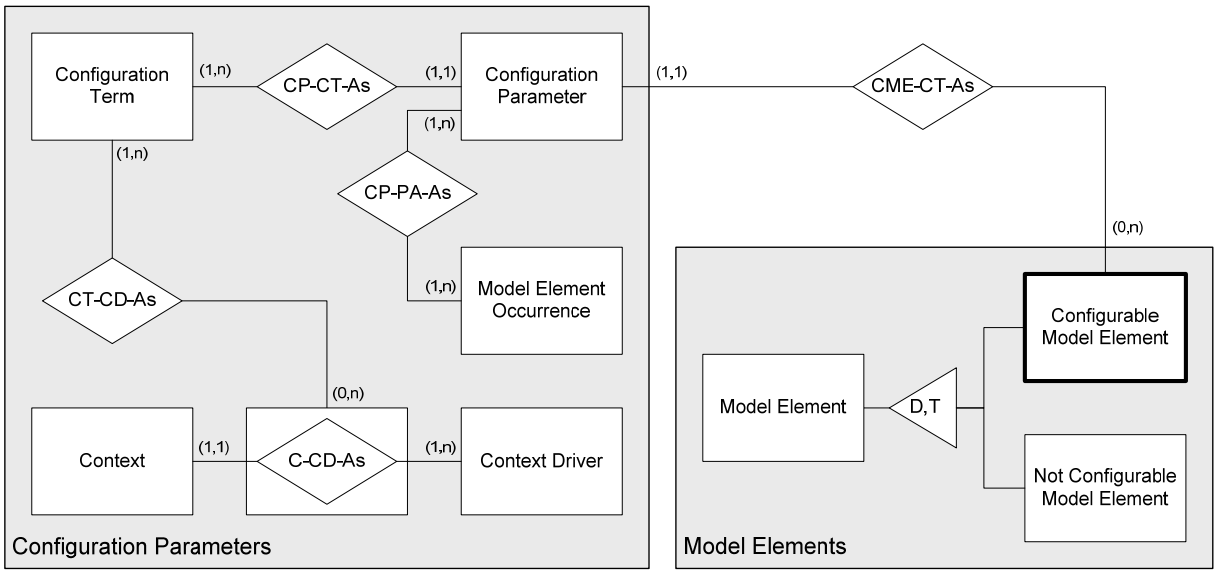

Figure 3. Configuration Parameters.

Not all model elements are configurable. Certain entity types and relationship types cannot be altered since they provide necessary structural information that is crucial for the consistency of the model. *Not configurable model elements* are, e.g., *AS-AG-As* or core component types. The latter provide the basis for all business documents and, thus, ensure their consistency. Changing this basis would create a model dialect that interferes with the consistent storage of the business documents. The former provides an unambiguous reference from association component to aggregate component; it is not possible to sever this relation since one cannot be used without the other.

Any *configurable model element* can be associated with multiple *configuration parameters*. A configuration parameter always consists of a *configuration term* (condition) and the associated *model element occurrence* (conclusion), i.e. the "when" and "how" of configuration. In other words, the configuration term provides the circumstances under which the model element occurrence is applied while the latter stores either instruction on how to configure the element or the actual set-up of the element.

The model element occurrence contains mostly element-specific information which can be as simple as `NOT`, meaning that the element configured is not needed in the model variant. More likely, restrictions or instructions for change will be provided: Restrictions can be posed on data types of components or the multiplicity of components towards their subcomponents. A data type is of a fixed format and can only be constrained in forms of its range; e.g., the allowed number of characters or the selection of entries from a code list. Furthermore, the multiplicity can be constrained; i.e. a subcomponent can be made mandatory rather than voluntary or a component's cardinality can be constrained down from `n` to `1`. The renaming of elements can have two reasons: First, synonymous terms might exist where the actual does not suit the application context. Second, due to the context-based configuration it might be necessary to enhance the name with a qualifier or property term to make it distinguishable. This should be done semi-automatically in order to provide a uniform naming. However, due to the complex nature of multiple contexts being applicable at a time standardized naming alone does not seem to be sufficient. There should be a distinction between the human-readable business name and a kind of dictionary entry that provides a unique name for the variant. Element naming proceeds similar-

ly as restriction. Each synonym is annotated with the applicable configuration term or none at all to imply general ambiguity.

For each model element a configuration term is provided that may consist of *contexts* from the distinct *context driver* categories analog to CCTS or other schemas. As stated above, a context is a specific circumstance under which a configuration parameter is applicable. To consistently define these circumstances, existing code lists can be used that provide an unambiguous operationalization. Contexts are grouped in distinct categories called context drivers. CCTS distinguishes eight categories (Crawford, 2003). But since not all of them are currently operationalizable with standardized code lists (e.g. a description of system capabilities) and some of them mostly occur in conjunction (most laws apply only and/or differently for different regions, e.g. sales tax), their comprehensiveness has to be evaluated and justified in practice first.

2.4 Modeling Repository for Configuration and Management of Business Documents

In an ongoing research effort software has been developed, which allows the specification and management of hierarchical structures. It contains both, method specifications and the actual models, to allow for careful adaptations of the meta models while modeling. Consequently, the repository consists of two parts. The first part represents a container to create the method specifications (meta model) and, therefore, serves as a meta meta model related to the real world. The second part represents the actual models that are instances of a certain method which in turn are the models related to the real world (cf. Figure 1). A basic excerpt from the repository's data model and the method H2 can be conferred in (Becker et al., 2007a) and (Becker et al., 2007b).

The most significant problem, which results from a multiplicity of perspective-specific, i.e. contextualized and tailor-made, models, is the need to manage possible redundancies inside the model itself. This leads to increased modeling and maintenance cost and the danger of inconsistencies within the model base. In order to enable an efficient multi-perspective document modeling, redundancies have to be overcome. A modeling methodology, which enables the user to avoid redundancies and to consider multiple perspectives within one master model, is called configurative modeling (Becker et al., 2006). The approach is based on the concept of model projection. A configurable information model that provides all relevant information for each perspective contains constraints that determine to which perspective each model element belongs. By this means, redundancies are avoided and, simultaneously, multi-perspective modeling is made possible. When a configuration is performed, each element is hidden that does not belong to the selected perspective. This implies that the core modeling is conducted using one master model and as such can only be performed by method experts that are properly trained.

The general configuration process is as follows: In the course of modeling, models can to be annotated with configuration parameters that allow for an automated configuration of models according to the specific requirements of an enterprise. The annotation itself is conducted with terms that declare the applicability of model elements (conclusion) according to certain parameters (condition).

While the H2 modeling layer comprises the business logic to create modeling methods and models, its extensions, the *term editor* and the *configuration customizer*, make configurative modeling possible. The term editor enables the model expert to define terms which are subsequently used within the configuration customizer to annotate model element via drag and drop creating a configuration parameter in the sense of Figure 3.

Contexts, i.e. the parameters or values within terms, are core to all configurations within H2. Concerning business documents, they can belong to one of the following three categories: context driver, perspective, and combined parameters. Each of these three is of the following structure: A parameter (e.g. *context driver*) contains dimensions (e.g. the context driver *geopolitical*); each dimension comprises dimension occurrences (e.g. *Germany*). Perspectives represent the roles of the actual users that log onto the tool to tailor their specific variants of the method. Combined configuration parameters are

necessary e.g. to represent intersections of dimension occurrences (sets) with configuration parameters. For an exemplary term cf. Figure 4. In the example two terms are generated to specify that a model element is neither relevant for web-based IS in Germany nor internet shops in France.

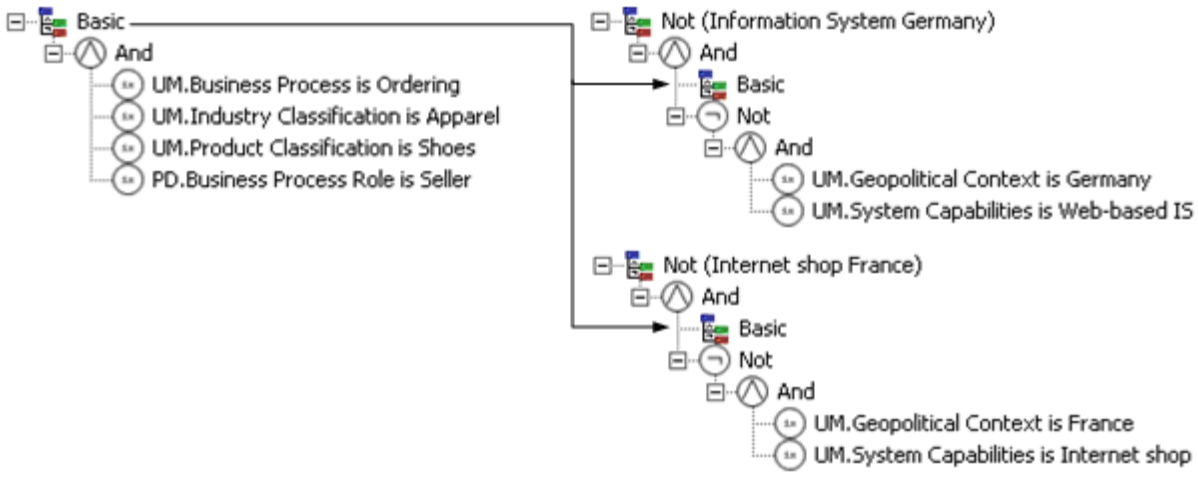

Figure 4. Configuration Term Definition in H2.

The entry and maintenance of configuration parameters is realized analog to the creation and maintenance of modeling methods with a hierarchical editor. Configuration parameters can be entered, altered, and assigned at any time during modeling. Reusing the common modeling component also entails that the structure of configuration parameters, i.e. their meta model, is editable as well.

Moreover, configuration parameters can have relations with each other. The selection of one configuration parameter can entail either that another parameter is selected as well (inclusion) or that another configuration may not be selected any more (exclusion). The software supports the user with ontologies that comprise this information.

3 EXEMPLARY APPLICATION

3.1 Scenario

Most major standards are currently offered as MS Word or MS Excel documents as well as XML schema. 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 actually work with on a higher level. In order to specify and maintain document specifications a proper modeling method is needed. The Unified Modeling Language (UML) is a usual suspect but does not provide the ease of use necessary to allow discussions with decision makers. It only allows convenient modeling on a type level, which requires the management's ability to abstract from instances. This is often not possible and a specialized hierarchical modeling method, which is based on common concepts such as abstraction and specialization as well as components, is required. It is necessary help to access very large models without disorienting the user.

The display of a business document in a manageable tree view built form components eases the understanding of the structure of the document. Cf. Figure 5 for a sample model of a business document as it is displayed in H2. For a legend of the constructs cf. the right hand side of Figure 6. Note the shaded elements, these components can only be changed in their respective views and are only linked to the root node. In the future, the tool enables controlling options to find obsolete components and identify major components as well as the export of configured documents to XML schema to allow the direct reuse as a business document specification.

Figure 5. Sample Business Document "OrderResponseSimple" of UBL 2.0 (Bosak et al., 2006).

3.2 Creating Views on Business Semantics

If, as part of a collaboration effort, an order needs to be sent, the following steps have to be taken to provide a view on the business document. With a requirements analysis the general set-up of the documents is compiled. In a first step the appropriate abstract business document (CC) from the repository needs to be retrieved. As a starting point, a coarse granular configuration can also take place via business scenarios, questionnaires etc. (Janiesch et al., 2006c). In a second step – the step conceptually introduced above – the actual business document is configured to the specific needs of the scenario. This includes the omission of elements that are not relevant to this task as well as their constraining.

The preparation of configuration in H2 is as described above: First, parameters are specified. From these parameters the ones required for this configuration project are selected. With the configuration customizer and the term editor these configuration parameters are assigned to model and meta model elements and their applicability is defined. The business document is now configurable. The configuration itself is checkbox-based and enables the user to choose the desired contexts manually. He is displayed a list of all parameters values, i.e. contexts, which are relevant for the model at hand (and

can be further constraint by his perspective). All configuration terms of the master model are evaluated against his selection as it determines the variant's final specification.

Thus, the configuration constrains the master model and creates a view on the business document. A subset of the document is the configuration result. Cf. Figure 6 for a minimal view on the "Order" document presented in Figure 5. Most BCC and ACC have been omitted as they are not mandatory to the design of a UBL order business document. It would also have been possible to create views on the meta model as well and omit meta model elements such as the data type for a high-level view or aggregate components for a simulation of a non-component approach. For reasons of clarity, only the configuration of the model is presented, but method configuration works in an analog way. Furthermore, configuration was limited to the omission of elements; among other cardinalities could have been constrained or synonymous terms could haven been substituted.

Figure 6. Configured Sample Business Document "OrderResponseSimple" from Figure 5.

The analysis of one of the simplest UBL 2.0 business documents, *OrderResponseSimple* (Bosak et al., 2006), reveals the possible impact of attaching configuration parameters at business documents and their components. Excluding recursive structures in the business document (i.e. *AgentParty*) OrderResponseSimple includes numerous components more than once, most strikingly *Address* occurring 36 times. If only this component was made configurable, it would already greatly assist variant handling. Table 1 is mainly of informative nature, there are more repetitive structures within the document, which provide even further justification by economies of scale for document configuration.

Abstract Component	#		**Instantiated Component**	#
Attachment	2	=	Document/ Signatory Attachment	2
Contact	6	=	Contact	6
DateTime	9	=	IssueDate	5
			IssueTime	4
Document Reference	3	=	Document Reference	3
Unique Identifier	5	=	GUID	3
			URI	2
Party	6	=	CustomerParty	3
			SupplierParty	2
			Party	1

additionally each Party consists of	→	Address	6 (6 x 6 = 36)
		Contact/ Person	2 (6 x 2 = 12)
		(Agent)Party	1 (6 x 1 = 6)

Table 1. Quantitative Analysis of "OrderResponseSimple".

This entails on a broader scale that making only some of the more important components configurable already has significant impact on the overall performance of the configuration. The specification of the regular *OrderResponse* about doubles the numbers of Table 1.

Furthermore, of the more than 100 basic components – this time even counting each aggregate component like *Address* only once – only five basic components are mandatory: An *OrderResponseSimple* can consist only of an *ID, IssueDate, AcceptedIndicator, OrderReference, Buyer_CustomerParty, Seller_SupplierParty* whilst the parties themselves can be empty and only *ID* of *OrderReference* is mandatory. This unveils huge potential for providing decision support on when (and how) to configure the optional elements to provide a suitable variant. It also points out the need for a structured way to maintain manageability of the diverse results. Moreover, as well the interdependencies with other business documents (such as Order) can be automated as the interdependencies with the respective business processes (Janiesch et al., 2006a, Janiesch et al., 2006b).

3.3 Comparison of Benefits and Drawbacks

To further justify the conceptual "overhead" proposed in the preceding sections, the approach's benefits have to be valued and contrasted with the drawbacks and limitations. Since no quantitative empirical data on this subject exists apart from the above, benefits and drawbacks can only be discussed argumentatively. While the example only hints at the benefits and mainly tries to illustrate the procedure, this section evaluates the overall approach. Cf. Table 2 for an overview of the benefits.

Top-down specification	Starting with a business problem, the appropriate models can be selected and subsequently configured in a consistent procedure from top to bottom. This procedure prohibits the proliferation of business document views, which are created based on a perceived need during implementation.
Consistent storage	The approach ensures implementation-independent storage of the conceptual specification. Furthermore, only master models and configuration parameters are stored to avoid the storage of countless variants that might contradict each other. Since only one master model exists per business document or process, all variants are derived therefrom. To ensure consistency over time all relevant objects are versioned.
Model-driven design	The complete process is model-based and abstracts from implementation issues that unnecessarily complicate things early in development. Model-based development enables faster comprehension of the problem at hand and, thus, facilitates faster and easier adjustment to the task and therefore faster produces results. Furthermore, a model-driven architecture approach cuts maintenance costs considerably according to OMG (Soley, 2005).
Maintainable model growth	Since only a reasonable number of business documents and processes exist, but a countless number of specific requirements surface, variant growth is inevitable. Growth occurs in terms of model size as well as number. By maintaining one versioned basic template of a master model from which variants are derived, variants can be controlled and their sheer number does not impair the overview.
Reuse	Inherent in the concept described above is component-orientation and reuse thereof.
Faster implementation	Due to faster adjustment to the task and a structured top-down approach, which allows the adaptation of master models to instantiated variants, the specification and implementation time can be shortened. Furthermore, it can be worked with by developers whose expertise does not include programming skills in every detail.
Further applications	The configuration mechanism can be used to simplify the display of the models by defining perspectives for distinct user groups so that only relevant parts of the repository are displayed.

Table 2. Benefits of Model-Driven Business Document Configuration.

Even though the approach offers many benefits, certain drawbacks and limitations are inevitable that have to be considered and weighed against the advantages. Cf. Table 3 for an overview of the limitations.

Configuration parameters	Full configuration of any master model involves a lot of parameterization. This entails an enormous initial effort on what to configure in which way. Detailed requirements will commonly arise only during the actual implementation and not beforehand.
Complex methodology	The maintenance of configuration parameters is complex. While high level configuration is fairly straight forward, the inherent complexity of the necessary granular configuration of business documents is rather high. Instead of managing countless variants a huge number of configuration parameters have to be maintained. It has to be ensured that the administration is economical. An ontology can support ensuring the consistency of model variant.
Implementation delays	While the aim is to shorten the general time of implementation, longer delays in implementation can be expected when new elements are introduced that are subject to an approval process.
Specific requirements	All configuration efforts do not suffice if the problem at hand is so product-specific or project-specific that it is not in any form transformable to a more abstract form. This entails in reverse that very specific requirements cannot be derived from the repository without (extensive) free modification so that the benefits of the approach cannot be realized.

Table 3. Drawbacks and Limitations of Model-Driven Business Document Configuration.

4 CONCLUSION

The design of standards for data exchange is influenced by other intermediate standards, which facilitate communication and standardization of business documents. The first notable layer is XML as a general exchange language and more recently CCTS is implemented as a method of business semantics standardization. The acceptance of one general business language is still due but might be the long-term solution to standardizing the building block concept proposed by CCTS.

However, if the specification of business documents (in the sense of BIE) is not restricted, these intermediate concepts are of no use since for any given problem a new document specification is assembled from scratch. This might be eased by the reuse of components that offer certain semantics, but ultimately leads to the same problems that currently exist. Proper variant management is an important issue to prevent this business document proliferation.

Implementing views on business semantics provides a promising concept for the management of variants of business documents especially in SOA environments. To allow reuse and adaptation even within models, a component-based approach is preferable as the configuration of certain central components has major impact on the whole configuration performance. Through context-based configuration mechanisms, the means of modifications are flexible and yet restrictive. Business documents can be configured to very diverse requirements but only in a predefined and traceable way. This controlled flexibility offers a feasible concept for business document variant management and other similar problems.

References

Becker, J., Delfmann, P. and Knackstedt, R. (2006). Adaptive Reference Modeling: Integrating Configurative and Generic Adaptation Techniques for Information Models. In Proceedings of the Reference Modeling Conference (RefMod2006), Passau.

Becker, J., Janiesch, C. and Pfeiffer, D. (2007a). Towards more Reuse in Conceptual Modeling: A Combined Approach using Contexts. In Proceedings of the 19th International Conference on Advanced Information Systems Engineering (CAiSE 2007) Forum, Trondheim.

Becker, J., Janiesch, C., Seidel, S. and Brelage, C. (2007b). A Framework for Situational and Evolutionary Language Adaptation in Information Systems Development. In Proceedings of the 15th International Conference on Information Systems Development (ISD 2006), Budapest.

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

Crawford, C. (2003). Core Components Technical Specification - Part 8 of the ebXML Framework. UN/CEFACT, http://www.unece.org/cefact/ebxml/CCTS_V2-01_Final.pdf.
Davenport, T. (1998). Putting the Enterprise into the Enterprise System. Harvard Business Review 76 (4), pp. 121-131.
Delen, D., Dalal, N. P. and Benjamin, P. C. (2005). Integrated Modeling - The Key to Holistic Understanding of the Enterprise. Communications of the ACM, 48 (4), pp. 107-112.
Dreiling, A., Rosemann, M., van der Aalst, W., Heuser, L. and Schulz, K. (2006). Model-Based Software Configuration: Patterns and Languages. European Journal of Information Systems, 15 (6), pp. 583-600.
Erl, T. (2005). Service-oriented Architecture - Concepts, Technology, and Design. Prentice Hall, Englewood Cliffs, NJ.
Frank, U. and Lange, C. (2007). E-MEMO: A Method to Support the Development of Customized Electronic Commerce Systems. Information Systems and E-Business Management, 5 (2), pp. 93-116.
Janiesch, C., Dreiling, A., Greiner, U. and Lippe, S. (2006a). Configuring Business Components: An Integrated Approach to Enterprise Systems Collaboration. In Proceedings of the 2006 IEEE International Conference on e-Business Engineering (ICEBE 2006) (Chung, J.-Y. Ed.), pp. 516-521, Shanghai.
Janiesch, C., Dreiling, A., Greiner, U. and Lippe, S. (2006b). Integrated Configuration of Enterprise Systems for Interoperability - Towards Process Model and Business Document Specification Alignment. In Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2006) (Hung, P. C. K. Ed.), pp. 445-448, Hong Kong.
Janiesch, C., Dreiling, A. and Seidel, S. (2006c). Document Variant Management: Facilitating Enterprise System Definition, Configuration, and Interoperability. In Proceedings of the 17th Australasian Conference on Information Systems (ACIS 2006), Adelaide.
Janiesch, C. and Thomas, S. M. (2006). 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), pp. 59-78.
Markus, M. L. (2004). Technochange Management: Using it to Drive Organizational Change. Journal of Information Technology, 19 (1), pp. 4-20.
Marttiin, P., Lyytinen, K., Rossi, M., Tahvanainen, V.-P., Smolander, K. and Tolvanen, J.-P. (1995). Modeling Requirements for Future CASE: Modeling Issues and Architectural Consideration. Information Resources Management Journal, 8 (1), pp. 15-25.
Ragowsky, A. and Somers, T. M. (2002). Special Section: Enterprise Resource Planning. Journal of Management Information Systems, 19 (1), p. 11.
Rosemann, M. and van der Aalst, W. M. P. (2007). A Configurable Reference Modelling Language. Information Systems, 32 (1), pp. 1-23.
Rossi, M., Ramesh, B., Lyytinen, K. and Tolvanen, J.-P. (2004). Managing Evolutionary Method Engineering by Method Rationale. Journal of the Association for Information Systems, 5 (9), pp. 356-391.
Scheer, A.-W. (2000). ARIS: Business Process Modeling. 3rd Edition, Springer-Verlag, Berlin.
Schroth, C., Janner, T., Schmidt, A. and Stuhec, G. (2006). From EDI to UN/CEFACT: An Evolutionary Path Towards a Next Generation e-Business Framework. In Proceedings of the 5th International Conference on e-Business 2006 (NCEB 2006), Bangkok.
Soffer, P., Golany, B. and Dori, D. (2005). ERP Modeling - A Comprehensive Approach. Information Systems Frontiers, 28 (6), pp. 673-690.
Soley, R. M. (2005). The Model Driven (R)evolution. In Keynote at 7th International Conference on Enterprise Information Systems (ICEIS 2005), Miami, FL, USA.
Stuhec, G. (2006). How to Solve the Business Standards Dilemma: The CCTS based Core Data Types. SAP Developer Network CCTS Article Series. Downloaded from https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/500db5c9-0e01-0010-81aa-d73cdd30df9a on 2006-11-21.
Zachman, J. A. (1987). A Framework for Information Systems Architecture. IBM Systems Journal, 26 (3), pp. 277-293.