Conceptual Business Document Modeling using UN/CEFACT’s Core Components

Philipp Liegl

Institute of Software Technology and Interactive Systems
Vienna University of Technology,
Favoritenstrasse 9-11/188, A-1040 Vienna,
Email: liegl@big.tuwien.ac.at

Abstract

Before two businesses can engage in a business-to-business process an agreement about the process execution order and the business documents exchanged in the collaborative process must be found. Although several initiatives from different industries have started standardization initiatives for business documents a set of shortcomings still remain: (1) the different standards do not have a common semantic basis causing inter-operability problems between them and they (2) try to include every possible element any industry might need into the business document standard. (3) Moreover, most of the standards are transfer specific and (4) do not provide a conceptual representation mechanism. In this article a new concept for the standardization of business documents called UN/CEFACT’s Core Components Technical Specification is presented which solves these shortcomings. Using Core Components the business document modeler can unambiguously define documents with a common semantic basis on a conceptual level. In order to allow for a better integration into UML modeling tools we introduce the UML Profile for Core Components. With the UML based components model and an XML schema generator the modeler can derive XML schema artifacts from the conceptual model.

Keywords: Business Document Modeling, Business Document Meta Modeling, UN/CEFACT’s Modeling Methodology

1 Introduction

If two businesses want to get involved in an automated B2B process they first have to agree upon a common choreography uniquely defining the exchange order of the different business documents. Several approaches for the standardization of a business choreography exist nowadays e.g. (TMG 2006), (Jung et al. 2004) or (Rinderle et al. 2006). While a process choreography describes the exchange order of business documents in detail, little to nothing is said about the harmonization of business documents which are being exchanged. One of the best known approaches for the standardization of exchanged data is UN/EDIFACT (Berge 1994) maintained by the United Nations Center for Trade Facilitation and Electronic Business (UN/CEFACT). The UN/EDIFACT standard provides a set of syntax rules used to structure business document data. The document format uses designated symbols and letter codes as delimiters between the different data fields.

Since XML was introduced in 1996 (W3C 2006) its popularity has constantly increased due to its versatility, flexibility and easy applicability. An additional boost has been brought by the introduction of Web Services and their related technologies such as WSDL (W3C 2007b), SOAP (W3C 2007a) and UDDI (OASIS 2007). In particular in the context of Web Services the clear and precise definition of a business document is of importance. Usually interfaces defined by WSDL import the appropriate XML schema defining the type of business document the interface accepts.

Given the popularity of XML as the representation format of choice for data, several initiatives have been started in order to standardize exchanged data using XML. An overview of different XML based standards for describing data and business documents is given by (Li 2000). However, the transition from a delimiter based approach such as UN/EDIFACT to an XML based solution did not solve the interoperability problems between business documents. We address the following shortcomings in regard to business document standardization:

(1) Standard incompatibilities. Due to the multiple initiatives which have been started several XML based business document representation mechanisms exist which are competing against each other, hence resulting in large incompatibilities between the different standards.

(2) All-in-one approach. Furthermore a lot of standards aim at the integration of every possible element into a standardized business document, resulting in a strong document overhead. E.g. a cross industry invoice which should be applicable in any industry context has to include every possible element any of the different industries might need. Whereas for instance number of nights per person is critical in a tourism context this attribute is rather unlikely to be needed in an oil industry context. However, in order to be cross-industry compatible every possible element has to be included in the standardized invoice. A partial solution is given by so called message implementation guidelines which cut down a standard to a set of agreed elements which are used between two business partners. This however results in a multitude of message implementation guidelines undermining the concept of a holistic business document standardization.

(3) Transfer syntax specific definition. Standards such as UN/EDIFACT or XML based solutions for business documents are tightly bound to the implementation syntax. Often the document semantics are defined on the logical level (e.g. XML schema) instead of being defined on an higher, conceptual level. Changes in the transfer syntax therefore result in re-engineering tasks for the standard, making it inflexible to future adaptations.
(4) Conceptual document description. Business document standards mostly lack a unique conceptual description model but merely focus on implementation details. Whereas this approach is sufficient for implementation it is hard to communicate a logical level model such as XML schema between different modelers. A standardized representation mechanism for the communication of business document concepts is needed.

Knowing these limitations UN/CEFACT started the development of the so called Core Components Technical Specification (TMG 2003). The idea is to develop a common ontological base of re-usable building blocks for business documents. Using these building blocks, a shared library is built from which modelers can retrieve artifacts in order to assemble a business document.

The development of the core components standard started in the late nineties as part of the the ebXML (OASIS 2001) initiative. The main goal of ebXML, was to provide a framework allowing potential business partners to engage in B2B processes in an interoperable, secure and consistent manner. One part of the ebXML technology stack were so called components, used to uniquely define the exchanged data between two enterprises. UN/CEFACT’s Technologies and Methodologies group, of which we are members of, continued the development of core components and today the standard is known as the Core Components Technical Specification (CCTS). The most recent version of the standard is 2.01 (TMG 2003) with the development of version 3.0 (TMG 2008) currently going on.

In this paper we present the UN/CEFACT Core Component Technical Specification and a UML profile which is build on the technical specification. It will be shown how core components can help to overcome the four limitations mentioned above and how XML schema specifications can automatically be derived from a core component model. The remainder of this article is structured as follows: section 2 gives an overview about related work in the field of conceptual XML schema modeling and section 3 introduces the core components approach. The UML Profile for Core Components is outlined in section 4 and an example is given in section 5. The derivation of XML schema artifacts from core components is shown in section 6 and section 7 concludes the paper and gives an overview about future work.

2 Background - Related Work

A general introduction into business document modeling is given by (Glushko & McGrath 2005). Glushko and McGrath provide a thorough and holistic overview about current approaches for business document modeling, document model interoperability and integration into business processes.

The conceptual modeling of data has existed for a while and forms an integral part of data engineering. One of the most important methodologies for data modeling is the so called entity relationship model (Chen 1976) used to design a relational database model. The entity relationship model (ER) provides its own modeling methodology consisting of entities, attributes belonging to entities and relationships between the different entities. A database modeler uses the entity relationship model to derive the appropriate data definition language (DDL) artifacts for creating the database model. The main goal of a database is to reliably store and retrieve information from it. If a hierarchical business document is stored in a database it is first broken up into the relational model and then stored in the appropriate database tables. Retrieving the document means querying the database for the relevant information parts and reassembling the business document. The relational database model has therefore less context than the business document model because its main goal is to store and retrieve pure information and avoiding inconsistencies.

Both, the business document model and the relational database model serve their own purpose. On the one hand the relational model focuses on a multitude of business documents and not on a single instance since its goal is the consistent storage of normalized data in the large scale. On the other hand a business document is assembled using a set of reusable components forming a hierarchical model. A relational database model cannot be used to depict hierarchies consisting of reusable parts. Finally the avoidance of data redundancy is an integral part of the relational model. In some cases business document models must deliberately allow data redundancy due to the requirements of a given business case. As an example an invoice bundle is taken which groups invoices of the same enterprise. The left hand side of figure 1 shows an invoice bundle containing multiple instances of invoices numbered 1, 2 and 3. Each instance of the invoice contains the same tax number (3) although all invoices are of the same enterprise and hence the tax number is the same for each invoice. Since the invoice itself has to be a self contained document, the tax number cannot be stored in the embracing invoice bundle but must be part of the invoice. In comparison the right hand side of figure 1 shows the relational model for the same scenario. An invoice bundle groups multiple invoices of the same enterprise. Since the data is stored normalized, the tax number is part of the invoice bundle and not of the invoice. It follows that the relational model is not suited for business document modeling.

Figure 1: Business document model vs. relational model

Related work in the field of conceptual XML schema modeling concentrates on two main fields. On the one side research is conducted in the area of forward engineering e.g. deriving XML schema artifacts from conceptual models such as UML. On the other side a lot of effort is invested in the reverse engineering approach e.g. generating conceptual models such as UML class diagrams from XML schema artifacts. An overview on research on the reverse engineering of XML schemas to conceptual models such as UML class diagrams is given in (Yu & Steele 2005). The authors examine several reverse engineering approaches and assess their applicability. Although several techniques for a forward engineering from conceptual models to XML representations exist today, only a few solutions are available for transformations in the opposite direction. The generation of UML models out of XML schema data proves to be difficult since not all of the features of an XML schema can be represented in a UML diagram. E.g. UML does not support the concept of inheritance by restriction as
XML schema does. Another open issue is the ordering of attributes which is important in an XML schema but not supported by UML class diagrams. A thorough solution for a reverse engineering approach is presented by (Salim et al. 2004). Using a set of transformation rules for the corresponding XML schema elements the authors present appropriate representation solutions in UML. However, the authors do not address the representation of an <xs:restriction> which cannot be demonstrated in UML.

In contrast to the reverse engineering of conceptual UML models from XML schema several approaches exist for the forward engineering approach. One of the first research propositions for the representation of XML using UML has been presented by (Skogan 1999). Build upon these research results (Combi & Oliboni 2006) introduce the so called UXS model (UML & XML Schema) based on UML. UXS is a methodology for designing XML documents using a set of graphical elements which correspond to the appropriate XML schema components. Furthermore a translation mechanism is introduced allowing the generation of XML schema artifacts according to the three well known patterns "Russian Doll", "Salami Slice", and "Venetian Blind" (Malik 2003).

Although several approaches for the conceptual modeling of XML schema exist, only a few of them consider the semantic data modeling aspects. A mutation analysis model used to verify the general semantic correctness of an XML schema is introduced by (Li & Miller 2005). Using their approach (Li & Miller 2005) compare different XML schema validators in regard to their effectiveness in finding semantic errors within XML schemas. A formalization for a data modeling approach is introduced by (Mani et al. 2001) also taking into account the semantic dependencies between the different elements within an XML schema. The introduced methodology called XGrammar allows for a precise definition of features necessary for data modeling such as n-ary relationships, generalizations etc. The application of the Active XML Schema approach for the semantic enrichment of XML schema documents is discussed in (Bernauer et al. 2003). The authors examine the trade-off between the semantic enrichment of an XML schema using the Active XML schema approach and the loss of XML schema interoperability caused by such an enrichment.

A similar approach to UN/CEFACT’s Core Components is pursued by OASIS and has become known as the Universal Business Language (OASIS 2006) (UBL). UBL is a standard for XML document formats based on UN/CEFACT’s core components and provides a mapping of the syntax neutral core components to real XML constructs. The initiative follows a similar approach as the naming and design rules (UN/CEFACT 2006) (NDR) provided by UN/CEFACT. In order to overcome the redundancies in regard to standardization by a merger of the UBL initiative with the core components initiative of UN/CEFACT has been decided during the last UN/CEFACT forum meeting 2007 in Stockholm.

In regard to domain specific business standards several initiatives have been started in recent years. RosettaNet Business Documents (RosettaNet 2006) is an initiative of the electronic components and telecommunications industry. In the insurance domain the ACORD (ACORD 2007) standard plays a significant role and CIDX (Exchange 2007) is pursing document standardization for the chemical industry. Other initiatives include SWIFT (for Worldwide Interbank Financial Telecommunication 2007) from the finance industry, HL7 (Seven 2007) from the health care industry, Papinet (papiNet 2007) from the forest and paper industry, and PIDX (Committee 2007) from the oil and gas industry.

As outlined in this chapter several approaches to the conceptual modeling of business documents and XML and the forward and reverse engineering thereof exist. Although applicable to the general purpose of XML modeling, the different approaches do not consider the business semantics and business requirements necessary for business document modeling. The following section will introduce the core components standard as a methodology of choice for the concise modeling of business documents exchanged in an inter-organizational business process.

3 UN/CEFACT’s Core Components

3.1 The core component meta model

Core Components form the central building blocks of the Core Components Technical Specification (CCTS) (TMG 2003). By definition core components are syntax and platform independent and the standardization of core components is done using regular spread sheets. If a core component is used in a certain business context it becomes a so called business information entity. The two packages in the CCTS meta model in figure 2 show the two fundamental concepts of a core and a business context.

As shown on the left hand side of figure 2 the core components standard distinguishes between three different types of core components: aggregate core components (ACC), basic core components (BCC) and association core components (ASCC). An aggregate core component forms a self contained entity which consists of several basic core components. For example an address would be an aggregate core component whereas the different details of an address such as street, postal code etc. would be basic core components. In order to build relationships between different aggregate core components the concept of so called association core components is used. An association core component may for instance relate the two aggregate core components person and address. A basic core component such as postal code in address has a certain type - a so called core data type. Core data types are based on primitive types referred to as Core Component Types. Core Component Types are e.g. Integer, String.

![CCTS meta model](image)

**Figure 2: CCTS meta model**

Core components are standardized by UN/CEFACT and are independent of a specific industry context or business domain. In order to derive a business document solution for a specific business context the business modeler takes a generic core component and tailors it to the specific need of a business domain.
Hence core components serve as the generic basis for industry specific document formats. If core components are put into a specific business context they become so called business information entities. As shown on the right hand side of figure 2 the core components standard differentiates between three different types of business information entities: aggregate business information entities (ABIE), basic business information entities (BBIE) and association business information entities (ASBIE). Similar to the concept of core components an aggregate of business information entity forms a self contained block which consists of several basic business information entities. For example a cargo box would be an aggregate business information entity whereas the different details of the cargo box such as height or weight would be basic business information entities. A basic business information entity has a certain type - a so called qualified data type. A qualified data type is based on a core data type. Similar to the relationship between a generic core component and a business context specific business information entity, a business specific qualified data type is based on a generic core data type.

Using association business information entities the model builds relationships between different aggregate business information entities. An association business information entity could for instance relate the two aggregate business information entities cargo box and cargo good in order to indicate the content of the cargo box. The concept of a message assembly as shown on the lower right hand side of figure 2 is used to aggregate different business information entities together, forming the final business document. The relationship between the core and the business context is denoted by the four dependencies in figure 2. As already mentioned a qualified data type is based on a core data type. Likewise a basic business information entity is based on a basic core component and an association business information entity is based on an association core component. Finally an aggregate business information entity is based on an aggregate core component. Since the specific dependencies between core components and business information entities are rather difficult to conceive on the meta level a simple example will be used in order to elaborate the basic concepts of core components and business information entities.

3.2 A simple core component example

In order to allow for a better understanding of the core components methodology we already use the UML based notation in this chapter as specified in the UML Profile for Core Components (UN/CEFACT TMG n.d.). In order to give the reader a first impression about core components figure 3 shows a simple core components example. Aggregate core components are modeled using UML classes and basic core components are shown as attributes thereof. Association core components are denoted using compositions between UML classes. On the left hand side, two aggregate core components are shown - invoice and line item. In this example an invoice consists of three basic core components - invoice number, a country identifier, and a description. In a real world example the invoice would contain more basic core components - for presentation purposes however they have been left out as indicated in figure 3. An invoice furthermore has an association core component named item leading to the aggregate core component line item. Line item in turn also has three basic core components: identifier, net price and description. Due to space limitations only three basic core components are shown. In a real world example an aggregate core component would have a multitude of different basic core components.

Figure 3: Dependency between core and business context

The two example core components shown on the left hand side of figure 3 are standardized independently of any business context by UN/CEFACT. These generic core components are now used to derive an industry specific business document format. In this case an example from the United States tourism industry is used. On the right hand side of figure 2 the business context with two aggregate business information entities US invoice and US line item is shown. US invoice has two basic business information entities namely invoice number and description. Furthermore it has an association business information entity named US item connecting the two aggregate business information entities US invoice and US line item. The aggregate business information entity US line item has three different basic business information entities: identifier, net price and description. The relationship between core components and business information entities is as follows. If a modeler constructs a business document for a certain business context or industry he first searches the generic core component library for an appropriate document representation of his business case. After having found an appropriate core component the modeler restricts the core component to the specific needs of the business domain. Thereby the core component becomes a business information entity. A business information entity is always derived from a core component by restriction. Hence a business information entity cannot contain any attributes which are not specified in the underlying core component. As shown in figure 3 the aggregate business information entity US invoice contains only two attributes namely invoice number and description because it restricts the generic aggregate core component invoice to those types needed in the specific business context. The same applies to the aggregate business information entity US line item and its underlying aggregate core component line item. This specific relationship between aggregate core components and aggregate business information entities is denoted by the basedOn dependency in figure 3. Likewise a basic business information entity is based on a basic core component. No basedOn dependencies between basic business information entities and basic core components are shown in figure 3 since the aggregate core components and aggregate business information entities containing the basic core components and basic business information entities are already connected using a basedOn dependency. Implicitly given by this relationship is the connection between business information entities and basic
core components. The same basedOn relationship applies to association core components and association business information entities as shown in figure 3.

As business context for the business information entities in figure 3 we assume an example from the tourism industry. In order to help to distinguish core components from business information entities the concept of qualifiers is used for aggregate business information entities and association business information entities. The qualifier used in figure 3 is US invoice number as shown in the United States tourism industry. A qualifier can be chosen arbitrarily by the business document modeler and does not need to comply to any constraints.

Basic business information entities and basic core components are of a certain type. The basic core component invoice number of the aggregate core component invoice is of type Integer. We refer to data types of basic core components as core data types (CDT). The basic business information entity invoice number of the aggregate business information entity US invoice is of type US invoice identifier. We refer to data types of business basic information entities as qualified data types (QDT). Please note however, that a basic business information entity can also be of a type if needed. The example in section 5 further elaborates this necessity.

As indicated on the left hand side of figure 3 the basic core component invoice number of the aggregate core component invoice is of type Integer. The basic business information entity invoice number of the aggregate business information entity US invoice as shown on the right hand side of figure 3 however is of type US invoice number. Hence the modeler has the possibility to restrict the data type of the basic business information entity to the specific needs of a certain business context. Not shown in figure 3 is the fact, that the modeler can also restrict which association core components are becoming association business information entities. It follows, that an aggregate business information entity does not necessarily has to have all associations like the underlying aggregate core component.

This section has given an overview about the fundamental principles of the core components standard. Since core components are standardized independently of any implementation platform or technology a representation mechanism for core components had to be found. The next section will introduce the UML Profile for Core Components.

4 UPCC - A UML Profile for Core Components

Recent years have shown an increasing trend towards the usage of UML in the area of business process modeling and business document modeling. Several tool vendors have developed UML modeling tools supporting the most recent version of the UML meta model ( OMG 2007). In order to allow for an easy integration of the core components methodology into such tools a representation format for core components is available. This represents a strong contradiction to the initial purpose of core components: cross-industry alignment and reusability of business documents and business information.

Therefore a unique representation mechanism for core components in UML is necessary. The authors of this article together with other contributions submitted a UML representation format for core components to UN/CEFACT. Since then this proposal has become known as the UML Profile for Core Components (UPCC) standard ( UN/CEFACT TMG n.d.). A UML profile customizes the UML meta model to the specific needs of a certain application scenario. Using stereotypes, tagged values and OCL constraints the generic UML meta model is tailored to the specific needs of business document modeling. Figure 4 gives an overview of the different stereotypes used in the UML Profile for Core Components. Since the full names of the different stereotypes are rather long, abbreviations have been used. Stereotypes representing modeling artifacts are presented using a black background. In UPCC modeling artifacts are presented using the concept of packages. In the meta-model these packages are shown with a white background.

The main goal when developing the UPCC standard was the precise and unambiguous representation of core components in UML. Where possible native concepts of UML have been used to depict core component principles. The very basic stereotype is a primitive type (PRIM). A PRIM is used to express basic types such as String, Integer and is very similar to the UML concept of a type. The core component standard defines six primitive types partly overlapping with the types defined in UML. In order to restrict a primitive type to a specific set of values an enumeration (ENUM) is used. Thereby the modeler can restrict a primitive type to a specific set of valid values e.g. ISO 3166 (ISO 2007) for valid country codes. In UPCC an enumeration is represented using the UML concept of an enumeration.

In contrast to an enumeration or a primitive type a core data type (CDT) can express a more meaningful type. A core data type is modeled using classes and consists of multiple attributes. Thereof exactly one attribute is stereotyped as content component (CON) and multiple attributes can be stereotyped as supplementary components (SUP). The content component represents an atomic value. Supplementary components are used to provide meta information about the content component. An example core data type might for instance be measurement. The content component would be the number 12. Additional supplementary components could be measurement unit (Fahrenheit) and measurement unit (Celsius). Hence the three basic values are combined in order to form a more complex type - a core data type.

UML requires, that each attribute of a class has a certain type. In case of content components and supplementary components the valid type is either a primitive type (PRIM) or an enumeration (PRIM). As already outlined in the introduction to core components an aggregate core component (ACC) is modeled using UML classes. The attributes of an aggregate core component are stereotyped as BCC and so called basic core components. A basic core component (BCC) attribute has a certain type - a so called core data type (CDT). In order to build a hierarchical structure between different aggregate core components it is possible to use the concept of a UML composition stereotyped as association core component (ASC). By definition every association core component must
5 Core Components by example

In figure 5 the example package structure of a business document model using the UML Profile for Core Components is shown. The scenario is taken from a purchase order process taking place between a buyer and a seller where the buyer sends a purchase order to the seller. The example used in this article has been created using the UML modeling tool Enterprise Architect (Systems 2008). The relevant diagrams contained in the different packages in figure 5 are shown in figure 6. Using the alphabetically numbered dots the packages in figure 5 are connected to the pertaining diagrams in figure 6.
different aggregate core components in (A) have not become part of the final business document model (B). Furthermore, qualifiers such as purchase_order are used for the aggregate business information entities in (B) in order to allow for a better differentiation between core components and business information entities in the overall model.

In order to comply with the specific requirements of a purchase order the modeler derives a qualified data type purchase_order_identifier from the generic core data type identifier in (D). Similar to the relationship between a core component and a business information entity a qualified data type is always derived from a core data type by restriction.

The core data type identifier is used several times in the core component model (A) as indicated by an exemplary dotted line. For the specific needs of the aggregate business information entity product unit_identifier the specialized qualified data type purchase_order_identifier is used instead of the generic type identifier. Hence the modeler can restrict the type of a basic core component when transferring it to a basic business information entity. Please note, that some of the basic core components and their types remain unchanged and are simply taken over from the core component model (A) to the business information entity model (B). Where no specialization of a data type is necessary, the basic business information entities simply use the core data type of the underlying basic core component. An aggregate core component address (A) becomes a tendering_address when used in the purchase order context. The aggregate business information entity tendering_address restricts the generic aggregate core component to two basic business information entities namely country name and postcode. While the type of postcode remains a core data type and therefore unchanged (code), the country name gets a qualified data type in (B) namely country code. The qualified data type country code is not shown in figure 6.

Primitive types (E) are used to set the type of supplementary components and content components in core data types (CDT) and qualified data types (QDT) as shown in (D). The enumeration country codes (F) is used to restrict the supplementary component code of the qualified data type purchase_order_identifier as shown in (D). Finally the business document is assembled in (C). The business message is a purchase order request message which has a standard business document header and a regular header. Both header elements carry additional meta information about the actual business document purchase_order. Header and the purchase_order are connected to the purchase order request message using two association messaging business information entities (ASMBIE). The concept is the same as the one of an association business information entity, only the naming is different when used in the context of a business message.

The message type purchase_order request message as well as the header and standard business document header as shown in (C) in figure 6 are forming the embracing part of the actual business document. The actual payload of the purchase order request message is the aggregate business information entity purchase_order and all its relating aggregate business information entities as shown in (B) in figure 6.

The next section will introduce the usage of naming and design rules in order to uniquely derive XML schema artifacts from a core component model.

6 Deriving XML artifacts from Core Components

The previous sections introduced the core components methodology and the UML profile for core components. Using the core components methodology the modeler can define a business document on a conceptual model in a unique and semantically precise manner. For the exchange of business documents between companies and B2B systems however a logical level representation of business documents is needed.

This section will outline how the conceptual core components model can be used to derive XML schema artifacts. These XML artifacts form the logical level business document model to which every document instance exchanged between two B2B systems must comply to.

In order to allow for a unique representation of core components in XML UN/CEFACT suggests the use of so called Naming and Design Rules (NDR) (UN/CEFACT 2006). Along with each new release of the Core Components Technical Specification and its UML Profile UN/CEFACT delivers pertaining Naming and Design Rules.

Since a real world core components example can easily become extensive and complex a manual transformation of core components represented in UML to the appropriate XML schema artifacts is not efficient. In order to overcome the limitations of a manual transformation we have build an XML schema generator. The XML schema generator is part of a larger set of tools supporting the modeler in inter-organizational business process and business data modeling known as the UMM Add-In (Austria 2007).

As already outlined in figure 5 a core components model is defined using stereotyped packages which follow a rigid structure. Using the UMM Add-In the user simply clicks on a package and initiates the transformation of core components to the appropriate XML schema representation. The XML schema generator automatically detects dependencies in the core components model and generates additional XML schema files containing data type definitions, core component definitions etc.

Listing 1 shows the XML schema representation of the example model shown in (B) in figure 6. As outlined by the dotted arrow from (B) to (C) in figure 6 this code is attached to the final purchase order request message in (C). The XML schema generator iterates over every aggregate business information entity in (B) and constructs a sequence for each. The six complex types are shown in line 5, 14, 22, 30, 37 and 43 in listing 1. The root element of the business document is shown in line 4. For each basic business information entity an element is created in the sequence of the embracing aggregate business information entity’s complexType. Since every basic business information entity has a certain type (either core data type or qualified data type) the necessary complexTypes have to be referenced. As shown in figure 6 the qualified data types and core data types are defined in a separate library to the business information entity library. The generator automatically detects these dependencies, creates the necessary auxiliary schemes and imports them into the final schema. In listing 1 the core data type library is imported in line 2 and the qualified data type library is imported in 3.

For each association business information entity an element in the complexType’s sequence of the aggregate business information entity is created as well. As outlined in figure 6 the aggregate business information entity purchase_order has two association business information entities namely ship from (tendering_address) and product (purchase order, line, item). As
shown in line 10 and 11 in listing 1 or each association business information entity the correct complexType is set.

Listing 1: Purchase_Order XML schema

```xml
of the qualified data type is depicted using an extension (line 81) and the supplementary components are specified using attributes (line 82-83).

Line 83 shows how the generic code attribute is restricted to a set of country codes.

Listing 4: Qualified Data Type Schema

```xml
<xs:complexType name="CountryCodesType">
  <xs:attribute name="Code" use="required" type="xs:string"/>
  <xs:attribute name="CountryCodes" use="optional" type="enum:CountryCodesType"/>
</xs:complexType>
```

The XML artifacts for a core component library are constructed similarly to a business information entity library. Due to space limitations however these generation artifacts are not shown in this article. This section has shown how the XML schema generator can be used to automatically generate a 1:1 representation of the conceptual core components model in XML schema syntax.

7 Conclusion and Future Trends

In this article we presented UN/CEFACT’s core components technology. Using the concept of core components business documents can be defined on a conceptual and syntax independent level. In order to allow for an integration of the core components technology into a UML modeling tool, the UML Profile for Core Components has been developed. Using the UML profile the modeler can retrieve existing generic core components from a standardized library. The generic core components are further restricted in order to create industry and context specific business information entities.

We introduced our tool based XML generator which follows the guidelines specified in the Naming and Design Rules of UN/CEFACT. Using a single mouse click the XML generator automatically iterates over a complete core components model and generates XML schema files. The schema files are used to validate the exchanged business documents in an actual B2B collaboration. Furthermore they are needed for the definition of entry/exit points in a service oriented architecture e.g. the generated schema is imported into a WSDL file. If both B2B partners have the same business document definitions the entry/exit points of both business partners match.

The contribution of our approach in regard to the four stated shortcomings of current business document modeling approaches is therefore as follows:

(1) **Standard incompatibilities.** Since core components define a business document on a conceptual level and not on the logical level (e.g. XML schema file) there cannot be any inconsistencies in regard to the representation of a specific business document. This is a major advantage over standards based on pure implementation logic (e.g. XML).

(2) **All-in-one approach.** Similar to EDI approaches UN/CEFACT defines a set of normative core components which are defined for a whole industry domain e.g. steel industry. Due to the derivation by restriction mechanism of core components no overhead occurs in a well defined business context. The generic industry specific core component is tailored to a business information entity for a given business domain. Since every business information entity has a based on dependency to its underlying generic core component the common semantic basis for every business information entity is given hence allowing a matching even between business information entities from different business contexts.

(3) **Transfer syntax specific definition.** Core components are defined on a conceptual meta level, independent of any transfer syntax. With every release of the core components standard UN/CEFACT releases a set of well defined Naming and Design rules allowing the mapping to a logical level implementation of core components. In case of a change in the transfer syntax only the mapping rules have to be altered instead of a complete standard re-engineering.

(4) **Conceptual document description.** Core components are standardized independent of any transfer syntax. In order to allow business models author to communicate business document models in an efficient way, UN/CEFACT provides the UML Profile for Core Components together with every release of the Core Components standard. Using the UML profile a business modeler can easily assemble business documents with UML artifacts and communicate the final UML business document model. Such a model can be handed over to other business modelers, software developers and stakeholders in order to communicate the structure and semantics of a specific business documents. Eventually the conceptual UML model can be used to derive artifacts for the logical level implementation (e.g. XML schema).

Although the current implementation of the UML Profile for Core Components together with the XML generator provides a solid basis for business document modeling and schema artifact generation, several tasks are planned for the future. On the one hand the current UML Profile for Core Components (UPCC) is based on the Core Component Technical Specification (CCTS) version 2.01 (TMG 2003). Since CCTS 3.0 will be released this year, the UML profile has to be updated to meet the requirements of the new core component version. Every core component release does also include updated Naming and Design Rules, the XML schema generator has to be updated accordingly as well.

In order to further foster the dissemination of the core component technology the usability of the UML Profile for Core Components must be enhanced. A first step allowing inexperienced users to use the core components technology is the implementation of a validation routine for the core component’s UML profile.

Since the core component model of a business document is independent of any implementation syntax the derivation of other artifacts than XML schema could be possible as well. Following modified Naming and Design Rules the core components model may be used to derive Relax NG (OASIS 2001a), UBL (OASIS 2006) or EDIFACT (UN/CEFACT 2007) artifacts as well.

References

ACORD (2007), ACORD Insurance Data Standards, ACORD.

URL: http://www.acord.org/
Austria, R. S. (2007), The UMM Add-In 1.0, Research Studios Austria. URL: http://ummaddin.researchstudio.at


ISO (2007), ISO 3166 List of country names and code elements, ISO. URL: http://www.iso.org/iso/country_codes.htm


OASIS (2007), Universal Description, Discovery, and Integration, OASIS. URL: http://uddi.xml.org/


Seven, H. L. (2007), Health Level Seven, Health Level Seven. URL: http://www.hl7.org/


TMG, U. (2003), Core Components Technical Specification - Part 8 of the ebXML Framework, UN/CEFACT TMG.


TMG, U. (2008), Core Components Technical Specification 3.0 - draft, UN/CEFACT TMG.

UN/CEFACT (2006), Naming and Design Rules, UN/CEFACT.


UN/CEFACT TMG, organization = UN/CEFACT, m. O.y. p. U. n. C. (n.d.), UPCC UML Profile for Core Components based on CCTS 2.01.


