On Mapping Business Document Models to Core Components *

Christian Pichler, Michael Strommer
Research Studios Austria
Thurngasse 8/3/20
1090 Vienna, Austria
{firstname.lastname}@researchstudio.at

Philipp Liegl
Vienna University of Technology
Favoritenstrasse 9-1/188
1040 Vienna, Austria
liegl@big.tuwien.ac.at

Abstract

Today, there exists a huge variety of business document models for electronic data exchange among business partners used in business-to-business, business-to-government, as well as government-to-government business transactions. Thus, it becomes inevitable to process differing document models in daily business, unless a commonly agreed standard is developed. However, even if a commonly agreed standard exists, a multitude of different document definitions will still be existent in the short-term. Incompatibilities between old document formats and the newly introduced document models are the consequence. To cope with various document models and formats we propose a heuristic approach that leverages interoperability by mapping common document models to the conceptual Core Components model introduced by UN/CEFACT. To successfully complete this mapping task we define a set of rules that may be applied to XML Schema based document models.

1 Introduction

Nowadays, business document modelers are confronted with a multitude of often heterogeneous business document standards. The development of standards is driven by various organizations and institutions, having different needs and motivations. A promising approach, aiming at global interoperability, is currently pursued by the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) named Core Components Technical Specification (CCTS) [21]. The goal of CCTS is the definition of semantic building blocks for assembling shared libraries of reusable business document definitions. The standard definition may be used in a two-fold manner, namely in a forward engineering approach and a backward engineering approach. In a forward-engineering approach new business document definitions are created from scratch from an existing CCTS compliant model. Such a scenario may be applied for new solutions. Forward engineering approaches are rather straightforward and may be implemented without any major difficulties. In case existing business document definitions already exist, they must be mapped to the CCTS compliant model.

In a backward engineering approach existing business document definitions are aligned to a common document format. In contrast to forward engineering, backward engineering approaches impose a set of challenges on the business document modeler. In our case the common document format is represented by a core component compliant model. In order to allow for a seamless mapping of existing business document definitions to a common conceptual core component model, a set of predefined rules is necessary. Such mapping mechanisms are still missing and not provided by UN/CEFACT. In this paper we close this gap by analyzing potential mappings from existing document definitions to conceptual core component models. We first introduce basic mapping mechanisms and consequently elaborate on advanced concepts using pertinent examples. For a detailed discussion on mapping of schemas see for example Legler and Naumann [13]. They present a classification of correspondences that may arise, as well as proper tooling available in this research area.

We motivate our mappings using an accompanying example from the e-Government domain. In the European Union (EU) cross-border transport of waste requires the exchange of transport documents between the exporting, importing, and transit countries. Thereby, documents are exchanged between the involved companies and local authorities in the different countries. Today, this informa-

*The work of Research Studios Austria is funded by the Austrian Federal Ministry of Science and Research. Furthermore, this work has been carried out under the research grant Public Private Interoperability (No. 818639) of the Austrian Research Promotion Agency (FFG).

tion exchange is entirely paper-based, causing potential errors when processing these documents. The EU sponsored project EUDIN (European Data Interchange for Waste Notification System) aims to supersede the paper-based information exchange by electronic data interchange. Thereby, EUDIN uses the Core Components Technical Specification (CCTS) concepts in order to model the exchanged information. Even if a central data model has been created with the use of Core Components, the different local authorities and involved companies still have their own, internal data structure for the different transport documents. In this paper we examine different strategies on how to map existing business document definitions to a globally defined, core component based model. Thereby, we use a simple core component scenario of a consignment item with different delivery and despatch parties, as it would be used in a cross-border waste transport process.

The remainder of this paper is structured as follows: Section 2 gives an overview of the Core Component technology and the related concept of business information entities. The different mapping mechanisms are introduced in Section 3. In Section 4 advanced mapping mechanisms such as metamodel layer mappings are introduced. Section 5 introduces related work in the domain of business document mapping. Finally, Section 6 concludes the paper and gives an outlook to future research directions.

2 CCTS at a glance

Our approach is based on the Core Components Technical Specification (CCTS) [21], maintained by UN/CEFACT. Figure 1 gives an overview of basic concepts of the Core Components Technical Specification. We distinguish between two major paradigms: Core Components (CC) and Business Information Entities (BIE). Core Components are context-independent building blocks for the creation of business documents. A core component does not have a specific business context, thus it may be used in any given scenario. In order to contextualize a given core component to the specific needs of a given business domain, the concept of business information entities (BIE) is used. Business information entities are derived from core components by restriction and may be used in a given business domain. Thus, a business information entity must not contain any attributes, which have not been defined in the underlying core component. Essentially, core components represent the common semantic basis for all business information entities. In a nutshell, business information entities are the same as core components except that they a) use a different naming convention in order to facilitate distinction from core components b) are used in a specific context c) are based on core components and thus restrict an existing core component definition.

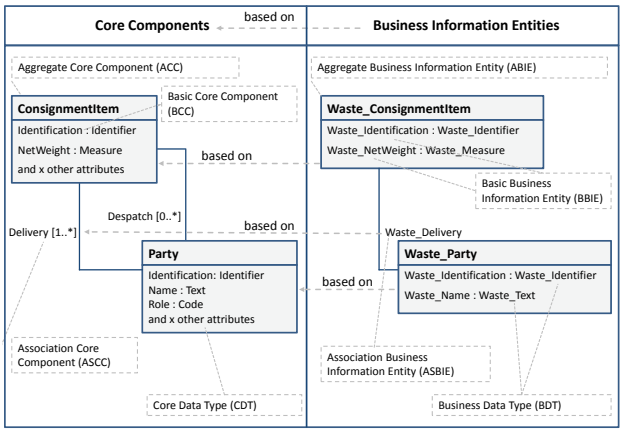

Figure 1: Overview of basic core component concepts

Having these concepts at hand, it is possible to model business documents first in a conceptual, syntax neutral, and platform-independent way using core components. The created core component libraries may then be used to create different libraries of business information entities. Due to the derivation-by-restriction concept all business information entities are based on a common base of core components. Therefore interoperability between different business information entities is always provided, as long as the link to the underlying core component definition exists.

Core Components. The basic concept behind core components is the identification of objects and object properties. Thereby, object properties are divided into two sub-groups: simple object properties (e.g. date, name, age) and complex object properties (references to other objects). An object is represented using an aggregate core component (ACC). Aggregate core components represent an embracing container for simple object properties. On the left hand side of Figure 1 the ACC `ConsignmentItem` includes two simple object properties - `Identification` and `NetWeight`. Simple object properties are referred to as basic core components (BCC). As indicated by the term *and x other attributes*, an aggregate core component has many different basic core components. Each simple object property has a data type known as core data type (CDT). The core data type of `Identification` is `Identifier`. A core data type defines the exact value domain of a given basic core component. In order to depict dependencies between different objects, the concept of association core components (ASCC) is used. In Figure 1 the ACC `ConsignmentItem` is connected to the ACC `Party` using the association core components `Delivery` and `Despatch`. Thus, `Party` is a complex object property of the ACC `ConsignmentItem` and indicates the fact, that each consignment item has one or more delivery parties and zero or more despatch parties.

Business Information Entities. If core components are used in a certain business context, they become so called business information entities. Business information entities are always derived from an underlying core component by restriction. This implies, that a business information entity must not contain any attributes or associations which have not been defined in the underlying core component. On top of Figure 1 a *based on* dependency between the business information entity artifacts on the right hand side and the core component artifacts on the left hand side underlines this strong relation. Due to this relationship, the underlying core component semantics of any business information entity may be retrieved easily.

Similar to the concept of core components, business information entities distinguish between three different elementary types. An aggregate business information entity (ABIE) is used to aggregate simple object properties. Simple object properties are depicted using the concept of basic business information entities (BBIE). On the right hand side of Figure 1 the ABIE `Waste_ConsignmentItem` aggregates the two BBIEs `Waste_Identification` and `Waste_NetWeight`. Please note, that the business information entity `Waste_ConsignmentItem` does not contain all attributes of the underlying core component `ConsignmentItem`. Thus, the ABIE restricts the underlying aggregate core component (ACC). The value domain for a given basic business information entity is defined using the concept of business data types (BDT). The BBIE `Waste_Name` contained in the ABIE `Waste_Party` is of type `Waste_Text`. Dependencies between different business information entities are shown using the concept of association business information entities (ASBIE). Please note, that a business information entity must contain the name of the underlying core component. However, so called qualifiers may be used in order to facilitate the distinction between core components and business information entities. For example, the qualifier `Waste_` is used in Figure 1 for all business information entity definitions. For a detailed discussion on core components and business information entities see [14].

A similar relationship as it is established between core components and business information entities is also established between core data types and business data types. Essentially, a business data type must always be based on a core data type and may only use attributes which have already been defined in the underlying core data type. Since we do not elaborate on data type mappings in this paper, we do not further dwell on these concepts.

Core Component Library. Since it is imperative that all business information entities are based on a core component, a consolidated library of predefined core components must be provided. A global library of core components is currently developed and maintained by UN/CEFACT and has become known as Core Component Library (UN/CCL) [19]. Business document modelers may search and retrieve core components from the core component library. Furthermore, anybody interested in the definition of core components may submit new core component definitions to UN/CEFACT, where the harmonization and standardization of core components is organized. Updated versions of the core component library are released twice a year, containing new core component definitions and updates of existing core components. The core component definitions in the Core Component Library represent the common semantic basis of all business information entities exchanged among business partners. Business information entities are themselves organized in libraries to which we refer to as business information entity libraries. Since business information entities are a domain specific concept, standardized business information entitites are provided per different domain rather than on a global level.

A UML Profile for Core Components. Core components in the core component library are defined using spreadsheets and thus integration into modeling tools is difficult. In order to apply the core component concepts as a means of modeling business documents we have created a UML profile, that allows using core components with standard UML modeling tools. This standard has been developed at the Research Studios Austria, Studio Inter Organizational Systems [1] and has consequently been submitted to UN/CEFACT for standardization. Today the standard has become known as the UML Profile for Core Components (UPCC) [22]. The UPCC fully represents all necessary concepts from the Core Component Technical Specification including business information entities and data types.

		CCTS Space	XML Space	Mapping Space	Example
NDR based	**Metamodel Layer**	CCTS	XML Schema	Metamodel Mapping	BBIE ↔ element
Modeling based	**Model Layer**	CCTS Model	XML Schema Instance	Model Mapping	Person Concept ↔ Party Concept
	Instance Layer	XML Instance	XML Instance	Instance Mapping	Michael ↔ Research Studios

Table 1: Mapping spaces in CCTS based document model integration.

3 Basic Mapping

As described above, the CCTS is a conceptual model that is syntax neutral and therefore not tied to any specific technology. For the exchange of models it is necessary to use

[1] http://ios.researchstudio.at

serialization standards such as XML. For core component models UN/CEFACT chooses to use XML Schema [24] to persist the defined models. XML Schema is also a widely accepted technology in the domain of business document engineering [15]. Table 1 illustrates the mapping spaces relevant for CCTS based model integration. In this paper we will not dwell on the instance layer and the accompanying task of mapping and transformation of instances. The mapping between CCTS and XML Schema is basically specified by the UN/CEFACT's Naming and Design rules [20] and happens on the metamodel layer as depicted in Table 1. This mapping specification ensures, that all conforming models may be imported and exported by tools building upon the CCTS. In Table 2 we have summarized the basic mappings specified by [20]. One may easily see by looking at the concepts involved in column XML Schema, that this is only a minimal set of XML Schema concepts available. Therefore, we will provide additional mapping definitions apart from the Naming and Design rules for CCTS in the following Chapter.

BIE Concept	mapsTo	XML Schema	Comment
Business Data Type (BDT)	<->	simpleType or complexType	
Basic Business Information Entity (BBIE)	<->	element	local declaration
Aggregate Business Information Entity (ABIE)	<->	complexType and element	global declaration
Association Business Information Entity (ASBIE)	<->	element	global and local declaration

Table 2: Mapping of CCTS to XML Schema components.

Note, that there are only mappings for BIE concepts defined by the Naming and Design rules. The underlying CC concepts are not participating in the serialization process except as optional annotations. For the generation of an XML Schema, corresponding core components may be neglected. But for the use of these schemas or the import in a tool one needs to know the corresponding dependencies in order to retrieve the semantic of the given business information entities. Without this knowledge we maintain meaningless data capsules loosing every advantage of the general top-down approach of CCTS. The very basic method for retrieving the underlying core component definition of a business information entity is to decompose the name of a business information entity into single terms. Please recall, that a business information entity's name must contain the name of the underlying core component as well. Thus, traceability between core components and business information entities is also ensured on the XML Schema level. By simple name matching we may then get an understanding of the contextualized core components by looking at the defined semantics in the core component library.

The second mapping space (cf. Table 1 Modeling based) we are interested in is concerned with the models themselves, which constitutes the actual mapping task carried out by users manually. By mapping two models, one of which is CCTS in our case, we have to ensure, that model structures and semantics are preserved. Heuristics for model mapping are presented in Section 4.2.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:tns="http://foo.bar/doc"
            xmlns:bdt="http://foo.bar/doc"  2
            targetNamespace="http://foo.bar/doc"
            ... >

  <xsd:include schemaLocation="BusinessDataType_1.xsd"/>

  <xsd:complexType name="Waste_ConsignmentItemType">
    <xsd:sequence>
      <xsd:element name="Waste_Identification"
                   type="bdt:Waste_IdentifierStringType" />
      ...  1
      ...
      <xsd:element name="Waste_DeliveryWaste_Party"  3
                   type="tns:Waste_PartyType" />
    </xsd:sequence>
  </xsd:complexType>

  <xsd:element name="Waste_ConsignmentItem"
               type="Waste_ConsignmentItemType"/>

  <xsd:complexType name="Waste_PartyType">
    <xsd:sequence>
      <xsd:element name="Waste_Name"
                   type="bdt:Waste_TextStringType" />
      ...
    </xsd:sequence>
  </xsd:complexType>

  <xsd:element name="Waste_Party"
               type="tns:Waste_PartyType"/>
</xsd:schema>
```

Figure 2: Distinguish BBIEs from ASBIEs.

Another problem occurs when it comes to identifying ASBIE's in XML Schema because both BBIEs and ASBIE's local declarations map to the schema concept `element` (see Figure 2, mark 1). Generally, this problem may not be resolved very elegantly because the naming and design rules promote the use of a homogeneous namespace design approach with a single target namespace (see Figure 2, mark 2). Nevertheless, we solve the problem via the namespace prefix of the type property of the two Schema elements for the corresponding BBIE's and ASBIE's, respectively (see Figure 2, mark 3). We are currently implementing a tool called VIENNA Add-In (Visualizing INter Enterprise Network Architectures) [23], which is now capable of generating and importing XML Schema files following the naming and design rules specified by UN/CEFACT.

4 Advanced Mapping

In the previous section we introduced different approaches involved in CCTS driven mapping. However, the naming and design rules do not provide sufficient definitions on how to handle arbitrary business document standards based on XML Schema. Therefore, we examine how different XML Schema concepts may be mapped to CCTS

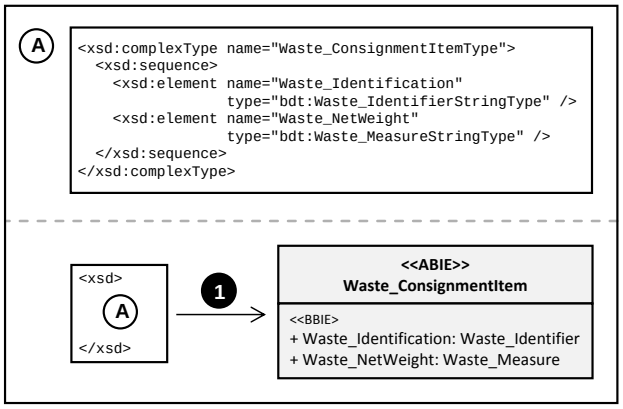

Figure 3: Reusable Group *sequence*.

and evaluate how to preserve semantics of model elements during the mapping of two schemas.

4.1 Metamodel Layer Mapping

To illustrate the shortcomings of the naming and design rules we identify a number of XML Schema concepts and evaluate their applicability in regard to being mapped to core components. Furthermore, we introduce a set of rules to map XML Schema concepts to core components.

4.1.1 Reusable Groups

Reusable groups are used for defining a group of reusable elements or attributes. An example is the use of a model group to define a sequence of elements which in turn may be used to define the content of a complex type. In general there are three different kinds of model groups which include `sequence`, `choice`, and `all`. Each listed model group is elaborated in more detail in the following.

sequence. The model group `sequence` is used for defining elements in a certain order such as the order of elements within a complex type definition. An example illustrating the use of the model group `sequence` within a complex type definition is illustrated in Figure 3, letter A.

The approach of creating an adequate model representation is straightforward, since all elements of the model group may be mapped to BBIEs (see Figure 3, mark 1). Therefore, the resulting model representation of the illustrated example equals an ABIE named `Waste_ConsignmentItem` containing the BBIEs `Waste_Identification` and `Waste_NetWeight`.

choice. The model group `choice` allows the grouping of elements whereas only one of the elements defined may occur. Again, the model group may be used for defining the content of a complex type. An example utilizing the model group is illustrated in Figure 4, letter A.

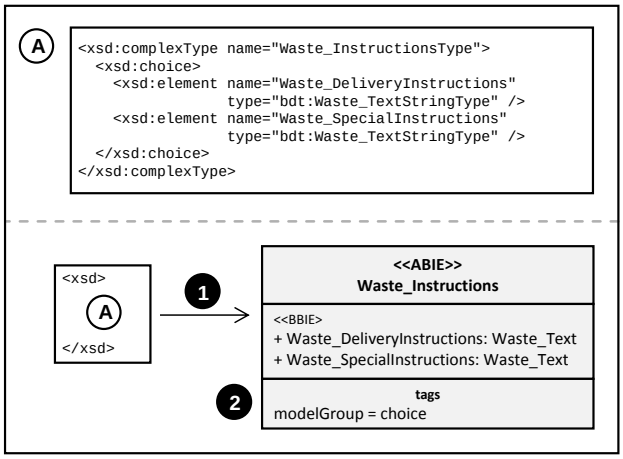

Figure 4: Reusable Group *choice* (Alternative 1).

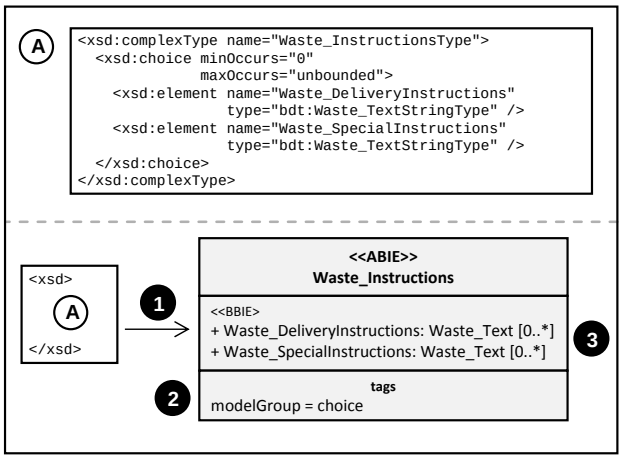

Figure 5: Reusable Group *choice* (Alternative 2).

However, creating an adequate model representation is not feasible without the loss of semantic meaning regarding the characteristics of the model group `choice`. Although the model representation allows specifying minimum and maximum cardinalities, the concept of an exclusive-OR is currently not supported in CCTS. Overcoming these limitations maybe achieved by extending CCTS the following way. The extension may allow that both elements within the model group are represented as BBIEs (see Figure 4, mark 1). In addition we introduce an additional tagged value named *modelGroup* in ABIEs, indicating that the BBIEs within the ABIE are exclusive (see Figure 4, mark 2).

Another common use case of the model group `choice` is to define element groupings enabling all elements to appear in any desired order as well as in an unlimited number of occurrences. The key to define such a model group is setting the model group's maximum number of occurrences to `unbounded`. An according example is illustrated in Figure 5, letter A.

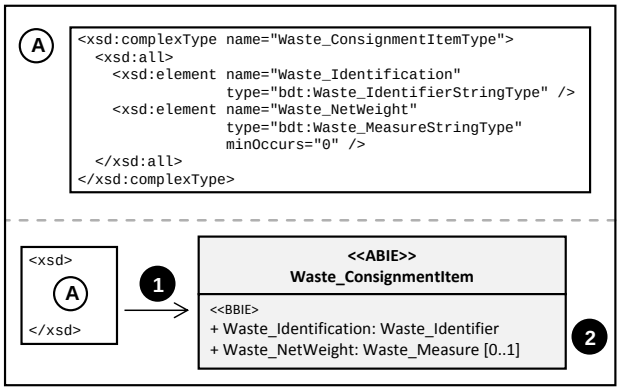

Figure 6: Reusable Group *all*.

The approach to create an adequate model representation is similar to the approach introduced for the model group `sequence`. All element declarations may be mapped directly to BBIEs (see Figure 5, mark 1). The definition of the maximum number of occurrences for the model group itself may be reflected through the cardinalities of BBIEs. Therefore, each BBIE has a minimum cardinality of zero as well as an unlimited maximum cardinality (see Figure 5, mark 3).

However, one drawback of the proposed solution is that when serializing the model representation to an XML Schema not all semantics are preserved. According to the naming and design rules an ABIE is serialized into a complex type definition utilizing the model group `sequence` with according minimum and maximum occurrences for each BBIE. Hence, the resulting XML Schema is not the same as the schema used to create the model representation. However, using our introduced tagged value "modelGroup" as illustrated in Figure 5, mark 2, the correct semantics are preserved.

all. The third kind of model group named `all` allows to define that all element declarations within the model group may appear in any order, but each element may occur only once. An example is shown in Figure 6, letter A.

The process for creating an adequate model representation of the model group is proposed the following way. First of all each element declaration within the model group may be mapped directly to a BBIE (see Figure 6, mark 1). Furthermore, the minimum occurrences and maximum occurrences of the element declarations may be reflected through the cardinalities of the BBIEs (see Figure 6, mark 2). Applied to the example above this would mean that the element `Waste_Identification` is represented through a BBIE with a minimum cardinality of one and the element `Waste_NetWeight` is represented through a BBIE as well whereas the minimum cardinality of the BBIE `Waste_NetWeight` equals zero.

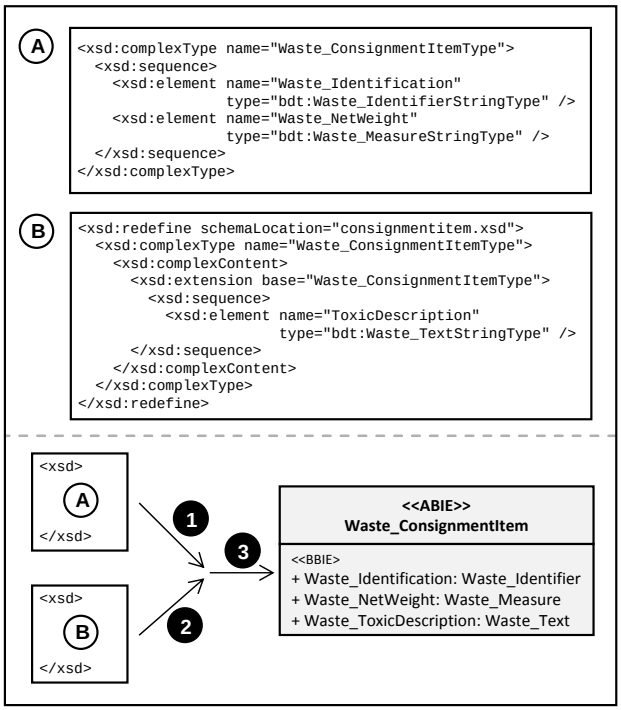

Figure 7: *redefine* Mechanism.

4.1.2 Extensions and Restrictions

redefine. The redefine mechanism may be applied to types and groups and is used to alter existing types and groups. The types include simple types as well as complex types, whereas redefine must extend or restrict the type. Groups include attribute and model groups. Applied to groups, a redefine must subset or superset the original group.

An example utilizing the redefine mechanism is illustrated in Figure 7, letter A. In addition to the complex type definition, Figure 7, letter B, shows the redefinition of the complex type.

The complex type definition of `Waste_ConsignmentItemType` consists of the original type definition as well as the type redefinition. Therefore, in order to create a correct and complete model representation of the complex type, it is necessary to process both, the original type definition as well as the type redefinition. The suggested approach is to first process the complex type definition (see Figure 7, mark 1) and second adapt the complex type definition according to the redefinition (see Figure 7, mark 2). The cumulated complex type may then be mapped to an ABIE, whereas each element of the complex type is simply mapped to a BBIE of the ABIE (see Figure 7, mark 3).

Substitution Group. The purpose of substitution groups is to add flexibility to content models, which are for example used in complex type definitions. A substitution group

A

```
<xsd:element name="Waste_Consignment"
             type="Waste_ConsignmentType" />
<xsd:complexType name="Waste_ConsignmentType">
  <xsd:sequence>
    <xsd:element ref="Waste_ConsignmentItem" />
  </xsd:sequence>
</xsd:complexType>

<xsd:element name="Waste_ConsignmentItem"
             type="Waste_ConsignmentItemType" />
<xsd:complexType name="Waste_ConsignmentItemType">
 <xsd:sequence>
    <xsd:element name="Waste_Identification"
                 type="bdt:Waste_IdentifierStringType" />
    <xsd:element name="Waste_NetWeight"
                 type="bdt:Waste_MeasureStringType" />
  </xsd:sequence>
</xsd:complexType>
```

B

```
<xsd:element name="Waste_ExportConsignmentItem"
         type="Waste_ExportConsignmentItemType"
         substitutionGroup="Waste_ConsignmentItemType" />
<xsd:redefine schemaLocation="consignmentitem.xsd">
  <xsd:complexType name="Waste_ExportConsignmentItemType">
    <xsd:complexContent>
      <xsd:extension base="Waste_ConsignmentItemType">
        <xsd:sequence>
          <xsd:element name="ImportCountry"
                       type="bdt:Waste_TextStringType" />
        </xsd:sequence>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:redefine>

<xsd:element name="Waste_ImportConsignmentItem"
         type="Waste_ImportConsignmentItemType"
         substitutionGroup="Waste_ConsignmentItemType" />
<xsd:redefine schemaLocation="consignmentitem.xsd">
  <xsd:complexType name="Waste_ImportConsignmentItemType">
    <xsd:complexContent>
      <xsd:extension base="Waste_ConsignmentItemType">
        <xsd:sequence>
          <xsd:element name="ExportCountry"
                       type="bdt:Waste_TextStringType" />
        </xsd:sequence>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:redefine>
```

C

```
<xsd:element name="Waste_ConsignmentItem"
             type="Waste_ConsignmentItemType"
             abstract="true" />
```

Figure 8: *Substitution Group* Mechanism (1/3).

consists of one head element as well as one or more member elements, which together form a hierarchy. One constraint these hierarchies must fulfill is, that all member elements are derived from the head element either by restriction or extension. A well defined substitution group then allows to substitute every head element used in a content model by one of the defined member elements. An example illustrating a head element and substitutable member elements is illustrated in Figure 8, letters A and B. In addition, substitution groups also allow the use of abstract head elements. Compared to non-abstract head elements described above, the use of an abstract head element enforces that the head element is replaced by one of the member elements. An example for an abstract head element definition is illustrated in Figure 8, letter C.

First, we dwell on a substitution group defined using an abstract head element. To represent the relationship between the abstract head element and its derived member elements the following approach is suggested. The head element is mapped to an ACC (see Figure 9, mark 1) and all member elements are mapped to ABIEs, which are based on the particular ACC (see Figure 9, mark 2). Careful readers might notice that the approach suggested is not valid since CCTS supports derivation by restriction only, whereas the member elements are derived from the head element by restriction or extension. Another encountered problem is the

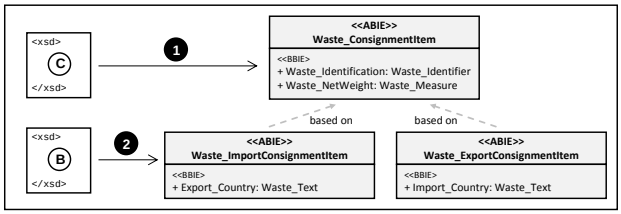

Figure 9: *Substitution Group* Mechanism (2/3).

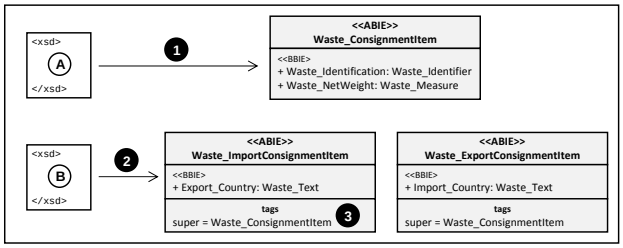

Figure 10: *Substitution Group* Mechanism (3/3).

issue of representing hierarchies consisting of multiple levels. An example is that a member element is at the same time the head element for another member element. These hierarchies are also not supported in CCTS, hence they cannot be represented at all.

On the contrary, substitution groups may be defined using a non-abstract head element. In this case the suggested approach is that the head element as well as the member elements are represented as ABIEs in the model representation. However, also the second approach is not entirely suitable, since hierarchy information is lost.

An alternative approach to allow the extension and restriction of member elements as well as to preserve the hierarchies within a substitution group is introduced in the following. The suggested approach introduces an additional tagged value for ABIEs named *super*. At first, the head element as well as the member elements are mapped to ABIEs in the model representation (see Figure 10, marks 1 and 2). Second, a tagged value named *super* is used to maintain the hierarchy information by storing a reference from each member element to it's head element in the tagged value (see Figure 10, mark 3).

4.2 Model Layer Mappings

In the previous Section we described how XML Schema constructs, which do not explicitly occur in CCTS, may be dealt with. Additionally, we now focus on concrete model elements and how their semantics may be preserved during the mapping of two schemas, i.e., some XML Schema based document model and CCTS. The following problem statements may not only be seen as mapping heuristics, but also to a certain extend as modeling guidelines for CCTS.

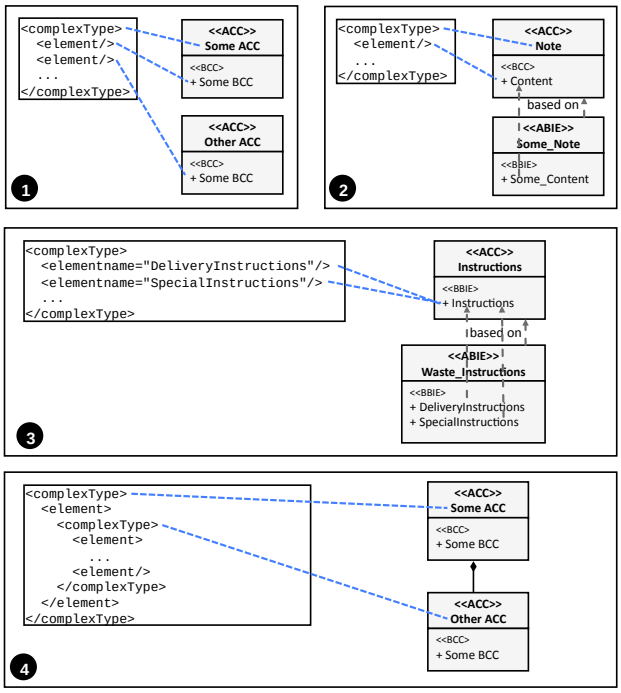

Figure 11: Conceptual illustration of model mappings.

For now, there do not exist modeling guidelines for CCTS, raising also issues for future work in this area. The aim of this Section is to show how XML Schema based document parts may be mapped to the existing Core Components Library (CCL) and how extensions to the CCL may be incorporated by some lightweight extension mechanisms. The general mapping heuristic for mapping arbitrary XML Schema models to CCL both, core components and business information entities, comprises the following steps:

1. Find an ACC, which includes most of the concepts modeled within a complex type.

2. Create a corresponding ABIE, which restricts on those BBIEs needed.

3. In case 1. and 2. cannot be fully applied, the following specific heuristics and problems may be considered:

ACC Combination Mappings. The mapping from one XML Schema complex type to only one corresponding ACC may sometimes not be sufficient. Instead, we may have to map one XML Schema complex type to more than one ACC. Thus, the mapping on the CCL side involves a combination of predefined building blocks. In practice this will most likely entail mappings from element declarations to BBIEs (cf. Figure 11, mark 1 for a generic visualization). For example, the information for a complex type called `ConsignmentItem` may be contained in an ACC `Consignment` and ACC `Item` from the CCL. This combination information must be stored within a mapping model for later use in XML Schema generation, in order to produce well defined schema documents. Otherwise, more than one complex type would be generated in the schema.

Generic Content Containers. The CCL provides some generic ACCs like `Note` with BCCs for content, subject, identification, and others. Wrapped in a business context of a BIE library, these components may be mapped to XML Schema elements (see Figure 11, mark 2). However, one has to be careful with data types of such BCCs as they might not match the ones used in the XML Schema. This may lead to type conversion problems. Also, generic core components have to be used with care and should not lead to constructs whose meaning is no longer clear or ambiguous.

Semi-Semantic Loss. Some BCCs in the CCL are so generic that they may be used for more than one contextualized BBIE. Multiple identifiers of some sort in an invoice document are a good example, where the reuse of one BCC makes sense. The distinction of the BBIEs based on just one BCC is achieved through qualifiers stating the specific business context. By using this BCC *overloading* we are able to map two different elements on one BCC that evolves into two BBIEs on the business information entity layer. Thus, the basic semantic of each BBIE is defined through the based on dependency to the underlying core component, but the semantic of the specifics of that element is not defined. Hence, it is lost during the mapping and modeling task. An example in the waste management context is given in Figure 11, mark 3.

Negative Mapping. In some cases concepts defined within a schema may not be mapped to any of the reusable building blocks provided by the CCL. In such cases we propose to allow the creation of completely new business information entities, which are not based on any existing ACCs. The meaning of the newly created and mapped ABIEs is not specified. This extension mechanism should be used wisely or interoperability issues may arise.

XML Schema Nesting. Both XML and XML Schema documents often show deeply nested structures. Especially XML Schema documents may contain complex types, which contain elements with complex types, which may again contain complex types, and so on. The same applies to sequences and choices, which further complicates the mapping to CCTS concepts. The problem here is basically the inverse of the problem statement *ACC Combination Mappings*. The mapping task is to find suitable core components for each of the complex types if possible in order to reproduce the structure of the XML Schema. The nesting of the mentioned XML Schema elements equals the use of aggregation and composition associations in the CCTS model (cf. Figure 11, mark 4). If some mappings cannot be established, the structure will be flattened on the CCTS side, resulting in less ACCs. Again, this information has to be

stored in a suitable mapping model, capturing the individual structure of an XML Schema based document model.

CCTS Nesting. Assume we encounter some composite structures in XML Schema, which are differently nested and composed through the aggregation relationship of ACCs in the CCL. Strictly following the CCTS approach it would not be possible to pertain these relationships because ASBIEs need to be based on some ASCC. However, in practice we may relax this restriction of the CCTS standard. For the composition of document parts on the business information entities layer, ASBIEs without corresponding ASCCs may be created.

4.3 Implementation and Case Study

This mapping approach is currently the basis for implementation in our business document engineering tool VIENNA Add-In [23]. Our work is accompanied by a broader case study involving the Austrian e-Billing standard ebInterface [1] for small to medium sized companies and public services. First findings and an evaluation of the applicability of our mapping model are presented and discussed at an international workshop [7].

5 Related Work

Mapping and interoperability issues, especially in the domain of E-Government, are not new and a considerable amount of work has already been done in this field. Concerning our work presented in this paper we may identify three different areas of related work, i.e, schema mapping and ontology alignment, metadata standards and interoperability frameworks.

Schema Mapping and Ontology Alignment. Schema mapping has its roots in database integration and the need to map heterogeneous data. A formal foundation for relational schema mapping is for example presented in [2, 8, 12]. A more general perspective of schema mappings is captured by [3], who define schema mappings upon some sort of formalism, which may be a relational schema or a UML model, and present a practical mapping language fostering the generation of transformation models. Another practical approach has been implemented by [9, 16] within their tool Clio, which is able to map relational schemas and XML Schema in order to generate transformations based on several technologies for the concrete data. When dealing with ontologies, mappings are in general not the basis for transformation and created manually, but computed automatically based on similarities - see [6]. Here, the mapping task results more or less in a matching task. All this work is closely related to ours in the sense that semantic equivalent elements are to be precisely mapped. However, we focus on preserving the structure of the mapped schemas to support roundtrip engineering. Because of the very different structures of the incorporated schemas we apply heuristics upon the mappings to transform instances of the schemas.

Metadata Standards. There exist several technologies and frameworks for the support of metadata and metadata registries [10]. Most related to the UN/CEFACT's CCTS is the ISO 11179 [11] standard for metadata registries. In fact the basic concept of a generic data element in ISO 11179 has been adopted by CCTS in terms of BCC and BBIE properties. To support ASCC and ASBIE properties CCTS extends the concept of a generic data element. However, the concept of a conceptual data element is not supported by CCTS. One of the advantages of using CCTS is the availability of the CCL, which provides a commonly agreed set of reusable document building blocks with defined semantics. Furthermore, the UML Profile for Core Components facilitates the integration of core component concepts in widely available UML modeling tools.

A popular metadata framework in Model Driven Development to support modeling and registry tasks is the Eclipse Modeling Framework with its metamodeling language Ecore [18]. The framework is closely related to MOF, another metadata standard, managed by the OMG.

In the ontology domain there exist also approaches to manage metadata. See for example the W3C recommendation SAWSDL [25] or the Dublin Core Metadata Initiative [5]. However, these technologies do not provide a standardized library of reusable, on a conceptual level defined, elements.

Interoperability Frameworks. Interoperability issues in G2G or B2B arise because of the use of different technologies and data models. Therefore, lots of research to solve these issues has been undertaken. Saekow et al. [17] present an interoperability framework focusing on their tool, which shall close the gap between conceptual and practical interoperability approaches. In [4] the authors describe how the ISO 11179 standard may be interpreted and extended in order to overcome certain restrictions of this standard and maintain multiple views on the same data elements.

6 Conclusion and Future Work

In this paper we introduced the concepts of the Core Component Technical Specification (CCTS) and showed how these concepts may be used to create global data definition models. Based on context independent core component models, domain specific business information entities may be derived. We also showed that the classic forward engineering approach, where business document models are created from scratch from a core component/business information entity definition, is rather straightforward. The more challenging approach is the adherence of existing, XML based business document definitions to an existing

core component model. We referred to this task as backward engineering. However, without appropriate rules and definitions a successful backward engineering of existing XML Schema constructs to a Core Component model is not possible.

In this paper we closed the gap between existing definitions and Core Components by introducing a well defined heuristic for the mapping of arbitrary document models based on XML Schema to the CCTS conforming document models. We have shown how arbitrary XML Schema constructs may be mapped to the predefined structures of the Core Component Technical Specification. Our experiments, using a real world example from the domain of cross-border waste transport, have proved that these mappings leverage the integration task of two existing document models.

References

[1] AustriaPRO. ebInterface. http://www.ebinterface.at/.

[2] P. A. Bernstein and H. Ho. Model Management and Schema Mappings: Theory and Practice. In *Proceedings of the 33rd international conference on Very large data bases (VLDB '07)*, pages 1439–1440. VLDB Endowment, 2007.

[3] A. Blouin, O. Beaudoux, and S. Loiseau. Malan: A Mapping Language for the Data Manipulation. In *Proceedings of the 2008 ACM Symposium on Document Engineering, Sao Paulo, Brazil*, pages 66–75, 2008.

[4] J. Davies, S. Harris, C. Crichton, A. Shukla, and J. Gibbons. Metadata Standards for Semantic Interoperability in Electronic Government. In *Proceedings of the 2nd International Conference on Theory and Practice of Electronic Governance (ICEGOV '08)*, pages 67–75, New York, NY, USA, 2008.

[5] DCMI. *Dublin Core Metadata Initiative Abstract Model*. `http://dublincore.org/documents/abstract-model/`, Last Visit: August 2009, 2007.

[6] M. Ehrig and Y. Sure. Ontology mapping - an integrated approach. In *Proceedings of the First European Semantic Web Symposium*, volume 3053, pages 76–91, Heraklion, Greece, 2004.

[7] C. Eis, P. Liegl, C. Pichler, and M. Strommer. An Evaluation of Mapping Strategies for Core Components. In *Proceedings of the International Workshop on Services Computing for B2B, Bangalore, India, To Appear0*, 2009.

[8] R. Fagin, P. G. Kolaitis, A. Nash, and L. Popa. Towards a Theory of Schema-Mapping Optimization. In *Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS '08)*, pages 33–42, New York, NY, USA, 2008.

[9] L. M. Haas, M. A. Hernández, H. Ho, L. Popa, and M. Roth. Clio Grows Up: From Research Prototype to Industrial Tool. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data (SIGMOD '05)*, pages 805–810, 2005.

[10] S. Harris, J. Gibbons, J. Davies, A. Tsui, and C. Crichton. Semantic Technologies in Electronic Government. In *Proceedings of the 2nd International Conference on Theory and Practice of Electronic Governance (ICEGOV '08)*, pages 45–51, New York, NY, USA, 2008.

[11] ISO/IEC. *ISO/IEC 11179-1, Information Technology - Metadata Registries (MDR) - Part 1: Framework*, 2004.

[12] P. G. Kolaitis. Schema Mappings, Data Exchange, and Metadata Management. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS '05)*, pages 61–75, New York, NY, USA, 2005.

[13] F. Legler and F. Naumann. A Classification of Schema Mappings and Analysis of Mapping Tools. In *Proceedings of Datenbanksysteme in Business, Technologie und Web (BTW 2007)*, volume 103 of *LNI*, pages 449–464. GI, 2007.

[14] P. Liegl. Conceptual Business Document Modeling using UN/CEFACT's Core Components. In *Proceedings of Conceptual Modelling 2009, Sixth Asia-Pacific Conference on Conceptual Modelling (APCCM 2009), Wellington, New Zealand*, pages 59–69, 2009.

[15] J.-M. Nurmilaakso. EDI, XML and e-Business Frameworks: A Survey. *Comput. Ind.*, 59(4):370–379, 2008.

[16] L. Popa, Y. Velegrakis, M. A. Hernández, R. J. Miller, and R. Fagin. Translating Web Data. In *Proceedings of the 28th international conference on Very Large Data Bases (VLDB '02)*, pages 598–609, 2002.

[17] A. Saekow and C. Boonmee. Towards a Practical Approach for Electronic Government Interoperability Framework (e-GIF). In *Proceedings of the 42nd Hawaii International Conference on System Sciences (HICSS '09)*, pages 1–9, Washington, DC, USA, 2009.

[18] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks. *EMF: Eclipse Modeling Framework 2.0*. Addison-Wesley Professional, 2009.

[19] UN/CEFACT. *UN/CEFACT's Core Component Library UN/CCL*. `http://www.unece.org/cefact/codesfortrade/codes_index.htm`.

[20] UN/CEFACT. *XML Naming and Design Rules 3.0, ODP5*. United Nations Center for Trade Facilitation and Electronic Business, 2008.

[21] UN/CEFACT. *Core Components Technical Specification 3.0*. `http://www.untmg.org/ccts/spec/3_0`, Last Visit: May 2009, 2009.

[22] UN/CEFACT. *UML Profile for Core Components Technical Specification 3.0*. `http://www.untmg.org/upcc/spec/3_0`, Last Visit: May 2009, 2009.

[23] VIENNA Add-In. *A B2B collaboration modeling Add-In for Sparx Systems Enterprise Architect*. `http://code.google.com/p/vienna-add-in/`.

[24] W3C, `http://www.w3.org/TR/xmlschema-0/`, Last Visit: June 2009. *XML Schema Part 0: Primer Second Edition*, 2004.

[25] W3C. *Semantic Annotations for WSDL and XML Schema*. `http://www.w3.og/TR/sawsdl/`, Last Visit: August 2009, 2007.