A registry model for UN/CEFACT's Core Components

Christian Huemer, Philipp Liegl
Institute of Software Technology and Interactive Systems
Vienna University of Technology
Vienna, Austria
{huemer, liegl}@big.tuwien.ac.at

Christian Pichler
Research Studios Austria
Vienna, Austria
christian.pichler@researchstudio.at

***Abstract*—Business documents exchanged in a service-oriented context play a crucial role in the definition of service interfaces. Only if both partners have a common agreement on the data exchanged, automated business interactions are possible. The United Nations Center for Trade Facilitation and Electronic Business (UN/CEFACT) provides a reliable and interoperable solution for conceptual business document definitions with UN/CEFACT's Core Components Technical Specification. However, document definitions aren't available in a centralized manner to all interested business partners. In order to allow for an easy search and retrieval of core component business document definitions, a registry is needed. Business partners may retrieve service interface definitions from the registry and adapt their software accordingly in order to engage in automated business interactions. We specify a registry meta model on top of ebRIM, registering core component artifacts and defining their interdependencies.**

I. Introduction and Motivation

Today business processes must be designed to support the value exchanges between companies, and IT applications must quickly adapt to changing business processes. This paradigm is also known as business/IT alignment. With their potential to provide a new level of flexibility in regard to the adaptation of the affected IT systems, service oriented architectures are a promising solution for the business/IT alignment problem. In former days change requests to the IT resulted in rigorous reengineering tasks of existing IT implementations. Nowadays service oriented IT departments face the challenge of aligning their service interfaces. This term is also known as service alignment and refers to the reconcilement between business partners in order to provide complementary services.

Before two business partners can engage in an automated business interaction an agreement on the exchanged data i.e. business document structure has to be found. In most SOA based scenarios XML schemas are used to define the interface of each business partner. Thus, it are the business document definitions, which essentially define what type of XML instances a service interface accepts. If both business partners support the same XML schema, XML instances may be exchanged between the different systems via service interfaces. However, even if predefined business document standards [1] are used, it is not guaranteed that all participating business partners share a common understanding of a particular business document. Most business document standards are overloaded with optional elements, since they try to meet as many different user requirements as possible. As a result, costly and error prone mappers have to be implemented. We argue that a single approach for the definition of service interfaces is needed, where every business partner may easily retrieve the necessary definitions from a repository. Additionally, business partners may store their existing business document definitions for a service interface in the repository in order to share them with other potential business partners.

Figure 1 gives an overview of our motivating business scenario.

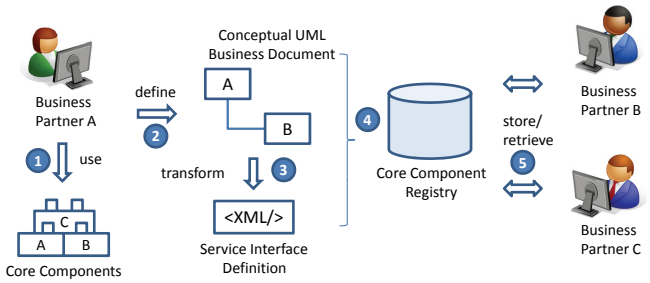

Fig. 1: Motivating Business Scenario

Business partner A uses predefined building blocks (1) in order to assemble a conceptual business document definition (2). These so called core components are standardized by the United Nations Center for Trade Facilitation and Electronic Business (UN/CEFACT) and follow the Core Components Technical Specification (CCTS) [2]. Using the core component concept business documents are built in a semantically unambiguous manner, following a globally defined standard. However, core components are standardized in an implementation neutral manner, making integration into modeling tools and machine processing difficult. In order to overcome this limitation we have introduced the UML Profile for Core Components (UPCC) [3] and consequently submitted it to UN/CEFACT for standardization. One may now use the UML Profile and assemble business document definitions on a conceptual, UML based level (2). A UML based business document definition is easy to communicate between different developers and IT architects. In particular in a service oriented context, where several negotiation steps between the participating partners are

necessary, a UML based approach is superior compared to a sole XML based document definition.

However, service interfaces are defined using XML Schema artifacts. Using the UN/CEFACT's Naming and Design Rules (NDR) [4] a conceptual business document model may be transformed into an equivalent XML syntax (3). Both representations serve their specific purpose. Conceptual service interface definitions ease the communication between developers and IT architects, but cannot be used directly for interface definitions in IT systems. XML based service interface definitions are difficult to communicate between different developers, but may be directly used in IT systems. Thus, we argue that it is necessary to store both - the conceptual and the XML based representation of a service interface in a registry (4). Other business partners may search the registry for service interface definitions and retrieve both, the UML based and XML based business document definitions (5).

We argue, that in order to foster reuse of business document definitions, which serve as interface definitions in a SOA, a registry approach is needed. However, such a registry approach is still missing. In this paper we show how a core components registry may be built on top of the ebXML Registry Information Model (RIM) [5]. We outline how the registry serves as the storage, search, and retrieval point for both, conceptual service interface definitions based on UML and XML Schema artifacts. We also stress the importance of a registry federation concept in order to allow different levels of interoperability.

We already introduced a registry approach in [6], covering the business, business process, and deployment artifact perspective of an electronic business interaction between business partners in a service oriented context. For the business process layer artifacts the used a dedicated business collaboration registry model [7]. However, our introduced registry model did not consider the business document information that is being exchanged in an electronic business interaction. This paper closes this gap and provides an extension to the concepts presented by [6] in order to allow for support of business document information in an e-Business registry as well.

The remainder of this paper is structured as follows: Section II introduces the basic core component concepts necessary for further conception of the paper. The registration of service interface definitions in a registry is discussed in Section III and different registration federation concepts are introduced in Section IV. Finally Section V concludes the paper.

II. UN/CEFACT's Core Components

The Core Components Technical Specification (CCTS) [2] distinguishes between two elementary concepts: core components and business information entities. Core components are reusable building blocks for assembling business documents in an implementation neutral manner. Thereby, core component are context independent, and due to their generic nature may be used in any given application scenario. A business document modeler distinguishes between three different types of core components: (i) *basic core components (BCC)* are used to represent simple properties such as name or description (ii) *aggregate core components (ACC)* are used to aggregate different basic core components to a more complex type e.g. address, consignment etc. *(iii) association core components (ASCC)* are used to define relationships between different ACC e.g. indicate the fact, that a party has a private and a work address.

If core components are used in a certain application domain, they become so called business information entities. Business information entities are derived from core components by restriction. Thus, a business information entity must not contain any attributes or associations, which have not been defined in the underlying core component. Similar to the concept of core components a distinction is made between *basic business information entities (BBIE)*, *aggregate business information entities (ABIE)*, and *association business information entities (ASBIE)*. Figure 2 gives an overview of the basic concepts of the core component standard.

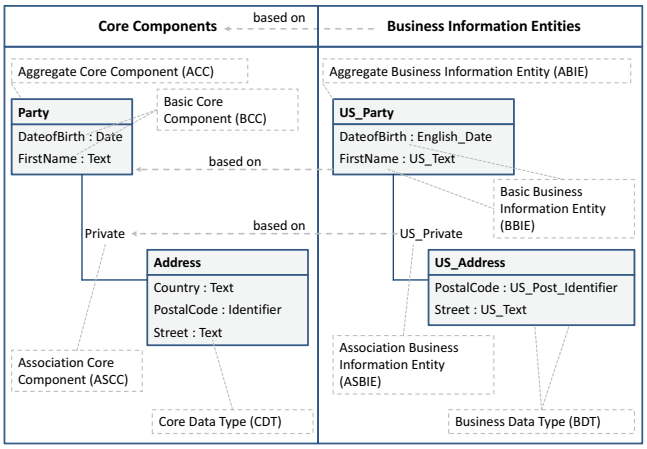

Fig. 2: Overview of basic core component concepts

The value domain of basic core components and basic business information entities is defined using the concept of data types. A *core data type (CDT)* is used to set the value domain of a basic core component and a *business data type (BDT)* is used to set the value domain of a basic business information entity e.g. `US_Post_Identifier` defines the value domain for the basic business information entity `PostalCode` on the lower right hand side of Figure 2. Similar to the relationship between core components and business information entities, a business data type is derived from a core data type by restriction. Both, CDTs and BDTs are composed of exactly one *content component (CON)*, which carries the actual value of the data type e.g. 12. Zero or more *supplementary components (SUP)* provide additional meta-information about the content component e.g. *unit code = Fahrenheit* of the content component's value.

A major shortcoming of core components is the fact that they are defined in an implementation neutral manner. Thus, integration into modeling tools is difficult. In order to apply the core component concepts as a means of modeling business documents, We have developed the UML Profile for Core Components (UPCC) [3] and consequently submitted it to

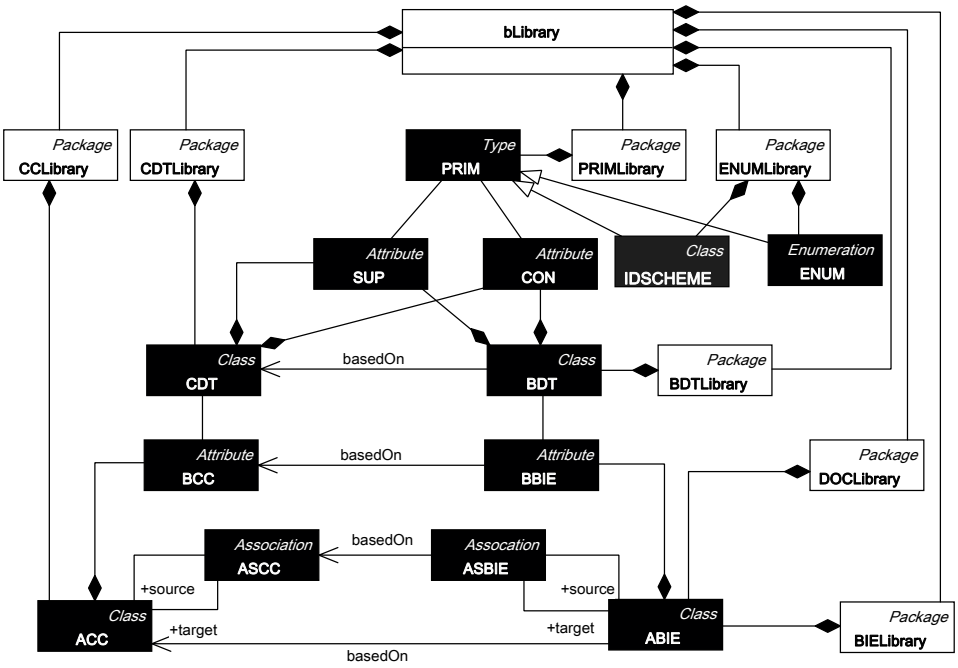

Fig. 3: UML Profile for Core Components Overview

UN/CEFACT for standardization. The UPCC allows using core components with standard UML modeling tools. Conceptual core component models may be easily exchanged between different business partners and stakeholders in order to communicate business document definitions. Figure 3 gives a brief overview of the structure of the UML Profile for Core Components. Denoted with a white background are packages which embrace artifacts of a certain type. Artifacts such as core components and data types are shown with a black background.

The central package and embracing container for all other component related packages is a business library (`bLibrary`), shown on top of Figure 3. A particular role plays the package `DOCLibrary`, shown on the lower right hand side of Figure 3. In a `DOCLibrary` different business information entities are aggregated to business document definitions. For a detailed discussion of the UML Profile for Core Components see [8].

For transforming UML based core component models to XML Schema deployment artifacts, UN/CEFACT developed a specification named the XML Naming and Design Rules (NDR) [4]. The specification describes rules and guidelines for representing conceptual models comprised of core components, business information entities, as well as business document definitions through XML Schema. Figure 4 gives an overview of the basic transformation mechanisms as mandated by the Naming and Design Rules.

In particular, the figure describes rules for the business information entity concepts including business data types (BDT), basic business information entities (BBIE), aggregate business information entities (ABIE), as well as association business information entities (ASBIE). The concept of a BDT is represented through the XML Schema construct `simpleType`

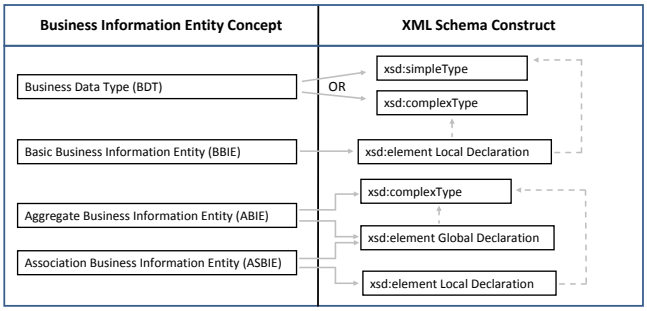

Fig. 4: Transformation of CCTS to XML Schema components.

or `complexType`. In case a BDT only contains a content component, it is represented through a `simpleType`. If a BDT contains supplementary components as well, it is represented using a `complexType`. An ABIE is represented through the XML Schema construct `complexType`, which contains local element declarations representing the BBIEs of the ABIE. The BBIEs are either typified through a `simpleType` or a `complexType` representing a particular BDT. In addition to the complex type definition, a global element declaration is created for each ABIE. For representing ASBIEs, which are used for defining associations between ABIEs, either local or global element declarations are used. In case the association is of type shared, a global element is defined for the ASBIE which is referenced from within the complex type definition of the ABIE. The transformation of core components to XML Schema constructs is similar to the approach described for business information entities. For a detailed discussion of the XML Schema derivation mechanism for conceptual core component models we would like to direct the interest reader

to [8].

Another aspect of generating XML schemas from conceptual models is the structure of the resulting XML schemas. Core component artifacts are, as introduced earlier, organized in different libraries. The libraries relevant in respect of BDTs, ABIEs, ASBIEs, as well as business document definitions, are the packages named `BDTLibrary`, `BIELibrary`, and `DOCLibrary`. According to the package structure and following the Naming and Design Rules (NDR), the generated XML Schema representation is organized into three separate XML Schema files. An overview of the generated deployment artifacts is provided in Figure 5 showing that a separate XML Schema file is generated for each library.

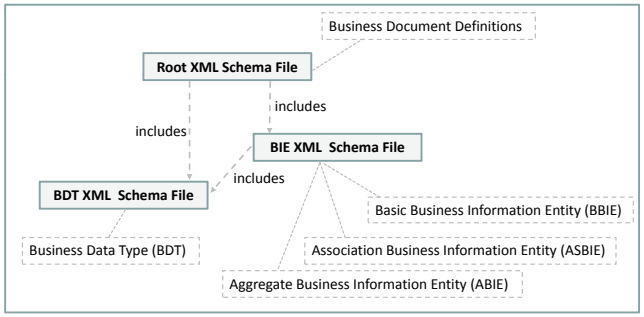

Fig. 5: Deployment Artifacts

III. The Core Component Registry Model

Having introduced the basic core component concepts we now discuss how both, conceptual core component artifacts based on UML and logical level core component artifacts based on XML Schema are managed in a registry. For this purpose we provide a registry meta-model based on the ebXML registry information model [5], which supports the specifics of the conceptual and logical layer. The registry meta-model has the purpose to define which artifacts are maintained in the registry. An ebXML registry stores these artifacts as *extrinsic objects*, which are XMI (XML Metadata Interchange) and XML artifacts in our case, but may in principle be any format of choice. It is important to notice, that the content of an extrinsic object is encapsulated - this means a query to the registry does not access the content of an extrinsic object directly. It follows, that an extrinsic object must be annotated with pertinent meta-data in order to allow for an effective search. Additionally, the different artifacts and their meta data have dependencies on each other. Our registry meta model defines the required links between the extrinsic objects of the different artifacts and also between their meta data if required.

Figure 6 shows the resulting meta-model of our core component registry. Due to space limitations we do not show all artifacts of the meta model, but limit our discussion to the business information entity part of the meta model. Extrinsic objects are denoted with a thick border and classes referring to logical level artifacts are denoted with a gray background. Each class of our core component registry is based on an existing meta class of ebRIM. The meta-class is denoted in the upper-right corner of each class.

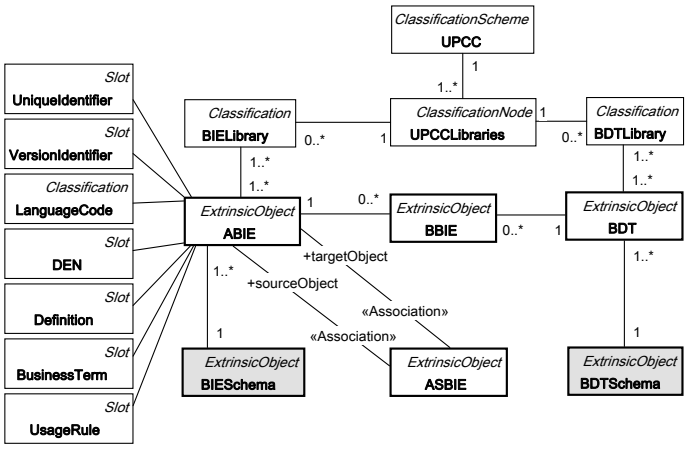

Fig. 6: Cut-out: Core Component registry meta model

As shown in Figure 6 our meta model cut-out contains four extrinsic objects for the conceptual layer, namely `ABIE` (aggregate business information entity), `BBIE` (basic business information entity), `ASBIE` (association business information entity), and `BDT` (business data type). Each of the four mentioned extrinsic objects has several associated *slots* and *classifications*. The concepts of *slots* and *classifications* is used to annotate an extrinsic object with the pertinent meta data information in order to allow for search and retrieval of the artifact.

Due to space limitations we only show the slots and classifications for the extrinsic object `ABIE` in detail, which however also apply to `BBIE`, `ASBIE`, and `BDT` artifacts. For the logical level layer we define exactly two extrinsic objects namely `BIESchema` and `BDTSchema`. The extrinsic objects `ABIE` and `BDT` are associated with the *classification* artifacts `BIELibrary` and `BDTLibrary`, respectively. *Classification scheme* and *classification node* are concepts built into the ebRIM and are used for defining taxonomies of meta data.

A. Registering conceptual core component models

In our proposed scenario a conceptual core component model is based on the UML syntax. The graphical UML syntax may also be represented in XMI (XML Metadata Interchange). The XMI representation of a core component model is used to store it as extrinsic object in our registry. Thereby, we distinguish between two different use cases: storing an entire core component model and storing a single core component.

Figure 7 gives on overview of how the business information entity artifacts shown in Figure 2 are stored in the core component registry. We denote *classification* and *slot* artifacts using a dark background in order to foster distinction from *extrinsic objects*. In most use cases a user may want to retrieve or store a single business information entity artifact from the registry. In case an aggregate business information entity such as `US_Person` is stored in the registry, its XMI representation is stored in the extrinsic object `ABIE`. The basic business

information entities DateofBirth and FirstName are stored in the respective extrinsic objects BBIE. The same applies for the association business information entity US_Private, which is stored in the extrinsic object ASBIE. Business data types, defining the value domain of basic business information entities are stored in their respective extrinsic object BDT. In order to indicate to which library a given business information entity or business data types belongs, we use the classification BIELibrary and BDTLibrary.

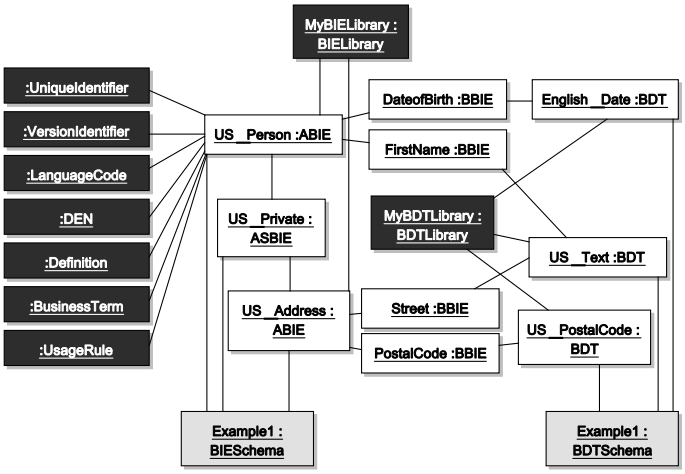

Fig. 7: Core Component registry example

In case an entire business information entity model is stored in the registry the same steps as for storing a single business information entity are applied. However, the registry has to check whether a given business information entity already exists in the registry, before inserting a new definition. For this purpose the unique identifier slot is used.

If core components are stored in the registry the same principles as shown for business information entities in Figure 7 are applied. Of particular importance is the establishment of the correct dependencies between core components and business information entities in the registry. For example a business information entity must not be inserted into the registry if no connection to an underlying core component definition can be provided, since that would violate one of the main principles of the core component approach.

B. Registering logical level core component artifacts

In the previous paragraph we showed how conceptual core component artifacts are stored in the registry. Consequently we show how to map deployment XML Schema artifacts to the core component registry model and link them to the business information entity artifacts. As outlined in Section II the UML based core component model may serve as the basis for the derivation of XML Schema artifacts, which are used to define an interface in a SOA scenario. For the derivation of XML Schema artifacts from core component models transformers such as our freely available VIENNA Add-In [9] are necessary.

However, in certain cases a user may want to retrieve a ready to use XML schema file with business information entity definitions instead of retrieving conceptual business information entity definitions. A user may also want to avoid the creation of XML Schema artifacts from newly created business information entity models, but use predefined and ready-to-use XML Schema artifacts instead. In Section II we have shown how a single library of business information entities results in exactly one XML Schema file. Consequently, the business data types used for the business information entities also result in a single XML Schema file. Therefore, in case a user submits an entire business information entity library to the registry, it is recommended to submit the underlying XML schema files as well. As shown in Figure 7, we use the two extrinsic objects BIESchema and BDTSchema in order to store business information entity schema files and business data type schema files, respectively. Both extrinsic objects are associated with the business information entity artifacts and business data type artifacts they belong to.

Since users are able to store XML Schema files together with conceptual definitions an easy search and retrieval of XML Schema artifacts using the definitions on the conceptual level such as the slots UniqueIdentifier, Definition, etc. is possible. However, users may also search for pertinent business information entities by retrieving all core components from a certain business information entity library using the classification BIELibrary. In either case the user is able to retrieve the right business information entities from the registry together with their XML Schema equivalent.

IV. Registry Federation

One of the early and most successful attempts to provide a single set of business document definitions is the UN/EDIFACT standard. However, a single central registry or repository of reusable business document artifacts has a set of shortcomings. Several hundred interest groups and stakeholders from different industries actively contribute to the UN/EDIFACT standard, making the resulting message definitions quite complex. Furthermore, the standard definition is rather rigid in nature and change requests usually have to go through several review cycles by all involved partners.

In order to avoid these shortcomings we propose a federated registry approach for core components. Figure 8 gives an overview of our approach. Core components are standardized and harmonized by UN/CEFACT, serving as the single entity shown on top of Figure 8. Different interest groups such as SWIFT (Society for Worldwide Interbank Financial Telecommunication) [10] or CIDX (Chemical Industry Data Exchange Standard) [11] and entire industry sectors (Automotive industry) represent the needs of their involved companies and stakeholders. Each interest group maintains its own core component library, which is aligned to the UN/CEFACT library. Companies such as Shell or BP retrieve their core component definitions directly from their interest group registry (CIDX), instead of the generic UN/CEFACT library. Additionally, industry sector specific libraries such as for the automotive industry are created. This ensures core component compatibility for sub-groups of the industry domain such as

AIAG (North American car industry) and ODETTE (European car industry).

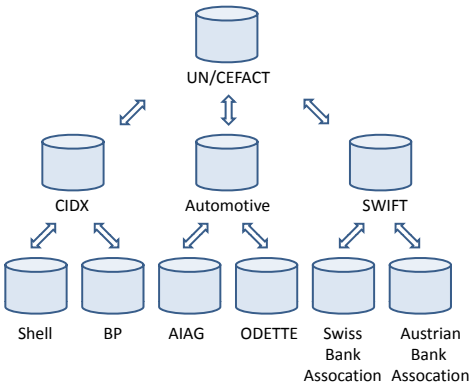

Fig. 8: Federated registry approach

The advantages of such a federated approach are apparent. Companies do not need to use generic and overloaded core component definitions, but may use core component definitions which are already tailored to their specific industry domain. If industry groups such as CIDX align their industry specific registries to the generic UN/CEFACT core component library, business document definitions may easily be mapped between different industries e.g. between SWIFT and CIDX as shown in Figure 8. However, an alignment of industry specific libraries towards the generic UN/CEFACT library is not imperative. In certain scenarios industries may choose to create their own core components. An automotive industry representation such as AIAG may chose to create its own core component for the automotive supply chain, because alignment of these core components to the financial industry may be considered as unnecessary. However, in such a case interoperability of core components at a cross-industry level is not feasible any more. Nevertheless, core components from the upper level (UN/CEFACT) may still be used if interoperability to other industries is required.

Finally, each enterprise may choose to implement its own core component definitions in a dedicated enterprise-wide registry. As shown at the bottom of Figure 8, Shell may for instance choose to implement its own core component definitions, which are valid for the whole enterprise. Such a step ensures interoperability between different company departments and sub-groups such as Shell Asia and Shell Europe.

V. Conclusion

As outlined in this paper an efficient service interface alignment is crucial for the successful implementation of a SOA. Only if both business partners are able to find a common agreement on their service interface definitions, an automated data exchange between the business partners is possible. Service interfaces are essentially defined by the type of business document they accept. Thereby, UN/CEFACT's Core Components are a powerful and easy to use concept for the unambiguous definition of business documents. In order to process implementation neutral core components we have introduced the UML Profile for Core Components (UPCC). Using the UPCC it is possible to define service interfaces on a conceptual, UML based level. The UML based service interface definitions may be easily communicated between the different stakeholders in a service oriented environment. With the use of Naming and Design Rules the conceptual service interface definitions may be transformed to XML Schema. These XML artifacts are used for the definition of service interfaces on a technical level. Thus, both representations of service interfaces are needed - conceptually for the communication between stakeholders and in an XML format for the respective IT systems.

Although the technical prerequisites for service definitions exists with the UML Profile for Core Components, an efficient alignment of different service interfaces is still not possible. With the core component registry, introduced in this paper, we tackle this problem and provide a central access point for service interface definitions. Stakeholders may retrieve interface definitions from the core component registry and reconcile their service interfaces. We built our registry approach on the well accepted electronic business registry information model (ebRIM). Thereby, we provide a registry meta model supporting both - conceptual and XML based service interface definitions. Since we base our approach on the global core component standard, a maximum of interoperability is provided.

Currently the versioning of different service interface definitions is not reflected in our proposed solution. As time evolves, the requirements of business partners most likely change and thus and adoption of the corresponding business documents becomes necessary. The aspect of proper versioning as well as business document evolution strategies remains unresolved issues and will be subject to future research.

References

[1] H. Li, "XML and Industrial Standards for Electronic Commerce," *Knowledge and Information Systems*, vol. 2, no. 4, pp. 487–497, 2000.
[2] UN/CEFACT, *Core Components Technical Specification 3.0*, 2009, http://www.untmg.org/ccts/spec/3_0.
[3] UN/CEFACT, *UML Profile for Core Components Technical Specification 3.0*, 2009.
[4] UN/CEFACT, *UN/CEFACT's Naming and Design Rules 3.0*, 2009.
[5] OASIS, *ebXML Registry Information Model (RIM) 3.0*, 2005.
[6] C. Huemer, P. Liegl, R. Schuster, and M. Zapletal, "A 3-level e-Business Registry Meta Model," in *Proceedings of the IEEE International Conference on Services Computing (SCC08), July 8-11, Honolulu, HI, USA*, 2008, pp. 441–450.
[7] B. Hofreiter, C. Huemer, and M. Zapletal, "A Business Collaboration Registry Model on Top of ebRIM," in *IEEE International Conference on e-Business Engineering (ICEBE06), October 24-26, Shanghai, China*, 2006, pp. 392–400.
[8] P. Liegl, "Conceptual Business Document Modeling using UN/CEFACT's Core Components," in *Proceedings Sixth Asia-Pacific Conference on Conceptual Modelling (APCCM), January 20-23, Wellington, New Zealand*, 2009, pp. 59–69.
[9] VIENNA Add-In development team, "The VIENNA Add-In," 2009, http://code.google.com/p/vienna-add-in/.
[10] SWIFT, *Society for Worldwide Interbank Financial Telecommunication*, 2009, http://www.swift.com/.
[11] CIDX, *Chemical Industry Data Exchange Standard*, 2007, http://www.cidx.org.