Towards a global business document reference ontology

Philipp Liegl, Christian Huemer, Marco Zapletal
Vienna University of Technology
Favoritenstrasse 9-11/188
1040 Vienna, Austria
{firstname.lastname}@tuwien.ac.at

Abstract

In the field of business document standardization a multitude of different standardization efforts exist. Unfortunately, most of the developed standards are designed for a specific application domain or industry and do not consider cross-standard interoperability. This results in several incompatible standard definitions. Without the provision of a common semantical basis for business document definitions, cross-domain interoperability cannot be achieved. In this paper we provide a methodology for building a global reference ontology based on the Core Components Technical Specification (CCTS) and Web Ontology Language (OWL). Using our approach a common semantic business document basis is developed. New document definitions may be derived from this basis and existing document definitions may be aligned to it. Using our 'derivation-by-restriction' mechanism, instances derived from a common semantical basis are interoperable to each other. Thus, mapping mechanisms between different standard definitions may be implemented in an easier and semantically correct manner.

1. Introduction

In the field of business document standardization several different approaches and standards have emerged over the past few years. Most of these standardization approaches focus on the definition of a common syntax definition for business documents. In most cases the common syntax is based on XML Schema. A multitude of different business document standards exist nowadays, which are mostly incompatible to each other [2]. In case two business partners want to engage in an automated B2B interaction they either have to support the same business document standard or they have to implement a costly syntactical mapping mechanism from standard *A* to standard *B*. As a consequence of the sole focus on a syntactical format and the need for implementing costly mappers, the requirement for a common semantic basis for business document standards emerges.

In the field of semantic business document interoperability we identify two important use case scenarios. On the one hand side, business documents are standardized in a top-down manner. Thereby, a global reference ontology to which business partner *A* and business partner *B* have to adhere to must be provided. Thus, any local ontologies aligned with the global reference ontology are compatible to each other and both business partners have a common understanding of what they actually exchange in a business transaction. On the other hand side, business partners might already have their own local implementations and ontology definitions. In such a bottom-up scenario business partner *A* and business partner *B* map their existing local ontologies to the global reference ontology. In both cases the provision of a global reference ontology is of utmost importance in order to provide interoperability in a heterogeneous environment.

In the paper at hand we propose an approach for defining a global reference ontology based on the Core Component Technical Specification (CCTS) [7] from the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) and the Web Ontology Language (OWL). Core components are reusable building blocks, for defining business documents. However, they have two factors making a concrete implementation difficult. Core components are defined independent of any implementation syntax and the standard itself is defined using English prose. This non-formalization makes automated processing of core component concepts impossible. Our paper will show how core component concepts are formalized using Web Ontology Language in order to provide means for building a global reference ontology. Thereby we go beyond existing approaches based solely on RDFS [1] and use the mightiness of Web Ontology Language as the representation format of choice for a reference ontology.

The remainder of this paper is structured as follows: Section 2 gives an overview of the basic concepts of core components. Section 3 introduces the core component reference ontology in detail and Section 4 shows how a document instance is mapped to the reference ontology. Finally Section 5 concludes the paper with a final assessment.

2. Core Components at a glance

The central idea of core components is the definition of semantic building blocks for assembling business documents in a reusable manner. Different business document definitions are aggregated in a central library, independent of any application domain. Industry specific solutions are able to use the shared components from the library and tailor them to the specific needs of their application domain. The tailoring mechanism only allows a derivation of domain specific artifacts from core components by restriction. Thus, it is guaranteed that all industry specific artifacts are based on the same semantical document base - namely core components.

The research presented in this paper is based on the most recent version of the standard definition - UN/CEFACT's Core Components Technical Specification 3.0 (CCTS 3.0) [7]. In contrast to other standardization approaches the core component standard is defined independent of a specific implementation format. Furthermore, core components are defined using English prose. Thus, they are missing a formalized representation allowing automatic processing of core component information. Other business document standards mostly use XML schema as their representation format of choice. In regard to the formalization of the core component standard at present there exists only one approved and semi-formalized representation for Core Components - the UML Profile for Core Components (UPCC) [8] [3]. We will use the UML representation for visualization purposes in this Section.

The core components standard distinguishes between two elementary concept families: Core components and business information entities. Core components are independent of any specific context and are defined in a generic and reusable manner. As such they serve as the basis for industry specific artifacts - so called business information entities.

Core Components The main idea behind core components is the identification of objects and of object properties. Object properties are classified in two categories: simple object properties (text, number, date) and complex object properties. We refer to object types as aggregate core components - so called ACCs. An aggregate core component serves as a container for simple properties - so called basic core components (BCC) - and aggregates them to a higher entity. Each simple property has a data type known as core data type (CDT). A core data type defines the exact value domain of a given basic core component. Association core components (ASCC) represent complex properties and are used to pin out dependencies between different aggregate core components.

The left hand side of Figure 1 shows a cut-out from a UML representation of a core component model. The sample model includes two aggregate core components (ACC) `Invoice` and `LineItem`. The ACC `Invoice` contains several basic core components (BCC) such as `InvoiceIdentifier`, `CountryIdentifier`, `Description` etc. In a real world example an invoice would contain more basic core components - for presentation purpose, however, they have been left out as indicated by the term 'and x other attributes'. Moreover, `Invoice` has exactly one association core component (ASCC) named `item` which denotes, that an `Invoice` contains zero to many `LineItems`.

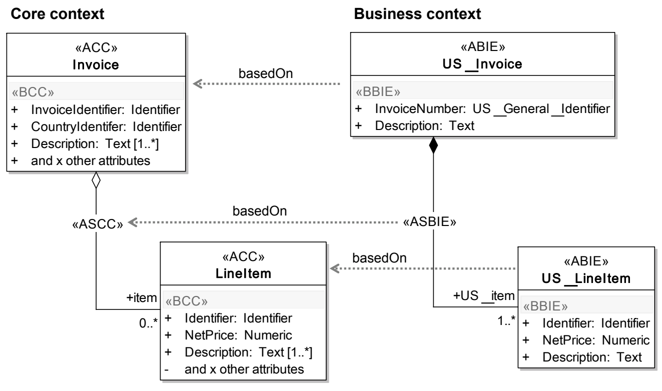

Figure 1: Core Components Overview

Each basic core component has a certain value domain, which defines the data type of the attribute. For basic core components core data types (CDT) are used in order to set the value domain. The concept of a core data type in the domain of core components distinguishes between two different types - content components (CON) and supplementary components (SUP). Each core data type has exactly one content component and may have zero to many supplementary components. Thereby a content component holds the actual information value (e.g. 12) and supplementary components provide additional meta information about the actual information (e.g. type = temperature, measurement = Celcius etc.).

As shown in the example in Figure 1, the basic core component (BCC) `InvoiceIdentifier` in ACC `Invoice` has the core data type `Identifier`. The exact outline of the core data type `Identifier` is shown on top of Figure 2. `Identifier` has exactly one content component named `Content` and four supplementary components providing additional information about the actual identifier value. Both, content components and supplementary components have a specified data type we refer to as primitive type (PRIM). Primitive types are defined by UN/CEFACT in the core component data type catalogue [9] and denote the exact value domain of a content or supplementary component (e.g. String, Integer etc.). The concept of enumerations (ENUM) is used to restrict a primitive type to a set of allowed values. An example for such a restriction would be a country code enumeration in order to restrict a String primitive type to a list of allowed country codes. Enumerations are not used in the examples of this paper.

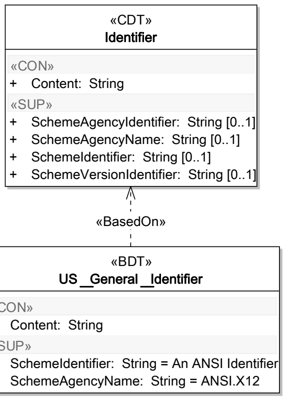
«CDT»
Identifier
«CON»
+ Content: String
«SUP»
+ SchemeAgencyIdentifier: String [0..1]
+ SchemeAgencyName: String [0..1]
+ SchemeIdentifier: String [0..1]
+ SchemeVersionIdentifier: String [0..1]
«BasedOn»
«BDT»
US _General _Identifier
«CON»
+ Content: String
«SUP»
+ SchemeIdentifier: String = An ANSI Identifier
+ SchemeAgencyName: String = ANSI.X12

Figure 2: Data Types

Business Information Entities If core components are used in a specific business context, they become so called business information entities. Thereby, a core component definition is taken a tailored to the specific needs of the current business domain. A derivation-by-restriction mechanism is used to create a business information entity out of a core component. Thus, a business information entity must not contain any attributes, which haven't been defined in the underlying core component. Similar to the concept of core components, business information entities are used to describe objects and properties of objects. We also distinguish two kinds of properties: simple object properties and complex object properties. Simple properties refer to elementary object values such as text, date etc. and complex properties are used to show relations to other object types. An aggregate business information entity (ABIE) is used to define the type of a certain object. Thereby, an object may have several simple properties, so called basic business information entities (BBIE). The value domain for a given basic business information entity is defined using the concept of business data types (BDT). In order to define relationships between different business information entities the concept of association business information entities (ASBIE) is used.

The right hand side of Figure 1 shows an example of business information entities. The aggregate business information entity `US_Invoice` has two simple properties denoted by the two basic business information entities `InvoiceNumber` and `Description`. Furthermore `US_Invoice` has exactly one association business information entity `US_Item` in order to show that an `US_Invoice` contains one to many `US_LineItems`.

Similar to the concept of basic core components each basic business information entity has a certain value domain defining the data type of the attribute. For basic business information entities so called business data types are used (BDT). The concepts of business data types and core data types are the same, only their application domain is different. Core data types are only used in the context of core components and business data types are only used in the context of business information entities. Similar to a core data type a business data type has exactly one content component (CON) and zero to many supplementary components (SUP). Equivalent to the derivation-by-restriction mechanism between business information entities and core components a business data type is always derived from a core data type by restriction. Thus, a business data type must not use any attributes which have not been defined in the underlying core data type.

As shown in the example in Figure 1 the basic business information entity `InvoiceNumber` in ABIE `US_Invoice` has the business data type (BDT) `US_General_Identifier`. The exact outline of the business data type `US_General_Identifier` is shown at the bottom of Figure 2. The BDT has exactly one content component and two supplementary components providing additional meta information about the content component. Thereby, the business data type restricts the core data type and uses only two of the four supplementary components of the underlying core data type.

3. The core components reference ontology

In the last section the basic core component concepts have been elaborated and we learned that core components are independent of a specific implementation format and are standardized in a non-formalized way. Since core components are agreed upon by a broad industry spectrum and other standardization organizations and interest groups, they provide the ideal basis for a reference ontology. However, in order to allow for core components to represent a global ontology, a formalized representation of core components is needed. In this chapter we formalize the core components methodology using Web Ontology Language (OWL).

Figure 3 gives an overview of the global reference ontology. The top class in the ontology is `owl:Thing`, serving as the superclass for all other classes. For a better legibility Figure 3 has been divided by three squares embracing core component, business information entity, and

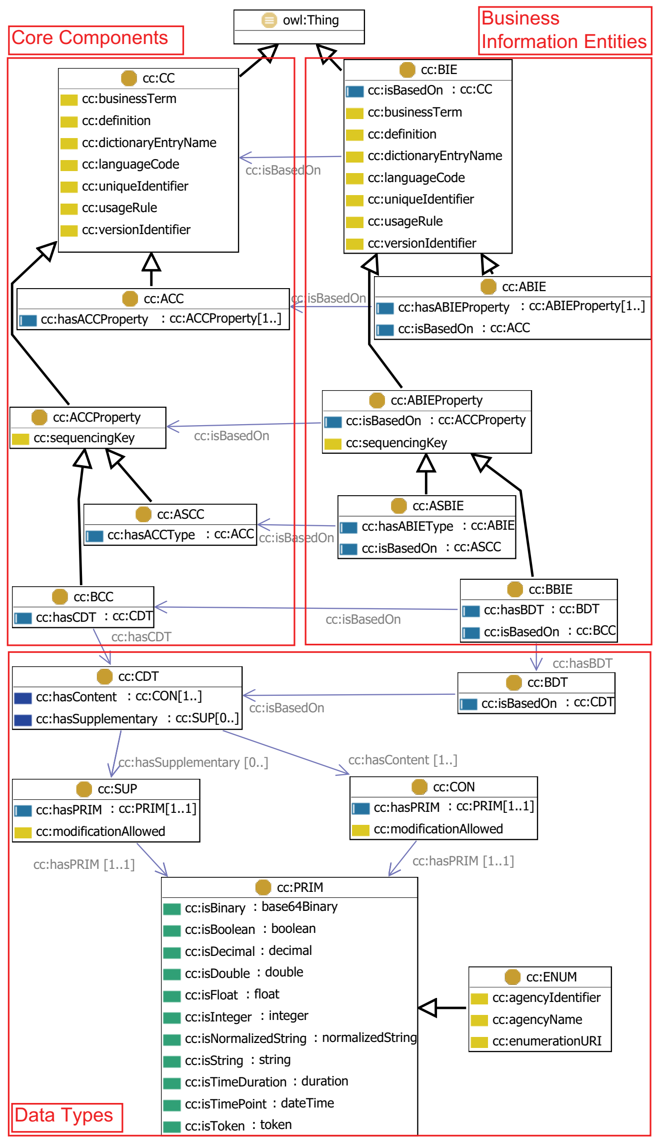

Figure 3: Overview of the reference ontology

data type specific artifacts respectively. As namespace for the reference ontology specific classes, properties etc. `http://www.umm-dev.org/owl/ccts3#` with the prefix `cc` has been chosen.

The upper left square of Figure 3 represents all core component specific artifacts of the global reference ontology. The superclass of every core component artifact is `cc:CC`, which has seven `owl:AnnotationProperty` values representing specific core component properties according to the CCTS [7]. `cc:businessTerm` is a term under which the core component is commonly known and used in business and `cc:definition` is used to store the unique semantic meaning of the core component. If core components are stored and retrieved from business registries they require a unique name. This name is represented by `cc:dictionaryEntryName`. In order to support multilingualism `cc:languageCode` defines the language used for the core component. `cc:usageRule` defines constraints in free-form text on the usage of core components. `cc:uniqueIdentifier` and `cc:versionIdentifier` are additional meta-information fields, required for registry storage and retrieval. `cc:CC` has two sub-classes namely `cc:ACC` representing aggregate core components and `cc:ACCProperty`, which is a newly introduced superclass for basic core components and association core components. Both, association core components and basic core components are represented by their respective `owl:Classes` `cc:ASCC` and `cc:BCC`. The annotation property `cc:sequencingKey` in `cc:ACCProperty` is used to assign an arbitrary order to an `cc:ACCProperty`. Similar to the core component concepts the elements in the square on the right hand side of Figure 3 represent the business information entity specific artifacts of the ontology. The superclass of all business information entities is `cc:BIE`. Since business information entities are based on core components and traceability between these two artifacts must be guaranteed at any time, a `cc:isBasedOn` object property is defined between `cc:CC` and `cc:BIE`. Similar to a core component a business information entity also has a set of annotation properties such as `cc:businessTerm` etc. Their meaning is to be read according to those of a `cc:CC`.

`cc:BIE` has two sub-classes namely `cc:ABIE` representing an aggregate business information entity and `cc:ABIEProperty`, which is the superclass for basic business information entities and association business information entities. For both, association business information entities and basic business information entities the respective `owl:Class` elements `cc:ASBIE` and `cc:BBIE` are defined. Similar to a `cc:ACCProperty` a `cc:ABIEProperty` has an annotation property `cc:sequencingKey` in order to specify an arbitrary sequence order. Please note the different `cc:basedOn` object properties between artifacts from the core component and business information entity square in Figure 3. These object properties help to trace business information entity artifacts back to the core component artifacts they are based on.

The lower square in Figure 3 shows the data type specific elements of our reference ontology. We learned earlier, that the value of basic core components is set by core data types (CDT) and the value of basic business information entities is set by business data types (BDT). This fact is reflected in the ontology as well and a `cc:isBasedOn` object property is defined between a `cc:BDT` and a `cc:CDT`. Both, a core data type and a business data type always consist of exactly one content component and zero to many supplementary components. This fact is reflected by `cc:hasSupplementary` and `cc:hasContent` object properties. Please note, that these two properties are not shown for `cc:BDT` on the lower hand side of Figure 3. Since a `cc:BDT` is based on a `cc:CDT` these dependency properties are implicitly given.

Finally the concept of primitive types is reflected by `cc:PRIM`. As outlined on the lower side of Figure 3 a primitive type has eleven data type properties. Each data type property has a predefined data type defined as `rdfs:range` e.g. `cc:isTimeDuration` has `rdfs:range xsd:duration`. A specialization of a primitive type is represented by a so called enumeration type `cc:ENUM`. An enumeration represents a predefined code list or identifier schemes e.g. a list of predefined country codes. The three annotation properties `cc:agencyIdentifier`, `cc:agencyName`, and `cc:enumerationURI` are used to provide meta-information about an enumeration type.

Figure 4 shows the core component ontology in detail. The top core component class `cc:CC` is shown on the right hand side of Figure 4. It serves as the superclass for the two direct sub-classes `cc:ACC` and `cc:ACCProperty`. In turn an `cc:ACCProperty` is the super-class of `cc:BCC` and `cc:ASCC`, representing basic core components and association core components, respectively. The dependency between a `cc:ACC` and its ACC properties is defined by the two object properties in the upper left corner of Figure 4. The object property `cc:hasACCProperty` indicates that an ACC has ACC properties. Additionally the cardinality of the object property is set to 1. Thus, there cannot be an ACC without ACC properties. The allowed values for the object property are restricted to `cc:ACC` and `cc:ACCProperty` using `rdfs:range` and `rdfs:domain`.

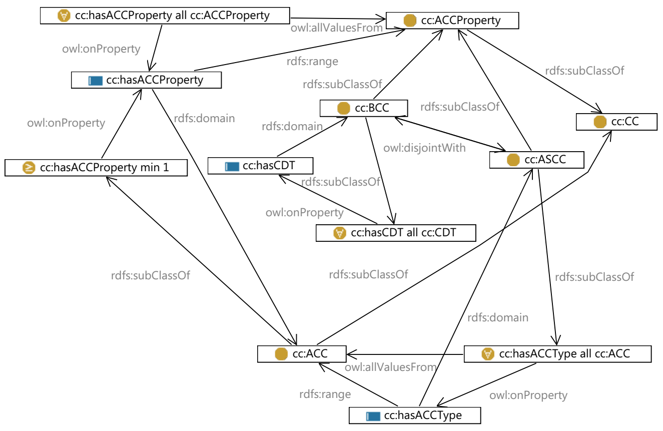

Figure 4: Core component ontology in detail

A basic core component has an assigned core data type which is reflected by the object property `cc:hasCDT` in the center of Figure 4. Allowed values of `cc:hasCDT` are restricted to `cc:CDT` and `cc:BCC` using `rdfs:range` and `rdfs:domain`. An association core component points to an aggregate core component. Using the `cc:hasACCType` each `cc:ASCC` is assigned with the appropriate `cc:ACC`. As with all other object properties the allowed domain and range is restricted as well. An ACC property must be either an `cc:BCC` or an `cc:ASCC`. Thus, an `owl:disjointWith` property exists between the two classes. In the following section we dwell on how to map specific document instance to the reference ontology.

4. Mapping instances to the reference ontology

Our global business document reference ontology will be the starting point for integrating heterogeneous business document standards. As shown in Figure 5 we distinguish between three different levels in regard to business document interoperability according to the first three Meta-Object-Facility (MOF) layers [5]. As an example we assume a Universal Business Language (UBL) [4] and a OAGi [6] business document schema and instances.

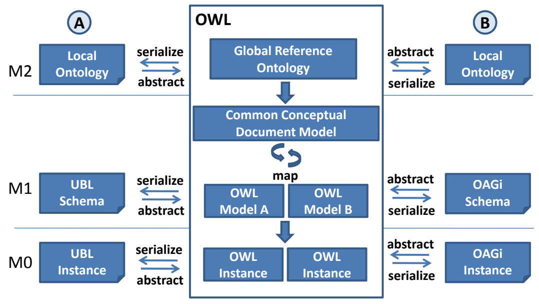

Figure 5: Overview of mapping

In a typical interoperability scenario partner *A* and partner *B* exchange business document instances based on an XML based schema definition. Figure 5 shows the main principles of our interoperability approach based on OWL. On the lowest level (M0) business partners exchange business document instances which are usually incompatible. The different document instances are valid against their Schema definitions - in our example against the UBL and OAGi XML Schema definitions files. Accordingly a mapping between the OWL representation of the UBL schema/OAGi schema and the OWL based common conceptual document model is required. In a first step the schema definitions are abstracted from one format to an OWL representation (resulting in OWL Model A and OWL Model B) in Figure 5. The resulting OWL representations of UBL and OAGi are mapped to the common conceptual document model based on business information entities. These business information entities are valid against the global reference ontology based on core components.

In principle the OWL representation as shown in Figure 5 serves as the intermediary format between different business document standards. If appropriate serialization mechanism and mapping mechanisms against the global reference ontology are provided, instances of different business document standards may be transformed seamlessly. Although in principle any XML representation may be chosen as the intermediary format, our proposed approach has two major advantages. First, it is based on the most complete global reference ontology provided by UN/CEFACT. Second, the OWL representation of the common conceptual document model allows for a machine interpretable processing.

As an example Figure 6 shows how specific OWL representations of both, OAGi and UBL, are mapped against our common conceptual model. On the M1 level, shown in the center of Figure 6, we define our common conceptual document model. Due to space limitations we do not show all elements of the reference ontology, but focus on the main concepts for explanatory purposes. Our common conceptual model consists of one association business information entity (`abie:My_Item`), one basic business information entity (`abie:My_InvoiceIdentifier`), and one aggregate business information entity (`abie:My_Invoice`). Concepts of the common conceptual document model use the namespace prefix `abie`, concepts from the global reference ontology the prefix `cc`. UBL and OAGi ontology concepts have the namespace prefix `ubl` and `oagi`, respectively.

Using the concept of `owl:equivalentClass` we map the ontological representation of UBL and OAGi against our global conceptual document model. In our conceptual document model we use the concept of `abie:My_InvoiceIdentifier` to represent the ID of an invoice. In UBL this concept is named `Invoice.Identifier` and in OAGi `ID`. Both are mapped to the global concept `abie:My_InvoiceIdentifier`. On the lowest level exemplary instances of the ontology are shown.

The core component concepts which are used to build our common conceptual document model are shown on top of Figure 6. Since all business information entities must be based on core components an equivalent representation of the common conceptual document model as outlined on the M1 level in Figure 6 must be given. Please note, that we only show the business information entity perspective in the example, and left out the core components view due to space limitations. This brief example has shown how different business document ontologies are mapped to a common conceptual document model based on OWL. A global reference ontology based on core component concepts serves as the basis for the common conceptual document model.

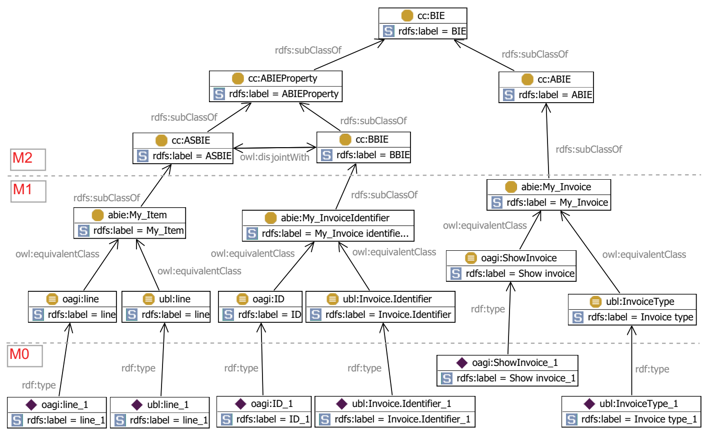

Figure 6: Mapping UBL and OAGi

5. Conclusion

In this paper we presented an approach to build a global reference ontology for business documents. We based our approach on UN/CEFACT's Core Components and the Web Ontology Language (OWL). Using our approach a flexible interoperability solution coping with heterogeneous business document standards is provided. The major advantage of our approach, compared to other efforts in this area, is twofold. First, we base our solution on a global reference ontology standardized by the United Nations. Second, we provide an OWL based representation, which allows for machine based interpretation using Semantic Web technologies.

References

[1] B. Hofreiter. Binding UMM Business Documents to a Business Document Ontology. In *Proceedings of the Inaugural Conference on Digital Ecosystems and Technologies (DEST '07)*, pages 666–671. IEEE, 2007.
[2] H. Li. XML and Industrial Standards for Electronic Commerce. *Knowledge and Information Systems*, 2(4):487–497, 2000.
[3] P. Liegl. Conceptual Business Document Modeling using UN/CEFACT's Core Components. In *Proceedings of the 6th Asia-Pacific Conference on Conceptual Modeling (APCCM2009)*. Australian Computer Society, 2009.
[4] OASIS. *Universal Business Language 2.0*, 2006.
[5] OMG. *Meta Object Facility*, 2006.
[6] Open Applications Group. OAGIS Canonical Model for Integration, 2009.
[7] UN/CEFACT. *Core Components Technical Specification 3.0*, 2009.
[8] UN/CEFACT. *UML Profile for Core Components Technical Specification 3.0*, 2009.
[9] UN/CEFACT. *UN/CEFACT Core Components Data Type Catalogue 3.0*, 2009.