Abstract
UN/CEFACT’s modeling methodology (UMM) is used to develop business collaboration models independent of the IT-platform. The information being exchanged is modeled by class diagrams. It is expected that later on these class diagrams are mapped to business document standards. Since it is hard to predict which standards will be dominant in the future, we do not propose direct mappings from UMM business information class diagrams to the specific business document languages. Instead we propose an ontology layer to represent the information identified in UMM. This ontology layer is based on UN/CEFACT’s core components.

1 Motivation
The development of any good system of a certain size requires a well defined analysis and design process. For the specific focus of developing B2B systems, the United Nation’s Centre for Trade Facilitation and Electronic Business (UN/CEFACT) has developed the UN/CEFACT modeling methodology (UMM) for analyzing both dynamic and structural B2B aspects [UN06]. A UMM model concentrates on the business aspects only. It is not bound to a specific platform. In order to implement a B2B system a UMM model must be transformed to platform specific specifications. This means the dynamic aspects will be transformed to a business process modelling language (e.g. BPEL, BPSS). We describe these mappings in [HoHu04] and [HoHK06]. The structural aspects will be mapped to business document standards which is a focus of this paper.

This leads to the question which business document standard should be supported? Although most implementations run on UN/EDIFACT, all latest implementations are based on XML. Therefore, a mapping to an XML-based business document standard is preferred. Unfortunately, plenty of these business document standards exist. Although some of the business document standards have become the first choice within a vertical industry, there are still some competing efforts. Popular business document standards include those of market place providers, like Commerce One’s xCBL and Ariba’s cXML, the Open Application Group’s OAGI interconnecting ERP systems, OASIS’s UBL and domain-specific solutions like RosettaNet in the IT sector. An overview of business document standards is provided in [Li00].

After the e-commerce hype only a few new business document standards appeared and there is a trend that some standards merge or disappear. Today we are not in a stable situation and it is hard to forecast which will be the most commonly accepted business document standard in the long run. For example, five years ago xCBL was one of the most accepted business document standards. Today its support is rather limited.

Instead of developing a mapping from UMM class diagrams to a specific business document language, of which we do not know how it will look like tomorrow, we prefer an approach that will survive industry trends. Therefore, we propose a business document ontology layer that is first of all independent of the XML-schema of a business document language. This ontology layer is based on UN/CEFACT’s core components. We use RDFS to describe the core components based ontology. Furthermore we use RDF’s XML serialization to denote the ontology in a machine-readable format. This is a precondition that the mapping from UMM class diagrams to our business document ontology is useful for a further transformation to a business document language of choice.

The remainder of this paper is structured as follows: In section 2 we show how the core components-based business document ontology works. By the example of a quote in our purchase order management case study, we demonstrate a mapping from a UMM class diagram to our business document ontology in section 3. In section 4 we define what is required for a mapping from the business document ontology to a business document language.

2 Related Work
The rapid growth in information volume makes it more and more difficult to find, organize, access and maintain the information required by users. The Semantic Web proposes enhanced information access based on machine-processable meta-data. This resulted in an increased interest of computer scientists in ontologies. However, the term ontol-
ology is used for a much longer time in philosophy. There
ontology means the study of being or existence as well as
the basic categories thereof. In computer science, an ontol-
ogy is the attempt to formulate an exhaustive and rigorous
conceptual schema within a given domain, a typically hier-
archical data structure containing all the relevant entities
and their relationships and rules within that domain. Ontol-
ogies are commonly used in artificial intelligence and
knowledge representation. T.R. Gruber has described an
ontology as “a formal, explicit specification of a shared
centralization” [Grub93]. A conceptualization is an
abstract, simplified view of the world that we wish to rep-
resent. It should be shareable and re-usable.

An ontology is always developed for a certain purpose.
As a result, ontologies developed independently for differ-
ent purposes will differ significantly from each other. An
ontology which is subject-independent is known as a foun-
dation ontology or upper ontology. An upper ontology is
limited to concepts that are meta, generic, abstract, and
hence are general enough to address (at a high level) a
broad range of domain areas.

No upper ontology has yet become a de-facto standard.
Different organizations are attempting to define standards
for specific domains. For practical reasons, ontologies
must be expressed in a concrete notation. An ontology lan-
guage is a formal syntax by which an ontology is built.
Over the last years a number of ontology languages have
been developed, both proprietary and standards-based. The
most well-known ontology languages are the following:

The Resource Description Framework (RDF) is a World
Wide Web Consortium (W3C) standard recommendation
for describing machine-processable semantics of data rep-
resented by subject-property-object triples [MaMi04].
RDFS is also a W3C recommendation and provides a
means to define the vocabulary for RDF properties and
specifies the kinds of objects to which these properties may
be applied. This means, RDFS provides a basic type sys-
tem for RDF data models [BrGu02]. DARPA Agent
Markup Language (DAML) was created as part of a
research program started in August 2000 by DARPA, a US
governmental research organization; and Ontology Infer-
ence Layer (OIL) is an initiative funded by the European
Union programme for Information Society Technologies as
part of some of its research projects. DAML and OIL are
standardized schema languages and not data models like
RDF or topic maps [W3C01]. The Web Ontology Lan-
guage (OWL) has been developed by the Web Ontology
Working Group as part of the W3C Semantic Web Activity
and has the status of a recommendation since February
2004 [McGH04]. It is designed for use by applications that
need to process the content of information instead of just
presenting information. The working group proposes that
OWL facilitates greater machine inter-pretability of Web
content than that supported by RDF and RDFS. XML
Topic Maps (XTM) are a reformulation of topic maps in
XML syntax based on XLink. Topic maps have a long and
complicated history. They originally started as a process
to create a standard SGML DTD for software documentation.

3 A Core Components Based Ontology

The goal of our B2B business document ontology is to
describe the semantic concepts to be represented in docu-
ment types exchanged between business partners [HoHu02]. It is important that document types are not
developed in isolation from each other. Semantic concepts
that are shared between different document types must be
re-used. This means, e.g., that a line item should be based
on the same concept regardless if it is a quote document
type from one or the other business environment, or a pur-
chase order, an invoice, etc.

In order to develop a business document ontology, two
main approaches are introduced by Ontoprise’s Semantic
B2B Broker [Onto00]: a top-down (Fig. 1a) and a bottom-
up approach (Fig. 1b). A top-down approach is used if no
XML structure of a business document standard is given
and such a structure is secondary. In this case, business
experts will first define a document ontology that describes
their shared understanding of a given business document
type. The conceptual model of this document ontology
builds the foundation of the development of a new business
document standard (represented as DTD or XML schema).
Vice versa, a bottom-up approach takes DTDs or XML
schemas from existing business document standards to
analyze their semantic content. The conceptual models of
all considered business document standards have to be har-
monized in order to define a unified document ontology.

The industry-wide recognized development of compo-
ents independent of a business document standard is cur-
rently undertaken by UN/CEFACT. The work started as
part of the ebXML initiative and is called UN/CEFACT’s
core components. It is our goal to take advantage of a
future pool of core components. Hence, our ontology layer
is not defined by reverse engineering of existing business
document standards like in the bottom-up approach.
Instead, our ontology layer is based on UN/CEFACT’s
core components. However, we do not use a pure top-down
approach, because we will not develop a new business doc-
ument standard. We have to mediate the core components-
based ontology layer with existing business document stan-
dards. Since we are coming from top as well as from bot-

![Figure 1. 3 ways to develop a Document Ontology](image-url)
The resulting RDFS is depicted as a graph in Fig. 2 and (RDFS) [BrGu02] for the core components meta model. For this purpose we have developed an RDF schema (UN/CEFACT's core components specification [UN03]. In the first step we develop an ontology that follows the domain range

Figure 2. Meta model for Core Components in RDFS
Each UN/CEFACT core component contains the following dictionary information: A dictionary entry name is the unique official name of the core component. It corresponds to the RDFS property label. The definition of the unique semantic business meaning of the core component is given in the RDFS property comment. Therefore, we do not have to redefine these two properties in our code. The property remark (line 8) is used to further clarify the definition, to provide examples and/or to reference a recognized standard. If there exist further synonym terms under which the core component is commonly known and used in the business, the property business term (line 12) is used to define them. For reasons of completeness, we assign the properties representation term (line 16), object class (line 20) to core components, and property term (line 34) only to basic core components as defined in the UN/CEFACT specification.

The RDF library of core components must be populated with all core components currently being developed by UN/CEFACT. Each core component will be expressed as an RDF model that follows the RDFS of the core components’ meta model. The code line 53 to line 108 demonstrates the population of the RDF library by the means of the aggregate core component product service details. A representation of this core component as UML class could be represented as follows:

![UML Class for Product Service](image)

The resource representing this aggregate core component (line 97) is marked #000155 according to its UN/CEFACT core component ID. This represents only the fragment identifier of a URI uniquely identifying core components. The unique identifications (UID) assigned to core components are used as fragment identifier. The label of the resource is equal to the data dictionary entry name product service details (line 98). The comment states the definition of the product service details (line 99). The business term good which is commonly used to refer to product service is assigned to it (line 100). Each of the components aggregated within product service details is assigned as core component child (line 102 to line 107).

As an example, we have a detailed look on the component product service identifier. This basic core component is defined in line 65 to line 71. Its identifier is #000156 (line 65). It is referenced in the coreComponentChild statement of the aggregate core component service product details (line 102). Similarly to aggregate core components, the product service identifier captures label, comment and objectClass (line 66 to line 68). Since it is a basic core component it also carries a propertyTerm (line 69) and is assigned to a specific type (line 70). Since product service identifier is of type identifier the elementType statement in line 70 references the ID #000101 of the core component type identifier type (line 62). In the example code we just show the label of the core components type (line 63) and omit to present all its other characteristics.

All the other basic core components that are children of the product service details (and correspond to the attributes in Fig. 3) are rudimentarily outlined in line 72 to line 96. All their data types are defined in line 53 to line 64.

![Image](image)
Besides product service details all the other core components must be described by RDFS and provided in a library. Currently, UN/CEFACT is in the process of delivering a first full set of core components. In the future, these are the ones that must be available for real world use. For demonstration purposes we use the preliminary set of core components as issued by UN/CEFACT.

4 From a UMM Class Diagram to the Business Document Ontology

The core components stored in a core components library are the foundation for building business document types. In order to develop a business document type the question is which core components are used and how are they assembled? In order to identify the core components it is a good idea to start with UMM. The UMM model defines the business information being exchanged during a business transaction. The business process oriented approach guarantees that the business information is kept to the minimum necessary. The resulting business document semantics are modeled in a class diagram. This class diagram is already built from classes that correspond to core component semantics. Therefore, classes and their attributes give a hint to those core components that must be selected from the library in order to build the corresponding business document type.

Core components are by definition free of a business context. Business document types are always used in a certain business context. Accordingly, a building block actually used in a business document type is not called a core component anymore. This building block is now called business information entity. A business information entity adopts a core component to the business context of its usage and, thus, is always based on a core component. A business document type is always an assembly of business information entities. This means that it is necessary to describe the relationships of the business information entities within the assembly. Therefore, it is necessary to define how business information entities are associated with other ones.

In order to meet these requirements we have to extend our meta model of Fig. 2. The necessary extensions are depicted in Fig. 4 and are listed in code line 109 to line 135.

A business information entity is defined as a subclass of the abstract concept building block (line 109). Accordingly, building block is the abstract super class for both core component and business information entity. A business information entity is based on a core component. However, it is not necessary that it is directly based on a core component. This means that a business information entity might be based on another business information entity. Hence, the property based on points to the abstract building block a business information entity is based on (line 112). This allows to point to both a core component and a business information entity.

The property associated with is used for referencing the business information entity to which an association exists. Nevertheless, the property does not directly point to another business information entity. For a business information entity instance it is important to define how many instances of the related business information entity are associated with it. RDFS by itself does not support the specification of lower and upper limits for statements of a certain type. To overcome this limitation we define the class component (line 121). The property associated with points to a component (line 117). Component carries the min (line 124) and the max (line 128) property to define the lower and upper bound of the related business information entity. Of course the component must point to the business information entity that is related to the business information entity in question. We define on which building block a component is based upon. For this purpose, we use the property based on (line 112) again. Thus, the domain of based on is not only business information entity (line 113), but also component (line 114).

The next constraint we have to consider is that not all of the attributes of a class are relevant in a certain business context. This type of constraint leads to the fact that a business information entity is not used exactly as the underlying core component. For this purpose we define the property includes (line 132) that is used to define the children of an business information entity. Similarly to the associated with property, the property includes does not directly reference the included core components or business information entities. It points to a component which in turn references the included building block (core component or business information entity). We have already explained that a component specifies the minimum and maximum occurrence of the included building block. A maximum value of 0 signals that the attribute is not used.

Figure 4. Meta Model for Business Information Entities
The model developed so far allows us to assemble business document types from existing core components in a library as well as to create business information entities that customize the underlying core components to the specific needs of a certain business context.

In order to demonstrate our approach we take the example of a quote document. We assume that a business process analyst has followed the UMM and identified the information depicted in the class diagram of Fig. 5 to be interchanged during a business collaboration. This class diagram is reduced to show only those elements we need to demonstrate our approach and, consequently, does not show attributes for most of the classes. As mandated by the UMM this class diagram is already based on classes that are equivalent to core components. A quick look on Fig. 5 (line 139, line 166 to line 170).

Figure 5. Class Diagram for a Context-Free Quote

As we said before, as soon as a core component is used in a business document type it becomes a business information entity. Therefore, we first have to select the core components we need from the library. For each core component we create (at least) one business information entity. A quote is a special type of document. Hence, we select the core component #000210 document details from the library and create a business information entity quote that is based on #000210 (line 136). Similarly, we create a business information entity quote request that is based on the same core component #000210 document details (line 141). We base the business information entity line on the core component #000135 line identifier (line 144). Furthermore, we create a business information entity unit charge price from the core component #000125 unit charge price details (line 149) and a business information entity product/service from the core component #000155 product/service details (line 152).

Having created the five business information entities quote, quote request, line, unit charge price and product/service, we have to establish the necessary associations between them. A quote is made as a response to exactly one quote request. In the terminology of our business document ontology this means a quote is associated with a component - named corresponding quote request - that is based on the business information entity quote request. This component is specified exactly one time, since the minimum and the maximum occurrence is 1 (line 138, line 161 and line 165). Furthermore, the business information entity quote is associated with 0 to n components - called quoted line - that are based on the business information entity line (line 139, line 166 to line 170).

The business information entity line is associated with exactly 1 component quoted price for line that is based on unit charge price (line 146, line 171 to line 175). Finally, a line is associated with 1 to n components products/service of line that are based on the business information entity product/service (line 147, line 176 to line 180). It should be noted that it is up to the document designer whether an association is created for both directions or only one. In absence of names of associations and roles in the UMM class diagram, it is up to the document designer to give names to the components that are pointed to by the associated property.

In our example the business information entity product service (line 152) is based on its corresponding core component (line 153, line 97). However, the business information entity uses only 2 out of the 6 children of the core component. Thus, the business information entity references a component for each of the 6 children (line 154 to line 159). Since our example uses only the product/service identifier and product/service description text the min and max property of these components are set to 1 (line 181 to line 185 and line 190 to line 194). Since the other four properties are not used in the book example, the corresponding max property is set to 0 (line 186 to line 189 and line 195 to line 206).
5 Language Binding for Business Document Standards

Our business document ontology should be used in a next step to provide language bindings to corresponding document types in a business document language. Since our “middle-out” approach and a bottom-up approach only differ in the way the ontology is built, the problem of defining a language binding remains the same. Thus, our framework considers the language binding defined in the bottom-up approach in [OmFe00].

![Diagram of Language Binding](image)

**Figure 6.** Binding between Ontology and Documents

The basic concept of this language binding is depicted in Fig. 6. It is based on the definition of a common conceptual model for document types of different business document standards. The conceptual models are described in RDFS. Thus, a mapping between the business document language’s DTD or XML schema on the one side and the conceptual model on the other side is required. This mapping on the schema level is defined by the means of XSLT. On the instance level, an incoming document is abstracted from its XML serialization and translated into its RDF data model. Vice versa, in order to create an outgoing document, the RDF data model of the target business document language is serialized according to the target XML format.

Furthermore, the conceptual data model of a business document language’s document type (expressed in RDFS) must be mapped to the document ontologies data model of the same document type (also expressed in RDFS). This means that the equivalent mapping between the terminologies requires a transformation from one RDFS model to the other. The mapping must be described by the means of an RDFS mapping language. The mappings must be automatically translated into an RDF transformation language. This transformation language is applied to translate the conceptual RDF model of an incoming document into an instance of the document ontology and, vice versa, to translate an instance of the document ontology into the conceptual RDF model of an outgoing document. It follows that all mappings between different business document standards will also be managed by mapping each standard to the core components-based document ontology. Thus, future work has to concentrate on a RDFS mapping language and the RDF transformation language.
6 Conclusion

In this paper we presented an approach to define a business document ontology using RDFS. The business documents are built upon the UN/CEFACT core components concept. Core components are re-usable building blocks, which we represent by the means of RDFS. The RDFS notation of core components provides a basis for a mapping to different XML-based business document standards.

An evaluation of our approach according to the set of quality criterias for ontologies discussed in Gruber [Grub93] results in the conclusion that we meet every single criteria. Clarity requires definitions to be context independent. Core components are said to be free of context. In fact, they are not limited to any business sector. They are only assigned to the very general domain of „business“. Coherence demands consistency of the definitions. Core components are based on a controlled vocabulary in order to ensure coherence. Extensibility is reached if new concepts can easily be composed without any changes to the ontological foundations. The usefulness of an ontology depends on its extensibility. This is very important in dynamic environments like business. Our ontology proposed might be extended to meet the requirements of future business processes.

Another criteria is a minimal encoding bias. An encoding bias results when representation choices are made only for the convenience of notation or implementation. The development of our ontology was not influenced by any programming or encoding environment. Finally, Gruber asks for minimal ontological commitment. An ontology should make as few claims as possible about the world being modeled, allowing the parties committed to the ontology freedom to specialize and instantiate the ontology as needed. In our proposed ontology parties will always have to specialize and use the ontology as needed in their business environment.

References