Putting Services in Context

Danijel Novakovic, Christian Huemer
Institute of Software Technology and Interactive Systems
Vienna University of Technology
Vienna, Austria
{novakovic, huemer}@big.tuwien.ac.at

***Abstract*—Business transactions between companies are more and more executed by a flow of well-defined electronic business documents. The resulting inter-organizational business processes are often realized by concepts known from service-oriented computing. Exchanging a business document corresponds to a service call and the input/output of the service calls commonly follows a certain business document standard. However, these standards typically present the superset of all required elements used in any business context. In a specific context (in a specific industry, in a specific geopolitical region, etc.) the input/output is adjusted to this context by constraints on the generic structure. Accordingly, a specific service is always used in a specific context. It follows that it is important to define the business context of a service (in a structured format). For this purpose we have developed the Business Context Ontology model (BCOnt). In this paper we elaborate on the theoretical concepts and the underlying algorithms as well as on their implementation in practice.**

Keywords*—*business context; ontology based business context model; (semi-) automatic generation of e-business documents

I. Introduction

Context is today widely exploited in pervasive systems, the most often in case when mobile devices apply a user location to perform different computations. However, in this paper we describe how a contextual knowledge could be harnessed in a different domain. It is the domain of inter-organizational business processes where business documents are exchanged between business partners thereby synchronizing their own private business processes. Therefore, development of new techniques which can be used to speed up the generation of these documents and to avoid heterogeneous interpretations of the exchanged data contents is essential for a reliable and efficient execution of the today's complex business ecosystems.

This paper describes our approach to apply the contextual knowledge for (semi-) automatically generating implementation guidelines of business documents. An implementation guideline represents a context specific constraint of the underlying generic document standard. More precisely, in the work explained in this paper we consider business documents built upon the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) document standard [1]. These documents are exchanged between business partners when executing inter-organizational business processes.

In our previous research [2], we have introduced the Business Context Ontology model (BCOnt) to formally represent business context (BC). Furthermore, in [3] we have established the theoretical foundations to contextualize already existing semantically interoperable data building blocks, so-called Core Components, by means of this model. In the following we additionally tailor this approach and prove that it does not hold only in theory, but in practice as well. Therefore, we develop the corresponding implementation algorithms and show how these algorithms can be integrated in an applicable system. The proposed service oriented architecture exploits business contextual information to re-use the already existing Core Components during the development of new BC aware business document implementation guidelines. Our concurrent research [4], [5] addresses the same problem using the Enhanced Unified Context (E-UCM) model. In order to easy the understanding and future comparisons between our two approaches, in the following we try, when it is possible, to describe the BCOnt model by adapting the corresponding terminology and relevant examples used in [4] and [5] for the explanation of the E-UCM model.

The remainder of the paper is structured as follows. First, Section II presents our BC definition, gives an overview of the UN/CEFACT document standard and describes the main pillars of the BCOnt model. In Section III we explain our approach to utilize the contextual information contained by Core Components and to (semi-) automatically model new implementation guidelines of business documents. In Section IV we present our implementation of the proposed conceptual solution. We elucidate clearly the key features of the most important services provided by the underlying architecture and show how these services can be implemented by our described algorithms. Finally, Section V concludes the paper and gives an outlook on future research directions.

II. Related and Previous Work

A. Business Context

The relevant scientific literature ([6], [7], etc.) describes context as an *enumeration of examples*, such as: location, time, temperature, or in terms of *relevant synonyms*, such as: user environment, application surroundings, user situation. Starting from the outcomes of our general survey on BC presented in [8] and considering one of the most applied context understandings proposed by Dey and Abowd [6] to the domain of the modern business where different business process are interrelated, we have defined BC in the following way: *BC is any information that can be used to characterize the situation of an entity within a scope where business operates. An entity is a person, place, or object that is considered relevant to the interaction between a business process and a business environment, including the business process and business environments themselves.*

The entities which are introduced by our BC definition can be described by different attributes, where each of these

attributes can be grouped into one of the primary BC categories. Our research [8] shows that we can distinguish between three primary BC categories which are particularly important for the characterization of BC, namely location, industry and activity. In the following these categories serve as a basis for providing contextual metadata on electronic business documents exchanged between inter-organizational business processes. For example, Japan and the Book industry, Austria and the DVD industry, or Canada and the Aircraft industry, can be used to describe the situation of the e-business documents which are involved within a particular user activity, such as invoicing, ordering and confirming goods receipt.

B. Business Document Standards

UN/CEFACT is an intergovernmental Standards Development Organization established by the United Nations. It proposes Core Components Technical Specification (CCTS) [1], the methodology whose main aim is the standardization of business documents for electronic interchange.

CCTS defines a Core Component business document modeling approach. Accordingly, every business document consists of business data which are encompassed by semantically interoperable data building blocks. CCTS distinguishes between two primary concepts: Core Components (CCs) and Business Information Entities (BIEs). The corresponding example is shown in Fig. 1.

CCs represent conceptual data model components for the creation of business documents that are not specific to any particular BC. Thereby, they can be used in any business scenario. CCs consist of three main entity types: Basic Core Components (BCCs), Aggregated Core Components (ACCs) and Association Core Components (ASCCs). A BCC is a piece of information which is located in a business document. Each ACC represents a collection of BCCs. Relations between ACCs are established by ASCCs. On the other hand, BIEs are logical data model components which have assigned BCs. Thereby, they are used in a context specific business scenario. Each BIE is *derived by restriction* from a CC. Corresponding to the CC concept, building elements of each BIE are: Basic Business Information Entities (BBIEs), Aggregated Business Information Entities (ABIEs) and Association Business Information Entities (ASBIEs). In the following of this paper we consider that communication models established between inter-organizational business processes conform to CCTS.

C. Business Context Model

Business Context Ontology (BCOnt) is our model used to manage representations and applications of BC under the scope of the UN/CEFACT standard. In the following we present only the tenets of this approach which we consider to be more important to understand the rest of the paper. The more complete description can be found in [3].

Business Context Ontology. BCOnt is the OWL DL based ontology whose corresponding model is presented in Fig. 2. Accordingly, this is the three level ontology model which comprises the upper, middle and lower level.

Each of the BCOnt levels is composed by the following elements: classes, individuals and properties. Classes are concrete representations of concepts or groups of concepts with

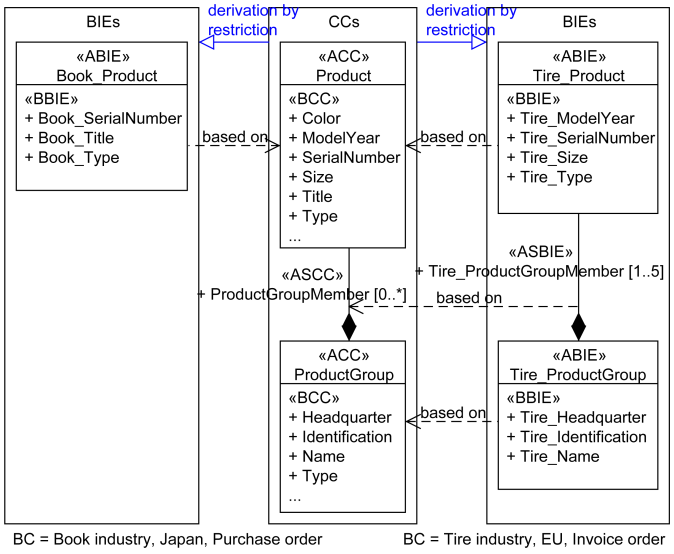

Fig. 1. Example - CCTS business document standard

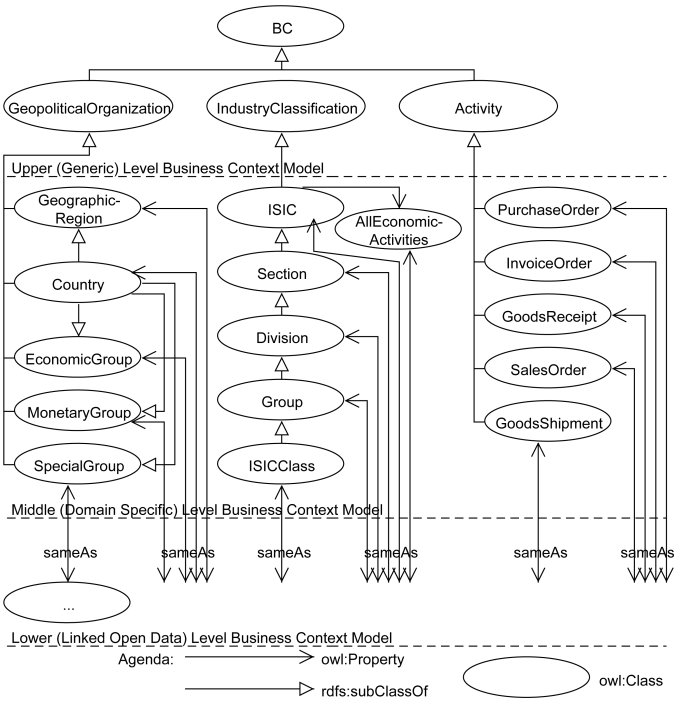

Fig. 2. BCOnt ontology model

similar characteristics. They are organized in a superclass-subclass hierarchy (taxonomy). Individuals are instances of classes. Every individual has an assigned BC value. A BC value is an atomic piece of knowledge that represents one aspect of the BC (industry, geopolitical region or activity). If an individual *A* belongs the class *ClassA*, and if an individual *B* belongs to the class *ClassB* which is the subclass of the *ClassA*, the BC value assigned to the individual *A* is restricted to the BC value assigned to the individual *B*. Relations between individuals are established by properties.

The upper level of BCOnt is a high level ontology which refers to the general concepts of BC. It is implemented by

the classes: *GeopoliticalOrganization*, *IndustryClassification* and *Activity*. Correspondent to our BC definition presented in Section II, these classes encapsulate domains restricted by the location, industry and activity BC categories, respectively.

The middle level of the ontology based BC model covers more domain specific subontologies which refine concepts introduced by the upper level of the model. It consists of three main subontologies, namely *BCFAO*, *BCISIC* and *BCActivity*. *BCFAO* is the middle level subontology of our model which refines the geopolitical domain of BC. It is based on the geopolitical classification introduced by the Food and Agriculture Organization of the United Nations (FAO) [9]. *BCISIC* is the middle level subontology of our BC model which refines the industry domain of BC. It is built in respect to the International Standard Industrial Classification of All Economic Activities (ISIC) [10], proposed by the United Nations Statistics Division. We have decided to use the FAO and ISIC foundations to develop the middle level subontologies of BCOnt due to the following reasons: (i) both of these approaches belong to the group of the most complete and today worldwide applied classifications of their corresponding domains, and (ii) FAO, ISIC and UN/CEFACT are all standardized and propagated by the same institution, the United Nations. Finally, as shown in Fig. 2, *BCActivity* is the third middle level subontology of BCOnt. It refines the activity domain of BC by providing a classification of all possible user activities, such as invoicing or purchase ordering.

The lower level of the BCOnt model is the collection of the subontologies which refer to the more specific details of the more general concepts implemented in the upper levels of the model. It is essential that this level has plug in/unplug capabilities. Therefore, additional subontologies can be dynamically plugged in or unplugged from the model depending on the current business scenarios. Finally, the lower level of the BCOnt model is an extension point to the external ontologies located in the scope of Linked Open Data (LOD) [11]. Thus, in case that some concept is not defined in the model, BCOnt can be interrelated to some external ontology, such as DBpedia [11], Geonames [11] and FOAF [11], where the missing concept is defined.

BCOnt Reasoning. The reasoning capabilities are essential benefits of the ontology based modeling. Thereby, in our research we try to harness them in order to derive new implicit business contextual knowledge. We apply two types of reasoning: (i) *ontology based* reasoning and (ii) *rule based* reasoning. Both of these techniques are implemented by the reasoning rules which are expressed using the DL based syntax [12].

The *ontology based* reasoning mechanism is applied to acquire an implicit business contextual knowledge by following the existing reasoning rules. These rules are integrated in respect to the semantics of the used OWL DL language, for example: subclass relation (*rdfs:subClassOf*), equality relation (*owl:sameAs*), and functional property (*owl:FunctionalProperty*). In our work we use ontology based reasoning to build class taxonomy and check consistency of the concepts. For example, (i) if the *ClassA* is the subclass of the *ClassB*, and (ii) if the *ClassB* is the subclass of the *ClassC*, the ontology based reasoning mechanism can infer that the *ClassA* is also the subclass of the *ClassC*. This can be formally expressed by the following rule: *(?A rdfs:subClassOf ?B)* $\sqcap$ *(?B rdfs:subClassOf ?C)* $\Longrightarrow$ *(?A rdfs:subClassOf ?C).*

The *rule based* reasoning follows the reasoning rules which are not included by the OWL DL semantics. These rules are explicitly defined by users. In our work we use this approach to infer a high level information from the low level information which holds in a specific BC. For example: (i) if two different documents ($BDoc_1$ and $BDoc_2$) are valid in two different countries (*Austria* and *Germany*, respectively) which are members of the same economic organization (*the European Union*), and (ii) if one of these documents ($BDoc_2$) has the BIE denoted by *StandVATRate*, the reasoning mechanism can infer that the other document ($BDoc_1$) also contains the same BIE denoted by *StandVATRate*. This is formally expressed by the following rule: *(?A bcont:hasMember ?B)* $\sqcap$ *(?A bcont:hasMember ?C)* $\sqcap$ *(?C bcont:hasBIE ?D)* $\Longrightarrow$ *(?B bcont:hasBIE ?D).*

Generally speaking, one of the most important shortcomings of the ontology based modeling approach is that reasoning involves calculation intense tasks. In particular, the performances of reasoning strictly depend on the size of the ontology knowledge base and CPU power. In our work we do not search for a new solution for better utilization of the CPU performances. However, we undermine the first obstacle in the following way. As already explained, BCOnt comprises the three level model structure which consists of the pluggable, domain specific subontologies. These subontologies can be interwoven with the dynamically pluggable LOD elements. Thereby, BCOnt contains only those conceptual elements which are relevant to the current business scenario. Thus, during runtime the BC knowledge database covers only the domain which is necessary for applying reasoning restricted to the particular inter-organizational business processes.

Example - Application of the BCOnt Model. We show the application of the BCOnt model on the CCTS entities (ABIEs, BBIEs and ASBIEs) in the following. The used BIEs are already introduced in our previous example described in Fig. 1. Every BIE is valid in a BC which is presented by our ontology based model. The particular BC values are specified using the DL based syntax. For reasons of simplicity, we discard the activity BC category and consider only the location and industry BC categories. The runtime BC of a BIE often is not the same as its assigned BC. Thereby, in the following we refer to runtime BC as overall BC.

As illustrated in Fig. 3, Mark 1, two entities (BBIE1 and BBIE2) are given. The BBIE1 is a piece of information which refers to the type of a tire valid in the European Union. The BBIE2 is a piece of information which refers to the size of a tire valid in Japan. Thus, the BBIE1 has the assigned BC $(\sqsubseteq EU) \sqcup (\sqsubseteq Automotive)$ and the BBIE2 has the assigned BC $(\sqsubseteq Japan) \sqcup (\sqsubseteq Automotive)$.

In the next step, Fig. 3, Mark 2, the BBIE1 and BBIE2 are covered by the ABIE1. The ABIE1 comprises the pieces of information which specify a tire product. Generally speaking, an ABIE does not have an assigned BC. The overall BC of an ABIE is dependent and, thus, calculated based on the union of the assigned BCs of the included BBIEs and the overall BCs of the included ASBIEs. This can be expressed by the

❶ BBIEs
BBIE1 = BBIE_EU_Tire_Type
$BBIE1_{assignedBC}$ = (⊑ EU) ⊔ (⊑ Automotive)
BBIE2 = BBIE_Japan_Tire_Size
$BBIE2_{assignedBC}$ = (⊑ Japan) ⊔ (⊑ Automotive)

❷ ABIE
ABIE1 = ABIE_Tire_Product
$ABIE1_{overallBC}$ = $BBIE1_{assignedBC}$ ⊔ $BBIE2_{assignedBCs}$
= ((⊑ EU) ⊔ (⊑ Japan)) ⊔ (⊑ Automotive)

ABIE1	ABIE_ Tire_Product	
	(overall) BC = ((⊑ EU) ⊔ (⊑ Japan)) ⊔ (⊑ Automotive)	
	BBIE1	BBIE_EU_Tire_Type
		(assigned) BC = (⊑ EU) ⊔ (⊑ Automotive)
	BBIE2	BBIE_Japan_Tire_Size
		(assigned) BC = (⊑ Japan) ⊔ (⊑ Automotive)

❸ ASBIEs
ASBIE1 = ASBIE_Europe_Tire_ProductGroupMember
$ASBIE1_{assignedBC}$ = (⊑ Europe) ⊔ (⊑ Automotive)
$ASBIE1_{overallBC}$ = $ASBIE1_{assignedBC}$ ⊓ $ABIE1_{overallBC}$ = (⊑ EU) ⊔ (⊑ Automotive)
ASBIE2 = ASBIE_Asia_Tire_ProductGroupMember
$ASBIE2_{assignedBC}$ = (⊑ Asia) ⊔ (⊑ Automotive)
$ASBIE2_{overallBC}$ = $ASBIE2_{assignedBC}$ ⊓ $ABIE1_{overallBC}$ = (⊑ Japan) ⊔ (⊑ Automotive)

ASBIE1	ASBIE_Europe_Tire_ProductGroupMember		
	(assigned) BC = (⊑ Europe) ⊔ (⊑ Automotive)		
	(overall) BC = (⊑ EU) ⊔ (⊑ Automotive)		
	ABIE1a	ABIE_ Tire_Product	
		(overall) BC = ((⊑ EU) ⊔ (⊑ Japan)) ⊔ (⊑ Automotive)	
		(effective) BC = (⊑ EU) ⊔ (⊑ Automotive) ❹	
		BBIE1a	BBIE_EU_Tire_Type
			(assigned) BC = (⊑ EU) ⊔ (⊑ Automotive)
			(effective) BC = (⊑ EU) ⊔ (⊑ Automotive) ❹
		BBIE2a	BBIE_Japan_Tire_Size
			(assigned) BC = (⊑ Japan) ⊔ (⊑ Automotive)
			(effective) BC = ⊥ ❹
ASBIE2	ASBIE_Asia_Tire_ProductGroupMember		
	(assigned) BC = (⊑ Asia) ⊔ (⊑ Automotive)		
	(overall) BC = (⊑ Japan) ⊔ (⊑ Automotive)		
	ABIE1b	ABIE_ Tire_Product	
		(overall) BC = ((⊑ EU) ⊔ (⊑ Japan)) ⊔ (⊑ Automotive)	
		(effective) BC = (⊑ Japan) ⊔ (⊑ Automotive) ❹	
		BBIE1b	BBIE_EU_Tire_Type
			(assigned) BC = (⊑ EU) ⊔ (⊑ Automotive)
			(effective) BC = ⊥ ❹
		BBIE2b	BBIE_Japan_Tire_Size
			(assigned) BC = (⊑ Japan) ⊔ (⊑ Automotive)
			(effective) BC = (⊑ Japan) ⊔ (⊑ Automotive) ❹

Fig. 3. BCOnt - application on the CCTS BIEs

following Formula:

$$BC\_ABIE_{overall} = (\sqcup_{i=0}^{k} BC\_BBIE_{assigned}) \sqcup (\sqcup_{i=0}^{l} BC\_ASBIE_{overall}) \,, \quad (1)$$

where k and l represent the numbers of the included BBIEs and ASBIEs, respectively. Hence, the overall BC of the ABIE1 is expressed as: $((\sqsubseteq EU) \sqcup (\sqsubseteq Japan)) \sqcup (\sqsubseteq Automotive)$.

As illustrated in Fig. 3, Mark 3, the ABIE1 is associated by the ASBIE1 and ASBIE2. These ASBIEs are derived by restriction from the same ASCC, and they relate the group of tire products with the specific tire products. However, in our case the ASBIE1 and ASBIE2 are valid in different geopolitical regions (Europe and Asia, respectively). Thereby, the ASBIE1 has the assigned BC $(\sqsubseteq Europe) \sqcup (\sqsubseteq Automotive)$ and the ASBIE2 has the assigned BC $(\sqsubseteq Asia) \sqcup (\sqsubseteq Automotive)$. Generally speaking, the overall BC of an ASBIE is dependent, and, thus, calculated based on the intersection of its assigned BC and the overall BC of the associated ABIE. This can be expressed by the following Formula:

$$BC\_ASBIE_{overall} = BC\_ASBIE_{assigned} \sqcap BC\_AssociatedABIE_{overall} \,. \quad (2)$$

Hence, the ASBIE1 has the overall BC $(\sqsubseteq EU) \sqcup (\sqsubseteq Automotive)$ and ASBIE2 has the overall BC $(\sqsubseteq Japan) \sqcup (\sqsubseteq Automotive)$. As an essential consequence, the overall BC of the ABIE1 and the overall BCs of its BBIEs are effectively narrowed. This is illustrated by the effective BCs shown in Fig. 3, Mark 4.

Finally, we can see that the effective BCs of the same ABIE can be different depending on the overall BC of the associating ASBIE. For instance, in case the ABIE1 is associated by the ASBIE1, its effective BC is $(\sqsubseteq EU) \sqcup (\sqsubseteq Automotive)$. However, in case the ABIE1 is associated by the ASBIE2, its effective BC is $(\sqsubseteq Japan) \sqcup (\sqsubseteq Automotive)$. The same conclusion holds for the effective BCs of the BBIE1 and BBIE2 which are contained by the ABIE1. In particular, the effective BC of some BIE may be null (denoted by $\perp$ in the example). Thereby, these BIEs are not relevant in the specified business scenario and, thus, they should be excluded from the corresponding business documents.

III. BC Aware Core Components Modeling

In our previous work [3] we have established the theoretical guidelines to calculate the content model of a business document implementation guideline (BDocIG) using the BCOnt model. In the following we additionally tailor our approach and present how it can be realized by applying a set of services of the proposed BC aware service oriented architecture.

A. Conceptual Solution

The existing BC knowledge, which will be used during the further processing, is embedded by the implementation guidelines of the already existing e-business documents (*ExistBDocIGs*). In Fig. 4, these documents are denoted as $BDoc_1$, $BDoc_2$, ..., $BDoc_m$. According to CCTS, the implementation guidelines consist of the semantically interoperable BIE blocks, where each BIE has its assigned, overall and effective BC ($BC_{assigned}$, $BC_{overall}$ and $BC_{effective}$, respectively).

In the first step of our approach (Fig. 4, Mark 1), all BIEs are extracted from the original *ExistBDocIGs* and embedded into a Generic Business Document Implementation Guideline (*GenBDocIG*). Therefore, the *GenBDocIG* encompasses the entire contextual knowledge collected from the already existing BDocIGs. Afterwards, in respect to the specific user requirements, only those BIEs which are valid in the required BC are extracted from the *GenBDocIG* (Fig. 4, Mark 2) and embedded into a new Customized Business Document Implementation Guideline (*CustBDocIG*). This is the new BDocIG which is relevant in the BC explicitly required by user.

B. BC Reasoning

We can streamline the conceptual solution described in the previous Subsection by introducing reasoning capabilities. The reasoning capabilities are underpinned by the reasoning techniques provided by the BCOnt model (they are already explained in Section II-C). In essence, BC reasoning can be

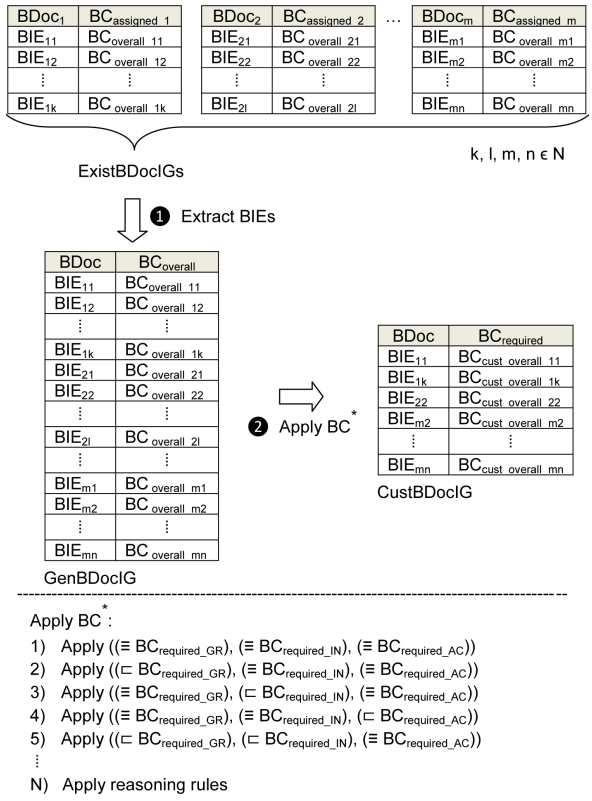

Fig. 4. Conceptual solution

achieved by following two main tenets: (i) learning from a BCOnt model and (ii) learning from a knowledge database.

The crux of the first BC reasoning tenet is the BC organization in the form of the BCOnt ontology, as described in Section II-C. Accordingly, the superclass-subclass property in our model restricts the BC in which the source concept of the property is valid to the BC in which its target concept is valid. Thereby, not only the contextual knowledge that originates from the particular concept, but also the knowledge indicated by its subconcepts could be harnessed for the further customization of the *GenBDocIG*. In Fig. 4 this is shown as the iterative application of the BC, denoted by $ApplyBC^*$.

The second BC reasoning tenet comes as a direct consequence of the application of the *rule based* reasoning technique provided by the BCOnt model. The new knowledge, thus, can be derived from the existing business contextual knowledge by following the explicitly provided reasoning rules. The more complete explanation and the corresponding example are already presented in Section II-C.

IV. Implementation

In this Section we show the implementation of the conceptual solution proposed in Section III. First, we describe the XML based representation of contextualized BDocIGs. Afterwards, we explain the service oriented architecture which generates new BDocIGs valid in the particular BC.

```
<xsd:element name="PostCode" type="bdt:StringType">
  <xsd:annotation>
    <xsd:documentation>
      <ccts:AcronymCode>BBIE</ccts:AcronymCode>
      <ccts:UniqueID>f0e2b9ad-7d05</ccts:UniqueID>
        …
    <xsd:documentation>
    <xsd:appInfo>
      <ccts:BC> 1
        <ccts:IndustryBC>(⊒ Aircraft)</ccts:IndustryBC>
        <ccts:RegionBC> (⊑ EU) ⊔ (⊑ USA)</ccts:RegionBC>  } 2
        <ccts:ActivityBC>(≡ Invoice)</ccts:ActivityBC>
      </ccts:BC>
    </xsd:appInfo>
  </xsd:annotation>
</xsd:element>
```

Fig. 5. Example - BDocIG presented using the contextualized NDR

A. Representation of BC Aware BDocIGs

The UN/CEFACT XML Naming and Design Rules (NDR) [13] is the specification proposed by UN/CEFACT. It formulates the set of rules necessary to develop XML schemas and XML schema based documents which conform to CCTS. Thereby, in the following we present BC aware BDocIGs following the principles defined by the NDR specification.

However, the standard NDR specification can not be directly applied to present contextualized business documents. Therefore, in order to provide an instrument to assign and to process business contextual information, the NDR specification must be enhanced. Our corresponding solution introduces the new XML DOM element which is denoted as: *<ccts:BC>*. It is used to specify the concrete BC in which some specific Core Component presented by the XML NDR schema is valid. The introduced element is integrated in the scope of the application information element (*<xsd:appInfo>*) defined by the standard NDR. The relevant example is shown in Fig. 5, Mark 1.

Furthermore, the *<ccts:BC>* element encompasses the following children elements: *<ccts:IndustryBC>*, *<ccts:RegionBC>* and *<ccts:ActivityBC>*. These are new XML DOM elements which are correspondent to our primary BC categories industry, geopolitical region and activity, respectively. Thus, the subdomains of the BC in which some specific Core Component is valid can be presented by the DL syntax based business context expression indicated within the corresponding BC category tags. This is shown in the example in Fig. 5, Mark 2.

B. Architecture

In the following we explain the simplified architecture which implements our approach to model BDocIGs valid in the required BC. The corresponding blueprint and the explanation of its graphical notation are shown in Fig. 6. All processing units and included libraries are developed using the Java programming language. The BDocIGs (*ExistBDocIGs*, *GenBDocIG* and *CustBDocIG*) conform to the enhanced NDR specification introduced in the previous Subsection.

Contextual information is presented by the BCOnt model. We implement this model by the Protégé modeling tool [14]. It is the free, open-source ontology editor and knowledge base framework. We have chosen Protégé due to its support to the OWL languages, plug in extension possibilities, built in

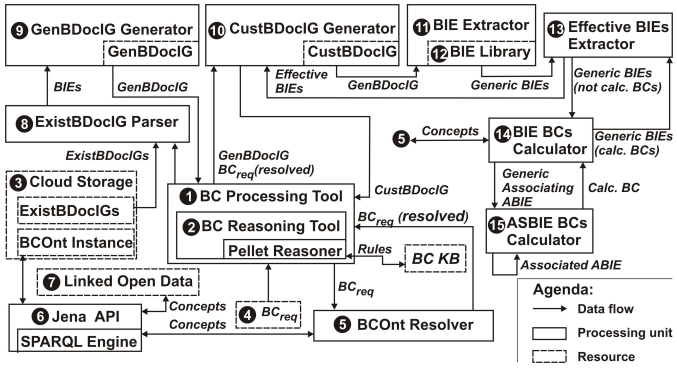

Fig. 6. Business Context Aware Service Oriented Architecture

reasoners, excellent documentation, user friendly interface and its ease of use.

The core of our proposed architecture is the *Business Context Processing Tool* (Fig. 6, Mark 1). It is the processing unit which initiates, controls and coordinates the services provided by the other elements in the system. As shown in Fig. 6, Mark 2, this tool comprises the *BC Reasoning Tool*. It is the processing unit which executes the BC reasoning techniques already explained in Section III. The reasoning is conducted involving the *Pellet reasoner* [15]. It is the open source, sound and complete OWL-DL reasoner written in Java.

Input Processing. The following input parameters are processed by the proposed architecture: (i) *instance* of the BCOnt model, (ii) already existing business document implementation guidelines (*ExistBDocIGs*), and (iii) BC in which the output *CustBDocIG* must be valid (BC_{req}).

The *BCOnt model instance* and *ExistBDocIGs* are provided by the system itself. They are stored on the cloud, as shown in Fig. 6, Mark 3. The BC_{req} (Fig. 6, Mark 4) is expressed using the DL based syntax [12]. It is provided directly by user.

The *BCOnt Resolver* (Fig. 6, Mark 5) is the processing unit which checks the syntax correctness of the BC_{req} and resolves the specified concepts. As shown in Fig. 6, Mark 6, the access to the BCOnt ontology is established by *Jena Semantic Web Framework* [16]. This is the Java framework used for building Semantic Web applications. It is integrated with the *Pellet reasoner* and includes the *SPARQL Engine* [17]. SPARQL is an RDF based query language which is applied to retrieve concepts specified by the BC_{req}. In case that some of these concepts is not defined by BCOnt, our ontology can be interconnected to external ontologies located in the scope of LOD where the missing concept is defined (Fig. 6, Mark 7). The corresponding piece of pseudo-code is presented in Alg. 1 and explained in the following.

If the current processing concept can not be resolved from the BCOnt ontology (Alg. 1, Line 2), a connection to DBpedia ontology [11] is established and the missing concept is queried through the SPARQL endpoint (Alg. 1, Line 6). In case that the missing concept can be refined invoking DBpedia (Alg. 1, Line 7), linking between BCOnt and DBpedia is established (Alg. 1, Line 9). More precisely, in the example implemented by Alg. 1, the relationship between corresponding instances of the class *Country* defined by BCOnt (shown in Fig. 2) and the corresponding instances of the class *City* defined by DBpedia are set up.

Algorithm 1 Example: BCOnt - LOD interrelation

Input: $conceptName$
Output: $resolved\ concept$

```
1:  ...
2:  if !BCOnt.contains(conceptName) then
3:      String service = "http://DBpedia.org/sparql";
4:      String DBpediaUniqueID =
        "<http://dbpedia.org/resource/" + conceptName;
5:      String query = "SELECT ?X WHERE { " +
        DBpediaUniqueID +
        "<http://dbpedia.org/ontology/country> ?X}";
6:      Result r = SPARQLService(service, query);
7:      if r! = null then
8:          OntConcept parentConcept =
            BCOnt.getConcept(r.getConceptName);
9:          linking(parentConcept, DBpediaUniqueID);
10:     else
11:         print("Concept is not defined in DBpedia.");
12:     end if
13: end if
14: ...
```

GenBDocIG Generator. The *GenBDocIG Generator* (Fig. 6, Mark 9) is the processing unit used to develop the *GenBDocIG*. It extracts BIEs located in the available *ExistBDocIGs* and embeds them into the generic guideline.

CustBDocIG Generator. The *CustBDocIG Generator* (Fig. 6, Mark 10) is the processing unit used to customize the previously developed *GenBDocIG* and to create the new *CustBDocIG* valid in the BC_{req}. It invokes the *BIE Extractor* (Fig. 6, Mark 11), the processing unit which extracts all BIEs encompassed by the *GenBDocIG*. The included *BIE Library* (Fig. 6, Mark 12) represents our Java implementation of the BIE models defined by CCTS and introduced in Section II-B.

Effective BIEs Extractor. The *Effective BIEs Extractor* (Fig. 6, Mark 13) is the processing unit used to extract only those BIEs from the previously created list of the generic BIEs which are valid in the BC_{req}. The corresponding pseudo-code is shown in Alg. 2 and explained in the following.

The ABIEs contained in the list of the generic ABIEs are processed within the loop initiated in Alg. 2, Line 2. As explained in Section II-C, the overall BC of an ABIE is calculated based on the union of the overall BCs of its included BIEs (BBIEs and ASBIEs). Therefore, if the currently processing ABIE is valid in the BC_{req} (checked in Alg. 2, Line 4), it is possible that it contains the BIEs which are valid in the BC_{req}. These BBIEs and ASBIEs are selected in Alg. 2, Lines 7 and 12, respectively. The non-selected BIEs are not relevant in the current business scenario and, thus, they are excluded from the further processing.

The new ABIE which contains only the previously selected BIEs is generated in Alg. 2, Lines 16-18. Thus, this newly created ABIE originates from the same ACC as the currently processing ABIE, but it is *derived by restriction* based on the different BC (BC_{req}). The whole list of the effective ABIEs is forwarded to the *CustBDocIG Generator* where it is embedded into a new BDocIG.

BIE BCs Calculator. The *BIE BCs Calculator* (Fig. 6, Mark 14) is the processing unit used to calculate the overall

Algorithm 2 Effective BIEs Extractor

Input: $genABIEList$
Output: $effABIEList$
1: $genABIEList = Alg\_3(genABIEList)$;
2: **for each** $abie : genABIEList$ **do**
3: $newBBIEList = null$; $newASBIEList = null$; $fReused = false$;
4: **if** $requriedBC \subset abie.overallBC()$ **then**
5: **for each** $bbie : abie.BBIEList$ **do**
6: **if** $requiredBC \subset bbie.overallBC()$ **then**
7: $newBBIEList.add(bbie)$; $fReused = true$;
8: **end if**
9: **end for**
10: **for each** $asbie : abie.ASBIEList$ **do**
11: **if** $requiredBC \subset asbie.overallBC()$ **then**
12: $newASBIEList.add(asbie)$; $fReused = true$;
13: **end if**
14: **end for**
15: **if** $fReused$ **then**
16: $newABIE = abie$;
17: $newABIE.set(newABIEList)$;
18: $newABIE.set(newASBIEList)$;
19: $effABIEList.add(newABIE)$;
20: **end if**
21: **end if**
22: **end for**
23: **return** $effABIEList$

BCs in which the generic (already existing) ABIEs are valid. It is invoked by the previously explained *Effective BIEs Extractor* in Alg. 2, Line 1. The corresponding pseudo-code is presented in Alg. 3 and explained in the following.

The ABIEs contained in the list of the generic ABIEs are processed within the loop initiated in Alg. 3, Line 1. According to Formula 1 explained in Section II-C, an ABIE is valid in the BC which is calculated as the union based on the following two components: (i) the union of the assigned BCs in which its included BBIEs are valid, and (ii) the union of the overall BCs in which its included ASBIEs are valid. The first component of the overall BC of the currently processing ABIE is calculated in Alg. 3, Lines 3-5. If this ABIE does not contain any ASBIE, the second component of its overall BC is null. Thus, its previously calculated component of the BC is equal to its overall BC (Alg. 3, Line 12). However, if the currently processing ABIE contains ASBIEs, the second component of its overall BC is not null, and it is calculated involving the *ASBIE BCs Calculator* (Alg. 3, Line 17).

ASBIE BCs Calculator. The *ASBIE BCs Calculator* (Fig. 6, Mark 15) is the unit used to calculate the overall BCs in which the ASBIEs contained by the generic ABIEs are valid. It is implemented by the recursive algorithm which pseudo-code is presented in Alg. 4 and explained in the following.

The ABIE which encompasses the currently processing ASBIEs (associating ABIE) is the input parameter of Alg. 4. The ASBIEs contained by the input ABIE are handled within the loop initiated in Alg. 4, Line 2. According to Formula 2 explained in Section II-C, the overall BC of an ASBIE is dependant and, thus, calculated based on the intersection of its assigned BC and the overall BC of its associated ABIE. Therefore, there are two options (checked in Alg. 4, Line 4) for the following execution steps of Alg. 4 : (i) the overall BC of the associated ABIE is still unknown, and (ii) the overall BC of the associated ABIE has already been calculated.

Algorithm 3 BIE BCs Calculator

Input: $ABIEList$ {BIE_overallBCs are not calculated.}
Output: $ABIEList$ {BIE_overallBCs are calculated.}
1: **for each** $abie : ABIEList$ **do**
2: $abie.overallBC = null$;
3: **for each** $bbie : abie.BBIEList$ **do**
4: $abie.overallBC = abie.overallBC \mid\mid bbie.assignedBC$;
5: **end for**
6: **if** $abie.hasASBIEs$ **then**
7: **for each** $asbie : abie.ASBIEList$ **do**
8: $asbie.isOverallBCCalculated = false$;
9: **end for**
10: $abie.isOverallBCCalculated = false$;
11: **else**
12: $abie.isOverallBCCalculated = true$;
13: **end if**
14: **end for**
15: **for each** $abie : ABIEList$ **do**
16: **if** $!abie.isOverallBCCalculated$ **then**
17: $abie.overallBC = Alg\_4(abie)$;
18: $abie.isOverallBCCalculated = true$;
19: **end if**
20: **end for**
21: **return** $ABIEList$

Algorithm 4 ASBIE BCs Calculator

Input: $abie$ {associating ABIE, overall BC is not calculated}
Output: $abie$ {associating ABIE, overall BC is calculated}
1: **if** $!abie.isOvelrallBCCalculated$ **then**
2: **for each** $asbie : abie.ASBIEList$ **do**
3: $r = asbie.associatedABIE()$;
4: **if** $!r.isOverallBCCalculated$ **then**
5: $asbie.overallBC = asbie.assignedBC \;\&\&\; Alg\_4(r)$;
6: **else**
7: $asbie.overallBC = asbie.assignedBC \;\&\&\; r.overallBC$;
8: **end if**
9: $asbie.isOverallBCCalculated = true$;
10: $abie.overallBC = abie.overallBC \mid\mid asbie.overallBC$;
11: **end for**
12: $abie.isOverallBCCalculated = true$;
13: $ret = abie.overallBC$;
14: **else**
15: $ret = abie.overallBC$;
16: **end if**
17: **return** ret

In this execution phase, the overall BC of the associated ABIE is unknown iff this ABIE contains at least one ASBIE which overall BC has not been processed yet. Therefore, Alg. 4 is recursively called (Alg. 4, Line 5) where the associated ABIE is indicated as the new input parameter. In case that the overall BC of the associated ABIE is already known, the exit condition of the recursion is reached, and the overall BC of the currently processing ASBIE is calculated (Alg. 4, Line 7). Finally, the previously calculated component of the BC in which the associating ABIE is valid is unionised with the overall BC in which the currently processing ASBIE is valid in Alg. 4, Line 10.

Output. The final output of our approach is the *CustBDocIG* which is presented following the contextualized XML NDR specification. It is valid in the BC_{req} which is expressed using our BCOnt model. As explained in the previous Subsections, the customization steps are controlled by the *CustBDocIG Generator* which forwards the resulting guideline to the *BC Processing Tool*.

C. Evaluation

The evaluation of the proposed architecture is the current phase of our research. The usability and functionality of the architecture have already been analyzed. In respect to the Design Science Research (DSR) methodology [18], this phase of evaluation was conducted as build and evaluate loop iterated a number of times before the final algorithms were developed.

First, the exemplary BCOnt which consisted of 21 classes, 1015 instances and 193 properties was built. Second, the BCFAO subontology was interconnected with the external DBpedia ontology. Third, the set of 20 already existing contextualized business document implementation guidelines based on the UN/CEFACT document standard and enhanced NDR specification was prepared. Before every iterative step of the evaluation, one of these guidelines (denoted by $BDocIG_{selected}$) was arbitrary selected to be a missing guideline. Therefore, the BC in which this guideline was valid was processed by the system as the $BC_{required}$ input parameter. The rest of the guidelines from the introduced set of guidelines was processed as the *ExistBDocIGs* input parameter.

The resulting set of the Core Components contained by the generated *CustBDocIG* was analyzed and compared with the set of the Core Components contained by the original $BDocIG_{selected}$. Finally, in the later iterations of this evaluation phase we could conclude that the corresponding data building blocks located in both guidelines were matched. Thus, usability and functionality of the approach were proved. In the current phase of the research we evaluate the BCOnt model against our concurrent Enhanced Unified Context (E-UCM) model [4] also applied to formally represent BC.

V. Conclusion and Future Work

In this paper we presented the implementation our approach to calculate the content model of the business context aware BDocIGs. The corresponding e-documents conform to the CCTS document standard and they are exchanged between business partners when executing inter-organizational business processes. The proposed solution harnesses the contextual knowledge which comprises the circumstances (industry, geopolitical region and activity) where the e-business documents are valid. The applied BC is represented by our ontology based business context (BCOnt) model. This is the hierarchical model which has reasoning capabilities and can be interrelated with external ontologies located in the scope of LOD.

First, the Core Components are extracted from the already existing documents and embedded into the *GenBDocIG*. Second, our algorithm detects only those Core Components from the *GenBDocIG* which are relevant in the required BC and build them into the *CustBDocIG*. Finally, we can apply the reasoning techniques (learning from a DAG and learning from a knowledge database) and use the derived contextual knowledge to iteratively repeat the previous step.

We implement our conceptual solution adapting the NDR specification and developing modules of the BC aware service oriented architecture presented in Fig. 6. The final outcomes of our approach are new, more homogeneous BDocIGs which are valid in the BC required by user. Based on the results of our usability and functionality evaluation, we conclude that our conceptual solution holds not only in theory, but in practice as well. In our current work we evaluate the results achieved by application of the BCOnt model in different real-world business scenarios against the corresponding results achieved in case when our concurrent E-UCM BC model [4] is applied. Because of space limitations, the status of this phase of our research is not in the scope of this paper, but it's current outcomes can be found in [19].

References

[1] UN/CEFACT. Core Components Technical Specification CCTS, version 3.0. http://www.unece.org/fileadmin/DAM/cefact/codesfortrade/CCTS/, September 2009. Last Visit: August 2013.

[2] Danijel Novakovic and Christian Huemer. Business context sensitive business documents: An ontology based business context model for Core Components. In *Proceedings of the 10th Intl. Conf. for Informatics and Information Technology (CIIT 2013), Bitola, Macedonia*, 2013.

[3] Danijel Novakovic and Christian Huemer. Contextualizing business documents. *To appear in the 10th IEEE Intl. Conf. on e-Business Engineering (ICEBE 2013), United Kingdom*, 2013.

[4] Danijel Novakovic and Christian Huemer. Business context sensitive business documents: Business context aware Core Components modeling using the E-UCM model. In *Proceedings of the 11th IEEE Intl. Conf. on Industrial Informatics (INDIN 2013), Bochum, Germany*, 2013.

[5] Danijel Novakovic and Christian Huemer. Context aware business documents modeling. *To appear in Modeling and Using Context - 8th Intl. and Interdisciplinary Conf., CONTEXT 2013, France*, 2013.

[6] Anind K. Dey and Gregory D. Abowd. Towards a better understanding of context and context-awareness. In *Workshop on The What, Who, Where, When, and How of Context-Awareness, as part of the Conference on Human Factors in Computing Systems (CHI 2000)*, April 2000.

[7] Thomas Strang and Claudia Linnhoff-Popien. A context modeling survey. In *First International Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp*, August 18 2004.

[8] Danijel Novakovic and Christian Huemer. A survey on business context. In *Proceedings of the Intl. Conf. on Advanced Computing, Networking, and Informatics (ICACNI), Raipur, India*, 2013.

[9] Caterina Caracciolo, Marta Iglesias Sucasas, and Johannes Keizer. Towards interoperability of geopolitical information within FAO. *Computing and Informatics*, 27(1):119–129, 2008.

[10] Department of Economic and Social Affairs Statistics Division. *International Standard Industrial Classification of All Economic Activities (ISIC), Revision 4*. United Nations Publications, 2009.

[11] Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Synthesis Lectures on the Semantic Web. Morgan & Claypool Publishers, 2011.

[12] Sebastian Rudolph. Foundations of description logics. In *Reasoning Web*, pages 76–136, 2011.

[13] UN/CEFACT. UN/CEFACT XML Naming and Design Rules technical specification version 3.0. http://www.unece.org/cefact/xml/UNCEFACT+XML+NDR+V3p0.pdf, 2009. Last Visit: August 2013.

[14] Daniel L. Rubin, Holger Knublauch, Ray W. Fergerson, Olivier Dameron, and Mark A. Musen. Protégé-OWL: Creating ontology-driven reasoning applications with the web ontology language.

[15] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical OWL-DL reasoner. *J. Web Sem*, 5(2):51–53, 2007.

[16] HP Labs. Jena - A Semantic Web Framework for Java. http://jena.sourceforge.net/, 2004. Last Visit: August 2013.

[17] W3C SPARQL Working Group. SPARQL 1.1 overview. World Wide Web Consortium, Working Draft WD-sparql11-overview-20111117, November 2011.

[18] Alan Hevner and Samir Chatterjee. *Design Research in Information Systems*. Springer US, 2010.

[19] Danijel Novakovic and Christian Huemer. Context aware modeling of business document implementation guidelines. http://web.student.tuwien.ac.at/~e1028296/phdwork/, June 2012. Last Visit: August 2013.