Mapping ebXML standards to ontology

Branko Arsić[1], Marija Đokić[1], Nenad Stefanović[1]
[1]*Faculty of Science, University of Kragujevac*

Abstract - *Finding the best business partner can be a real challenge due to the fact that it is often necessary to exchange large amounts of data and documents. For efficient and flexible B2B cooperation, in modern enterprise, ebXML standards can be applied. Collaboration Protocol Profile (CPP) and Collaboration Protocol Agreement (CPA) as a part of ebXML are used to make ad hoc agreements on the electronic document communication between two companies. In this paper, ontologically-enhanced ebXML is presented and the effort has been invested in translating CPPs to ontologies. The presented step pave the way for the future SPARQL based reasoning over these OWL documents aiming to create CPA document. Our mapping approach can be used for every ebXML standards and utilize all the benefits that ontology offers, such as reasoning, defining rules and computing additional property values.*

1. INTRODUCTION

In B2B e-commerce, there are many standards which provide means to exchange data between applications [1]. But, it does not guarantee interoperability. On the syntactic level, this requires a treaty on an e-business vocabulary and even more important, on the semantic level, business partners must share a common attitude unambiguously constraining the generic document types [2]. If we talk about benefits of using this standards, it is known that processing and communication between companies, using this systems, are quicker and less error-prune, because we have less human intervention and less paper work which reduces possibility for errors and so there will be better accuracy, stability, efficiency and dependability of company method. At the beginning, in a series of e-business standards we are mentioning some of them.

Electronic Data Interchange (EDI) [3] is the process by which companies and institutions exchange trade related documents in electronic form. EDI is based on the concept of transactions that comprise messages (business documents) in predefined formats. However, using of ad-hoc conducting business between companies without prior agreement of any kind was not possible.

One of the most popular standards, the Extensible Markup Language (XML) [4] has become the first choice for defining data interchange formats in business collaboration. XML provides a more open and flexible business of transactions unlike the EDI.

RosettaNet [5] defines common inter-company public processes and their associated business documents expressed in DTD or XML Schema in order to reducing of cost and extensibility benefits. But use of these documents does not resolve interoperability issues in B2B integrations, because they don't have expressive power to capture all necessary constraints.

However, globally accepted standard that will provide cooperation between companies is still missing. This standard should enable companies to find most suitable business partner. Agreement between companies should be established by using this standard in a short time and if possible automatically. Accepted standard should enable enterprises to collaborate even if they have different business applications.

One of the standards that meets the above conditions is ebXML (Electronic Business using eXtensible Markup Language) [6]. ebXML is a suite of specifications that enables enterprises to conduct business over the internet. The specifications cover the analysis of business processes and business documents, the documentation of capabilities of company and the document transfer to conduct e-business. A lot of large companies are involved in developing and evaluating this standard and this makes it very attractive for the others to embrace it. Collaboration-Protocol Profile (CPP) and Collaboration-Protocol Agreement (CPA) [7] as a part of the ebXML standard allow finding the most suitable business partner and make ad hoc agreements on the electronic document communication. A CPP defines the company's capabilities to involve in electronic business with other companies. CPA is a document that represents a formal agreement between collaborating companies and defines specific terms of message exchange that is agreed between the two companies.

Many approaches are proposed in order to standardization of the ebXML. However, they lack the semantic representation. The proposed ontologically-enhanced ebXML solution shows how existing ebXML challenges can be solved by using semantic technologies and the effort has been invested in translating CPPs to ontologies [8]. For CPP XML Schema we created a respective OWL model [9]. Further, we generate one out of the CPP instance document. Approach for improvement mapping scheme to present OWL concepts in terms of ebXML construct is described. This access exploits new ability and power for efficiently semantic characterization of documents.

The paper is structured so that the second section gives an overview of literature that is critically analyzed. The third section talks about motivation for this work and what benefits we can get at the end. The fourth section presents part of ebXML - CPP and CPA documents. In the fifth section focus is on the process of mapping the CPP documents into ontology. Last section is reserved for conclusions and future work.

2. RELATED WORK

The main goal of developing a standard for e-business is to create a basis for automated mapping system that would be able to convert the concepts of various standards in an independent manner. This entails that the meaning of terms, relationships, restrictions and rules in the standards should be clearly defined in the early stages of standard development.

Semantic Web [10] and ontologies [11] as core component of Semantic Web have a potential to deal with these problems. The idea of the Semantic Web and its related technologies are expressing information not only in natural language or in predominantly syntactical formats, but in a way that it can be read and used by software agents.

The real question which imposes is the aim of ontologizing. What system we need to obtain so it can be expandable? The solution should be flexible to adding new partners and new collaborative processes and support coping with inevitable variations that come when working with multiple partners. Companies have invested considerable amounts of money and resources to implement current B2B integrations based on existing B2B standards, and they have the supporting infrastructure largely in place. The papers [11, 12, 13] described several mapping processes from e-business standards to ontologies, trying to resolve heterogeneities, not structurally and semantically covered by the specifications of these standards.

The step of mapping is in our focus, and it is only one step of our unified semantic B2B collaboration model which is now being developed. It comprises the complete B2B process automation cycle starting from creation of CPPs, and all the way to the semi-automatic generation of CPA.

Several strategies for mapping XML technologies to OWL have been proposed [14, 15, 16, 17]. Some papers focused more on a general mapping between XML and RDF, others aim at mapping XML Schema to OWL without considering XML instance data.

Our work improves the previous approaches of mapping in creating individuals for every CPP document element. In previous works, with Object Properties also come the individuals, but without naming convention and solving conflicts with the same-type elements and same-level individuals. Especially, we pay attention to this part of mapping process, because it is crucial in our model. Our present research is based on SPARQL reasoning [18] over individuals with main goal to get CPA document as final product. Further, we are using the names of attributes for detecting semantic in CPP documents, so we are able to present it in ontology. Other approaches didn't deal with the discovering semantics in and out of the document structure, so this is another improvement of previous works.

Our approach proposed the method for translating CPPs documents to ontology. The representation of standards into ontology allows automation of mutual mapping process by using different computational algorithms and possibility to avoid all disadvantages of the XML representation. This paper is also an application of some existing and novel approaches, but in a new domain (ebXML CPP), because there is not any comprehensive paper with similar topic. Previous studies have mainly focused on improving the ontological ebXML registry and repository [19].

3. MOTIVATION

It is known fact that messages have been designed only syntactically. With an ambiguity of natural languages come real problems on the precise meaning of the words. As a consequence the set-up time is quite involved as it often takes developers a long time to agree on the precise meaning of the message content. Frequently mentioned example in the literature is the shipping date. Does the shipping date refer to the date the supplier transfers the goods to the shipper or the date when the transport vehicle departs the premises?

Ontologies represent a core pillar of the Semantic Web idea, as they define a set of concepts within a domain and the relationships between those concepts. More formally, an ontology defines the vocabulary of a problem domain, and a set of constraints (axioms or rules) on how terms can be combined to model specific domains. An ontology is typically structured as a set of definitions of concepts and relations between these concepts. Ontologies are machine-processable, and they also provide the semantic context by adding semantic information to models, thereby enabling natural language processing, reasoning capabilities, domain enrichment, domain validation, etc.

Current EDI, RosettaNet and ebXML specifications are not represented in a languages which support reasoning about the objects within that domain. Ontology Engineering (OE) is sometimes seen as the next level in knowledge modeling, aiming at avoiding conceptual ambiguities, advocating reuse and standardization, and serving as building blocks for more complex automated-reasoning systems [20]. In addition, it is possible to define property domains, cardinality ranges, and reasoning rules. Some reasoning engines, such as Pellet, or Jess can be used to infer additional facts about the knowledge explicitly included in OWL ontologies. Reasoning in OWL can be performed at a class, property or instance level. For example, it is possible to define rules for checking class equivalence, for classifying individuals, or for computing additional property values using transitiveness.

The analysis of Appendix E of the CPPA specification showed how CPPA elements and attributes are inter-dependent. For example, if non repudiation is required, then the necessary child elements must be present. XML schema cannot express all possible rules, so an additional

rules definition in the forms of ontology could be of value and natural choise.

4. EBXML STANDARDS - CPP AND CPA

Enterprises must collaborate with each other largely if they want to complete their cooperation. Exchanging data and different types of documents between them can be extremely difficult process. Many large companies and enterprises are involved in developing and evaluating ebXML standard in order to carry out these processes.

Main parts of ebXML standard are Collaboration Protocol Profile (CPP) and Collaboration Protocol Agreement (CPA). Major purpose of these documents is to help find the most relevant business partner and make ad hoc agreements on the electronic document communication as specified in [7].

The CPP defines the party's opportunities to engage in electronic business with other parties. Opportunities include business processes that the party supports as well as particular technical details of supported means of message exchange. CPP is an XML document containing elements that describe the processing of some business unit such as *PartyInfo* and *Packaging*. The first element describes supported business collaborations, role in the business collaboration and the technology details about message exchange (Fig.1) [7]. The second element provides specific information about how the Message Header and payload constituent(s) are packaged for transmittal over the transport.

Process specification layer defines interaction between partners, actually role that each company plays. *DeliveryChannels* layer represents the characteristics of Message-receiving. CPP can have several delivery channels. *DocExchange* layer accepts the business document of one company from *ProcessSpecification* layer, encrypts and/or adds a digital signature (based on specification), and sends it to transport layer for transmission to another partner. *Transport layer* is responsible for message delivery using the selected transport protocol. *Packaging elements* ensures information about message header packaging for transfer over the transport layer.

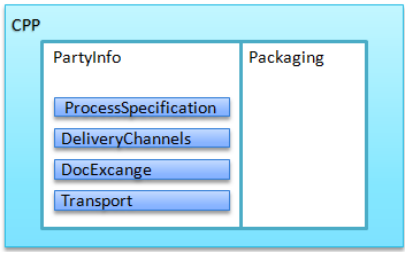

Figure 1. Layered structure of CPP

The CPA is a document that represents special terms of message exchange that is agreed between the two parties. It represents a formal management between of collaborating parties. CPA is actually result of intersection of two CPPs. This document contains common or compatible elements between partners. CPA document consists of general information about this document, two *PartyInfo* elements – for each business partner and one *Packaging* element. PartyInfo elements have the same structure like particular *PartyInfo* element in CPP. If negotiation involved values of some *PartyInfo* elements must be changed. Packaging element must be the same as appropriate element in CPP.

The example (Fig.2) shows how one company can engage in ebXML, how an already registered ebXML company searches for a new trading partner and how both then engage in electronic business.

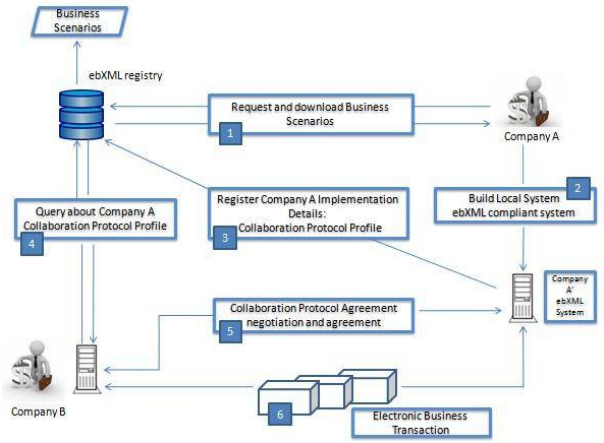

Figure 2. ebXML Overview (adapted from the ebXML Technical Architecture specification)

1. Company A browses the ebXML Registry to see which collaborative business processes and business document schemas are already available.
2. Company A needs a local ebXML system to communicate with trading partners.
3. Company A has to create a CPP which describes the supported collaborative business process capabilities. After that Company A registers its CPP at the ebXML Registry.
4. Company B is already registered at the ebXML registry and is looking for new trading partners. Company B queries the ebXML Registry and receives company A's CPP. Company B then has two CPP's: Company A's CPP and its own. The two companies have to come to an agreement on how to do business (CPA).
5. The CPA template has to be accepted by both parties. A CPA negotiation finalizes the CPA template to a final CPA if there is anything left to be negotiated.
6. The companies then use the underlying ebXML system to exchange business documents conforming to the CPA.

5. MAPPING OF EBXML STANDARDS TO ONTOLOGY

The ontologies have the potential to improve the quality of standards and lead to infer additional facts about the knowledge. In our research we decided to make the mapping of CPP electronic business standard by developing individual ontologies. In the focus of our research are documents which are based on Collaboration Protocol Profile and Agreement specification version 2.0. In CPP there are several *Certificate* parts, several *Transport* and *Documents Exchange* ways, a few *Security* parts which company offers in its business.

ebXML standards cover the area of electronic procurement and a mapping between them is a real and practical need because with OWL we get some advantages in comaprision with XML. Further, we present our algorithm steps for mapping CPP documents to OWL. Same steps could be used for any ebXML standard.

FIRST STEP: All XML elements of CPP are mapped to **OWL Classes**.

The name of the class is of type Internationalized Resource Identifiers (IRI). Each IRI is defined by its name and namespace taken from original CPP document. We use the same namespaces as defined by the CPP XML Schema [7].

SECOND STEP: Hierarchy between CPP elements is mapped by using the **rdfs:subClassOf** property. (Fig.6)

THIRD STEP: Creating the **individuals** for every element using new name convention:

a) If mapping element has its own **unique id number**, we take it for name

b) Otherwise, parent name + capitalize_first_letter (acronym of element name)

For example,

```
<tp:Transport tp:transportId="transportA1">
  <tp:TransportSender>...</tp:TransportSender>
  <tp:TransportReceiver>...</tp:TransportReceiver>
</tp:Transport>
```

Listing 1. *Transport* layer in CPP document

we get following individuals: "*transportA1*" for class *Transport*, "*transportA1Ts*"for *TransportSender* class and "*transportA1Tr*" for *TransportReceiver* class. In this way we can create individuals with different names and concrete values to make them SPARQL [21] aware for comparisons.

Exception for individuals is *PartyInfo* class whose individual take name of the *PartyName* attribute for simplicity, instead of *PartyId* text node.

A very important fact is that all same-type individuals have fixed suffix name. For every same-type element, in any CPP document, we create the individual using parent name as basename and abbreviation of element name as suffix according to rule b). The only difference is a basename. Conflicts for same-type elements at the same level, within same CPP document, are solved by using different element's ID attribute according to rule a) and CPP XML Schema. At any time we know the classes to which individuals belong, so we have a rule for future individual comparisons. For example, *transportA1* → *transportA1Ts*, *transportA1TsTcs, etc*; *transportA2* → *transportA2Ts*, *transportA2TsTcs, etc*. Same classes individuals are pairs (*transportA1, transportA2*), (*transportA1Ts, transportA2Ts*), (*transportA1TsTcs, transportA2TsTcs*), etc.

FOURTH STEP: Beside OWL classes and individuals, every child element is mapped into OWL syntax as **OWL Object Property**. The name of the object property is formed as an IRI, where the name has the prefix "**has**".

The OWL Class to which a particular property belongs, is mapped by using *rdfs:domain*. The OWL subclass for Object Property are mapped as *rdfs:range*. There is no other way to connect individuals in different levels.

FIFTH STEP: If one element in CPP has several ID numbers, we only use relevant ID for naming, and the other ID's are used as **OWL Object Property** for linkage among them.

```
<tp:DeliveryChannel tp:channelId="syncChannelA1"
 tp:transportId="transportA1"
 tp:docExchangeId="docExchangeA1">
 ...
</tp:DeliveryChannel>
```

Listing 2. *DeliveryChannel* layer in CPP document

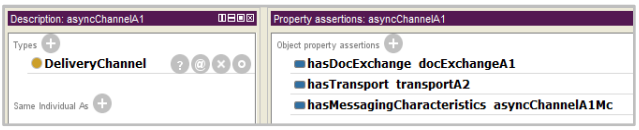

Figure 3. Object properties for *asyncChannelA1* individual

SIXTH STEP: Element attributes are mapped as **OWL Datatype Properties**.

The attribute's OWL Class is mapped by using **rdfs:domain**. **rdfs:range** is used for defining primitive data types like built-in data types of XML Schema.

SEVENTH STEP: If element has **text node** we mapped it as „**hasValue**" OWL Datatype Property.

```
<tp:TransportProtocol tp:version="1.1">
  HTTP
</tp:TransportProtocol>
```

Here, except creating class and individual we mentioned earlier, we create also two Datatype Properties and for *TransportProtocol* individual we associated next values:

`hasValue = "HTTP"` and `version="1.1"`

According to CPP XML Schema the restriction for the attribute values is made by **rdfs:range** of OWL Datatype Property. Datatype Properties *hasValue* has range *xsd:string* and *uri* has range *xsd:anyUri* etc.

EIGHTH STEP*:* There are also attributes which have semantically important values set in advance, with indicator „**type=name.type**“ in schema. We mapped attribute to a particular **Datatype Property** with „**name**“ from the previous alias. Element's OWL class is **rdfs:domain** and resticted values are **rdfs:range**.

```
<tp:DeliveryChannel
tp:channelId="asyncChannelA1"
tp:transportId="transportA2"
tp:docExchangeId="docExchangeA1">
<tp:MessagingCharacteristics
   tp:syncReplyMode="none" tp:ackRequested="always"
   tp:ackSignatureRequested="always"
tp:duplicateElimination="always"/>
</tp:DeliveryChannel>
```

Listing 3. *DeliveryChannel* layer in CPP document

For *syncReplyMode* attribute we got the following note:

Figure 4. Datatype Property *SyncReplyMode*

Datatype properties with the same values set are mapped as equivalent datatypes. From the Listing 3, attributes *ackRequested*, *ackSignatureRequested* and *duplicateElimination* are equivalent to *perMessageCharacteristics* Datatype Property.

There are also other exeptions which we have to deal with. If some element has resticted values set for *hasValue* property (described in seventh step) we create it as subproperty. For naming subproperty we use attribute name (first letter written in lowercase) and *rdfs:range* with possible values.

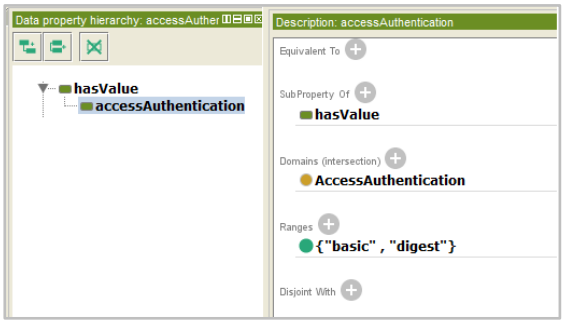

Figure 5. *AccessAuthentication* attribute with two possible values, *basic* and *digest*

The proposed mapping process is shown in the example of mapping from CPP to OWL. For simplicity of the presentation, definition of the CPP presented bellow does not include all the elements from the original definition. *PartyInfo* element point out to certain *DeliveryChannel* element, further, *DeliveryChannel* element point out to certain *TransportProtocol* and *DocExchange* elements.

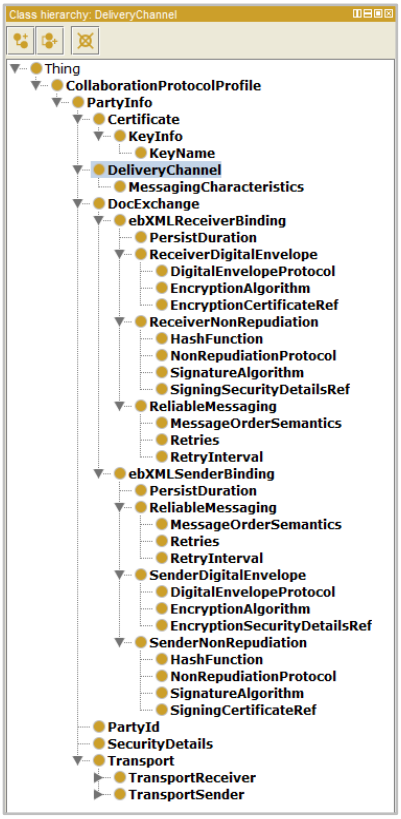

Figure 6. OWL Classes hierarchy

The CPP XML Schema is mapped in OWL only once. Further, we atomatically generate one out of the CPP instance document with individuals, following obtained ontology. The Listing 4 present the obtained ontology from one CPP document by using mapping process.

```
<tp:Transport tp:transportId="transportA1">
<tp:TransportSender>
<tp:TransportProtocol tp:version="1.1">
  HTTP
</tp:TransportProtocol>
<tp:AccessAuthentication>
  basic
</tp:AccessAuthentication>
<tp:AccessAuthentication>
  digest
</tp:AccessAuthentication>
<tp:TransportClientSecurity>
<tp:TransportSecurityProtocol tp:version="3.0">
SSL
</tp:TransportSecurityProtocol>
<tp:ClientCertificateRef
tp:certId="CompanyA_ClientCert"/>
<tp:ServerSecurityDetailsRef
tp:securityId="CompanyA_TransportSecurity"/>
</tp:TransportClientSecurity>
</tp:TransportSender>
<tp:TransportReceiver>
<tp:TransportProtocol tp:version="1.1">
  HTTP
</tp:TransportProtocol>
<tp:AccessAuthentication>
  basic
</tp:AccessAuthentication>
<tp:AccessAuthentication>
  digest
</tp:AccessAuthentication>
```

```
<tp:Endpoint tp:uri="https://www.CompanyA.com/servl
ets/ebxmlhandler/sync" tp:type="allPurpose"/>
<tp:TransportServerSecurity>
<tp:TransportSecurityProtocol tp:version="3.0">
SSL
</tp:TransportSecurityProtocol>
<tp:ServerCertificateRef tp:certId="CompanyA_Server
Cert"/>
<tp:ClientSecurityDetailsRef tp:securityId="Company
A_TransportSecurity"/>
</tp:TransportServerSecurity>
</tp:TransportReceiver>
</tp:Transport>
```

Listing 4. Generated ontology from CPP document

In this way we get mechanism to connect related individuals for SPARQL queries.

6. CONCLUSION

This paper presented a mapping of ebXML standards based on CPPA 2.0 specification into ontology using OWL. Our work was based on CPP and CPA standard, but the same mapping rules can be applied to all ebXML standards.

Our primary goal in the future will be the automation of negotiationing between different company's CPPs, and an important step towards its achievement is the mapping e-business standards to ontologies. Using our new approach we will try to match the ontologies parts that are the most relevant in negotiations using SPARQL queries and form document with best matches and document with conflicts. In this way we made this standards OWL aware and we are able to create reasoning rules because there exist inter-relationships among elements which are not presented in CPP format.

In parallel, we are building a system for the automatization of the CPA formation process, including the CPA composition in a bound environment. De facto, we will try to expand and elaborate early mentioned step 5 in Fig. 2.

7. REFRENCES

[1] Liegl, P., et al. "State-Of-The-Art In Business Document Standards".Industrial Informatics (Indin), 2010 8th Ieee International Conference On. IEEE, 2010.

[2] Hofreiter, B., and Christian, H. "B2B Integration - Aligning Ebxml And Ontology Approaches". Eurasia-Ict:Information And Communication Technology. Springer Berlin Heidelberg, 2002. pp. 339-349, 2002.

[3] Becker, M. "Electronic Data Interchange (EDI) (Interoperability Case Study)". Berkman Center Research Publication(2012-5), 2012.

[4] Boone, Keith W. "Extensible Markup Language".The Cda Tm Book. Springer London, pp. 23-34, 2011.

[5] Damodaran, S. "B2B Integration over the Internet withXML: RosettaNet Successes and Challenges", Proceedings of the Thirteenth World Wide Web Conference, pp. 188-195, 2004.

[6] UN/CEFACT, OASIS. ebXML Website. http://www.ebxml.org/

[7] Oasis Ebxml Collaboration Protocol Profile And Agreement Technical Committee, Collaboration-Protocol Profile And Agreement Specification Version 2.0.OASIS and UN/CEFACT, 2002.

[8] Oberle, D., Guarino, N., and Staab, S., "What Is An Ontology?". In: Handbook On Ontologies, Springer, 2nd Edition, 2009.

[9] "OWL 2 Web Ontology Language Document Overview". W3C. 2009-10-27.

[10] Berners-Lee, T., Hendler, J. and Lassila, O., "The Semantic Web". Scientific American, Retrieved March 26, 2008.

[11] Anicic, N., Nenad I., and Albert J., "An Architecture For Semantic Enterprise Application Integration Standards" . In Interoperability Of Enterprise Software And Applications, pp. 25-34. Springer London, 2006.

[12] Foxvog, D., and Bussler, C., "Ontologizing EDI Semantics". In Proceedings of the Workshop on Ontologising Industrial Standards, pp. 301-311. Springer, Tucson, AZ, USA, 2006.

[13] Haller, A., Gontarczyk, J., and Kotinurmi, P. "Towards A Complete Scm Ontology: The Case Of Ontologising Rosettanet". In Proceedings Of The 2008 Acm Symposium On Applied Computing (pp. 1467-1473). ACM, 2008.

[14] Ferdinand, M., Christian, Z. and David, T. "Lifting XML schema to OWL." In Web Engineering, pp. 354-358. Springer Berlin Heidelberg, 2004.

[15] Bohring, H. and Sören, A. "Mapping XML to OWL Ontologies." Leipziger Informatik-Tage 72, pp. 147-156, 2005.

[16] Anicic, N., Ivezic, N. and Marjanovic, Z. "Mapping XML schema to OWL." Enterprise Interoperability. Springer London, pp. 243-252, 2007.

[17] Bedini, I., et al. "Transforming XML schema to OWL using patterns." Semantic Computing (ICSC), 2011 Fifth IEEE International Conference on. IEEE, 2011.

[18] Coppens, S., Vander Sande, M., Verborgh, R., Mannens, E., and Van de Walle, R. "Reasoning over SPARQL". In Proceedings of the 6th Workshop on Linked Data on the Web, 2013.

[19] Dogac, A., et al. "Enhancing ebXML Registries To Make Them OWL Aware". Distributed And Parallel Databases 18(1), pp. 9-36, 2005.

[20] Eiter, T., Ianni, G., Polleres, A., Schindlauer, R., and Tompits, H., "Reasoning With Rules And Ontologies". In Reasoning Web. Springer Berlin Heidelberg. pp. 93-127, 2006.

[21] Eleven Sparql 1.1 "Specifications Are W3C Recommendations". W3.Org. 2013-03-21. Retrieved 2013-04-25.