Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 62 (2015) 457 – 466
Procedia Computer Science

The 2015 International Conference on Soft Computing and Software Engineering (SCSE 2015)

DTD2OWL^2: A new approach for the transformation of the DTD to OWL

Mokhtaria Hacherouf*, Safia Nait Bahloul[b]

*LSSD Laboratory, Faculty of mathematic and computer science, USTO-MB, BP 1505, EL-M'Naouer,31000 Oran, Algeria
[b]LITIO Laboratory, University of Oran 1,Ahmed Ben Bella, BP 1524, El-M'Naouer, 31000 Oran, Algeria

Abstract

The expansion of data sources existed in the web affects on the quality of research information. The correct answer (answer specific) of a request is all depend terms selected for their construction. Such as these terms sometimes mean more sense, the intended meaning may not be found. Fort this need, the Semantic Web has come to cover the semantic level, it proposed an ontological representation of data sources. This representation is implemented by OWL (Web Ontology Language). The current challenge of the Semantic Web is the transformation of data formats exist (SQL, XML ...) to the form of ontology (RDF, OWL ...), in order to facilitate the integration of data sources exist in the semantic web. Our target is to provide a series of extension concepts of DTD2OWL method, a simple and effective method for transforming XML documents into OWL ontologies.

Keywords: DTD, XML, mapping, OWL Ontology, DTD mixed.

1. Introduction

The publication of information on the current web is only interested by the data structure for facilitates their recovery, without taking into account, the context of use. The most search engines recover results without distinguishing the meaning of the search terms. Ex. "chat" is cat (animal) and cat (Internet chat)?. Sometimes you should open multiple web pages to find the search meaning. Ontology plays a very important role in the

* Mokhtaria Hacherouf. Tel.: +213 7 71 64 25 90.
E-mail address: Mokhtaria.hacherouf@univ-usto.dz

representation and the use of data. It can be used to describe the semantics of information sources and thus make their content explicit. Generally, the information published on the web is represented by XML [12], since it is the standard format for data exchange. The reuse of this information in the semantic web requires the transformation of XML to an ontology language. OWL [13] is just a language of ontology because it offers a wider vocabulary and a real formal semantics. This transformation can be achieved by creating a mappings between XML constructs (element , attribute, ...) and OWL constructs (classes , data property , object property , ...).

The big majority of approaches of transformation XML documents into OWL ([3], [4], [5], [6], [7], [8], [9], [10]) make a transformation from XSD schema. Other approaches proposed ([1], [2]) a direct transformation from XML instances. The objective of the transformation is to reuse the XML documents existed in the Semantic Web. These documents can be validated by the XSD schemas, as they can be validated by the DTD schemas. Then it is important to focus on the transformation of the two schemes (XSD and DTD).

The aim is to develop and expand the DTD2OWL approach [11], in order to give a complete transformation aimed at addressing all of the DTD schema structure.

In the rest of this paper, we begin with a brief overview of the approaches focus on the generation of OWL ontology from XML documents, second, the principle of DTD2OWL approach, in the third step, we give our detailed proposal, finally, the conclusion of our work.

2. Related Work

Mapping approaches are classified in terms of robustness and complete treatment of all constructions document (XML or XSD), and the ability to add more semantics to the XML document.

The approach proposed in [4], called XML2OWL was extended by the approach [9]. This approach treat imports (inclusion schemes), manages internal references in XSD schema construction, and offers controls on ontology generated. Both approaches have the meaning to generate the OWL model from the XSD schema that can be generated automatically, if it does not exist. The schemes automatically generated are not complete, for instance the lack of typing and options of attributes (required, optional, fixed value). Therefore, the generated ontologies are not semantically rich.

The approaches [1] and [2] add more semantics for the exiting XML schema**,** they manipulated the XML documents directly no need to sound their schema. They define notations to specify the mapping between XML and OWL. The problem arises when the mapped XML documents have a large size and it have a very long overlaps which make the notations longer and more complicated.

The most of approaches have proposed methods to mapped XML or XML-schemas with OWL. The approach [11] provides the mapping between the DTD and OWL. This approach aims to create mapping rules between the constructions of the DTD and OWL constructs, but it has not considered all constructions of the DTD, as: sequence, choice, enumeration…

These different approaches are distinguished by:

- The established correspondences between: XML instances and OWL, or between validation schema constructs and OWL constructs.
- The ontology generated from the XSD schema or DTD.
- The profile used OWL: OWL Lite, OWL-DL, OWL-FULL.
- The generation process of ontology and the generation process instances that can run things: parallel, sequential or independent.

According to these distinctions, we classify the approaches discussed in the following table:

Table1. Approaches classification of transforming XML to OWL.

Approach	Use of instance/schema XML	Generation schema	OWLlanguage profile	Generation model of ontology and individual
XML2OWL [4]	Validation schema	XSD	DL	Sequential
EXCO [9]	Validation schema	XSD	DL	Parallel
JXML2OWL [1]	XML instances	XPath expression	DL	Sequential
Martin et Amar [2]	XML instances	XPath expression + XML-Manchester	DL	Sequential
DTD2OWL [11]	Validation schema	DTD	DL	Sequential

3. DTD2OWL approach

The method DTD2OWL opted for XML document transformation to an OWL ontology by making two phases. The transformation at the DTD level is trying to convert all the elements and attributes of a DTD to classes or predicates in the target ontology. XML instances are transformed to individuals.

In the following table, we summarize the transformation rules of DTD constructs to OWL constructs:

Table 2. Mapping rules of DTD2OWL approach.

DTD Constructions	OWL Constructions
<!ELEMENT elt x> : x is the sequence of elements or "elt" have a lists of attributes.	<owl:class rdf:about="#elt"> <owl:disjointWith> <-- declaration of other classes> </owl:disjointWith> </owl:class>
<!ELEMENT elt1 (elt2)>	<owl:objectProperty rdf:ID="has_elt2"> <rdf:domain rdf;ressource="#elt1"> <rdf:range rdf;ressource="#elt2"> </owl:objectProperty>
<!ELEMENT elt2 (#PCDATA)> : the ancestor element of elt2 is elt1.	<owl:DataProperty rdf="elt2"> <rdf:domain rdf;ressource="#elt1"> <rdf:range rdf;ressource="#String"> </owl:DataProperty>
<!ATTLIST elt att CDATA>	<owl:DataProperty rdf="att"> <rdf:domain rdf;ressource="#elt"> <rdf:range rdf;ressource="#String"> </owl:DataProperty>
	Constraints mapping
#FIXED value, <!ENTITY entity-name "entity-value">	owl:haseValue
#REQUIRED	owl;minCardinality (=1)
#IMPLID	owl:cardinality (=0)
NOTATION	rdfs:Comment
+	owl;minCardinality (=1)
?	owl;minCardinality (=0)
*	owl;minCardinality (=0) owl;maxCardinality (=undounded)

4. Contribution

4.1. Global schema of DTD2OWL²

The DTD2OWL² approach is an extension of the DTD2OWL approach. In Fig. 1, we present the two modules (mixedDTD to simpleDTD and DTD2OWL² transforming) incorporated in the process of the method DTD2OWL [11]. The first step is to generate the DTD, if it does not exist, then, transform it on OWL ontology. Our contribution in this step is to add a very important phase before transforming the DTD. This phase is introduced when the DTD to be transformed is mixed type. For this type of DTD, a process is started to reorganize it to give a not-mixed DTD and ready to turn on OWL. The reorganization of the DTD consists to fusing the internal part of the DTD with the external part, by taking into account the redefined constructions. At the end, the mapping process transforms all DTD constructs to OWL ontology. Our contribution in this stage is to consider other constructions that were not treated with DTD2OWL. Once the ontology is created, the method DTD2OWL generates the individuals of this ontology from XML instances.

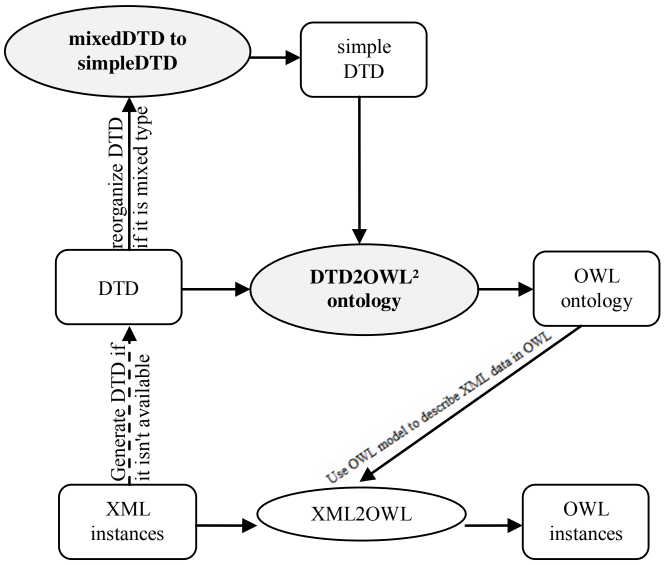
mixedDTD to simpleDTD
simple DTD
reorganize DTD if it is mixed type
DTD
DTD2OWL² ontology
OWL ontology
Generate DTD if it isn't available
Use OWL model to describe XML data in OWL
XML instances
XML2OWL
OWL instances

Fig. 1. Global architecture of DTD2OWL².

4.2. Transforming DTD schema to OWL ontology

The method DTD2OWL is simple and easy to implement. Conserving specific transformations for basic DTD constructs. However, there are other important constructions which are not considered by this method. Our work extends the DTD2OWL approach to treating the following questions:

- How to transform a mixed DTD: internal DTD referenced to an external one?
- How to express enumerations in OWL?
- How to express Choice/ Sequence elements in OWL?

What are the OWL constructs that express the type of attribute: ID and IDREF?

4.2.1. Sequences of elements

A sequence of elements is an ordered list of elements that should appear in the same order in the XML document. In a full declaration, the children must also be declared. However, even in OWL, the property "owl:intersectionOf" links a class to a group of classes declared in the same document OWL. [14]

We transform the DTD fragment defined in Fig. 2 by the OWL specification represented in Fig. 3.

```
    <!ELEMENT elt (elt1, elt2,)>
```

Fig. 2. Definition of sequence in DTD.

```
<owl:class rdf:about="#elt1"/>
<owl:class rdf:about="#elt2"/>
<owl:class rdf:about="#elt">
  <owl:IntersectionOf rdf:ParseType="#Colection">
    <owl:class rdf:resource="#elt1">
    <owl:class rdf:resource="#elt2">
  </owl:IntersectionOf>
</owl:class>
```

Fig. 3. Transforming of sequence into OWL.

4.2.2. Choice of elements

The construction of the choices in the DTD gives a choice from a list of several possible elements. However, in OWL there is a property "owl:unionOf" which allows to define a class as the union of other classes. This class can take the description of at least one of the classes in the list. Fig. 4 shows the structure of choice in the DTD. The OWL specification corresponding is given in Fig. 5.

```
<!ELEMENT elt2 (elt3| elt4)>
```

Fig. 4. Definition of choice in DTD.

```
<owl:class rdf:about="#elt3"/>
<owl:class rdf:about="#elt4"/>
<owl:class rdf:about="#elt2">
  <owl:UnionOf rdf:ParseType="#Collection">
    <owl:class rdf:resource="#elt3">
    <owl:class rdf:resource="#elt4">
  </owl:UnionOf>
</owl:class>
```

Fig. 5. Transforming of choice into OWL.

4.2.3. Attribute types: enumeration, ID and IDREF

Before dealing with these types of attributes, we don't forget that in DTDT2OWL approach, the attributes are conversed in OWL by "DatatypeProperty". This transformation can be maintained in the case where the attribute type is "CDATA". But when the type of attributes is: enumerate, ID and IDREF, the transformation will be changed.

Enumeration type. The attribute that limited by a list of possible attribute values is defined in OWL as a class. "ObjectProperty" is derived as follows: "has_name_of_attribute". Their domain is the class corresponds to the element containing this attribute. The range is the class itself. Class corresponds to the attribute is defined as an item listed with the construct "owl:oneOf".

In Fig. 6, the attribute "alt" can take one of the values: "val1" or "val2", otherwise the default value "val2". OWL construction corresponding to this DTD construction is given by Fig. 7.

```
<!ATTLIST elt att (val1, val2) “val2”>
```

Fig. 6. Definition of enumerated attribute in DTD.

```
<owl:class rdf:about="#elt"/>
<owl:class rdf:about="#att">
  <owl:OnOf rdf:ParseType="#Colection">
    <owl:Thing rdf:about="#val1">
    <owl:Thing rdf:about="#val2">
  </owl:OnOf>
</owl:class>
<owl:objectProperty rdf:ID="has_att">
  <rdf:domain rdf;ressource="#elt">
  <rdf:range rdf;ressource="#att">
</owl:objectProperty>
```

Fig. 7. Transforming of enumerated attribute in OWL.

ID type. An attribute of type "ID" allows to identify uniquely an element of the XML document. We define this attribute in OWL as a class. "ObjectProperty" is derived as follows: "has_name_of_l'attribut". This property is characterized by the specific property "FunctionalProperty". The domain of this property is the class corresponding to the element containing this attribute. The range is the class itself.

The structure of the ID attribute type is given by the Fig. 8. The transformation of this structure to OWL is presented in Fig. 9.

```
<!ATTLIST elt att ID>
```

Fig. 8. Definition of the ID attribute in DTD.

```
<owl:class rdf:about="#elt"/>
<owl:class rdf:about="#att"/>
<owl:objectProperty rdf:ID="has_att">
  <rdf:domain rdf;ressource="#elt">
  <rdf:range rdf;ressource="#att">
  <rdf:type rdf;ressource="&owl;FunctionalProperty">
</owl:objectProperty>
```

Fig. 9. Transforming of ID attribute into OWL.

IDREF type. An XML document can have a links between their data. The attributes of ID and IDREF type is jointly used to realize these links in a document. ID type attribute can be referenced by other elements through IDREF attributes. The value of IDREF attribute must bean XML name. This name must also be the value of an ID type attribute of an (or other) element.

We transform the attribute of IDREF type with the same monitoring mechanism for the transformation of the attribute ID, without the characterization of the property. To express the relationship between the type of ID and IDREF attributes in OWL, we use the restriction "owl:allValuesFrom". This restriction impose, for each instance of the class with instances of the specified property, that property values are all members of the class designated by the clause "owl:allValuesFrom" [15]. However, the element containing the IDREF type attribute is converted to a class defined as a subclass of an anonymous class defined by the universal quantifier.

Fig. 10 shows an example of an element declaration "elt2" contained an attribute "att2" of type IDREF. We believe that this declaration is fined with the example shown in Fig. 2. The OWL construction corresponding to this statement is given in Fig. 11.

```
<!ATTLIST elt2 att2 IDREF>
```

Fig. 10. Definition of IDREF type in DTD

```
<owl:classrdf:about="#elt2"/>
<owl:classrdf:about="#att2"/>
  <owl:objectPropertyrdf:ID="has_att2">
  <rdf:domainrdf;ressource="#elt2">
  <rdf:rangerdf;ressource="#att2">
</owl:objectProperty>
<owl:classrdf:about="#elt2">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onPropertyrdf:resource="#has_att2" />
      <owl:allValuesFromrdf:resource="#att" />
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Fig. 11. Transforming of IDREF construction into OWL.

4.2.4. Reorganization of mixed DTD

The Mixed DTD is formed by internal declarations followed by external declarations. It is possible to declare new elements in the mixed DTD, but it is forbidden to redefine elements of the external part in the internal part. We can redefine the entities and attributes in the internal part. Since this part is before the external part, then the redefined attribute is in priority. The following figure shows a recorded DTD under the name "library.xml". This DTD reference to another fragment of an external DTD registered under the name "biblio.dtd" (see Fig. 13).

InternalDTD: library.xml
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?> <!DOCTYPE library SYSTEM **"biblio.dtd"** [<!—modification of entity--> <!ENTITY %dtp ", datePublication?"> <!—Redefining of the element: datePublication --> <!ELEMENT datePublication (#PCDATA)> <!—Redefining of the attribute : name --> <!ATTLIST author name CDATA #REQUIRED>]>

Fig. 12. Declaration of mixed DTD.

ExternalDTD: biblio.dtd
<?xml version="1.0" encoding="iso-8859-1"?><!ENTITY %dtp""> <!ELEMENT library (book)> <!ELEMENT book (author %dtp;)> <!ELEMENT author (#PCDATA)> <!ATTLIST author name CDATA #IMPLID>

Fig. 13. External DTD declarations.

Conversing mixed DTD to OWL ontology cannot do directly. Then we propose a new method called "reorganization of the mixed DTD". This method executed on the mixed DTD before the transformation. It unfolds in three phases; each phase is piloted by a pseudo algorithm. the main algorithm which calls for these algorithms is:

```
Algorithm mixedDTDReorganisation(mixedDTDFile)
        externalDTD = returneExternalDTD(mixedDTDFile)
        externalDTDCleaned = cleanexternalDTD(externalDTD, internalDTD)
        internalDTDCleaned = cleaninternalDTD(mixedDTDFile)
        simpleDTD = consolidateDTD(externalDTDCleaned, internalDTDCleaned)
end algorithm
```

Fig. 14. Main algorithm of mixed DTD reorganization.

Step 1: Recovery of external DTD file. The process of this step makes a copy of the file containing the fragment of external DTD. The location of this file is returned from the URL that comes after the keyword "SYSTEM" in the internal DTD or in the second string after the word "PUBLIC". In the following example, the file to be recovered up is placed locally under the name "biblio.dtd":

Example:

1st case : <!DOCTYPE library SYSTEM "**biblio.dtd**"

2ed case : <!DOCTYPE library PUBLIC "FPI" "**biblio.dtd**"

The following fig. 15 shows the algorithm corresponding to this phase:

```
Algorithm returnExternalDTD(mixedDTD)
    fileURL = String that contained an URL
    externalDTDFile = file that contained external construct
    if DTD type is private
        fileURL is after "SYSTEM" keyword
    else if DTD type is public
        fileURL is the second String after "PUBLIC" keyword
    end if
    returnexternalDTDFile from fileURF
    returnexternalDTDFile
end algorithm
```

Fig. 15. Algorithm to return the external DTD.

Steps 2: Cleaning DTD. Once the process of the first phase recovers a copy of the external DTD, the process of the second phase makes another copy on the internal DTD. We make these copies to preserve the autonomy of the source. Entities and attributes redefined in the inner part are a priority. For this, the cleaning process removes this constructs from the external part. In the same part, the header is deleted. Another Cleaning process runs on the copy of the internal file. Their header is modified to become a non-mixed DTD, so the standalone = "no" is changed by standalone = "yes". The party "SYSTEM" "URL" is then deleted. Both algorithms given by Fig. 16 summarizes the treatments provided in this phase:

```
Algorithm cleanExternalDTD(externalDTD, internalDTD)
    delete header from externalDTD
    entitiesExt = all entities contained in the externalDTD
    entitiesInt = all entities contained in the internalDTD
    attributesExt = all attributes contained in the externalDTD
    attributesInt = all attributes contained in the internalDTD
    for each entity of entitiesExt
        if entity is different to entity of entitiesInt
        delete entity from externalDTD
    end for
    for each attribute of attributesExt
        if type or option of attribute are is different to type or option of attributesInt
            delete attribute from externalDTD
        end if
    end for
    returnexternalDTD
end algorithm

algorithm cleanInternalDTD(internalDTD)
    standaloneValue = value of standalone of internalDTD; standalone = "yes"
    delete “SYSTEM "URL" construct from internalDTD
    returninternalDTD
end algorithm
```

Fig. 16. Cleaning algorithm.

We apply the cleaning algorithms above on the files represented by Fig. 12 and 13, we obtain as a result two cleaned files and ready to be merged:

```
Simple DTD : library.xml
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
<!DOCTYPE library [
<!ENTITY %dtp  ", datePublication?">
<!ELEMENT datePublication (#PCDATA)>
<!ATTLIST author name CDATA #REQUIRED> ]>
```

Fig. 17. The internal DTD after cleaning.

```
ExternalDTD: biblio.dtd
<!ELEMENT library (book)>
<!ELEMENT book (author %dtp;)>
<!ELEMENT author (#PCDATA)>
```

Fig. 18. The external DTD after cleaning.

Steps 3: Fusion of DTDs. After the cleaning process, the process of the fusion copies the declarations of the external part in the internal file after their declarations before the closing"])". The following algorithm of fig. 19 is supported by the treatment of this phase:

```
algorithm consolidateDTD(externalDTD, internalDTD)
        constructs = all constructs contained in externalDTD
        {copy function permits to copy a constructs in the internalDTD after their constructs }
        copy (constructs, internalDTD)
        returninternalDTD
and algorithm
```

Fig. 19. Consolidate algorithm.

This final algorithm gives a result as a simple DTD, non-mixed and ready to be transformed into OWL. We apply this algorithm (fig. 20) on two resulting file of the preceding phase, we obtain the following DTD:

```
Simple DTD: library.xml
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
<!DOCTYPE library [
<!ENTITY %dtp  ", datePublication?">
<!ELEMENT datePublication (#PCDATA)>
<!ATTLIST author name CDATA #REQUIRED>
<!ELEMENT library (book)>
<!ELEMENT book (author %dtp;)>
<!ELEMENT auteur (#PCDATA)> ]>
```

Fig. 20. The DTD ready for transformation into OWL.

5. Conclusion

We have seen in this article how to transform the elements of the DTD to the OWL elements. The transformation of the basic elements already provided by the DTD2OWL method. We made an extension of this approach to complete the transformation of other elements, such as: typing attributes (ID, IDREF and enumeration), the choice and sequence of elements. We also provide a method named by the reorganization of the mixed DTD. It aims to retrieve the fragment of an external DTD, clean both DTD (internal, external) and fuse the internal DTD with the external DTD in order to give a single DTD file ready for the transformation into OWL. With DTD2OWL[2] approach, any DTD file can be completely transformed into OWL file. Such that the most methods focus on generating ontologies from XSD schema, we have opted in this paper an approach using the DTD whose interest is to use XML documents already validated by the DTD schema for ultimate integration.

References

1. Rodrigues T, Rosa P, Cardoso J. Mapping XML to Existing OWL ontologies. *Proc. of the International Conference on WWW/Internet*; 2006; 72-77.
2. O'Connor M J, Das A K. Acquiring OWL Ontologies from XML Documents. *Proc. of the 6th International Conference on Knowledge Capture (K-CAP)*; 2011; 17-24.
3. Ferdinand M, Zirpins C, Trastour D., Lifting XML Schema to OWL. *Proc. of the 4th International Conference on Web Engineering (ICWE)*; 2004; 354-358.
4. Bohring H, Auer S. Mapping XML to OWL Ontologies. *Leipziger Informatik-Tage*; 2005; 72:147-156.
5. Tsinaraki C, Christodoulakis S. Interoperability of XML Schema Applications with OWL Domain, Knowledge and Semantic Web Tools. *On the Move to Meaningful Internet Systems, Lecture Notes in Computer Science*; 2007; 4803: 850-869.
6. Cruz C, Nicolle N. Ontology Enrichment and Automatic Population From XML Data. *Proc. of the 4th International VLDB Workshop on Ontology-based Techniques for DataBases in Information Systems and Knowledge Systems, ODBIS*; 2008; 17-20.
7. Ghawi R, Cullot N. Building Ontologies from XML Data Sources. *Proc. of the 1st International Workshop on Modelling and Visualization of XML and Semantic Web Data (MoViX '09), held in conjunction with DEXA*; 2009; 480-484.
8. Bedini I, Matheus C, Peter F. and Nguyen B. Transforming XML Schema to OWL Using Patterns. *Proc. of the 5th IEEE International Conference on Semantic Computing*; 2011; 102-109.
9. Lacoste D, Sawant K P, Roy S. An efficient XML to OWL converter. *Proc. of the 4th International Conference on Software Engineering*; 2011; 145-154.
10. Yahia N, Mokhtar S A, Ahmed A. Automatic Generation of OWL Ontology from XML Data Source. *International Journal of Computer Science Issues*; 2012; 9:1694-0814.
11. Pham T T T, Lee Y. and Lee S. DTD2OWL: Automatic Transforming XML Documents into OWL Ontology. *Proc. of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*; 2009; 125-131.
12. Fallside D, Walmsley P. XML Schema Part 0: Primer. *W3C Recommendation 2001*. http://www.w3.org/TR/xmlschema-0/.
13. Smith M, Welty C, McGuinness D. OWL Web Ontology Language Guide. *W3C Recommendation 2004.* http://www.w3.org/TR/owl-guide/.
14. Mike Dean. Guus Schreiber. OWL Web Ontology LanguageReference. *W3C Recommendation 200.* http://www.w3.org/TR/owl-ref/#intersectionOf-def.