Journal of Information Science
2015, Vol. 41(2) 242–259
© The Author(s) 2015
Reprints and permissions:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/0165551514565972
jis.sagepub.com

Transforming XML documents to OWL ontologies: A survey

Mokhtaria Hacherouf
Université de Science et de Technologie d'Oran, Faculté des mathématiques et informatique. Laboratoire LSSD, BP 1505, EL-M'Naouer, 31000 Oran, Algeria

Safia Nait Bahloul
Université d'Oran1, Ahmed Ben Bella, Laboratoire LITIO, BP 1524, El-M'Naouer, 31000 Oran, Algeria

Christophe Cruz
Université de Bourgogne, Laboratoire LE2I, UMR CNRS 6306, 9 Avenue Alain Savary, 21078 Dijon CEDEX, France

Abstract
The aims of XML data conversion to ontologies are the indexing, integration and enrichment of existing ontologies with knowledge acquired from these sources. The contribution of this paper consists in providing a classification of the approaches used for the conversion of XML documents into OWL ontologies. This classification underlines the usage profile of each conversion method, providing a clear description of the advantages and drawbacks belonging to each method. Hence, this paper focuses on two main processes, which are ontology enrichment and ontology population using XML data. Ontology enrichment is related to the schema of the ontology (TBox), and ontology population is related to an individual (Abox). In addition, the ontologies described in these methods are based on formal languages of the Semantic Web such as OWL (Ontology Web Language) or RDF (Resource Description Framework). These languages are formal because the semantics are formally defined and take advantage of the Description Logics. In contrast, XML data sources are without formal semantics. The XML language is used to store, export and share data between processes able to process the specific data structure. However, even if the semantics is not explicitly expressed, data structure contains the universe of discourse by using a qualified vocabulary regarding a consensual agreement. In order to formalize this semantics, the OWL language provides rich logical constraints. Therefore, these logical constraints are evolved in the transformation of XML documents into OWL documents, allowing the enrichment and the population of the target ontology. To design such a transformation, the current research field establishes connections between OWL constructs (classes, predicates, simple or complex data types, etc.) and XML constructs (elements, attributes, element lists, etc.). Two different approaches for the transformation process are exposed. The instance approaches are based on XML documents without any schema associated. The validation approaches are based on the XML schema and document validated by the associated schema. The second approaches benefit from the schema definition to provide automated transformations with logic constraints. Both approaches are discussed in the text.

Keywords
Description logics; DTD; enrichment; XML; ontology; OWL; populating; XML schema

1. Introduction

The main purpose of ontologies in computer science is to provide formal models for the representation of knowledge with logical constraints. As a consequence, the knowledge represented formally is exploitable by both machines and users. Business knowledge is modelled using domain ontologies formalized with the help of controlled vocabularies describing a domain of application or a universe of discourse. In addition, the universe of discourse is related to an

Corresponding author:
Mokhtaria Hacherouf, Université de Science et de Technologie d'Oran, Faculté des mathématiques et informatique, Laboratoire LSSD, BP 1505, EL-M'Naouer, 31000 Oran, Algeria.
Email: mokhtaria.hacherouf@univ-usto.dz

organization or a community that benefits from the explicit specification of these controlled vocabularies. Three major methods are described in the literature to create domain ontologies. First, these methods consist in constructing ontologies *ex nihilo*. Second, they exploit available data that contain expert knowledge. The latter is a hybrid method of the previous ones to take advantage of both. Regarding the second method, the reuse process mainly computes many existing heterogeneous data sources. These data sources are relational databases, XML documents, tabular documents, textual documents, etc. Furthermore, the reuse process runs through two main phases, eventually the enrichment phase and the population phases [1]. The enriching phase consists of adding axioms to the target ontology. Structured and unstructured data sources provide these axioms to the enrichment phase. In order to ease the process, schemes of structured or semi-structured data should preferentially be selected. Since the axioms such as concepts, relationships, attributes and data types are formalized in the target ontology, the populating phase provides individuals for the previously defined concepts, relationships, attributes and data types.

This paper focuses mainly on the second method, which deals with XML data for the ontology enriching and populating phases. The ontology languages mainly used are the OWL language (Ontology Web Language) and the RDF language (Resource Description Framework) from the Semantic Web (World Wide Web Consortium; www.w3c.org). These languages are based on Description logics, providing the foundation of formal semantics. In contrast, XML data sources do not provide semantics. They are used to store, export and share data between processes able to process the specific data structure [2]. However, even if the semantics is not explicitly expressed, data structure contains the universe of discourse by using qualified and controlled vocabulary with regards to a consensual agreement. In order to formalize this semantics, the OWL language provides rich logical constraints, which are evolved in the transformation of an XML document into OWL ontologies. The result of this transformation makes it possible to index the XML data used for its integration and retrieval. To design such a transformation, the current research field establishes connections between OWL constructs (classes, predicates, simple or complex data types, etc.) and XML constructs (elements, attributes, element lists, etc.). The goal of this paper consists in presenting the different approaches and methods for the transformation of XML documents into OWL ontologies by classifying them, comparing them with a metric and underling the quality of these transformation processes. Two different approaches for the transformation process are exposed in the following sections. The instance approaches are based on XML documents without any schema associated. The validation approaches are based on the XML schema and documents validated by the associated schema. The second approach benefits from the schema definition to provide automated transformations with logic constraints. Both approaches are discussed below.

This document is organized as follows. Section 2 defines the various technologies discussed in the rest of the paper. Section 3 exploits different approaches to the transformation of XML data to OWL ontologies by grouping them into three main classes previously discussed. Section 4 provides a comparison of these classes. Section 5 concludes this paper.

2. Definitions

This section introduces languages for data structure that are discussed later in this document. It starts with the Semantic Web languages, including the XML language. It also provides details about the correspondence rules or mapping rules between XML data and OWL ontologies.

2.1. Key XML technologies

XML (Extensible Markup Language) is a simple text format derived from SGML (ISO 8879). It was designed to flexibly structure information using markup. XML is a suitable language for exchanging a wide variety of data on the Web [3]. An XML document is valid if it respects the grammar defined in a schema. The purpose of a schema is to define a class of XML documents. The term 'document instance' means that an XML document conforms to a particular schema. At the first age of the XML language, many schema formats were available with specific properties. Today, two major schema formats are used: the Document Type Definition (DTD) and the W3C XML Schema (XSD). A DTD [4] is a grammar that aims at describing the elements and attributes accepted in an XML document. It allows the declaration of new tags and the specification of constraints on them. Unfortunately, DTDs are not XML format. That means that they do not support 'namespaces' and provide very limited data typing. Hence, XSD is designed to provide more advanced features. XML schemes permit description of the structure of an XML document in a much more comprehensive way. XML schemes take advantage of the XML format while managing a variety of types of simple or complex data. In addition, XSL (eXtensible Stylesheet Language) is a specific language for defining style sheets associated with an XML document. An XSL style sheet is a file that describes how to transform a specific kind of XML document to another format. XSL includes three languages, XSLT, XPath and XSL-FO, but only the first two languages fit our requirements. Consequently, we quote only the first two languages. XSLT (eXtensible Stylesheet Language Transformation) is used to transform XML documents using stylesheets containing rules called 'template rules'. XSLT uses XPath for designating a part of an XML tree [5]. XPath is a non-XML language used to address nodes in an XML document. Thus, XPath is a query language commonly used in XSLT and other languages to specify paths in XML documents [6].

2.2. RDF and OWL

The RDF (Resource Description Framework) and OWL languages (Web Ontology Language) are XML-based Semantic Web languages. These languages include a strong semantic definition which puts these languages at a higher level regarding usage of the XML language. XML was created to structure, store and exchange data between processes. The Semantic Web Group incorporated in RDF and OWL semantics that do not appear in most applications. This semantics is a vector of interoperability [2].

The RDF language is a graph-based model that aims to describe Web resources formally and their metadata, such as the title, author and publication date of a web page. It is considered as a basic language for the Semantic Web. In addition, the RDF language can represent information about identifiable resources on the Web, even if these resources are not directly retrievable [7]. An RDF triple encodes a statement that is a simple logical expression or assertion about the world.

The OWL language [8] is used to specify ontologies that are intended for publication and sharing on the Web with a higher level of logical expressivity with regards to the RDF language. The second version of the language OWL is available now. OWL 1.0 consists of three sublanguages (OWL-Lite, OWL-DL and OWL-Full) of increasing expression. Each one is employed for specific users and requirements. In addition, each language is an extension from its simpler predecessor regarding the semantic richness. The OWL-Lite language is the simplest; being less expressive, it meets the requirements for a classification hierarchy and functionality constraints for relationships. The OWL-DL language has semantic expressivity of the Description Logics. It is characterized by the completeness of the calculation and the decidability of the reasoning system. OWL-Full is characterized by the maximum expressiveness. It cannot guarantee the completeness and decidability of calculations, which is the reason why it was not adopted by the Semantic Web community. The OWL 2.0 language is actually the OWL-DL 2 language [9]. This language is characterized by a certain expressive power of efficiency of reasoning. It increases the expressiveness with the guarantee of computability. It covers three profiles: OWL 2 – EL, OWL 2 – QL and OWL 2 – RL. Each of these profiles is intended to structure ontology and for specific reasoning tasks. OWL 2 – EL (Existential Language) is required for large ontologies of complex structural descriptions. It corresponds to the family of Description Logics. The verification algorithms of satisfiability (all the concepts admit individuals), classification (deduction of the concept hierarchy) and realization (find the most specific subsumant for an individual) are polynomial. OWL 2 – QL (Query Language) is intended for simple ontologies with a large number of entities. It can be easily integrated with relational databases, since it has a similar power to that entity–relationship schema and reasoning using rewriting SQL queries. OWL 2 – RL (Rule Language) is the most expressive profile. It requires some limitations on the expressiveness in order to keep some effectiveness. The reasoning is carried out using rule systems and inference engines.

2.3. Transformation process

The correspondence or matching rules are involved throughout the transformation process of XML data to OWL ontologies. They achieve three main objectives, which are the generation, enrichment and population of an OWL ontology. The ontology-enriching process from an XML document adds new constructors (classes, object attributes or data types, etc.) to the schema of an existing ontology. In the case of a nonexistent ontology, the ontology is generated directly from the XML documents using predefined rules. The process is named the ontology generation process. The ontology population process adds individuals or attributes to available individuals from an XML data to the ontology. Ontology generation and enrichment can be processed using XML instances or validation schemes (DTD or XSD-Schema). Consequently, two transformation approaches are distinguished to process the generation and the enrichment of ontologies, namely the instances approach and the validation approach. Regarding the population correspondence rules, they mainly require XML document instances. Figure 1 shows the different strategies for transforming XML to OWL used by distinct approaches. Two levels are described. The lower level is the instance level, and the upper level is the schema level. On the left, the figure shows the different kinds of XML data, and on the right, it shows the impact on the ontologies; the generation of the ontology, the enriched target ontology and the creation of instances in the target ontology. The arrows symbolize the different processes using correspondence rules, and the two different approaches, the validation and the instance approach. It has to be noted that only XML instances are used for correspondence rules.

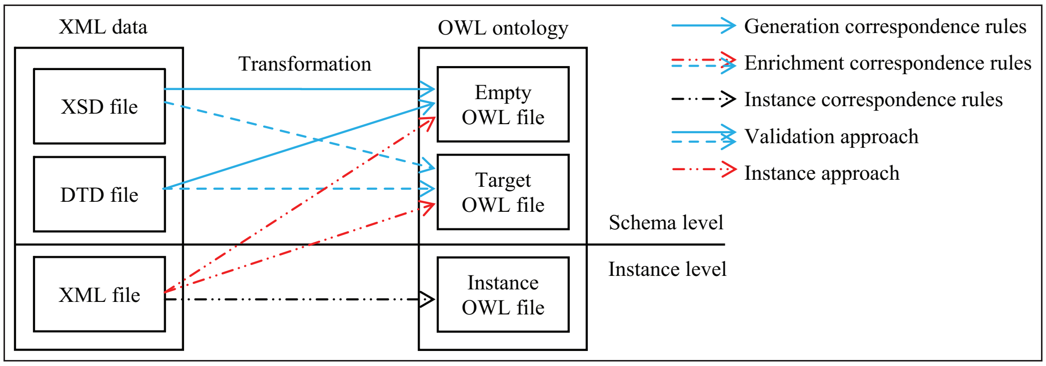

Figure 1. Different strategies and rules to transform XML data to OWL ontologies.

Table 1. Predefined notations by JXML2OWL.

Mappings	Notations
Class	(OWL Class URI, expression XPath)
	(OWL Class URI, expression XPath, expression XPath ID)
Data type Property	(OWL Data type Property URI, Domain Class Mapping, Expression XPath)
Object Property	(OWL Object Property URI, Domain Class Mapping, Range Class Mapping)

3. Different approaches to transformation

This section presents the different approaches for transforming XML documents into OWL ontologies by grouping them into two classes according to the scheme from which the ontology is generated. A comparative study is made between the approaches of each class and between classes themselves. The approaches of each class are set in order of the year of publication and are named according to the tool/prototype name, or the authors' name.

3.1. The instance approach

In this section, we present two methods for transforming XML documents to an OWL ontology. The first method [10] generates an ontology semi-automatically, while the second [11] permits the automatic creation of an ontology or the enrichment of an existing ontology with the new content mapped.

3.1.1. JXML2OWL. This approach [10] aims at reducing the gap between XML data and OWL ontologies without resorting to an XSD schema. Consequently, this method utilizes XPath expressions to help distinguish multiple XML nodes with the same name but with different tags. Thus, it is possible to define correspondence rules on these tags. To specify the correspondence rules between the elements of an XML document and resources defined by the created OWL ontology (classes, predicates, data types), the authors propose the notation shown in Table 1.

Figure 2 represents a fragment of an XML instance, which is used to show how to transform XML tags to OWL constructs utilizing the JXML2OWL approach.

Using the notation of Table 1, the following mappings are valid:

- cm1 = (reading:book, /library/book);
- cm2 = (reading:author, /library/book/author);
- cm3 = (reading:article, /library/article);
- op1 = (reading:hasAuthor, cm1, cm2);
- dp1 = (reading:article_laboratory, cm3, /library/article/author/@laboratory).

```
<library>
  <book>
    <author>
      <name> Meriem </name>
    </author >
  </book>
  <article>
    <author laboratory="Information system">
      <name> Sarah </name>
    </author>
  </article>
</library>
```

Figure 2. Fragment of an XML document instance.

```
(a)  Class: @/library/book[1]/author/name/text()SubClassOf: author

(b)  Class: @/library/book/name(mm:trim)SubClassOf: book
```

Figure 3. Inspired expressions of the XMLMaster language [11].

'cm1/2/3' corresponds to the definition of a class, 'op1' refers to a relationship between objects of type 'Object Properties' and 'dp1' represents an attribute of type 'DataTypeProperty' relationship. Once the OWL ontology is generated, the second step consists of populating this ontology from XML instances. A unique identifier is generated for each individual. The population process detects and filters redundant instances that have the same ID.

3.1.2. XMLMaster. Regarding this approach, the authors have developed a domain-specific language named XMLMaster [11] that supports XML documents that are not necessarily linked to an XML schema for validation. This language handles texts as unstructured data, that is, text processing elements or attributes values. The XMLMaster is constructed by combining a part of the Manchester OWL syntax, a language widely used to describe declaratively the OWL ontology, and secondly the XPath query language to describe XML content. Figure 3 from O'Connor and Das [11] shows some expressions of XMLMaster language. The fragment of the XML document of Figure 2 is used to define the properties of the OWL ontology. Expression (a) declares an OWL class named by the element's content of the first book. Expression (b) is another class declaration nominated by the element name 'book' using the 'mm:trim' function to remove spaces before and after.

Regarding the implementation of this mapping language, the authors have developed an analyser, a publisher, a processor and a debugger as an open source plug-in for the development environment Protégé (http://protege.stanford.edu/).

3.1.3. Discussion. Both approaches make use of the XPath language to refer to contents of XML documents. However, the techniques are different. JXML2OWL makes use of XPath expressions to generate a set of predefined correspondence rules. This pinpoints the expressiveness limits of the resulting ontologies. Even the XMLMaster language gives a richer possibility of expression. Nevertheless, only OWL classes can be named by an XML element or its contents. In addition, XMLMaster is characterized by its ability to process texts using functions.

In the approach of JXML2OWL, just the enriching phase is made, and only from XML elements (e.g. article, author and laboratory). In the approach of XMLMaster, the enriching phase can be made from elements or content (e.g. article, author, name, laboratory and information system). The populating phase is done at the same time as the enriching phase (e.g. Sara). Figure 4 illustrates the two cases. The XMLMaster approach is rather employed for applications that require the creation of an ontology *ex nihilo,* was the JXML2OWL approach serves schema integration application. Table 2 summarizes the points discussed for both approaches.

3.2. The validation approach

Most approaches that will be listed in this section are intended to generate an ontology from an XSD or a DTD schema. The only exception is the XSD2OWL approach [1], which provides an enrichment of an existing ontology from XSD

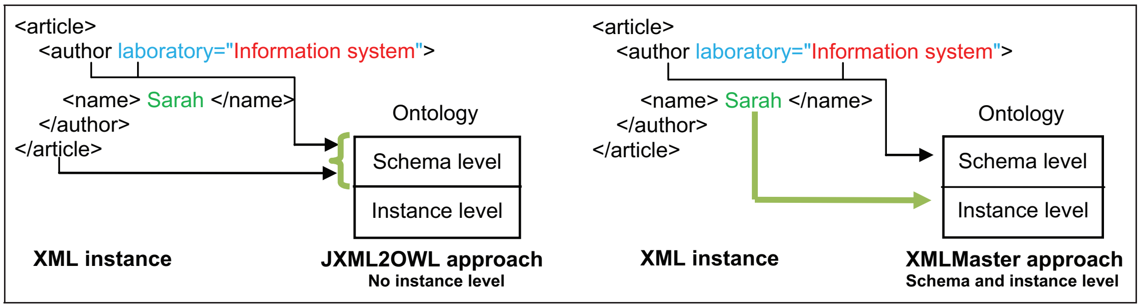

Figure 4. The enriching process of approaches JXML2OWL and XMLMaster.

Table 2. Comparison of XML instance approaches.

Approach	Expressivity	Technology used	Ontology existence
JXML2OWL [10]	Limited	XPath	Existing
XMLMaster [11]	Higher	XPath + OWL Manchester	Existing or no

Table 3. Mapping rules defined in the method OWLMAP.

XSD	OWL
ComplexeType/model group/attribute group	owl:Class
SimpleType/Attribute	owl:DataTypeProperty
Element of complex type	owl:ObjectProperty, rdfs:subClassOf
SubstitutionGroup	owl:subClassOf
Element Type (or attribute) local	owl:allValueFrom
minOccurs/maxOccurs	owl:minCardinality/maxCardinality/cardinality
xsd:sequence, xsd:all	owl:intersectionOf
xsd:choice	Boolean Expression with owl:intersectionOf, owl:unionOf and owl:complementOf

Table 4. Transformation rules of XML instances to RDF graphs.

XML component	RDF component
Composed element	Ressource
Subelement/attribute	RDF property
Simple element	RDF literal

schema without an automatic generation process of an ontology schema. The transformation process proposed by the XML2OWL [12], XS2OWL [13], X2OWL [14], Janus [15], EXCO [16] and Yahia et al. [17] approaches are fully automated. Approaches [1] and [18] conduct the processing semi-automatically. Approaches [1, 13, 18 and 19] realize the populating phase, while approaches [13–15] do not.

3.2.1. OWLMAP. This OWLMAP approach [18] proposes an automatic transformation of XSD schema constructors to OWL constructors, and provides XML instances in the RDF graph. The first transformation is performed from a predefined set of mapping rules. We summarize these rules in Table 3. XML instances are converted to RDF graphs to meet the application requirements that make use of inferences and semantic validations of XML data. Once translated into RDF, the data can be classified (e.g. *find the most specific classes of an individual*) by an OWL reasoner engine like Pellet. In Table 4, we summarize the representation of XML components (with RDF components such as element, attribute, property, etc.). The script in Figure 5 is the corresponding RDF graph of the XML fragment of Section 3.1.1 based on Table 4.

```
@prefix ex: <http://www.example.org/>.
        ex:library ex:book ex:author.
        ex:author ex:name ex:Meriem.
        ex:library ex:article ex:author.
        ex:author ex:name ex:Sarah.
        ex:author ex:laboratory ex: "Information system".
```

Figure 5. The RDF graph corresponding to the XML instances in Section 3.1.1.

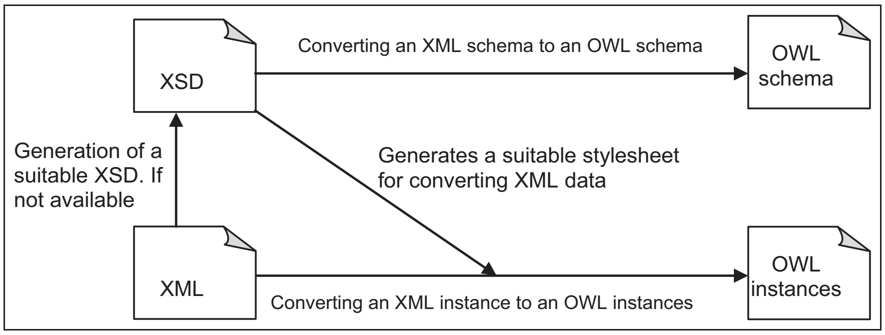
XSD
Converting an XML schema to an OWL schema
OWL schema
Generation of a suitable XSD. If not available
Generates a suitable stylesheet for converting XML data
XML
Converting an XML instance to an OWL instances
OWL instances

Figure 6. The sequence of operations for the application XML2OWL [12].

3.2.2. XML2OWL. Bohring and Auer [12] present an OWL ontology creation process, including individuals from XML documents (Figure 6). These XML documents were generated from a relational database. If the XML document has no XSD schema, then a step of generating XSD schema is required. In addition, the OWL schema is created from the XSD schema. The generation of OWL individuals makes use of the XSD schema, and the corresponding XML document, as the following schema shows. The matching rules proposed in this project are the same as those defined by the approach of Ferdinand et al. [18]. The transformation of XML instances in OWL individuals is made with the help of an XSL stylesheet. This stylesheet is created from the XSD schema at the same time as generating OWL ontology. The XSL file will then be used with the corresponding XML document to generate the OWL individuals.

3.2.3. XS2OWL. With regards to other approaches, the XS2OWL tool [13] transforms only XSD schemes in OWL-DL ontology schemes without the populating phase. However, the transformation model has been implemented as an XSLT stylesheet like XML2OWL [12]. The system XS2OWL takes as input an XSD schema and automatically generates a model comprising the three following modules:

(1) An upper ontology contains essentially the OWL constructs. The matching rules are almost the same as those proposed by XML2OWL [12] and OWLMAP [18].
(2) Simple data types defined in the XML schema are used to specify the upper-level ontology.
(3) Mapping ontology allows transformation of XML schema elements, which cannot be generated directly into the upper ontology, for example, the XML schema elements bearing the same name (see Table 5).

The XS2OWL approach aims to support the interoperability between OWL and XML environments. The XS2OWL framework has been integrated with the ‘SPARQL2XQuery framework’ [20], allowing the translation of SPARQL queries in XQuery.

3.2.4. XSD2OWL. This XSD2OWL approach [1] handles the enriching and populating phases (Figure 7). The enriching process is done manually by users establishing mapping rules. This manual process ensures equivalence between the XML schema elements and the concepts of the target ontology. Once this phase is completed, the ontology is populated automatically using XML documents validated by the mapped XML schema. The set of rules defined for the enriching process allows the automation of this phase. These rules are defined using the RDF language.

The rules used by the converter can be divided into two categories:

Table 5. Classes constitute the ontology mapping defined by XS2OWL.

The classes of mapping ontology	Explanation
ComplexTypeInfoType	Mapping complex types of XML schema that cannot be directly represented by OWL constructs
ElementInfoType	Mapping of the XML schema elements that cannot be represented directly by OWL, for example the order of sequence elements
DatatypePropertyInfoType	Specification of the default value as possible and type of XML schema elements represented by properties type
SequenceInfoTyp/ChoiceInfoType	Mapping sequences and choices that cannot be directly represented by OWL constructs
ComplexTypeInfoType	Mapping complex types of XML Schema that cannot be directly represented by OWL constructs

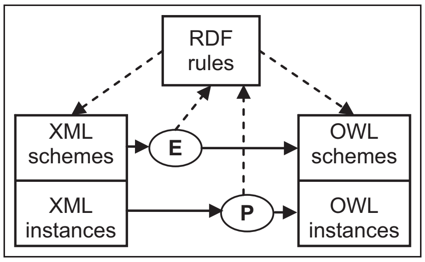

Figure 7. The enriching and populating processes [1].

(1) *Visual links* – a visual connection is the relationship between two elements belonging to the same language. There are two kinds of links:
   (1.1) 'Inter-xsd' – a hierarchical relationship between two XSD elements as SubClassOf.
   (1.2) 'Inter-owl' – a three-layer hierarchical relationship, OWL: owl: Range, owl: Domain and owl: SubClassOf.
(2) *Advanced mapping rules* – these rules are links between elements of XSD schema and ontology elements. There are two types:
   (2.1) 'is-a' link – the OWL class pointed by this link is defined as equivalent to the XSD element pointed.
   (2.2) Conversion link – the simple or complex rules of conversion realize bridges between XML data and data contained in the ontology.

This method is the only method to use RDF to model the matching rules. These rules are defined using a graphical interface. A Protégé plug-in is dedicated to this purpose (Figure 8). The authors stated that, even if the automation of the transformation is powerful and helpful in certain circumstances, only the experts are able to qualify at the end the relevance of the transformation rules. This assertion explains the creation of a graphical tool to ease the development and the creation of ontologies from several XSD schemes. Thus, the populating process of the target ontology associated with XML instance documents is fully allowing automaticity. A semi-automatic enriching process should be of benefit for users when XSD schemes are large and numerous.

3.2.5. X2OWL. Ghawi and Cullot [14] proposed a tool called X2OWL with a distributed system approach. This work focuses on information integration where local ontologies are used to describe the semantics of local information sources. The method only generates an OWL schema from XSD schema. However, XML instances are retrieved and converted in response to user requests. The X2OWL approach uses the same notations as those used in JXML2OWL [10] for mapping XML to OWL. In this approach [10], the notations are generated from XML instances. In the X2OWL approach, the notations are generated from the XML schema graph (XSG) with the same notation. An XSG is a directed acyclic graph where nodes represent an element, attribute, a group of elements and a group of attributes. Edges represent the links between: (1) each element and its type; and (2) each group of elements or attributes and their associated attributes or elements. An XSG tree is a graph where the elements and type declarations are not re-used. The X2OWL method is based

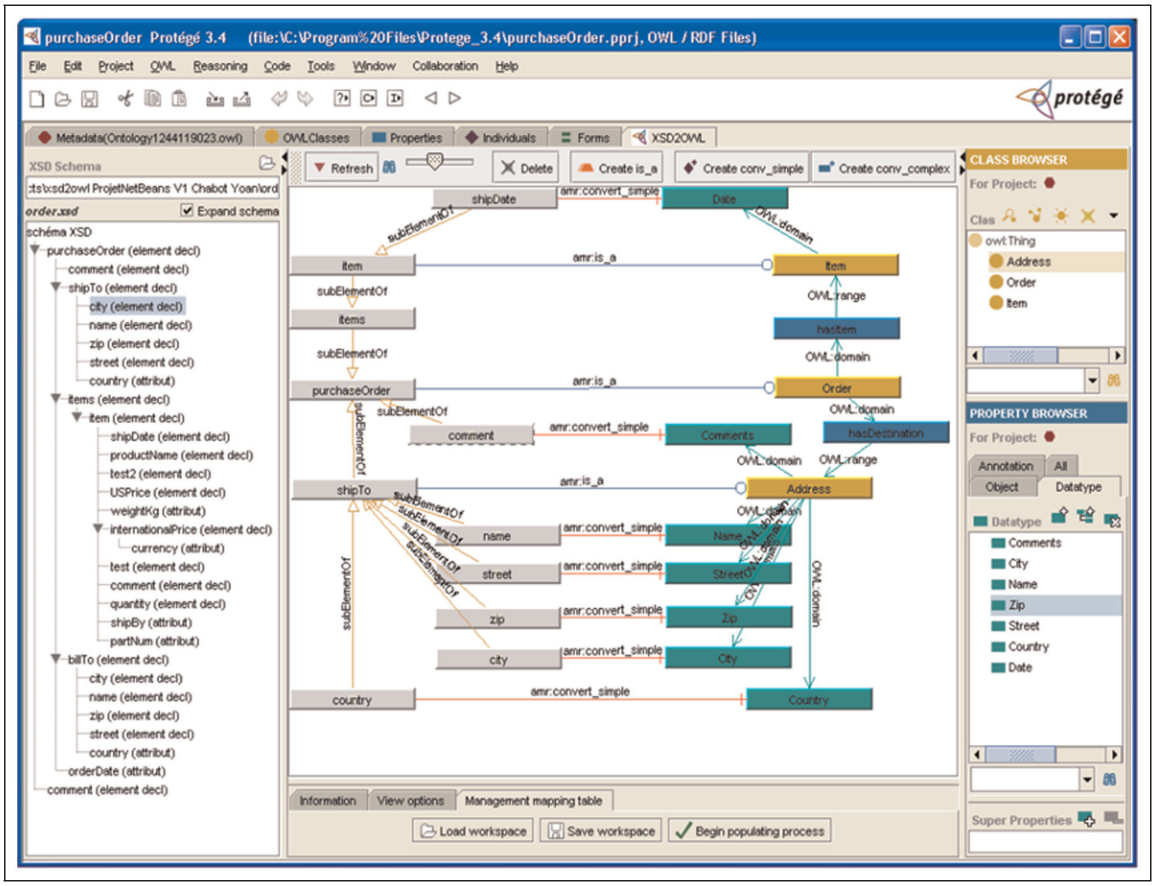

Figure 8. Snapshot of the Protégé XSD2OWL plug-in. On the left, the XML Schema graph is represented and on the right the class and property browser of Protégé is available. In the middle, the graphical representation of the correspondence rules with the associated links allows rules to be declared.

on this XSG graph and generates XPath expression for each node. If a node is visited more than once, then there may be several XPath expressions. The resulting expressions are then processed by a refinement process. This step allows the invalid correspondence rules to be cleaned and the generated ontology to be restructured. A matching rule is called invalid if it contains invalid Xpath expression or references another invalid rule. The X2OWL tool solves the double occurrence of an element and an attribute in the same XSD schema. On the example of Figure 10, we apply the concepts Cullot and Ghawi [14] to mapping the XML schema of Figure 9.

The rules 'dm2' in the example are invalid rules. They will be removed by the refinement process. In this example, the book author has no laboratory (see the XML instance of Section 3.1.1.).

3.2.6. DTD2OWL. Concerning the transformation from DTD, the DTD2OWL approach [19] automatically generates an OWL ontology from the DTD. Processing at the DTD tends to convert all elements and attributes of a DTD to a class or a predicate in the target ontology. The whole DTD2OWL framework combines the mapping of a DTD to an OWL ontology, and transforming the XML document into OWL instances. The the procedure was implemented using the XSL stylesheet.

The result of this transformation is an OWL ontology capturing the semantics of the DTD structure. In Table 6, we summarize the matching rules related to the DTD constructs. The rules generate OWL constructs that are shown on the left of Table 6. The XML language allows elements with the same name, which is not allowed in OWL. This problem is overcome by renaming the second element by concatenating the parent node + '_' + identical element. The DTD2OWL approach is simpler from the viewpoint of the appropriate OWL constructs with regard to XML schema elements. However, it is incomplete as it does not treat all of the constructs of the DTD definition. For example:

```
<xs:element name="library">
      <xs:complexType>
            <xs:sequence>
                  <xs:element ref="book"/>
                  <xs:element ref="article"/>
            </xs:sequence>
      </xs:complexType>
</xs:element>
<xs:element name="book">
      <xs:complexType>
            <xs:sequence>
                  <xs:element maxOccurs="unbounded" ref="author"/>
            </xs:sequence>
      </xs:complexType>
</xs:element>
<xs:element name="article">
      <xs:complexType>
            <xs:sequence>
                  <xs:element maxOccurs="unbounded" ref="author"/>
            </xs:sequence>
      </xs:complexType>
</xs:element>
<xs:element name="author">
      <xs:complexType>
            <xs:sequence>
                  <xs:element name="name" type="xs:string"/>
            </xs:sequence>
            <xs:attribute name="laboratoire" type="xs:string" use="optional">
      </xs:complexType>
</xs:element>
```

Figure 9. An XML schema example.

Class Mapping	**Object Property Mappings**
cm1 = (library, /library)	om1 = (hasBook, cm1, cm2)
cm2 = (book, /library/book)	om2 = (hasAuthor, cm2, cm3)
cm3 = (author, /library/book/author)	om4 = (hasArticle, cm1, cm4)
cm4 = (article, /library/article)	om5 = (hasAuthor, cm4, cm5)
cm5 = (author, /library/article/author)	

Datatype Property Mappings
dm1 = (name, cm3, /library/book/author/name/text())
dm2 = (laboratoitre, cm3, /library/book/author/laboratoire/text())
dm3 = (name, cm5, /library /article/author/name/text())
dm4 = (laboratoitre,cm5, /library/article/author/laboratoire/text())

Figure 10. Inspired resulting mapping rules [14].

- Attribute types: enumeration, ID, IDREF.
- Different entity types: general, parametric, external, internal.
- The hierarchical relationship between the declarations of elements: sequence, choice.
- Mixed DTD.

These constructions are the most commonly used in the DTDs. Consequently, it is necessary to extend this conversion to limit the loss of information.

3.2.7. Janus. Bedini et al. [15] developed a tool called Janus that automatically generates an OWL ontology from an XML schema. The proposed approach defines 40 correspondence rules to match most of the XSD constructs with the OWL2-RL constructs. For instance, the construction 'restriction', derived from restriction of a simple type, allows the creation of several simple types from simple predefined types by the XML schema language. Janus provides a

Table 6. Mapping rules defined by DTD2OWL.

DTD constructs	OWL constructs
< !ELEMENT elt x > x: element has an	< owl:class rdf:about="#elt" >
attribute or composed another element.	< owl:disjointWith > < – declaration of another 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 (#CDATA) >	< owl:DataProperty rdf="#elt2" >
	< rdf:domain rdf:ressource="#elt1"/ >
	< rdf:range rdf:ressource="#String"/ > < /owl:DataProperty >
< !ATTLIST elt att CDATA >	< owl:class rdf:about="#elt" >
	< rdfs:subClassOf >
	< owl:Restriction >
	< owl:OnProperty >
	< owl:DataTypeProperty rdf:ID="att" >
	< /owl:OnProperty >
	< /owl:Restriction >
	< /rdfs:subClassOf >
	< /owl:class >
#FIXED value	owl:haseValue
< !ENTITY entity-name "entity-value" >	
#REQUIRED	owl:minCardinality (=1)
#IMPLID	owl:cardinality (=0)
NOTATION	rdfs:Comment
+	owl:minCardinality (=1)
?	owl:minCardinality (=0)
*	owl:minCardinality (=0)
	owl:maxCardinality (=undounded)

transformation method for each type defined in a model, including derivations by restriction. The 'Turtle' syntax is used to express OWL entities. Among the proposed correspondence rules, Table 7 presents a subset that has not been considered in other projects. However, even if the Janus approach defines many correspondence rules, no populating process is available.

3.2.8. EXCO. The EXCO approach [16] aims at developing a fully automated method that can handle the imported schemes and manage internal references in the construction of an XSD schema. The authors propose a tool called EXCO (An Efficient XML to OWL COnverter). This tool handles the enriching and populating phases. The ontology creation from an XSD schema is based on the conversion table proposed by the XML2OWL approach [12]. In addition, the particularity of the EXCO tool remains its ability to handle internal and external references, and enumerations. EXCO proposed a method called 'consolidation schema'. The method is composed of three steps:

- *Collecting schema files* – this step collects all XSD schemes to be imported and included. These schemes are saved with the reference of their locations in the hash table. In addition, it saves all namespaces and references, avoids circular and multiple references.
- *Merging schema files* – this step merges all of the imported schemes by unify the namespace prefixes. The prefix associated with a different name then 'xsd' in the merged schema are replaced this one.
- *Reorganizing schema* – the process of this step consists of reorganizing the internal references. The referred elements are added to the current element. This is done recursively to the descendants. Finally, unnecessary elements will be eliminated. Figure 11 shows the reorganization of the element named 'book' from the previously presented XML schema, Figure 9.

The creation of individuals employs the same strategy proposed by XML2OWL [12]. It consists of making use of an XSL file. Finally, this is the unique approach that adds a process of verifying the conversion accuracy.

Table 7. A Janus subset of correspondences rules.

XML schema construct	OWL2-RL construct (Turtle syntax)
< xsd:simpleType name="st_name" >	:st_name owl:equivalentClass
< xsd:restriction	[rdf:type rdfs:Datatype;
base="xsd:nativDataType" >	owl:oneOf
< xsd:enumeration value="value1" >	("value1"^^xsd:nativDataType ...)] .
…	
< xsd:complexType name="ct_name" >	
< xsd:simpleContent >	:ct_name rdf:type owl:Class .
< xsd:extension	:has_ct_name rdf:type owl:DatatypeProperty ;
base="xsd:nativeDataType" >	rdfs:domain :ct_name ;
…	rdfs:range xsd:nativeDataType .
< xsd:complexType name="ct_name" >	
< xsd:all >	:has_elt_name1 rdf:type owl:FunctionalProperty ;
< xsd:element name="elt_name1"	rdfs:domain :ct_name ;
type="elt_type1"/ >	rdfs:range [rdf:type owl:Restriction ;
…	owl:onProperty :has_elt_name1 ;
< /xsd:all >	owl:onClass :elt_type1 ;
< /xsd:complexType >	owl:maxQualifiedCardinality
	"1"^^xsd:nonNegativeInteger] .
	…

```
<xsd:element name="book">
        <xsd:complexType>
                <xsd:sequence maxOccurs="unbounded">
                        <xsd:element name="author">
                                <xsd:complexType>
                                        <xsd:sequence>
                                                <xsd:element name="name" type="xsd:string"/>
                                        </xsd:sequence>
                                                <xsd:attribute name="laboratory" type="xsd:string"
                                                use="optional">
                                </xsd:complexType>
                        </xsd:element>
                </xsd:sequence>
        </xsd:complexType>
</xsd:element>
```

Figure 11. The XSD schema reorganized by the EXCO approach from Figure 9.

3.2.9. Yahia et al. The approach of Yahia et al. [17] was published in 2012. This work is more recent compared with other propositions and does not provide a significant contribution. The authors propose an automatic process to generate an OWL ontology from different XML data sources. The method uses the same notation as used in the X2OWL approach [14] with some modifications to handle multiple XML data sources. It is based on XML schemes or on generated schemes if not available. The generating and analysing process uses the XML Schema Object Model (XSOM https://xsom.java.net/) and the Java Universal Network/Graph Framework (JUNG http://jung.sourceforge.net) to handle the XSG graphs.

3.2.10. Discussion. All of these approaches propose to find out constructors (class, ObjectProperty, DataProperty, etc.) that best match the constructors of schema validation (xs:element, xs:attribute, xs:complexType, etc.). They are distinguished by the points summarized in Table 8. Each of the following bullet points discusses one of the columns of this table.

- *Input* – the inputs comprise two kinds, XML schemes or DTDs, and eventually include XML instances. Approaches that make use only of XML schemes serve indexing applications. The XML instances are taken into

Table 8. Comparison of approaches based on the XSD schema.

Approaches	Inputs	Outputs	OWL type	Generation of ontology and individuals	Rules	Integrity constraints	Existing ontology
OWLMAP [18]	XML schema + XML instances	OWL schema + RDF graph	OWL-DL	Sequential	Automatic	Yes	No
XML2OWL [12]	XML schema + XML instances	OWL schema + individual	OWL-DL	Sequential	Automatic	Yes	No
XS2OWL [13]	XML schema	OWL schema	OWL-DL	n/a	Automatic	Yes	No
XSD2OWL [1]	XML schema + XML instances	OWL ontology + individual	OWL-DL	sequential	Semi-automatic	No	Yes
X2OWL [14]	XML schema	OWL schema	OWL-DL	n/a	Automatic	No	No
Janus [15]	XML schema	OWL schema	OWL2-RL	n/a	Automatic	Yes	No
EXCO [16]	XML schema + XML instances	OWL schema + individual	OWL-DL	parallel	Automatic	Yes	No
Yahia et al. [17]	XML schema	OWL schema + individual	OWL-DL	n/a	Automatic	No	No
DTD2OWL [19]	DTD + XML instances	OWL schema + individual	OWL-DL	Sequential	Automatic	Yes	No

account to create the ontology individuals for approaches used in broader contexts such as indexing, generating or enriching ontologies, and creating adapters for warehouse data [21]. Approaches proposing an ontology generation process are subsidiaries. Tools are available for transforming DTDs to XML schemes.

- *Output* – all transformation processes provide an OWL schema as output. However, a better process consists of enriching this schema with individuals to provide a complete description of the XML document. The OWLMAP approach [18] has addressed the phase of the instance transformation using an RDF graph. This graph does not respect the ontology generated from XML schema.
- *OWL type* – all approaches use OWL DL as a result of the generation and enriching process. Janus [15] is an exception because it uses OWL-RL. On the one hand, the OWL-DL ontologies are more expressive than the OWL-RL ontologies. On the other hand, the OWL-RL meets the requirements of reasoning applications.
- *Generation* – the enriching and populating phase can be sequential or parallel. The EXCO approach [16] is the one that is not sequential. This execution time optimization is simply ignored by other approaches for large data sources.
- *Rules* – the generation of correspondence rules can be automatic or semi-automatic. The XSD2OWL approach [1] is the unique approach focusing on a semi-automatic transformation, that is, the user graphically performs the enriching process. This transformation provides an ontology that is more accurate and reliable, since it is controlled by a human. However, it is preferable to offer a semi-automatic process by providing a proposition for large data sources.
- *Integrity constraints* – the integrity constraints are, for instance, maxOccurs, All, Choice, etc. These constraints avoid the loss of information from XML data and allow the generation of a richer ontology by including logic constraints.
- *Ontology's existence* – some approaches enrich an existing ontology; others generate ontology from nothing. If an application requires a gradual integration of XML documents, then the enriching approach category allows that process. The XSD2OWL approach [1] is the unique approach focusing that process by providing a manual definition of the correspondence rules. The second category first generates the ontology that corresponds to the XML documents and the integrity. This category serves well the applications of indexing XML documents.

Validation approaches using DTDs or XML schemes provide richest results and a larger construct set. However, the selection of certain constructs for the transformation process may depend on the application domain. For instance, the Janus approach selects XML schema constructs based on a statistical analysis of the B2B (Business to Business) domain. This analysis shows practically which are the constructs most used in this domain. Table 9 shows the validation schema elements processed by each approach. The following diagram shows the number of schema validation elements automatically transformed into OWL for each approach in order of the publication year. An average of eight constructs is used in the different approaches, which are the most commonly employed.

Each automatic approach tries to achieve one of the two following tools. The first is a tool to transform all the constructs of a validation schema. However, neither approach has been realized full yet. Therefore, this tool will be more complex for processing a single document. The second is a tool to transform the validation schema elements related to an application domain (e.g. Janus). This transformation tool is not universal and is constrained by its application domain. The XSD2OWL [1] approach allows users to establish itself correspondence rules between XML schema elements with OWL constructs. Consequently, this approach cannot be compared with the other according to the number of construction schema validations transformed (Figure 12).

3.3. General discussion

We compared and analysed the approaches presented in this paper according to the two families they belongto: the family of instance approaches and family of validation approaches. Mainly our analysis summarizes the comparison points in Table 10. This section aims at underling the different possible criteria for quality of the transformation process.

3.3.1. Richness of the transformation results. The ontology generated from validation schemes is more structured than that generated from the XML instances. It contains, for instance, restrictions on the properties (Cardinality, maxCardinality, maxCardinality) and operators of class combinations (unionOf, complementOf, intersectionOf). In addition, the ontology generated from the XML instances contains only classes, data properties and object properties. Consequently, this ontology is not adapted for reasoning applications.

Table 9. The construction schema validation transformed by each approach.

Construction of schema validation		OWLMAP	XML2OWL	XS2OWL	X2OWL	Janus	EXCO	Yahia et al.	DTD2OWL
XML schema	DTD								
Element	< !ELEMENT x …	×	×	×	×	×	×	×	×
Attribute	< !ATTLIST x att …	×	×	×	×	×	×	×	×
AttributeGroup		×			×	×		×	
SubstitutionGroup		×				×			
ComplexType	< !ELEMENT x (y …	×	×	×	×	×	×	×	×
SimpleType	< !ELEMENT x (#CDATA) >	×	×	×	×	×	×	×	×
Group		×			×	×		×	
Sequence		×	×	×		×	×		
Choice		×	×	×		×	×		
All		×	×			×	×		
Union						×			
Any						×			
Extension		×			×	×		×	
Restriction		×			×	×		×	
SimpleContent						×			
ComplexContent						×			
Import						×	×		
Include						×	×		
Max/minOccurs	REQUIRED/IMPLID/ + /?/*	×	×			×	×		×
Mixed					×			×	
Annotation	NOTATION			×					×
Namespace		×							
	#FIXED value								×
	< !ENTITY entity-name "entity-value" >								
Total number		14	8	7	9	19	10	9	7

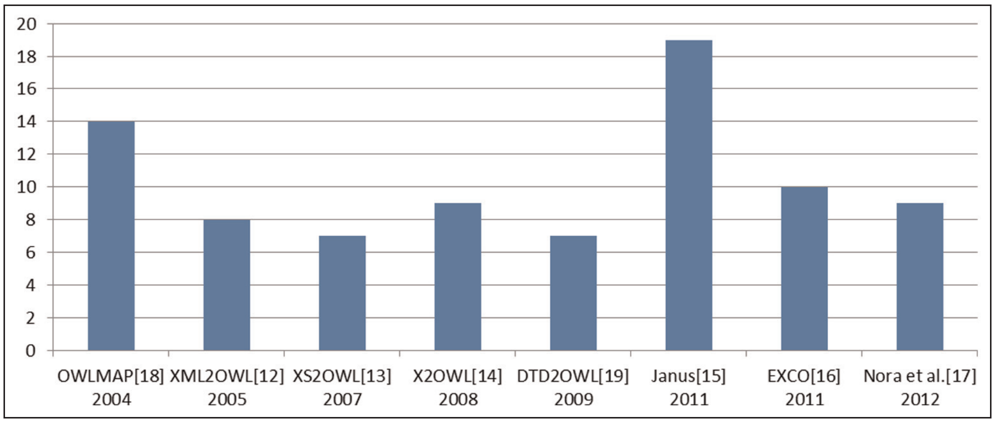

Figure 12. The comparison of validation approaches w.r.t. the number of schema elements processed.

Table 10. Comparison of approaches based on XML schemes.

Approach family	Complexity of correspondence rules	Information loss	Multiple XML data	Complexity of implementation
Instance approaches	More complex	No	One ontology for each XML instance	Easy
Validation approaches	More simple	May	One ontology for several XML instances	Hard

3.3.2. The complexity of the output results. The instance approaches are problematic when the XML documents are large and admit a deep imbrication. As a consequence, the mapping rules long and complicated. In contrast, the validation approaches offer predefined mapping rules that are short and simple. In addition, the instance approaches generate an ontology for each XML instance, even if they are validated by the same schema. Regarding the valuation approaches, they generate a single ontology schema and use it for the populating phase. This mechanism prevents reprocessing of the ontology schema.

3.3.3. Information loss. The instance approaches exploit all XML instance constructs that are elements or attributed. Consequently, information is not lost and completely transformed. The set of XML schema constructs is larger than that of the XML instances. This fact complicates the implementation of the validation approaches. For example, Janus offers 40 patterns, which are not easily implemented. As a consequence, the transformation process implies a loss of information. For instance, the information contained in the building extension or restriction shall not be taken into account in the process of transforming XML2OWL approaches XS2OWL, EXCO (see Table 9).

The comparisons of the different approaches are generally difficult to establish in order to determine the proper transformation process. Each approach overcomes the shortcomings of the other, and each is designed for a usage profile. However, there are some aspects that can be taken into account to achieve a quality transformation process:

- the selection of the most suitable OWL constructs with regard to XML constructs;
- the comprehensive treatment of all XML constructs to avoid information loss;
- the effectiveness of correspondence rules, which should not be too complex;
- the transformation process time should be considered;
- the possibility of user interventions for a certain configuration.

4. Conclusion

We have presented throughout this paper a survey on the different approaches to transform XML documents into OWL ontologies. We tried to maximally simplify the presentation of the principles regarding each approach. To reach this goal, the advantages of each approach were presented using examples. These approaches are classified in two main classes:

the instance approaches that make use of XML instances and the validation schema approaches that exploit XML schemas or DTDs. A comparative study is presented between these two classes and between the approaches of each class.

In addition, this paper focuses on ontology enriching and populating processes. The target ontologies are formal ontologies using the OWL or the RDF languages of the Semantic Web. This study underlines the richness of the validation approaches. This is due to the capabilities of XML schemes to specify integrity constraints. Thus, these constraints are eventually modelled in the resulting ontology. The selection of specific XML constructs for the transformation process may depend on the application domain. For example, the Janus approach selects many XML schema constructs in order to compute statistical analysis on the B2B domain.

This survey has identified eight properties to describe the approaches: the type of input data, the type of output data, the type of the OWL languages used, the generation of the schema ontology and the individuals is sequential or parallel, the generation of the corresponding rules which are automatic or semi-automatic, the use of the integrity constraints, and reuse of existing ontology. The selection of these properties remains for the domain application.

Finally, we have identified from this work a set of quality criteria that should be considered. The set is composed of three criteria: the richness of the transformation results, the complexity of the output results and the information loss. These criteria impact the expected transformation results. However, for the same application, all of these criteria are cost-demanding. Even so, these criteria are able to help developers in the selection of the most appropriate transformation properties.

As a future development, we are considering developing a semi-automated process based on XML2OWL [1] for the design of correspondence rules. These rules should be proposed automatically. However, it remains for the user to validate the rules. The populating process should be fully automatic and based on the correspondence rules. This development should additionally be based on the Janus approach [15] in order to consider all the XML schema constructs. Lastly, we should consider other data formats such as CSV, Excel and RDBMS for a more general data integration like an ETL software package (Extract Transform Load).

Acknowledgement

The author would like to thank the University of Science and Technology of Oran, the University of Oran 1, the University of Burgundy and the ‘Conseil Régional de Bourgogne’ for their support.

Funding

This research received no specific grant from any funding agency in the public, commercial or not-for-profit sectors.

References

[1] Cruz C and Nicolle N. Ontology enrichment and automatic population from XML data. In: *International VLDB workshop on ontology-based techniques for databases in information systems and knowledge systems, ODBIS*, New Zealand, 2008, pp. 17–20.
[2] Vanlande R, Nicolle C and Cruz C. IFC and building lifecycle management. *Automation in Construction* 2008; 18: 70–78.
[3] Manning CD, Raghavan P and Schütze H. Introduction to information retrieval. Word Processors & Editors Software, http://www-nlp.stanford.edu/IR-book/ (2008, accessed December 2013).
[4] Bray T, Paoli J, Sperberg-McQueen CM, Maler E and Yergeau F. Extensible Markup Language (XML), http://www.w3.org/TR/REC-xml/ (2008, accessed November 2013).
[5] Clark J. XSL transformations (XSLT) version 1.0, http://www.w3.org/TR/xslt (1999, accessed November 2013).
[6] Clark J. XML Path language (XPath) version 1.0, http://www.w3.org/TR/xpath/ (1999, accessed November 2013).
[7] Klyne G, Carroll JJ and McBride B. RDF 1.1 concepts and abstract syntax, http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/ (2004, accessed November 2013).
[8] Smith M, Welty C and McGuinness D. OWL web ontology language guide, 10 February 2004, http://www.w3.org/TR/owl-guide/
[9] Hitzler P, Krötzsch M, Parsia P, Patel-Schneider PF and Rudolph S. OWL 2 Web ontology language primer, http://www.w3.org/TR/owl2-primer/ (2012, accessed November 2013).
[10] Rodrigues T, Rosa P and Cardoso J. Mapping XML to existing OWL ontologies. In: *International conference on WWW/Internet*, 2006, pp. 72–77.
[11] O’Connor MJ and Das AK. *Acquiring OWL ontologies from XML documents*, 6th ed. Canada: Knowledge Capture (K-CAP), pp. 17–24, 2011.
[12] Bohring H and Auer S. Mapping XML to OWL ontologies. *Leipziger Informatik-Tage* 2005; 72: 147–156.

[13] Tsinaraki C and Christodoulakis S. Interoperability of XML schema applications with OWL domain. In: *Knowledge and semantic web tools, on the move to meaningful internet systems*. Lecture Notes in Computer Science, Vol. 4803. Berlin: Springer, 2007, pp. 850–869.

[14] Ghawi R and Cullot N. Building ontologies from XML data sources. In: *International workshop on modelling and visualization of XML and semantic Web data (MoViX '09)*, Linz, held in conjunction with DEXA, 2009, pp. 480–484.

[15] Bedini I, Matheus C, Peter F and Nguyen B. Transforming XML schema to OWL using patterns. In: *IEEE international conference on semantic computing*, Washington, DC, 2011, pp. 102–109.

[16] Lacoste D, Sawant KP and Roy S. An efficient XML to OWL converter. In: *India software engineering conference*, New York, 2011, pp. 145–154.

[17] Yahia N, Mokhtar SA and Ahmed A. Automatic generation of OWL ontology from XML data source. *International Journal of Computer Science Issues* 2012; 9: 77.

[18] Ferdinand M, Zirpins C and Trastour D. Lifting XML schema to OWL. In: *International Conference on Web Engineering (ICWE)*, Germany, 2004, pp. 354–358.

[19] Pham TTT, Lee Y and Lee S. DTD2OWL: Automatic transforming XML documents into OWL Ontology. In: *International conference on interaction sciences: Information technology, culture and human*, New York, 2009, pp. 125–131.

[20] Stavrakantonakis I, Tsinaraki C, Bikakis N, Gioldasis N and Christodoulakis S. SPARQL2XQuery 2.0: Supporting semantic-based queries over XML data. In: *Proceedings of SMAP*, 2010, pp. 76–84.

[21] Reynaud C and Safar B. Construction automatique d'adaptateurs guidée par une ontologie pour l'intégration de sources et de données XML. *Revue Technique et Science Informatiques* 2009; 28: 199–228.