Instance-level mapping of XML data to OWL ontologies

Ana Šaša[[1]](#footnote-2), Marjan Krisper

# Abstract

Information Systems Laboratory, Faculty of Computer and Information Science

University of Ljubljana

Tržaška 25, 1000 Ljubljana

Slovenia

With increasing quantities of data in business systems and potential benefits that ontologies can bring to a business system, the importance of automated mapping from XML documents to OWL ontologies has been recognized by many researchers. The study of related work in this field has shown that different types of XML to OWL mappings have not yet been researched in detail and that existing approaches do not support many mapping situations types that are very common in real world environments. This hinders the full extent of potential benefits that XML to OWL mappings could enable, such as improving the process of ontology evolution, information sharing, enabling a higher level of data reuse, more complete description of ontology domain, and, consequently, better ontology-based inference and decision support. In order to alleviate this problem, in this paper we identify different types of mapping situations and define a systematic way to address them. Based on different potential mapping requirements, we specify mapping definition types with an XML Schema in an implementation independent way. On the one hand, this allows for standardized mapping support of a large set of potential requirements, and on the other hand it allows for incorporation of implementation details specific for the target business system. We have considered all OWL-ontology instance level concepts that can be a result of XML to OWL mapping and supported them with the proposed mapping definition types. We have used our approach in two real-world environments and we were able to satisfy all mapping requirements. Automated XML-to-OWL mappings not only improved the role of the ontology-based knowledge bases, but actually enabled the fulfilment of their primary functions, such as decision support. Data reuse was improved and maintaining up-to-date, accurate and complete information in the ontology was considerably facilitated.

# Keywords

Ontologies, XML, Mapping, Transformation, XML Schema

# Introduction

Ontologies are increasingly important as a means of enabling communication and knowledge sharing of domains of interest, to build knowledge bases, and to support decisions [1][2][3][4][5][6][7]. In organizations, large quantities of data usually exist. These data carry various information that may be relevant to be captured in an ontology. However, there is a large gap between data representation and ontologies, especially due to differences in semantic expressivity between them. One of the most widespread data format standards has become XML, and as the W3C recommendation for building ontologies, the Web Ontology Language (OWL) has become one of the most important ontology languages. The importance of automated mapping from XML documents to OWL ontologies has been recognized by many researchers in the field of ontologies and knowledge management [8][9][10][11][12][13][14][15]. An important goal of such mappings is to assist organizations to move from a syntactic data infrastructure defined in XML to a semantic data infrastructure using OWL [8]. It can increase reuse of data that exist within a business system, improve information sharing, and facilitate development and evolution of ontologies, which is a tedious and costly challenge [16]. Our approach is concerned with mappings from XML documents to existing OWL ontologies. The study of related work in this field has shown that even though mappings from XML to existing OWL ontologies have been previously discussed, they have focused mainly on proposing a set of predefined one-for-all type mappings or on proposing methods and tools to develop custom mappings. However, different types of mappings have not yet been researched in detail and existing approaches do not allow for support for many mapping situations that are very common in real world environments. We see this as the key drawback in capturing the information carried in XML documents with OWL ontologies and in applicability of existing mapping proposals. This hinders the full extent of potential benefits that XML to OWL mappings could enable, such as improving the process of ontology evolution, information sharing, enabling a higher level of data reuse, more complete description of ontology domain, and, consequently, better inference and decision support. The approach proposed in this paper strives to alleviate this problem. For this purpose, we identify the types of mapping situations that have and that have not yet been supported and define a systematic way to address them. Furthermore, we propose to standardize the way of defining XML to OWL mappings in a way that supports the mapping situation types and is implementation independent. We have tested our approach in two real-world environments and we were able to satisfy all mapping requirements as well as allow for different specifics into the implementation of the mappings with the same standard mapping definitions.

# Related Work and Motivation

## Ontologies

Ontology is an explicit formal specification of a shared conceptualization of a domain of interest [1][17]. Lai defines an ontology as a means of enabling communication and knowledge sharing by capturing a shared understanding of terms that can be used both by humans and by programs [2]. This concept extends the idea of XML by providing not only the structure to a document, but by providing a way of describing the machine understandable meaning of a document [18]. There are several languages and formalisms used to express ontologies, such as Ontolingua [19], KIF [20], conceptual graphs [21], DAML, OIL, DAML+OIL [22], and OWL (OWL 1.0 [23], 1.1 [24], and the latest version OWL 2 [25]). The OWL language is a W3C’s Semantic Web Activity effort and has become one of the most widely used ontology languages. OWL 2 ontologies consist of entities, expressions, and axioms. The main OWL entities are Individuals, Properties and Classes. Individuals (also known as instances) represent objects in the domain that we are interested in. Properties state relationships between individuals or between individuals and data values. An OWL class can be defined as a set of individuals. OWL expressions represent complex notions in the domain being described. OWL axioms are statements that are asserted to be true in the domain being described. OWL ontologies and their elements are identified using Internationalized Resource Identifiers (IRIs) [25]. OWL designers have provided three increasingly expressive sublanguages named: OWL Lite, OWL DL and OWL Full. OWL Lite primarily supports classification hierarchy and simple constraint features. OWL DL corresponds with description logic and supports great expressiveness with decidable inference. OWL Full offers maximum expressiveness without computational guarantees. Our approach is based on OWL 2 DL [25], due to its ability to represent a useful group of ontology features, favourable computational properties, a high level of support, and its XML foundations, which make it appropriate to be used in conjunction with other Web technologies.

## Ontology development methodologies

Mapping of XML data to an existing OWL ontology should be used with a chosen ontology development methodology that encompasses the whole ontology life-cycle. Several methodologies for ontology development have been proposed. Staab et al. [5] have proposed a methodology for development of ontology-based knowledge management (KM) systems. The methodology distinguishes between two knowledge management processes: the knowledge process and the knowledge metaprocess. The knowledge metaprocess addresses aspects of introducing a new KM solution into an enterprise as well as maintaining it. The knowledge process addresses the handling of the already set-up KM solution. The authors point out that the two processes should be kept separate in order to have a clear separation of issues. The knowledge metaprocess represents an extension of the CommonKADS methodology [26], with the main difference that the CommonKADs methodology focuses only on early phases of an ontology life cycle. The On-To-Knowledge methodology [27] builds on the distinction of the two knowledge management processes, and comprises a full lifecycle of ontology based knowledge management systems. Maedche et al. [3] discuss how to manage ontology evolution. Ontology evolution is timely adaptation of the ontology to changing requirements and consistent propagation of changes to the dependent artefacts. It encompasses six phases: discovery, representation, semantics of change, propagation, implementation and validation. Lai [2] proposes a knowledge engineering approach to knowledge management which is based on conceptual graphs. The main activities comprise knowledge modelling, knowledge verification based on the formal semantics of the knowledge representation, knowledge storage in a hierarchical ontology system, knowledge querying and knowledge update. Methontology [28] is a methodology for ontology construction, which defines the development process based on the activities identified in the IEEE standard for software development. It defines three categories of activities: ontology management, ontology development-oriented activities, and ontology support. DILIGENT [29] is an argumentation ontology for distributed, loosely-controlled and evolving engineering process of ontologies. It focuses on the process of ontology engineering in which participants exchange arguments which may support or object to certain ontology engineering decisions. It enables tracking arguments and allows for inconsistency detection. A detailed survey of different ontology creation methodologies can be found in [30].

Our approach does not try to propose a new methodology for ontology development. Its purpose is to provide specifications that enable to define standard mappings from XML documents to existing OWL ontologies in a way that covers different mapping situations. It can complement ontology development methodologies in order to enhance instance level ontology development and evolution.

## Mapping from XML to OWL

Two main research streams can be identified in the field of mapping XML documents to OWL ontologies. The first is concerned with creation of a new ontology, for example [10][11] [12][13][31]. Ferdinand et al. [10] focus on extracting semantic information out of document schemas. They propose translations of XML documents to RDF graphs and lifting of XML schema to OWL. Their approach allows automated bootstrapping of ontology development from existing XML Schemas, which speeds up the adoption of Semantic Web technologies. Garcia et al. [11] developed an ontological framework based on journalism and multimedia standards, which are founded on XML technologies. In the scope of their work they also developed a generic XML Schema to OWL mapping. Bohring and Auer [12] present a mapping between XML and OWL and propose its implementation with an XSLT (Extensible Stylesheet Language Transformations [32]) framework. Their framework comprises the whole translation process: from a single XML instance document, over a generated XML schema to an OWL model with OWL instances. Anicic et al. [13] present a general method to map an XML Schema business document specification into OWL representation. They define mappings over RDF triple representation of superimposed models using production rules. They also define mapping rules to extract model information implicitly contained within a specific XML schema based document design and to represent this information using OWL concepts. In order to map an XML Schema to an OWL ontology, the GeRoMe metamodel and language developed by Kensche et al. [31] can be used. Their mappings support data translation between heterogeneous models, such as XML Schemas, relational schemas, and OWL ontologies. The approaches in the first research stream are concerned with automatic creation of a new OWL ontology based on the implicit semantics available in the structure of XML documents and definitions of predefined transformation rules. This can be very helpful in the process of ontology development. However, one of the main drawbacks is that an XML Schema is unable to accurately model the concepts and relations defined in ontology. These differences lead to possible information loss during the transformation from XML document to an ontology [9]. In case of such loss, the resulting ontology has to be supplemented in order to provide an accurate description of a domain of interest. In comparison to these approaches, we do not try to derive conceptual information from an XML schema or propose automatic translation to a new OWL ontology. We are concerned with mapping of XML document to an existing OWL ontology. Our approach is complementary to the ontology development and evolution methodologies and approaches. The most appropriate can be chosen for a given problem. Thus, it can be also used with automatic translations to new OWL ontologies, if this suffices for a given problem.

The second stream of research in this field is concerned with mapping of XML documents to existing OWL ontologies. Rodrigues et al. [8][33] propose an approach for mapping XML schema to existing OWL ontologies and transforming instances of the XML schema into individuals. Their main goal is to improve integration of disparate information spread over several data sources inside and outside the organization. They also propose a mapper tool and a solution to the semantic heterogeneity problem. The main motivation for their work was that the previously existing approaches had been focused on creation of new ontologies from XML and had not supported the incremental addition and mapping of new data sources. Cruz and Nicolle [14] extend the research presented in [33] and discuss mapping of several XML schemas into an existing OWL ontology. They present a method for enriching and populating an existing OWL ontology from XML data. Their enrichment process requires annotation knowledge which is contained in XML schemas. Both approaches are concerned with translations to instances and their properties in the ontology. Therefore, we refer to this type of mappings as XML to instance level ontology mappings.

Our approach belongs to the second stream of research. The main difference with the existing approaches in this group is that the focus of our research is to support a wide range of possible mappings that can satisfy different requirements which may arise in a business system. We try to improve instance level ontology evolution in an agile way by striving to exploit XML data that is being processed in the system to its maximum. For this, we have identified several limitations of existing approaches and grouped them into four categories (Mapping to all OWL assertion types not supported:*-*Predefined explicit definitions and input XML document based definitions:):

1. *Mapping to all OWL assertion types not supported:*

OWL 2 supports a rich set of axioms for stating assertions (axioms about individuals). Apart from class assertions, positive data and positive object property assertions, other types of assertions, such as equality and inequality assertions and negative property assertions, are not supported by previous approaches. The reason why negative data and object property assertions are not supported with previous approaches is mainly that they considered OWL 1.0 [23] and 1.1 [24] versions. These versions did not comprise all the assertion types of the latest version OWL 2.0 [25].

1. *Non-differentiation of existing and new individuals:*

Previous approaches do not take into consideration whether an individual that a mapping refers to already exists in the ontology or not. This may not be a drawback for some situations, such as if at design time it is not known whether if an individual will already exist in the ontology at design time or not. However, it may not be always clear for the mapping implementation mechanism whether an individual should be added to the ontology or not. If this information is present in the mapping definition, the mechanism can define rules regarding what to do in situations when an individual with the same naming already exists. If the mapping specifies that the individual refers to an existing individual, than the mechanism can identify it based on the naming. On the other hand, if the mapping specifies that it as a new individual and an individual with the same naming already exists in the ontology, then the mechanism may assign it a different name, consider it as a faulty mapping and maybe even notify the responsible knowledge engineer to resolve the issue. Thus, specifying if a mapping refers to an existing or a new individual is useful for many situations, especially if we want to achieve better control and governance over the instance-level ontology evolution.

1. *Assertion mapping dependencies:*

In previous approaches, one can define a mapping to an individual, to a data property value or to an object property value of an individual, only if a mapping to class membership for the individual is defined. This can be useful in the following two situations:

1) Individual’s class membership is not defined in the ontology and we have the information about its class membership at the mapping design time.

2) The individual already exists in the ontology and its membership is already defined, but we would like to assert that it belongs to another class, for which its membership was thus far unknown.

However, this also means that one cannot define a mapping to an individual or its property value independently of its class. This imposes several limitations:

1. If an individual already exists in the ontology, one cannot define a mapping to its property value without a mapping to its class membership, even if its class membership is already defined in the ontology. In this case, the mapping definition contains redundant information, which can lead to inconsistencies and maintenance complexity. Furthermore, this is especially inconvenient if the input XML document does not comprise information about membership of an individual, but only about its property values, and if the mapping designer does not know about class membership of the individual, e.g. due to separation of concerns. In this case, due to the assertion mapping dependency the mapping to a property value cannot be defined.
2. One cannot define a mapping to an object property of an individual, if a class mapping for its range individuals of the object property is not defined. This means, that we cannot define a mapping to an object property, if the input XML document does not comprise information about its class membership and the mapping designer is not familiar with it. This is especially inconvenient if the range individual is an existing OWL individual in the ontology.
3. We may not want to assert class membership for an individual with the mapping to an individual or its property value, because information about its class membership will be known at some later time.
4. We may not want to assert class membership for an individual explicitly, but rather deduce it based on individual’s property values.
5. *Predefined explicit definitions and input XML document based definitions:*

In previous approaches, different parts of mapping definitions are given either explicitly with an URI of an OWL entity, or based on input XML document. Which mapping part is given in which of these two ways is predefined for different mapping types:

* + In mappings to class assertions, the class is defined explicitly with its URI reference and the individual is defined by an XPath query [34] based on the input XML document. For each XML node that satisfies the query, one instance is created in the ontology and is asserted as a member of the specified class. This is useful in many situations. However, such an approach does not allow for dynamic class definitions based on expressions that take into account input XML document data. Thus, it does not apply to situations where every individual that corresponds to a node is not a member of the same OWL class. The corresponding class may be different for different individuals and in such cases it would be useful that the class URI is defined by an expression language.
  + Similarly, in property assertion mappings, the property is explicitly defined with its URI. Such an approach does not allow for dynamic property definitions based on expressions that take into account input XML document data.
  + In class and property assertion mappings, one can refer to an individual only by referencing data in input XML document. This means that one cannot define a mapping to a class assertion or a property assertion, if the input XML document doesn’t comprise data that defines its domain individual. For example, let us consider the following situation: 1) input XML document doesn’t comprise data that explicitly specifies its domain individual, 2) input XML document comprises data about an individual’s class membership, its data property value or about a range individual for its object property value, 3) we know at design time to which individual the data refers to. In such a situation, we cannot define a mapping that would assign the individual the corresponding class membership or a property value, even though the XML document comprises this information. Examples of such XML documents can be often found, such as data about a company, where we know that the data refers to “our” company, even though its name is not given explicitly in the document.
  + In mappings to data and object property assertions, one or more values may result from the XPath expression defining the property value. In previous approaches, in order to get the value used as the range of the property, the XPath expression in the class mapping is used together with the relative XPath expression from the property mapping. Based on this, a property value is then asserted for every individual resulting from the class mapping. Another possible requirement for a mapping that was not taken into account by previous approaches is that in some situations the XPath expression may not be required to be given relatively to the individual node. The difference is that resulting property values could be given for all domain individuals and would be asserted for every domain individual. Previous approaches do not allow for this possibility and several mappings would have to be defined to obtain the same result.

In order to surpass these limitations, the paper proposes several mapping definition types that support different mapping situations. We define an XML schema that specifies mapping definition types and allows for standardized XML to instance level OWL mapping definitions in a way that is implementation independent. In this regard, our approach extends previous approaches concerned with mapping of XML documents to existing OWL ontologies by widening the scope of the mappings and by proposing a standardized way to represent XML to OWL mappings.

# Fundamentals for XML to OWL instance level mapping

In this section, we describe the fundamental concepts for defining mappings from XML to instance level of an OWL ontology (XML-to-OWL). The structure of an XML document is most commonly defined by an XML Schema [35], which has become the most important XML data structure definition standard. Therefore, the basis for defining mappings are 1) an XML schema of input XML documents, and 2) the existing OWL ontology. An XML-to-OWL mapping definition document is comprised of one or more mapping definitions that translate an input XML document to different instance level concepts of an OWL ontology.

XML-to-OWL extends support for different mapping situations proposed by previous approaches in order to surpass the limitations identified in the previous section. In the following subsections, we first introduce the two general types of mapping types.. In the second subsection, we discuss the composition of mapping definitions and introduce the key concept of a mapping part. In Section 3.3, we consider two main ways to define a mapping part. Finally, in Section 3.4, we formally define the mapping part function, which will provide the basis for the concrete mapping rules defined defining in the next section, Proposed solution for XML to OWL instance level mapping.

## Mapping types

We distinguish between different types of mappings based on possible mapping results in the ontology. An instance level mapping can result in either a new individual or in an assertion (about one or more individuals) in the ontology[[2]](#footnote-3). Therefore, we group XML-to-OWL mappings into two main categories: mappings to OWL assertions (*assertion mappings*) and mappings to OWL individuals (individual mappings).

The OWL 2.0 specifications distinguish seven types of assertions [25]: class assertions, individual equality assertions, individual inequality assertions, positive and negative object property assertions, and positive and negative data property assertions. Based on these types and the specific structures required to support the mappings, we group assertion mappings into four assertion mapping types: 1) data property assertion mappings: mappings to positive and negative data property assertions, 2) object property assertion mappings: mappings to positive and negative object property assertions, 3) class assertion mappings, and 4) individual equality/inequality assertion mappings: mappings to equality and inequality assertions. We use the term *property assertion mappings* as a generalized term for both data and property assertion mappings. With the detailed specification of these mapping types given in the subsequent sections, we address the limitations in the group LG1.

The main purpose of individual mapping definitions is to add new individuals to the OWL ontology. However, note that assertion mappings might concern not only individuals which are new to the ontology (and so presumably added with an individual mapping earlier in the mapping document), but also those which already exist in the ontology (possibly having been added with an individual mapping in some previous mapping document). Furthermore, we may not know in advance which of these cases applies. Therefore, individual mapping definitions can actually be of three types: mappings to new individuals, mappings to existing individuals and mappings to individuals for which existence is not known at design time. With this we address the Non-differentiation of existing and new individuals: limitation. In addition, for convenience in cases where one or more assertion mappings applies to individuals from more than one mapping, a collection of individual mappings can also be defined to group together all the individuals from multiple individual mappings.

## Composition of Mapping Definitions

A mapping definition can be composed of one or more mapping parts that define the corresponding OWL entities or data. Figure shows different types of mapping definitions and their mapping part definitions using the composition and aggregation relations.

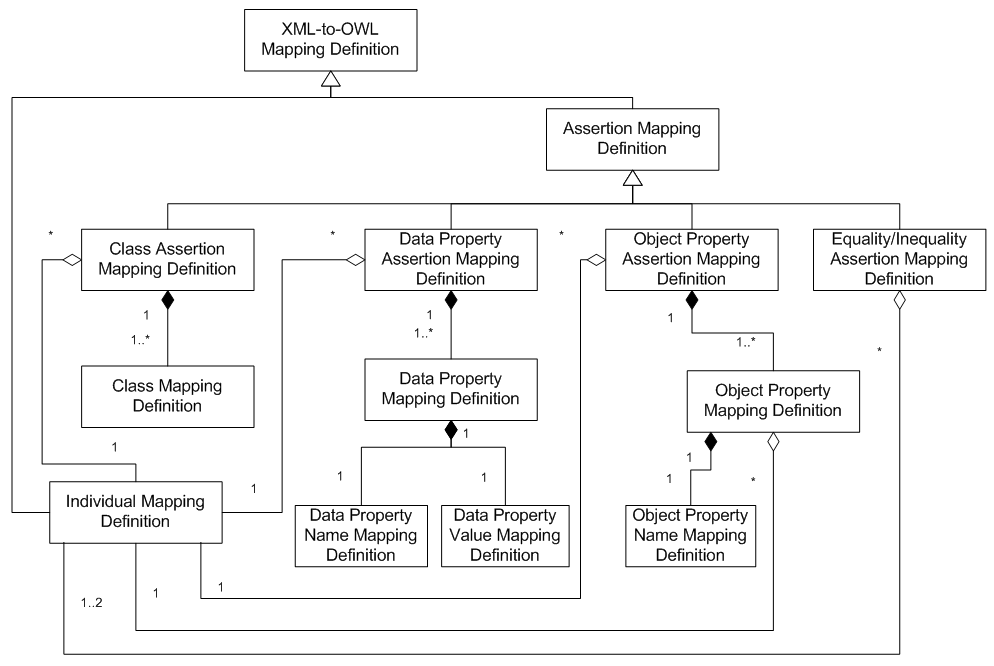


Figure : XML-to-OWL mapping metamodel

Every assertion mapping comprises at least one individual mapping definition part. Depending on the assertion mapping type, the assertion mapping definition can comprise one, two or more other mapping parts. As several assertion mapping definitions may concern the same individual mapping definition (or collection of individuals), an individual mapping can be defined independently and referenced by each assertion mapping that concerns it. Alternatively, it can be defined as part of an assertion mapping. This structure also allows for adding new individuals to the ontology independently of any assertions (see Assertion mapping dependencies: limitation category in Section Mapping from XML to OWL). Due to the fact that a mapping to a new individual comprises only one mapping part and due to similar data requirements, the same mapping definition type is used to define an individual mapping and to define a mapping to individual mapping part. It is referred to as an individual mapping definition.

Class assertion mappings are composed of two parts – a part that defines the class and a part that defines individuals that represent instances of the class. Property assertion mappings have three parts – one for the property name, a second for the property value (whose type depends on whether it's a data or object property), and a third for the individuals that the property applied to. Individual equality/inequality assertion mappings are composed of one set of individuals and optionally another (see Section ? for interpretation details). Metadata in a mapping definition is used to specify different mapping attributes such as if a data property assertion is negative or positive.

This organisation does not impose dependencies between different mapping definitions, while enabling different assertion mappings to refer to the same group of individuals. In this way, the L3 limitation is addressed.

## Dynamic and static mapping parts

In every XML-to-OWL mapping definition, at least one mapping part has to be based on the input XML document. Otherwise, none of the information for the mapping would come from the input XML document so the mapping would not represent a mapping from an XML document. In order to allow for a mapping part to be defined based on the input XML document, the corresponding XML nodes that contain relevant data are determined by an XML query language. Due to the structure of XML, a query can return multiple nodes. One mapping part can thus map to more than one OWL entity and consequently, one mapping definition can result in several mappings for one XML document. In order to define the IRI for the corresponding OWL entities, an XML expression language is used. At run-time, the expression is processed for each node that results from the query. The concept of a current node may be used in the expression, for example the use of “.” in the XPath language [34]. If it is used, the current node refers to the node resulting from the query for which the expression is processed.

Alternatively, if a mapping part is not based on the input XML document, it is given by an explicit IRI. In this case, the entity presumably already exists in the ontology, either due to a previous XML-to-OWL mapping or previously added without the use of XML-to-OWL.We refer to a mapping part definition that is given with a query and an expression as a *dynamic mapping part definition*, and to a mapping part that is given by an explicit IRI as a *static mapping part definition*.

We distinguish between assertion mapping situation types based on whether its mapping parts are defined statically or dynamically. Furthermore, if the domain individual mapping part and at least one other mapping part are defined dynamically, two possible situation types can be distinguished:

* all resulting domain individuals are related to all entities resulting from the other mapping part definition (all-to-all); or
* each resulting individual is related to distinct entities resulting from the other mapping part definition (all-to-some),

where *related to* refers to relations that assertions refer to, such as class membership or individual’s property value.

By allowing that every mapping type is defined either dynamically or statically as long as at least one of the mapping parts is given dynamically, and by supporting all-to-all and all-to-some situations, we address the Predefined explicit definitions and input XML document based definitions: limitation. Due to the combination of various possible mapping part definitions, a wide range of XML document structures can be mapped into an OWL ontology.

## Mapping part function

Every mapping part definition determines exactly one mapping part function. Let us define the following designations:

* … a set of XML documents,
* … a set of XML nodes in XML document ,
* … a generic representation for a set of OWL entity vectors which is different based on the mapping part type (it is concretized with the mapping type specifications).

A mapping part function can be defined as:

|  |  |
| --- | --- |
| , | (1) |

where is an input XML document, is the current node in , and is a vector of OWL entities of the type defined with the mapping. For example, in case of a class mapping definition it is a set of class vectors, and in case of a data property name mapping definition it is a set of data property vectors.

In a mapping definition, the current node may not be given explicitly. The default value for the current node is the root node of the input XML document.

If a mapping definition if given statically, the mapping part function is a constant and the resulting vector’s length is 1. If it is given dynamically, is a composed function:

|  |  |
| --- | --- |
| ,  ,  , | (2) |

where is a function determined by the mapping part query and translates an XML document to , a vector of certain nodes within the document, based on the current node ; is a function determined by the mapping part expression and translates XML nodes to OWL entities.

# Proposed solution for XML to OWL instance level mapping

Listing represents the overall syntax of XML to OWL ontology mappings definition. The mapping definitions are XML based. We have defined an XML Schema that unambiguously specifies the mapping structures. We have introduced a namespace URI *http://www.fri.uni-lj.si/xml2owl* for the XML Schema. The *queryLanguage* and *expressionLanguage* attributes specify the query language and the expression language used in the enclosing elements, respectively. The default value for both *queryLanguage* and *expressionLanguage* attributes is: “urn:fri-x2o:sublang:xpath2.0”, which represents the usage of XPath 2.0 within XML-to-OWL.

The proposed specifications allow subsequent extensions for potential other mapping situations that may be required. For this purpose, the *extensions* element is used to specify namespaces of XML-to-OWL mappings extension attributes and elements. The element is optional. If present, it must include at least one extension element. The *extension* element is used to specify a namespace of XML-to-OWL extension attributes and extension elements, and indicate whether they are mandatory or optional. Attribute *mustUnderstand* is used to specify whether the extension must be understood by a compliant implementation.

If the *prefixIRI* element is given, it defines the global prefix for the mapped OWL concepts’ IRIs. It can be given with an expression that references input XML document or explicitly with a string representation of the IRI. This value can be overridden in the individual mapping definitions. The string representation of the full IRI of an OWL concept corresponds to concatenation of the *prefixIRI* and the result of the name expression of the concept. The name expression is given by the corresponding mapping definition. For example, if “*http://www.oo.si/products.owl#*” represents a result of the *prefixIRI* expression and “*LCD\_TV\_ZX123*” is a result of a name expression for an OWL individual in the same mapping, then the full IRI of the individual is their concatenation, i.e. “*http://www.oo.si/products.owl# LCD\_TV\_ZX123*”.

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <ontologyMappingElements      name="NCName"     xmlns="http://www.fri.uni-lj.si/xml2owl"     targetNamespace="anyURI"      queryLanguage="anyURI"?     expressionLanguage="anyURI"?     xmlns:xsd="http://www.w3.org/2001/XMLSchema"      xmlns:tns="anyURI">      <extensions>?         <extension namespace="anyURI" mustUnderstand="yes|no" />+     </extensions>      <prefixIRI type="QName">?         expression     </prefixIRI>           [ <mapToOWLIndividual name="NCName"?>          ...          <mapToOWLIndividual> |          <mapToClassAssertion>         ...          </mapToClassAssertion> |             <mapToOWLDataPropertyAssertion>          ...         <mapToOWLDataPropertyAssertion> |             <mapToOWLObjectPropertyAssertion>         ...          </mapToOWLObjectPropertyAssertion> |           <mapToSameIndvs>         ...          </mapToSameIndvs> |           <mapToDifferentIndvs>         ...          </mapToDifferentIndvs> ]+       <ontologyMappingElements> |

Listing : Overall syntax of XML to OWL mapping specification

In the XML to OWL mapping specification there is one element for the individual mapping type (*<mapToOWLIndividual>).* Each of the four main assertion mapping types is represented by a distinct element (<*mapToOWLDataPropertyAssertion*>, *<mapToOWLObjectPropertyAssertion>*, *<mapToSameIndvs>, <mapToDifferentIndvs>*). Their subtypes are represented by attributes, for example to distinguish between negative and positive property assertion mappings.

The *mapToOWLIndividual* element supports the mapping to new OWL individuals mapping type and mapping to OWL individual mapping part definitions. This is due to the facts that 1) a mapping to a new individual can be a mapping in its own right and/or it can be referenced as a mapping part of an assertion mapping definition, 2) that a mapping to a new OWL individual comprises only one mapping part (that defined the OWL individual), and 3) that the same XML structure can support mappings to new and existing individuals.

Thus the main mapping elements: *mapToOWLIndividual*, *mapToClassAssertion, mapToOWLDataPropertyAssertion*, *mapToOWLObjectPropertyAssertion, mapToSameIndvs,* and *mapToDifferentIndvs* support the main mapping types. Mapping definitions are composed of mapping parts. The same XML element is used for both static and dynamic mapping part definitions of the same type. In order to define a static mapping part, the expression contains an explicit string representation of the OWL entity IRI, whereas the query element is omitted.

The proposed approach to defining different mapping types in a standardized format is given in detail in the following subsections.

## Individual Mapping Definition

The range of a mapping part function determined by a mapping to OWL individual definition is a set of OWL individual vectors:

|  |  |
| --- | --- |
| , | (3) |

where represent the *owl:Thing*, and is defined as in (1).

Thus, based on (1) the mapping function of a mapping to OWL individual definition:

|  |  |
| --- | --- |
| , | (4) |

maps an XML document to a vector of OWL individuals based on the current node , where and are defined as in (1).

### Identification of individual mapping situations

We differentiate situations of mappings to individuals based on information about their existence. A mapping to an existing individual should always define the IRI and should always be referenced by at least one assertion mapping definition. However, we can distinguish between several situations for mappings to new individuals. We categorize them based on two criteria:

* based on information about the new individuals at design time:

1. At design time, it is known that some data in an XML document represent a new individual in the ontology and the unique IRI for the new individual can be determined based on the data in the document.
2. At design time, it is known that some data in an XML document represent a new individual in the ontology, but the unique IRI for the new individual can not be determined based on the data in the document. In this case the individual IRI is not defined by the mapping, but should be ensured by the mapping mechanism in order to determine a unique IRI.
3. A mapping to a new individual may be defined also when it is not certain at design time whether an individual will exist in the ontology at run-time. In this case the IRI of an individual must be given. If at run-time the individual is found as existent in the ontology, the mapping is either ignored (for example, if it is not referenced by any assertion mapping) or the existing individual is used in assertion mappings that references it.

* based on assertion mappings that refer to the new individuals:

1. A mapping to a new individual can be defined even if there is no assertion mapping that references it. This may be useful if we would like to add assertions for the individual subsequently, for example based on data in other XML documents.
2. A mapping to a new individual can be referenced by or be a part of an assertion mapping. References are useful if several assertion mappings refer to the same individuals.

### Syntax of individual mapping definition

The *mapToOWLIndividual* element (Listing 2) is used to define a mapping to OWL individual which can support the identified situations (Section Identification of individual mapping situations). The *mapToOWLIndividualType* attribute can have three possible values: new, existing and unknown, which determine whether the mapping refers to a new or existing individual, or if its existence is unknown at design time.

Mapping to an OWL individual comprises one mapping part that represents the individual. If it is given as a dynamic mapping part, its query and expression represent the following:

* The query defines XML nodes that represent OWL individuals. If the query results in more than one XML node, then an OWL individual is created for every one of them.
* If the input XML document also contains data for the unique OWL individual name, an expression is used to define OWL individuals’ names. References to input XML document in the expression can be given relatively to XML node that represents an individual. If the name expression is not specified, the mechanism implementing the mappings needs to ensure that every OWL individual created by the mapping is assigned a unique IRI.

In a mapping to OWL individual definition, one can define either the XML node that represents the OWL individuals, or the OWL individual name, or both. If the *prefixIRI* element is specified, it overrides the global prefix IRI.

In a static mapping part definition, the *prefixIRI* and the name are given explicitly, whereas the *nodeForOWLIndividual* element is not present.

|  |
| --- |
| <mapToOWLIndividual name="NCName"? mapToOWLIndividualType="new|existing|unknown"?>                  [ <nodeForOWLIndividual type="QName">              query          </nodeForOWLIndividual>          <prefixIRI type="QName">?             expression         </prefixIRI>          <OWLIndividualName type="QName">?             expression         </OWLIndividualName> ] |          [ <prefixIRI type="QName">?             expression         </prefixIRI>          <OWLIndividualName type="QName">             expression         </OWLIndividualName> ]             </mapToOWLIndividual> |

Listing : Definiton of mapping to OWL individual

### Example

Let us consider the following input XML document:

|  |
| --- |
| <?xml version="1.0"?> <newProducts>     <month>October 2009</month>      <brand>         <brandName>ABC</brandName>          <product>             <productID>1552</productID>             <productName>LCD TV 32LH3000</productName>             <itemType>Electronics</itemType>             <quantity>10</quantity>             <salesInfo>                 <typeOfSales>Web</typeOfSales>                 <price>579.00</price>              </salesInfo>             <salesInfo>                 <typeOfSales>Shop</typeOfSales>                 <price>599.00</price>              </salesInfo>             <additionalInfo>                 <award>TheProductOfTheYear</award>                 <award>WhatHi-FiAward</award>                 <averageRating>3stars</averageRating>             </additionalInfo>         </product>         <product>            ...          </product>         ...     </brand>     <brand>         ...     </brand>  </newProducts> |

Listing : Example input XML document

The document contains data about new products that we would like to add to the ontology as OWL individuals. Listing gives an example of a mapping definition that specifies such a mapping.

|  |
| --- |
| <mapToOWLIndividual mapToOWLIndividualType="new" name="MapToNewProduct">         <nodeForOWLIndividual type="xsd:string">/newProducts/brand/product</nodeForOWLIndividual>         <prefixIRI type="xsd:string">"http://www.oo.si/products.owl#"</prefixIRI>         <OWLIndividualName type="xsd:string">fn:concat("Product\_",./productID)</OWLIndividualName>  </mapToOWLIndividual> |

Listing : Example mapping to OWL individual definition

## Class Assertion Mapping Definition

An OWL class assertion states that the individual is an instance of the class expression [25]. Thus, a class assertion mapping has two mapping parts: class mapping definition and an individual mapping definition. Let denote a set of all classes in an OWL ontology that the mapping refers to. A dynamic class mapping definition can be given relatively to distinct individual nodes or based on the input XML document as a whole.

The range of a mapping part function determined by a definition of a class definition mapping part is a set of OWL class vectors:

|  |  |
| --- | --- |
| , | (5) |

where represent a set of OWL classes in the OWL ontology that the mapping refers to. is defined as in (1).

Thus, based on (1) the mapping function of a class assertion mapping:

|  |  |
| --- | --- |
| , | (6) |

maps an XML document to a vector of OWL classes based on the current node , where and are defined as in (1).

### Identification of class mapping situations

Different class assertion mapping situations and resulting class assertions are given in Table (Table -A and 2-B).

|  |  |  |
| --- | --- | --- |
|  | SCMPD | DCMPD |
| SIMPD | Not relevant. | Situation identification: CA2  Situation description:  Input XML document contains data about classes that a certain (predefined) individual is a member of.  Resulting class assertions: |
| DIMPD | Situation identification: CA1  Situation description:  Input XML document contains data about individuals that are members of a certain (predefined) class.  Resulting class assertions: | Situation identification: CA3  Situation description:  Input XML document contains data about individuals and classes that the individuals are member of.  We distinguish between two CA3 situation subtypes: CA3.1 and CA3.2. They are given in more detail in Table -B. |

Table -A: Main class assertion mapping situations and resulting class assertions

|  |  |  |
| --- | --- | --- |
| Situation identification | CA3.1 (all-to-all) | CA3.2 (all-to-some) |
| Situation description | All individuals that the input XML document refers to are asserted as members of all classes that the input XML document refers to. | Every individual that the input XML document refers to is asserted as member of classes given relative to the individual node. |
| Resulting class assertions |  | : |

Table -B: Situation CA3 subtypes

Table : Class assertion mapping situations and resulting class assertions

Legend:

|  |  |
| --- | --- |
|  | SIMPD… static individual mapping part definition |
| …individual mapping part function | DIMPD… dynamic individual mapping part definition |
| …query function for dynamic mapping to individual definition: | SCMPD… static class mapping part definition |
| DCMPD… dynamic class mapping part definition |
| …class mapping part function | … root node of the input XML document |
| …number of elements in vector |  |

### Syntax of class assertion mapping definition

The *mapToClassAssertion* element (Listing ) is used to define a class assertion mapping which can support the identified situations (Section Identification of class mapping situations). Mapping to OWL individuals, which are to be asserted as members of a class, is given either with the *mapToOWLIndividual* element or with the *referenceToOWLIndividual* element (a reference to an individual mapping). Attribute *classAssertionType* is used to distinguish between CA3.1 (attribute value *allIndividuals*) and CA3.2 (attribute value *distinct*) mapping situations. Its default value is allIndividuals. For CA3.2 situations, the query is relative to XML node that represents an OWL individual. If an *OWLClassPrefixIRI* is given, it overrides the global *prefixIRI*. If an *OWLClassPrefixIRI* element is defined within a *mapToClassAssertion* node and within the *OWLAssertionClass* node, then the latter definition is used within the *OWLAssertionClass* node.

|  |
| --- |
| <mapToClassAssertion>                  <mapToOWLIndividual name="NCName"? mapToOWLIndividualType="new|existing|unknown"?>              ...         </mapToOWLIndividual> |          <referenceToOWLIndividual refName="NCName" />                  <OWLClassPrefixIRI type="QName">?             expression         </OWLClassPrefixIRI>           <OWLAssertionClasses>             <OWLAssertionClass classAssertionType="distinct|allIndividuals"?>+                 <OWLClassNode type="QName">                      query                  </OWLClassNode>                 <OWLClassPrefixIRI type="QName">?                     expression                 </OWLClassPrefixIRI>                  <OWLClassName type="QName">                     expression                 </OWLClassName>             </OWLAssertionClass>         </OWLAssertionClasses>                     </mapToClassAssertion> |

Listing : Class assertion mapping definition

### Example

Listing gives an example of CA3.1 mapping situation type, and Listing gives an example mapping definition for CA3.2 mapping situation type. Both example mapping definitions are based on the XML schema of the example input XML document shown in Listing , and they reference the mapping to individual definition given in Listing .

|  |
| --- |
| <mapToClassAssertion>         <referenceToOWLIndividual refName="MapToNewProduct"></referenceToOWLIndividual>         <OWLClassPrefixIRI type="xsd:string">"http://www.oo.si/products.owl#"</OWLClassPrefixIRI>         <OWLAssertionClasses>             <OWLAssertionClass classAssertionType="allIndividuals">                 <OWLClassName type="xsd:string">fn:concat("NewProductsFor",fn:substring-before(/newProducts/month,' '),fn:substring-after(/newProducts/month,' '))</OWLClassName>             </OWLAssertionClass>         </OWLAssertionClasses>  </mapToClassAssertion> |

Listing : Example of all-to-all type class assertion mapping definition

|  |
| --- |
| <mapToClassAssertion>         <referenceToOWLIndividual refName="MapToNewProduct"></referenceToOWLIndividual>         <OWLClassPrefixIRI type="xsd:string">"http://www.oo.si/products.owl#"</OWLClassPrefixIRI>         <OWLAssertionClasses>             <OWLAssertionClass classAssertionType="distinct">                 <OWLClassName type="xsd:string">fn:concat(./itemType,"Product")</OWLClassName>             </OWLAssertionClass>         </OWLAssertionClasses> </mapToClassAssertion> |

Listing : Example of all-to-some type class assertion mapping definition

## Property Assertion Mapping Definition

There are two types of property assertions: object and data property assertions. Both can be either negative and positive [25]:

* A positive data property assertion states that the individual is connected by the data property expression to the literal .
* A negative data property assertion states that the individual is not connected by the data property expression to the literal .
* A positive object property assertion states that the individual is connected by the object property expression to the individual .
* A negative object property assertion states that the individual is not connected by the object property expression to the individual .

Based on the above definitions, a data property assertion mapping has three mapping parts: data property name mapping definition, data property value mapping definition and individual mapping definition. Similarly, an object property mapping has three mapping parts: object property name mapping definition, and domain and range individual mapping definitions. We also refer to the object property range individual mapping part definition as the object property value mapping part definition.

The range of a property name mapping part function is a set of vectors:

|  |  |
| --- | --- |
| . | (7) |

If the property is a data property, then represents a set of OWL data properties in the OWL ontology that the mapping refers to. If the property is an object property, then represents a set of OWL object properties in the OWL ontology that the mapping refers to. is defined as in (1).

The range of a property value mapping part function is a set of value vectors:

|  |  |
| --- | --- |
| . | (8) |

If the mapping is a data property assertion mapping, then represent a set of OWL datatypes. If the mapping is an object property assertion mapping, then . is defined as above.

We denote a property name mapping part function by , a property value mapping part function by , and the corresponding query and expression functions by and , respectively. Based on (2), the following holds:

|  |  |
| --- | --- |
| ,  . | (9) |

If property name and property value mapping parts are given dynamically their expressions refer to the same XML node:

|  |  |
| --- | --- |
| . | (10) |

Since both expressions refer to the same XML node, the query function is the same for both mapping part definitions of the same property mapping definition.

### Identification of mapping situations

Analogous types of situations can be identified for both negative and positive property assertion mappings. These situation types and resulting property assertions are given in Table . Due to the analogous nature of the situation types, the term is used as a generalized term for positive and negative data property assertions and for positive and negative object property assertions. All four types of property assertions mapping types should support these mapping situations types.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | SPNMPD | DPNMPD | |
| SIMPD | SPVMPD | Not relevant. | | | Not relevant. |
| DPVMPD | Situation identification: PA1  Situation description:  Input XML document contains data about property P values for a predefined individual I. All values obtained as a result of the query and the expression are asserted as values for property P for the individual I.  Resulting property assertions: | | | Situation identification: PA2  Situation description:  Input XML document contains data for one or more properties about a predefined individual. These properties can be different for different XML documents, but can be addressed with the same XML query and the same XML expression.  Resulting property assertions: |
| DIMPD | SPVMPD | Situation identification: PA3  Situation description:  Input XML document contains data about individuals that have a certain property value. It is known at design time what property value the individuals represented by the XML document have and they are not defined by the input XML document.  Resulting property assertions:    ) | | | Not relevant. |
| DPVMPD | Situation identification: PA4  Situation description:  Input XML document contains data for individual names and property P values of these individuals. Property P unique IRI is known at design time and specified statically in the mapping definition.  We distinguish between two PA4 situation subtypes: PA4.1 and PA4.2. They are given in more detail bellow. | | | Situation identification: PA5  Situation description:  Input XML document contains data about different properties and their values for different individuals. These data properties may differ for different input XML documents. They are addressed by one mapping definition.  We distinguish between two PA5 situation subtypes: PA5.1 and PA5.2. They are given in more detail bellow. |
| Situation identification: PA4.1 (all-to-all)  Situation description:  All individuals are asserted all values obtained by data property value mapping part.  Resulting property assertions:      ) | | | Situation identification: PA5.1 (all-to-all)  Situation description:  All individuals are asserted all values for all data properties obtained by property name and value mapping parts.  Resulting property assertions: |
| Situation identification: PA4.2 (all-to-some)  Situation description:  Individuals are asserted distinct values obtained by data property value mapping part relative to the individual mapping part.  Resulting property assertions:      ) | | | Situation identification: PA5.2 (all-to-some)  Situation description:  Individuals are asserted values for distinct properties obtained by property name and value mapping parts relative to the individual mapping part.  Resulting property assertions: |

Table : Property assertion mapping situations and resulting property assertions

Legend:

|  |  |
| --- | --- |
|  | SIMPD… static individual mapping part definition |
| …individual mapping part function | DIMPD… dynamic individual mapping part definition |
| …query function for dynamic mapping to individual definition: | SPNMPD…static (data or object) property name mapping part definition |
| … (data or object) property name mapping part function | DPNMPD… dynamic (data or object) property name mapping part definition |
| … (data or object) property value mapping part function | SPVMPD… static (data or object) property value mapping part definition |
| … root node of the input XML document | DPVMPD… dynamic (data or object) property value mapping part definition |
| …number of elements in vector |  |

### Syntax of data and object property assertion mapping definition

The *mapToOWLDataPropertyAssertion* element (Listing ) and the *mapToOWLObjectPropertyAssertion* element (Listing ) are used to define data and object property assertion mappings, respectively. They support the identified situations (Section Identification of mapping situations). Within both elements, the attribute *propertyMappingType* is used to distinguish between all-to-some (attribute value *distinct*) and all-to-all (attribute value *allIndividuals*) mapping types. The *rangeOWLIndividualMapping* element is of the same type as the *mapToOWLIndividual* element (see Section Individual Mapping Definition). The attribute *propertyAssertionType* defines whether a property assertion is negative or positive. The innermost *prefixIRI* element (*prefixIRI*, *OWLDataPropertyPrefixIRI*, *OWLObjectPropertyprefixIRI*) is valid for a given XML node of a mapping definition.

|  |
| --- |
| <mapToOWLDataPropertyAssertion>                  <mapToOWLIndividual name="NCName"? mapToOWLIndividualType="new|existing|unknown"?>              ...         </mapToOWLIndividual> |          <referenceToOWLIndividual refName="NCName" />                  <OWLDataPropertyPrefixIRI type="QName">?             expression         </OWLDataPropertyPrefixIRI>           <OWLDPAssertionsForDomainIndividuals>              <OWLDPAssertionForDomainIndividuals propertyMappingType ="distinct|allIndividuals"? propertyAssertionType="positive|negative"?>+                  <OWLDataPropertyPrefixIRI type="QName">?                     expression                 </OWLDataPropertyPrefixIRI>                   <OWLDataPropertyNode type="QName">                      query                  </OWLDataPropertyNode>                  <OWLDataProperty type="QName">                     expression                 </OWLDataProperty>                  <OWLDataPropertyValue type="QName">                     expression                 </OWLDataPropertyValue>              </OWLDPAssertionForDomainIndividuals>         </OWLDPAssertionsForDomainIndividuals>                     </mapToOWLDataPropertyAssertion> |

Listing : Data property assertion mapping definition

|  |
| --- |
| <mapToOWLObjectPropertyAssertion>                  <mapToOWLIndividual name="NCName"? mapToOWLIndividualType="new|existing|unknown"?>              ...         </mapToOWLIndividual> |          <referenceToOWLIndividual refName="NCName" />                  <OWLObjectPropertyprefixIRI type="QName">?             expression         </OWLObjectPropertyprefixIRI>           <OWLOPAssertionsforDomainIndividuals>              <OWLOPAssertionforDomainIndividuals propertyMappingType="distinct|allIndividuals"? propertyAssertionType="positive|negative"?>+                  <OWLObjectPropertyprefixIRI type="QName">?                     expression                 </OWLObjectPropertyprefixIRI>                   <rangeOWLIndividualMapping name="NCName"? mapToOWLIndividualType="new|existing|unknown"?>                     ...                 </rangeOWLIndividualMapping> | <referenceToRangeOWLIndividual refName="NCName" />                   <OWLObjectProperty type="QName">                      query                  </OWLObjectProperty>               </OWLOPAssertionforDomainIndividuals>         </OWLOPAssertionsforDomainIndividuals>                     </mapToOWLObjectPropertyAssertion> |

Listing : Object property assertion mapping definition

### Example

Listing gives an example of positive data property assertion for the PA4.1 mapping situation type. Listing gives an example of a positive object property assertion for the PA5.2 mapping situation type. Both example mapping definitions are based on the XML schema of the example input XML document shown in Listing , and they reference the mapping to individual definition given in Listing .

|  |
| --- |
| <mapToOWLDataPropertyAssertion>         <referenceToOWLIndividual refName="MapToNewProduct"></referenceToOWLIndividual>         <OWLDataPropertyPrefixIRI type="xsd:string">"http://www.oo.si/products.owl#"</OWLDataPropertyPrefixIRI>          <OWLDPAssertionsForDomainIndividuals>             <OWLDPAssertionForDomainIndividuals propertyAssertionType="positive" propertyMappingType="allIndividuals">                 <OWLDataPropertyNode type="xsd:string">/newProducts/month</OWLDataPropertyNode>                 <OWLDataProperty type="xsd:string">"introducedIn"</OWLDataProperty>                 <OWLDataPropertyValue type="xsd:string">fn:concat(fn:substring-before(.,' '),fn:substring-after(.,' '))</OWLDataPropertyValue>             </OWLDPAssertionForDomainIndividuals>         </OWLDPAssertionsForDomainIndividuals> </mapToOWLDataPropertyAssertion> |

Listing : Example positive data property assertion

|  |
| --- |
| <mapToOWLObjectPropertyAssertion>          <referenceToDomainOWLIndividual refName="MapToNewProduct"></referenceToDomainOWLIndividual>         <OWLObjectPropertyprefixIRI type="xsd:string">"http://www.oo.si/products.owl#"</OWLObjectPropertyprefixIRI>          <OWLOPAssertionsforDomainIndividuals>             <OWLOPAssertionforDomainIndividuals propertyAssertionType="positive" propertyMappingType="distinct">                 <rangeOWLIndividualMapping mapToOWLIndividualType="existing">                     <nodeForOWLIndividual type="xsd:string">./additionalInfo/\*</nodeForOWLIndividual>                     <prefixIRI type="xsd:string">"http://www.oo.si/products.owl#"</prefixIRI>                     <OWLIndividualName type="xsd:string">.</OWLIndividualName>                 </rangeOWLIndividualMapping>                 <OWLObjectProperty type="xsd:string">fn:concat("has",fn:local-name(.))</OWLObjectProperty>             </OWLOPAssertionforDomainIndividuals>         </OWLOPAssertionsforDomainIndividuals>  </mapToOWLObjectPropertyAssertion> |

Listing : Example positive object property assertion

## Individual Equality/Inequality Assertion Mapping Definition

An individual equality axiom states that all of the individuals , are equal to each other. An individual inequality axiom states that all of the individuals , are different from each other; that is, no individuals and with can be derived to be equal [25]. Individual equality and inequality assertion mapping definitions comprise one or two mapping parts. Every mapping part is of the individual mapping definition type. The corresponding mapping part function is defined by (3) and (4).

### Identification of mapping situations

Analogous types of situations can be identified for individual equality and inequality assertion mappings. If individual equality/inequality assertion mapping definition comprises only one mapping part (Situation identification: IEA1), then the mapping part should be given dynamically. All individuals obtained by such mapping part are asserted as equal/different to each other, depending on the mapping type (equality/inequality).

If individual equality/inequality assertion mapping definition comprises two mapping parts, then we can distinguish between several different mapping situation types (Table ). Due to the analogous nature of the situation types, the term is used as a generalized term for both individual equality and inequality assertions. Both individual equality and inequality assertion mapping definition types should support these mapping situation types.

|  |  |  |
| --- | --- | --- |
|  | SIMPD | DIMPD |
| SIMPD | Not relevant. | Situation identification: IEA3  Situation description:  Situation IEA3 is equal to the situation IEA2 with reversed operands in the mapping definition. |
| DIMPD | Situation identification: IEA2  Situation description:  The statically given individual is different from/equal to all individuals obtained by the dynamic part mapping.  Resulting assertions: | Situation identification: IEA4  Situation description: Both individual mapping parts are given dynamically.  We distinguish between two IEA4 situation subtypes: PA4.1 and PA4.2. They are given in more detail bellow. |
| Situation identification: IEA4.1 (all-to-all)  Situation description:  All individuals resulting from the first mapping part are equal to/different from all individuals resulting from the second mapping part.  Resulting assertions: |
| Situation identification: IEA4.2 (all-to-some)  Situation description:  Every individual resulting from the first mapping part is equal to/different from distinct individuals resulting from the second mapping part. The individual nodes of the second mapping part are obtained relatively to the first mapping part individual nodes.  Resulting assertions:    : |

Table : Individual equality and inequality assertion mapping situations and resulting equality and inequality assertions

Legend:

|  |  |
| --- | --- |
|  | SIMPD… static individual mapping part definition |
| …individual mapping part function | DIMPD… dynamic individual mapping part definition |
| …query function for dynamic mapping to individual definition: | … root node of the input XML document |
| …number of elements in vector |

### Syntax of individual equality/inequality assertion mapping definition

The *mapToSameIndvs* element (Listing ) and *mapToDifferentIndvs* element are used to define individual equality and inequality assertion mappings, respectively. They are based on the same XML schema type, therefore the syntax for the *mapToDifferentIndvs* element is not shown explicitly. They support the identified mapping situations (Section Identification of mapping situations).

|  |
| --- |
| <mapToSameIndvs equlityMappingType="distinct|allIndividuals"?>                  <prefixIRI type="QName">?             expression         </prefixIRI>          <mapToOWLIndividual name="NCName"? mapToOWLIndividualType="new|existing|unknown"?>              ...         </mapToOWLIndividual> |          <referenceToOWLIndividual refName="NCName" />                  <OWLObjectPropertyprefixIRI type="QName">?             expression         </OWLObjectPropertyprefixIRI>           [ <mapToOWLIndividual name="NCName"? mapToOWLIndividualType="new|existing|unknown"?>              ...         </mapToOWLIndividual> |          <referenceToOWLIndividual refName="NCName" /> ]?                      </mapToSameIndvs> |

Listing : Individual equality assertion mapping definition

### Example

Listing shows an example of individual inequality assertion of the situation mapping type IEA1.

|  |
| --- |
| <mapToDifferentIndvs equlityMappingType="distinct">         <referenceToOWLIndividual refName="MapToNewProduct"></referenceToOWLIndividual> </mapToDifferentIndvs> |

Listing : Example individual inequality assertion

# Conclusion

In this paper, we presented a systematic approach for mapping XML documents to OWL ontologies. We have identified several limitations of previous approaches and addressed them with different mapping definition types and their specifications. Based on the different possible mapping requirements, we have defined mapping situation types and specified mapping definition types with an XML Schema in an implementation independent way. On the one hand, this allows for standardized mapping support of a large set of potential requirements, and on the other hand it allows for incorporation of implementation details specific for the target business system. In our approach we have considered all OWL-ontology instance level concepts that can be a result of XML to OWL mapping and supported them with the proposed mapping definition types. The mapping definition specification is structured in a way that supports possible extensions for other situations that may be relevant in different contexts.

We have used our approach in two real world projects. In both projects the XML-to-OWL approach has been demonstrated as very useful, because it enabled up-to-date information in the ontology and a higher level of information reuse. With this, it not only improved the role of the ontology-based knowledge bases of the two companies, but actually enabled the fulfilment of their primary functions, such as decision support. Before the implementation of the automatic XML-to-OWL mappings, the instance-level of their ontology was very difficult to maintain up-to-date in a consistent way, and the ontology was not reliable. On the other hand a lot of XML data was processed in the companies every day, especially through business processes automation. These data carried information, which was not used in the ontology, regardless that the ontology existed and that the information was relevant for the ontology. It was not reused to the extent to which it could be. With the implementation of our approach, a significant part of this information was mapped to the ontology. In this way, the instance-level ontology development and evolution that in the scope of XML-to-OWL mapping was implemented as a by-product of business process execution. Data reuse was improved and maintaining up-to-date, accurate and complete information in the ontology was considerably facilitated. With this, our approach represents an important step forward to achieving the goals of enabling a higher level of data reuse, of improving the process of ontology development and evolution, information sharing, more complete description of ontology domain, better inference and decision support.

**References**

[1] T.R. Gruber, “Toward principles for the design of ontologies used for knowledge sharing,” *Int. J. Hum.-Comput. Stud.*, vol. 43, 1995, pp. 907-928.

[2] L.F. Lai, “A knowledge engineering approach to knowledge management,” *Information Sciences*, vol. 177, Oct. 2007, pp. 4072-4094.

[3] A. Maedche, B. Motik, L. Stojanovic, R. Studer, and R. Volz, “Ontologies for Enterprise Knowledge Management,” *IEEE Intelligent Systems*, vol. 18, 2003, pp. 26-33.

[4] A. Sasa, M.B. Juric, and M. Krisper, “Service-Oriented Framework for Human Task Support and Automation,” *IEEE Transactions on Industrial Informatics*, vol. 4, 2008, pp. 292-302.

[5] S. Staab, R. Studer, H. Schnurr, and Y. Sure, “Knowledge processes and ontologies,” *IEEE Intelligent Systems*, vol. 16, 2001, pp. 26-34.

[6] M. Uschold and M. Gruninger, “Ontologies: Principles, methods and applications,” *Knowledge Engineering Review*, vol. 11, 1996, pp. 93--136.

[7] C. Lee, Y. Kao, Y. Kuo, and M. Wang, “Automated ontology construction for unstructured text documents,” *Data & Knowledge Engineering*, vol. 60, Mar. 2007, pp. 547-566.

[8] T. Rodrugues, P. Rosa, and J. Cardoso, “Moving from syntactic to semantic organizations using JXML2OWL,” *Computers in Industry*, vol. 59, Oct. 2008, pp. 808-819.

[9] K. Yang, A. Lo, and R. Steele, “Ontology mediated XML data translation,” *International Journal of Web Information Systems*, vol. 4, 2008, pp. 181 - 197.

[10] M. Ferdinand, C. Zirpins, and D. Trastour, “Lifting XML Schema to OWL,” *Web Engineering*, 2004, pp. 776-777.

[11] R. García, F. Perdrix, R. Gil, and M. Oliva, “The semantic web as a newspaper media convergence facilitator,” *Web Semant.*, vol. 6, 2008, pp. 151-161.

[12] H. Bohring and S. Auer, “Mapping XML to OWL Ontologies,” Lecture Notes in Informatics (LNI), 2005.

[13] N. Anicic, N. Ivezic, and Z. Marjanovic, “Mapping XML Schema to OWL,” *Enterprise Interoperability New Challenges and Approaches*, G. Doumeingts, J. Müller, G. Morel, and B. Vallespir, Eds., Springer London, 2007, pp. 243-252.

[14] C. Cruz and C. Nicolle, “Ontology Enrichment and Automatic Population From XML Data,” *Proceedings of ODBIS 2008*, Auckland, New Zealand: , pp. 17-20.

[15] L. Zhang and J. Gu, “Ontology based semantic mapping architecture,” *Proceedings of 2005 International Conference on Machine Learning and Cybernetics*, 2005, pp. 2200-2205 Vol. 4.

[16] D. Lonsdale, D.W. Embley, Y. Ding, L. Xu, and M. Hepp, “Reusing ontologies and language components for ontology generation,” *Data & Knowledge Engineering*, vol. 69, Apr. 2010, pp. 318-330.

[17] Y. Sure and R. Studer, “Semantic Web technologies for digital libraries,” *Library Management*, vol. 26, 2005, pp. 190 - 195.

[18] P.P. Chen, “XML and the Semantic Web: What is the future?,” *Proceedings of 36th Hawaii International Conference on System Sciences*, HI, USA: IEEE Computer Society, 2003.

[19] T.R. Gruber, “A translation approach to portable ontology specifications,” *Knowl. Acquis.*, vol. 5, 1993, pp. 199-220.

[20] M.R. Genesereth and R.E. Fikes, *Knowledge Interchange Format, Version 3.0 Reference Manual*, Computer Science Department, Standford University, 1992.

[21] J.F. Sowa, “Conceptual Graphs,” *Handbook of Knowledge Representation*, F. van Harmelen, V. Lifschitz, and B. Porter, Eds., Elsevier, 2008, pp. 213-237.

[22] I. Horrocks, D.L. McGuinness, and C. Welty, “Digital Libraries and Web-Based Information Systems,” *The Description Logic Handbook: Theory, Implementation and Applications*, F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, Eds., Cambridge, UK: Cambridge University Press, 2003.

[23] W3C, “OWL Web Ontology Language Overview,” *http://www.w3.org/TR/owl-features/*, 2004.

[24] W3C, “OWL 1.1 Web Ontology Language Structural Specification and Functional-Style Syntax,” *http://www.w3.org/Submission/owl11-owl\_specification/*, 2006.

[25] W3C, “OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax,” *http://www.w3.org/TR/owl2-syntax/*, 2009.

[26] G. Schreiber, H. Akkermans, A. Anjewierden, R.D. Hoog, N. Shadbolt, W.V.D. Velde, and B. Wielinga, *Knowledge Engineering and Management: The CommonKADS Methodology*, The MIT Press, 1999.

[27] Y. Sure and R. Studer, *On-To-Knowledge Methodology - Final Version*, EU-IST Project IST-1999-10132 On-To-Knowledge, 2002.

[28] M. Fernandez-Lopez, A. Gomez-Perez, and N. Juristo, “METHONTOLOGY: from Ontological Art towards Ontological Engineering,” *Proceedings of the AAAI97 Spring Symposium*, 1997, pp. 40, 33.

[29] C. Tempich, H.S. Pinto, Y. Sure, and S. Staab, “An Argumentation Ontology for DIstributed, Loosely-controlled and evolvInG Engineering processes of oNTologies (DILIGENT),” *Proc. Second European Semantic Web Conference, ESWC 2005*, Heraklion, Crete, Greece: Springer, 2005, pp. 241-256.

[30] M. Cristani and R. Cuel, “A survey on ontology creation methodologies,” *International Journal on Semantic Web & Information Systems*, vol. 1, 2005, pp. 69, 49.

[31] D. Kensche, C. Quix, X. Li, Y. Li, and M. Jarke, “Generic schema mappings for composition and query answering,” *Data & Knowledge Engineering*, vol. 68, Jul. 2009, pp. 599-621.

[32] W3C, “XSL Transformations (XSLT),” *http://www.w3.org/TR/xslt*, 1999.

[33] T. Rodrigues, P. Rosa, and J. Cardoso, “Mapping XML to existing OWL ontologies,” *Proceeding of International Conference WWW/Internet 2006*, Murcia, Spain: 2006, pp. 72-77.

[34] W3C, “XML Path Language (XPath) 2.0,” *http://www.w3.org/TR/xpath20/*, 2007.

[35] W3C, “W3C XML Schema,” *http://www.w3.org/XML/Schema*.

[36] OASIS, “Web Services Business Process Execution Language Version 2.0, OASIS Standard,” *http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html*, 2007.

[37] A. Sasa and M. Krisper, “Knowledge Management in Service-Oriented Systems,” *Information Modelling and Knowledge Bases XXI, 19th European-Japanese Conference on Information Modelling and Knowledge Bases (EJC 2009)*, Yasushi Kiyoki, T. Tokuda, H. Jaakkola, X. Chen, and N. Yoshida, Eds., IOS Press, 2009.

1. Corresponding author: (Tel) +386 1 4768 704, (Fax) +386 1 4768 198, (Email) ana.sasa@@fri.uni-lj.si [↑](#footnote-ref-2)
2. A new individual or an assertion in the ontology is a result of the mapping. If it is actually added to the ontology and what additional actions are taken due to this depends on the chosen ontology development methodology, e.g. how to proceed if there is a contradictory assertion in the ontology. It is important that consistency is maintained. [↑](#footnote-ref-3)