Using semantic web technologies for analysis and validation of structural markup

Angelo Di Iorio, Silvio Peroni* and Fabio Vitali

Department of Computer Science,
University of Bologna,
Mura Anteo Zamboni, 7, 40127 Bologna, Italy
E-mail: diiorio@cs.unibo.it
E-mail: speroni@cs.unibo.it
E-mail: fabio@cs.unibo.it
*Corresponding author

Abstract: An increasing part of research in the Semantic Web has been directed at making data become the main concept of the web. Plenty of languages and specifications support this transition and work by inserting additional (semantic) markup into web documents. Yet, little attention is being paid to the possibility of expressing the actual structures of the documents in a form suitable for the semantic web. EARMARK is a model for explicitly expressing structural assertions of markup and documents, allowing a straightforward integration of the semantics of the markup and the semantics of the content. The well-formedness of a hierarchy, for instance, becomes an explicit assertion and similarly the analysis of the validity of markup structures become matter for further semantic analysis. This paper describes EARMARK and shows a framework for using OWL ontologies, that implement particular markup properties, to demonstrate the compliance of EARMARK documents with those properties.

Keywords: document structure; extremely annotational RDF markup; EARMARK; OWL; overlapping; structural markup ontology; structural pattern for documents; validation.

Reference to this paper should be made as follows: Di Iorio, A., Peroni, S. and Vitali, F. (2011) 'Using semantic web technologies for analysis and validation of structural markup', *Int. J. Web Engineering and Technology*, Vol. 6, No. 4, pp.375–398.

Biographical notes: Angelo Di Iorio holds a PhD in Computer Science, from the University of Bologna. His thesis is positioned over markup languages and document engineering areas, being focused on design patterns for digital documents and automatic processes of analysis and segmentation. During his PhD, he has also worked on collaborative authoring, document versioning, content formatting, and semantic web technologies. His research interests have recently extended towards layout languages and algorithms. He is a member of the W3C XSL-FO working group, and the author of several conference and journal papers on markup languages, digital publishing and web technologies.

Silvio Peroni holds a degree in Computer Science at the University of Bologna. The main research interests in his current PhD career include semantic web technologies, markup languages for complex documents, design patterns for digital documents and ontology modelling, and automatic processes of analysis and segmentation.

Fabio Vitali is an Associate Professor in Computer Science at the University of Bologna, where he teaches web technologies and human-computer interaction. His interest lies in models and languages for document management and hypertext support, and has published more than 60 papers in national and international venues. He is a member of the W3C Working Group on XML schema, and member of the scientific committee of several conferences and journals in web engineering and technologies. He is the author of important standards in the legislative XML domain, and work on issues related to digital publishing, web technologies and semantic web technologies.

1 Introduction

Broadly speaking, *markup* is anything added to a text content with the intention to say something about it. Formerly, this definition has been applied on all those markup languages that used (and still use) to describe the syntactical structure of text contents: LaTeX, DocBook, Office OpenXML, OpenDocument, and HTML are languages that fit the above mentioned definition, since they allow to describe the structures – in some cases, associating them to a bit of presentational behaviour – of text.

One of the most important issues addressed by this kind of *structural* markup concerns the possibility to express and verify specific syntactical properties:

- The *well-formedness*, that depends to the syntax specified for the particular markup language considered – e.g., in XML-based languages the begin and end tags which delimit elements must correctly nest without overlapping.
- The *validity* against a vocabulary, that (formally) restricts the set of values we can use for naming elements and attributes – definable by using schema languages such as XML schema and RelaxNG for XML.
- The *validity* against a *content model*, that defines contexts in which a particular named element can or cannot be[1] – definable by using schema languages as well. Particularly interesting in this context is the validity of a document against a set of *patterns*, i.e., a set of recurring rules and content models.

By electing data, instead of documents, as the main concept of the web, the semantic web brought with it the will to mark up not only text content in a structural way, as before, but to mark up any kind of resource in a *semantic* way: we have attended to a shift of interest from a syntactic (HTML and the like) to a semantic level (RDF and OWL).

Similarly to the structural markup domain, in which we want to declare particular syntactical properties for it, here, in the *semantic markup*, we want to model these data around some particular needs, sometimes constraining their semantics around some limits by using proper tools, such as ontologies.

Everybody is now participating, as active or passive user, to this transition from a syntactic paradigm, in which documents are the subject of the discourse, to a semantic one, using data as main concept. Obviously, this transition has been possible also by means of a number of hybrid languages, such as RDFa (Adida et al., 2008), that use structural markup as the basis to build semantic assertions on text and resources in general.

Much research has been done to enrich text content with semantic assertions about it but, strangely enough, little attention is being paid to the possibility of describing the structures of documents in a form suitable for the semantic web, even when there are opportunities in bringing together the structural and the semantical domains.

With these ideas in mind, we developed *extremely annotational RDF markup* (*EARMARK*) (Peroni and Vitali, 2009) as a model for explicitly expressing structural assertions of markup onto documents, allowing a straightforward integration of the *semantics of the markup* (e.g., what is the meaning of all those markup elements *p* contained in a document *d*?) and the *semantics of the content* of a document (e.g., the string 'Fabio' represents a particular person having that name) (Renear et al., 2002). EARMARK can be used for fine tuning exactly the set of statements we need to assert on the content of a document and on its structures, and for verifying exactly the constraints we need to verify, and not others implied by the syntax we used.

In previous works, we compared EARMARK with other markup models (Di Iorio et al., 2009) – GODDAG (Sperberg-McQueen and Huitfeldt, 2004) – and languages – TeXMECS (Huitfeldt and Sperberg-McQueen, 2003), LMNL (Tennison and Piez, 2002), TEI (TEI Consortium, 2005) – and we shown (Di Iorio et al., 2010) how to use it for handling complex and very frequent scenarios of overlapping markup concerning Microsoft Word (JTC1/SC34 WG 4, 2008) and Open Office (JTC1/SC34 WG 6, 2006) XML formats. Although in XML the only way to encode overlaps is to use workarounds and to hide overlapping hierarchies within the main tree-based structure of a document; EARMARK is not obliged to depend on such (i.e., *XML well-formedness*) syntactic constraints and allows users to verify them only if needed. Different structures can be in fact encoded as different sets of RDF/OWL statements on the top of the same content. They are given the same relevance and they can be accessed separately.

In this article, we introduce the EARMARK model in detail and discuss how most correctness properties typical of the structural markup, such as the *validity* against a schema, can be expressed through OWL ontologies and verified upon EARMARK documents at a semantic level by means of reasoners, such as Pellet (http://clarkparsia.com/pellet), even when dealing with non-hierarchical or multi-hierarchical structures. In order to support this claim, we also introduce two different running examples, the former based on a simple markup schema and the latter on a specific meta-level theory for document structures based on structural patterns (Dattolo et al., 2007).

The paper is thus structured as follows: in Section 2, we introduce EARMARK and its core ontology. In Section 3, the general issue of using EARMARK for expressing and assessing properties over text content is introduced. In Section 4, as an example of properties to be assessed, patterns over content models are introduced in general and expressed in EARMARK terms. Section 5 contains a short but focussed overview of similar works dealing with using semantic web technologies for markup and validation, and in Section 6, we draw some lessons from our experience and detail further work we plan to perform on and with EARMARK.

2 An ontological approach for meta-markup languages: EARMARK

This section describes the model behind EARMARK (Peroni and Vitali, 2009; Di Iorio et al., 2009, 2010), *extreme annotational RDF markup*. The model itself is defined

through an OWL document (http://www.essepuntato.it/2008/12/earmark) specifying classes and relationships. We distinguish between *ghost classes* – that define the general model – and *shell classes* – that are actually used to create EARMARK instances.

2.1 Ghost classes

The ghost classes describe three disjoint base concepts – *docuverses*, *ranges* and *markup items* – through three different and disjoint OWL classes.[2]

The textual content of an EARMARK document is conceptually separated from its annotations, and is referred to through the *Docuverse* class.[3] The individuals of this class represent the object of discourse, i.e., all the containers of text of an EARMARK document.

```
Class: Docuverse

DatatypeProperty: hasContent
   Characteristics: FunctionalProperty
   Domain: Docuverse Range: rdfs:Literal
```

Any individual of the *Docuverse* class – commonly called a *docuverse* (lowercase to distinguish it from the class) – specifies its actual content with the property *hasContent*.

We then define the class *range* for any text lying between two locations of a docuverse. A *range*, i.e., an individual of the class *range*, is defined by a starting and an ending location (any literal) of a specific docuverse through the properties *begins*, *ends* and *refersTo*, respectively.

```
Class: Range
   EquivalentTo:
      refersTo some Docuverse and
      begins some rdfs:Literal and
      ends some rdfs:Literal

ObjectProperty: refersTo
   Characteristics: FunctionalProperty
   Domain: Range Range: Docuverse

DatatypeProperty: begins
   Characteristics: FunctionalProperty
   Domain: Range Range: rdfs:Literal

DatatypeProperty: ends
   Characteristics: FunctionalProperty
   Domain: Range Range: rdfs:Literal
```

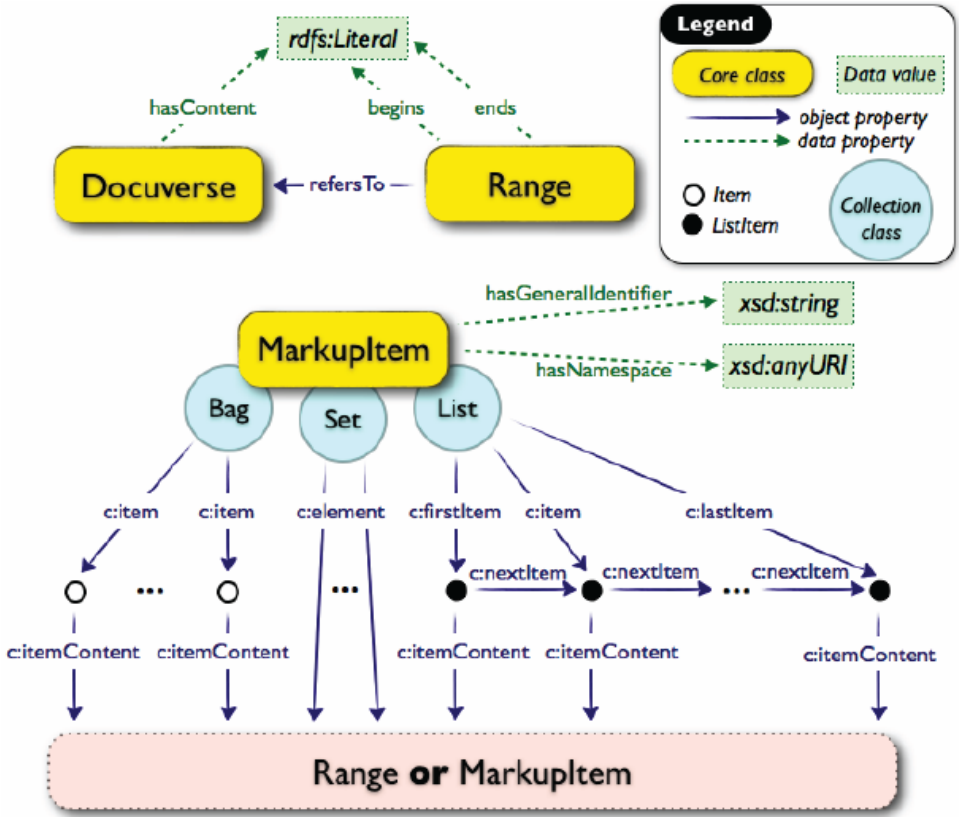
There is no restriction on locations used for the *begins* and *ends* properties. That is very useful: it allows us to define ranges that *follow* or *reverse* the text order of the docuverse they refer to. For instance, the string 'desserts' can be considered both in document order, with the *begins* location lower than the *ends* location or in the opposite one, forming 'stressed' (http://en.wikipedia.org/wiki/Palindrome#Semordnilaps). Thus, the value of properties *begins* and *ends* define the way a range must be read.

The class *MarkupItem* is the superclass defining artefacts to be interpreted as markup (such as elements and attributes).

```
Class: MarkupItem
   SubClassOf:
      (c:Set that c:element only (Range or MarkupItem)) or
      (c:Bag that c:item only
         (c:itemContent only (Range or MarkupItem))

DatatypeProperty: hasGeneralIdentifier
   Characteristics: FunctionalProperty
   Domain: MarkupItem Range: xsd:string

DatatypeProperty: hasNamespace
   Characteristics: FunctionalProperty
   Domain: MarkupItem Range: xsd:anyURI

Class: c:Collection
Class: c:Set SubClassOf: c:Collection
Class: c:Bag SubClassOf: c:Collection
Class: c:List SubClassOf: c:Bag
Class: c:Item
Class: c:ListItem SubClassOf: c:Item

ObjectProperty: c:element
   Domain: c:Collection

ObjectProperty: c:item SubPropertyOf: c:element
   Domain: c:Bag Range: c:Item

ObjectProperty: c:firstItem SubPropertyOf: c:item
   Domain: c:List

ObjectProperty: c:itemContent
   Characteristics: FunctionalProperty
   Domain: c:Item Range: not c:Item
```

```
ObjectProperty: c:nextItem
    Characteristics: FunctionalProperty
    Domain: c:ListItem Range: c:ListItem
```

Figure 1 A graphical representation of the EARMARK ontology (see online version for colours)



A *markupitem* individual is a collection (`c:Set`, `c:Bag` or `c:List`, where the latter is a subclass of the second one and all of them are subclasses of `c:Collection`) of individuals belonging to the classes *MarkupItem* and *Range*. Through these collections it is possible:

- to define a markup item as a set of other markup items and ranges by using the property *element*
- to define a markup item as a bag of items (defined by individuals belonging to the class `c:Item`), each of them containing a markup item or a range, by using the properties *item* and *itemContent* respectively
- to define a markup item as a list of items (defined by individuals belonging to the class `c:ListItem`), each of them containing a markup item or a range, in which we can also specify a particular order among the items themselves by using the property *nextItem*.

Thus, it becomes possible to define elements containing nested elements or text, or attributes containing values, as well as overlapping and complex structures. Note also that handling collections directly in OWL allows us to reason about content models for markup items, which would not be possible if we had used the corresponding constructs in RDF.[4]

A *markupitem* might also have a name, specified in the functional property *hasGeneralIdentifier* [recalling the SGML term to refer to the name of elements (Goldfarb, 1990)], and a namespace specified using the functional property *hasNamespace*. Note that we can have *anonymous* markup items – as it is possible in LMNL (Tennison and Piez, 2002) and GODDAG (Sperberg-McQueen and Huitfeldt, 2004) – by simply asserting that the item belongs to the class of all those markupitems that do not have a general identifier (i.e., `hasGeneralIdentifier exactly 0`).

A graphical summarisation of the ghost classes and their relations is shown in Figure 1.

2.2 Shell classes

The ghost classes discussed so far give us an abstract picture of the EARMARK framework. We need to specialise our model, defining a concrete description of our classes. These new *shell* subclasses apply specific restrictions to the *ghost* classes.

First of all, the class *Docuverse* is restricted to be either a *StringDocuverse* (the content is specified by a string) or an *URIDocuverse* (the actual content is located at the URI specified).

```
Class: StringDocuverse
    SubClassOf:
       Docuverse,
       hasContent some xsd:string
    DisjointWith: URIDocuverse

Class: URIDocuverse
    SubClassOf:
       Docuverse ,
       hasContent some xsd:anyURI
```

Depending on particular scenarios or on the kind of docuverse we are dealing with – it may be plain-text, XML, LaTeX, a picture, etc. – we need to use different kinds of ranges. Therefore, the class *Range* has three different subclasses:

- *PointerRange* defines a range by counting characters. In that case, the value of the properties *begins* and *ends* must be a non-negative integer that identifies unambiguous positions in the character stream, remembering that the value 0 refers to the location immediately before the 1st character, the value 1 refers to the location after the 1st character and before the 2nd one, and so on. By using the *hasKey* OWL property, we also assert that two pointer ranges having equal docuverse, begin and end locations are the same range.

- *XPathRange* defines a range considering the whole docuverse or its particular context specifiable through an XPath expression (Berglund et al., 2007) as value of the property *hasXPathContext*. Note that, by using these ranges, we implicitly admit that the docuverse it refers to must be an XML structure. Moreover, the properties *begins* and *ends* have to be applied on the string value obtained by juxtaposing all the text nodes identified by the XPath. By using the *hasKey* OWL property, we also assert that two XPath ranges having equal docuverse, XPath context, begin and end locations are the same range.
- *XPathPointerRange* is an *XPathRange* in which the value of the properties begins and ends must be a non-negative integer.

```
Class: PointerRange
   SubClassOf:
      Range,
      begins some xsd:nonNegativeInteger and
      ends some xsd:nonNegativeInteger
   HasKey: refersTo begins ends

Class: XPathRange SubClassOf: Range
   EquivalentTo: hasXPathContext some rdfs:Literal
   HasKey: refersTo begins ends hasXPathContext

Class: XPathPointerRange
   SubClassOf:
      XPathRange,
      begins some xsd:nonNegativeInteger and
      ends some xsd:nonNegativeInteger

DatatypeProperty: hasXPathContext
   Characteristics: FunctionalProperty
   Domain: XPathRange Range: rdfs:Literal
```

MarkupItem is specialised in three disjointed sub-classes: *element*, *attribute* and *comment*, that allow a more precise characterisation of markup items.

```
Class: Element SubClassOf: MarkupItem
Class: Attribute SubClassOf: MarkupItem
Class: Comment SubClassOf: MarkupItem

DisjointedClasses: Element, Attribute, Comment
```

In order to understand how EARMARK is used to describe markup hierarchies, let us to introduce an XML excerpt, using TEI fragmentation (TEI Consortium, 2005) to express overlapping elements upon the string ‘Fabio says that overlhappens’:

```
<phrase>
   Fabio says that
   <noun xml:id="e1" next="e2">overl </noun>
   <verb>h<noun xml:id="e2">ap<noun>pens </verb>
</phrase>
```

Here, the two elements *noun* represent the same element fragmented and overlapping with part of the textual content of *verb*, i.e., the characters 'ap'. The EARMARK translation of it is the following:

```
@prefix ex: <http:// www.example.com/>.
ex:doc :hasContent "Fabio says that overlhappens".
ex:r0-16 a : PointerRange; :refersTo ex:doc
   ; :begins "0"^^xsd:integer; :ends "16"^^xsd:integer.
ex:r16-21 a : PointerRange; :refersTo ex:doc
   ; :begins "16"^^xsd:integer; :ends "21"^^xsd:integer.
ex:r21-28 a : PointerRange; :refersTo ex:doc
   ; :begins "21"^^xsd:integer; :ends "28"^^xsd:integer.
ex:r22-24 a : PointerRange; :refersTo ex:doc
   ; :begins "22"^^xsd:integer; :ends "24"^^xsd:integer.
ex:phrase a :Element; : hasGeneralIdentifier "phrase"
   ; c:firstItem [c:itemContent ex:r0-16
   ; c:nextItem [c:itemContent ex:noun
   ; c:nextItem [c:itemContent ex:verb]]].
ex:noun a :Element; : hasGeneralIdentifier "noun"
   ; c:firstItem [c:itemContent ex:r16-21
   ; c:nextItem [c:itemContent ex:r22-24]].
ex:verb a :Element; : hasGeneralIdentifier "verb"
   ; c:firstItem [c:itemContent ex:r21-28].
```

2.3 The EARMARK framework and API

We have already designed and implemented a framework for the creation, validation and manipulation of EARMARK data structures. The API is hosted by SourceForge (http://earmark.sourceforge.net) under the Apache 2.0 license and fully implements the current EARMARK model.

All the code is written in Java and uses well-known libraries in the semantic web community such as Jena (http://jena.sourceforge.net). The implementation of the data structure follows exactly what is defined in the EARMARK ontology and uses ghost and shell classes, encoding OWL properties as methods of these classes.

The Java classes *EARMARKDocument*, *Range* and *MarkupItem* have been written taking into consideration a particular interface, called *EARMARKNode*, directly derived from the *Node* interface of the Java DOM implementation (http://java.sun.com/xml). This

choice has been made to maintain the EARMARK data structure as close as possible to a well-known and used model for XML documents.

The API makes it simple to create/load/store/modify EARMARK documents directly in a Java framework. Let us take again into consideration the example introduced in Section 2.2. In the followings excerpts, we illustrate how to build that document using the API.

Let us start creating a new EARMARK document and a docuverse, containing all the text content of our document:

```
EARMARKDocument ed=
   new EARMARKDocument (URI.create (“http://www.example.com”));

String ex = “http://www.example.com/”;

Docuverse doc = ed.createStringDocuverse(ex+”doc”,
   “Fabio says that overlhappens”);
```

The next excerpt shows how to create ranges starting from the above docuverse:

```
Range r0_16 = ed.createPointerRange(ex+”r0_16”, doc, 0, 16);
```

Finally, we define all the markup items we need to build the structure of the document. Usually, a markup item needs three different values in order to be created: a string representing its general identifier, an identifier for the item and the type for the collection it defines (either set, bag or list). In the following excerpt, we show the creation of three different elements, composed in order to define hierarchical relations among them by using the method *appendChild*:

```
Element phrase = ed.createElement(
   “phrase”, ex+”phrase”, Collection.Type.List);
ed.appendChild(phrase); phrase. appendChild(r0_16);

Element noun = ed.createElement(
   “noun”, ex+”noun”, Collection.Type.List);
phrase.appendChild(noun);

Element verb = ed.createElement(
   “verb”, ex+”verb”, Collection.Type.List);
phrase.appendChild(verb);

...
```

As seen, the API allows to create EARMARK structures with very simple and straightforward methods. Even rather complicated structures can be created with a few lines of Java code.

3 Assessing properties on EARMARK documents

In previous works, we discussed the expressivity of EARMARK (Peroni and Vitali, 2009) (Di Iorio et al., 2009) and we shown possible applications of it in different contexts, such as scholarly disciplines (e.g., linguistics) and mainstream applications (e.g., word processors) (Di Iorio et al., 2010).

In this paper, we focus on another important advantage of using EARMARK, i.e., the possibility of expressing structural assertions of markup, allowing:

- A straightforward integration of the *semantics of the markup* (e.g., what is the meaning of all those markup elements *p* contained in a document *d*?) and the *semantics of the content* of a web document (e.g., the string 'Fabio' of the example in Section 2.2, identifiable through a range, represents a particular person having that name) (Renear et al., 2002).
- To define and assess, in a semantic way, all those typical properties of the structural markup, such as validity. Moreover, by means of the intrinsic nature of EARMARK, we can easily assess these structural properties even in presence of very complex overlapping hierarchies.

The assessment of ontological properties in the semantic domain is in a way comparable and can be made to act as validation in the XML domain that is the process of verification of whether relevant markup items of a well-formed XML document satisfy the constraints embodied in the relevant components of a schema. In the world of XML, several schema languages have been introduced such as DTD, XML schema and RelaxNG. They have different expressive power but they share the same objectives and basic principles: the validator checks the structural properties of a well-formed document by verifying all the *syntactical* constraints expressed in the schema.

Moving from a syntactical perspective to a *semantical* one – as proposed by EARMARK – opens new perspectives for a general approach to assessment as well. A key point of such approach is the translation of many markup properties from a syntactical to an ontological level. In the case of XML schema validation, for instance, this means expressing:

a schema definitions as ontology classes and properties

b schema documents as ontology instances and assertions, that express hierarchies as semantic relations.

Starting from an ontological *TBox* representing the schema and an *ABox* representing the document, we can then conclude that the document is valid according to the schema if and only if the *ABox* is consistent with the *TBox*.

Our goal is to design a framework and to implement tools that verify if an EARMARK document is compliant to any property *P* over its syntax, structure and semantics. The same approach can thus be used for common validation as well as for all the other above-mentioned constraints we want to verify on our documents. Our approach – that is meant to be instantiated for each specific case – can be summarised as follows:

1 define an ontology fragment *O* that details the particular property *P* we want to verify

2 associate the EARMARK document instances to *O*, so as to obtain an ABox for *O*

3 use a reasoner to prove whether the ABox is consistent (i.e., that property *P* is held) or not (and *P* is not held).

3.1 Defining content-models on EARMARK documents

In order to illustrate how to assess properties on EARMARK documents defining specific ontologies, we first take into consideration simple syntactical property definitions, such as those specifiable for the content models of markup items using schema languages such as DTD, XML schema or RelaxNG.

For instance, let us consider the following seven sentences informal description of a schema:

1 it is only possible to use elements (e.g., no attributes are allowed anywhere)

2 no element is associated to a namespace

3 all elements must specify a general identifier

4 each element contains the other nodes in a specific order

5 no element with a general identifier different from ‘phrase’, ‘noun’ and ‘verb’ can be used

6 each element with general identifier ‘phrase’ can contain, in any order, only an unlimited number of elements with general identifier ‘noun’ or ‘verb’, and cannot contain text

7 each element with general identifier ‘noun’ or ‘verb’ can contain only text and no other elements.

Using a RelaxNG Compact-like syntax (Clark, 2002), opportunely extended to allow us to define more than one root element for our documents in the structure start (when applied to XML documents, RelaxNG only allows to define one root), the previous informal schema may be formally expressed as follows:

```
start = (e.phrase | e.noun | e.verb)*
e.phrase = element phrase {(e.noun | e.verb)*}
e.noun = element noun {text}
e.verb = element verb {text}
```

As we have previously introduced, in order to understand whether an EARMARK document is written according to the previous sentences and, consequently, to the previous schema, we have to develop an OWL ontology that implements them. In this case, we can obtain implicit associations between EARMARK document instances and the above property constraints to assess by extending the EARMARK ontology itself.

First of all, we can limit the use of elements (sentence 1) simply defining all markup items equivalent to elements only, as shown in the following excerpt:

```
Class: MarkupItem
    EquivalentTo: Element and not(Attribute or Comment)
```

This sentence also results in asserting, by inference, that *attribute* and *comment* are subclass of the OWL class *nothing*, which represents the empty set: no individuals can belong to it.

Then, we express sentences 2 to 4 by adding three subclass relations to the element class:

```
Class: Element
   SubClassOf:
      hasNamespace exactly 0, c:List,
      hasGeneralIdentifier some xsd:string
```

To express sentences 5 to 7, we create a new object property to describe the parent-child relations among EARMARK nodes (i.e., markup items and ranges):

```
ObjectProperty: hasChild
   SubPropertyChain:
      c:item o c:itemContent
```

Since it is defined by a property chain between the properties item and *itemContent*, we can say that if an element individual *e* has an item *i* that refers to a particular EARMARK node *m* – that is a typical parent-child relation concerning EARMARK elements expressed by bags or lists – then we can infer that *e* has *m* as child.

Then, using this property, we can easily cover the remaining sentences (4 to 7) by adding another subclass relation to the element class:

```
(hasGeneralIdentifier value 'phrase' and hasChild only
   (hasGeneralIdentifier some {'noun', 'verb'})) or
((hasGeneralIdentifier some {'noun' , 'verb'}) and
   hasChild only Range)
```

That is it: the above assertions are sufficient to assess whether an EARMARK document is valid against the schema presented.[5] For instance, when we try to verify, through a reasoner, whether the EARMARK document in Section 2.2 is consistent with the model introduced in this section, we will receive an inconsistency exception because the element *phrase* of the sample document contains free text where it is not allowed.[6]

Property definitions through OWL ontologies can be used to define schemas for EARMARK documents. Yet, rather than discussing validation, that is addressed in other works, such as Yang et al. (2007), Ferdinand et al. (2004) and Rodrigues et al. (2006), in the following sections, we will concentrate, as an extensive example, on the assessment of the properties connected to *structural patterns*, such as the ones discussed in Dattolo et al. (2007) and Di Iorio et al. (2005).

4 Using and verifying patterns in EARMARK documents

In this section, we elaborate on the topic of the validity of EARMARK documents against particular theories for document structures, introducing another, much more complex, example of other markup syntactic constraints that goes beyond the common markup content-model validation: the *structural patterns*. Then we verify these patterns on EARMARK documents with multiple hierarchies and overlapping elements.

Before showing the OWL implementation of these constraints and their use in EARMARK documents, we need to introduce the theory grounding these constraints themselves. Therefore, in Section 4.1, we present a meta-level theory for document structures based on specific patterns (Dattolo et al., 2007) and then, in Section 4.2, we implement it in an OWL ontology used for validating EARMARK documents against this theory.

4.1 Structural patterns

The idea of using patterns to produce reusable and high-quality assets is not new in the literature. Software engineers (Gamma et al., 1994), architects [as Alexander (1979) who first introduced this term] and designers very often use – or rather reuse – patterns to handle problems which recur over and over. Patterns have also been studied to modularise and customise web ontologies (Presutti and Gangemi, 2008). They guarantee the flexibility and maintainability of concepts and solutions in several heterogeneous scenarios.

We have been investigating patterns for XML documents for some time (Dattolo et al., 2007; Di Iorio et al., 2005). The overall goal of our research is to understand how the structure of digital documents can be segmented into atomic components that can be manipulated independently and re-flowed in different contexts. Instead of defining a large number of complex and diversified structures, we have identified a small number of *structures/patterns* that are sufficient to express what most users need. Our idea is that ten patterns, shown in Table 1, are enough to capture the most relevant document structures.

The two main characterising aspects of such pattern theory are:

- *Orthogonality* – each pattern has a specific goal and fits a specific context. The orthogonality between patterns makes it possible to associate a single pattern to each of the most common situations in document design. Then, whenever a designer has a particular need he/she has to only select the corresponding pattern and to apply it.
- *Assemblability* – each pattern can be used only in some locations (within other patterns). Although this may seem a limitation, such strictness improves the expressiveness and non-ambiguity of patterns. By limiting the possible choices, patterns prevent the creation of uncontrolled and misleading content structures. This characteristic still allows the presence of overlapping items – for example, a block that contains two different inlines that overlap upon the same segment continues to be a valid structure in terms of patterns because its content model is not violated, even though the presence of overlapping descendants.

These patterns allow authors to create unambiguous, manageable and well-structured documents. The regularity of pattern-based documents makes it possible to perform easily complex operations even when knowing very little about the documents' vocabulary. In fact, designers can implement more reliable and efficient tools, can make hypotheses regarding the meanings of document fragments, can identify singularities and can study global properties of sets of documents.

There are two main methods to check if (and how) a document uses patterns or can be normalised into a new pattern-based resource. A procedural approach requires *ad hoc* tools – written in a procedural programming language and running on a software platform – that are difficult to write, test, maintain, and extend. A declarative approach, on the other hand, guarantees more flexibility, extensibility and portability. The ontological model adopted by EARMARK is particularly suitable for such a context: verifying that a document meets the requirements given by patterns, in fact, only requires verification using a reasoner of some properties of an OWL ontology.

Table 1 A brief summary of the structural pattern theory of Dattolo et al. (2007) extended with two more patterns: *headed container* and *popup*

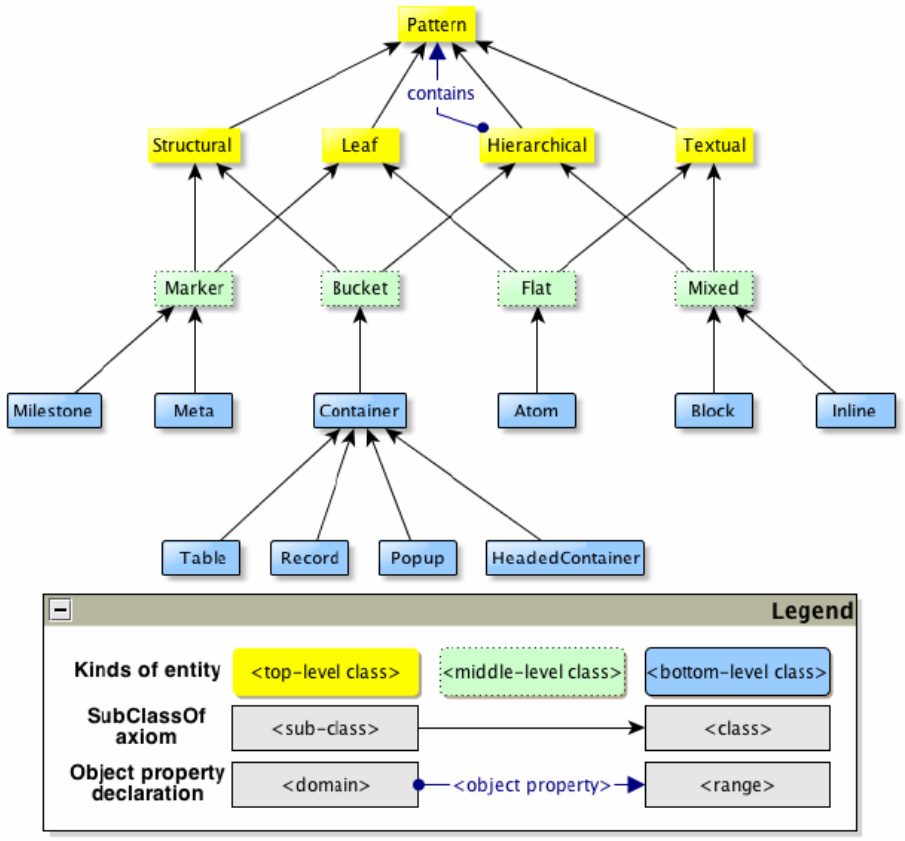
Pattern	*Description*	*Example (XHTML)*	*May contain*
Meta	An empty element whose meaning depends on its presence in a document (rather than on its position) and its attributes	meta	EMPTY
Milestone	An empty element whose meaning depends on its position	br	EMPTY
Atom	A unit of unstructured information	title	Text
Block	A block of text mixed with unordered and repeatable inline elements, with the same content model	p	Text, milestone, inline, popup
Inline	A block of text mixed with unordered and repeatable inline elements, with the same content model	i	Text, milestone, inline, popup
Container	A sequence of heterogeneous, unordered, optional and repeatable elements	body	Meta, atom, block, record, container, table
Headed container	A sequence of heterogeneous, unordered, optional and repeatable elements headed by a sequence of blocks	div that contains h1	Meta, atom, block, record, container, table
Record	A set of optional, heterogeneous and non-repeatable elements	html	Meta, atom, block, record, container, table
Table	A sequence of homogeneous elements	ul	Meta, atom, block, record, container, table
Popup	A sequence of heterogeneous, unordered, optional and repeatable elements	iframe	Meta, atom, block, record, container, table

In the next section, we will propose an example of this approach, made even more complex by the presence of an overlapping structure that makes property verification through XML technologies much more complex, and possibly even impossible.

4.2 Assessing structural patterns on EARMARK documents

The first step to verify patterns-related properties of EARMARK documents is obviously to build an ontology that describes such patterns and their relationships. We developed the ontology through a set of hierarchical class/sub-class relations, summarised in Figure 2.

Figure 2 The ontology describing the patterns theory (see online version for colours)



We have three levels of abstraction modelling different aspects of our theory:

- The *top-level* classes describe the general features of a pattern. In particular, they characterise their content model expressing the possibility for the element to contain text (*textual*) or not (*structural*) and to contain other elements (*hierarchical*) or not (*leaf*).
- The *middle-level* classes model all the combinations of top-level classes and express their disjointness.

- The *bottom-level* classes include all the 'instanceable' classes, i.e., all those classes which document elements can explicitly belong to.

This ontology allows us to verify whether or not an EARMARK document follows the structural patterns. In particular, it allows us to verify if a given association between elements and patterns (that assigns one pattern to each element) is valid. That, again, means checking whether the ontological association of each element to a particular pattern is consistent.

For assessing patterns on an EARMARK document *D*, we need to apply the following steps:

1 for each element in *D*, associate an instanceable pattern to the element

2 for each element in *D*, assert the element belongs to the class defined by the restriction `isContainedBy exactly 0 Pattern` whether it is not contained by any other element – where *isContainedBy* is the inverse property of property *contains* shown in Figure 2.

3 launch a reasoner to check if the pattern ontology with these added assertions is consistent (all the pattern constraints hold) or not (there are some errors when assigning patterns to elements).

Before presenting experimental results, let us show some excerpts from the pattern ontology developed, just to sketch out how we formally define those structural properties. First of all, we need to express containment relations among patterns through specific OWL object properties, as shown as follows.

```
ObjectProperty: contains
   Domain: Hierarchical Range: Pattern
ObjectProperty: isContainedBy
   InverseOf: contains
```

Expressing EARMARK markup item containments in terms of these two properties is straightforward by using particular rules to infer new assertions:

```
MarkupItem(x), c:Set(x), c:element(x,y), MarkupItem(y)
   -> contains(x,y)
MarkupItem(x), c:item(x,y), itemContent(y,z), MarkupItem(z)
   -> contains(x,z)
```

Taking into consideration the above properties and assertions, we illustrate, as example, the definition of *inline* and *block* patterns, discussing the main differences among them. We implement them through two OWL classes:

```
Class: Inline
   SubClassOf:
      Mixed and contains only
         (Inline or Milestone or Popup),
      isContainedBy some Block
```

```
Class: Block
   SubClassOf:
      Mixed and contains only
         (Inline or Milestone or Popup)
   DisjointWith: Inline
```

As shown, content models for these classes are defined by adding restrictions in form of superclasses, such as:

```
Mixed and contains only (Inline or Milestone or Popup)
```

All *Textual* individuals are inferred starting from markup items containing some ranges:

```
MarkupItem(x), c:Set(x), c:element(x,y), Range(y)
   -> Textual(x)
MarkupItem(x), c:item(x,y), c:itemContent(y,z), Range(z)
   -> Textual(x)
```

Although we cannot discuss any structural pattern implementation in this paper because of space limits, the entire pattern ontology definition is available online (http://www.essepuntato.it/2008/12/pattern) and uses SWRL (Horrocks et al., 2004) as the rule language.

4.3 Experimental results

In the following, we discuss an explicative example on the first three verses of the *Paradise Lost* by John Milton, which are often quoted as a standard example for enjambement[7] in poetry:

> Of Man's first disobedience, and the fruit
>
> Of that forbidden tree whose mortal taste
>
> Brought death into the World

We want to describe two different hierarchies, one for the verses and another one for the syntactical units giving rise to the enjambments, and we use an HTMLlike syntax for the general identifiers. Figure 3 shows the graph representation of a corresponding EARMARK document (http://www.essepuntato.it/2010/04/ParadiseLost).

Table 2 summarises our experiments running a reasoner to verify patterns on the previous example (http://www.essepuntato.it/2010/04/ParadiseLost/test). We tried eight different combinations of patterns and – for each combination – we checked whether it generated a consistent ontology.

The fourth column of the table shows the output of the reasoner. The answer 'yes' indicates that a specific combination does not violate the constraints of the relevant patterns. That combination is then valid and can be used for identifying structural roles of document's elements. Negative results are very relevant because the reasoner is able to identify that requirements are not respected, and why. For instance, the reasoner detects that an element *div* cannot be an inline, being the root of a hierarchy.

Figure 3 The EARMARK document, in the form of a graph, of the first three verses of the Paradise Lost by John Milton (see online version for colours)

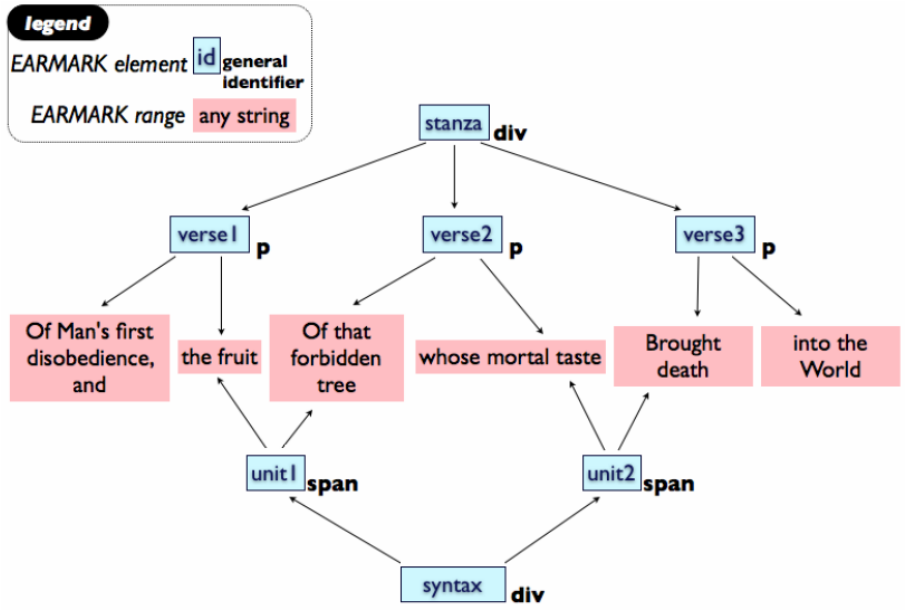


Table 2 Testing associations between elements and patterns on the 'Paradise Lost' example through an OWL reasoner

div	*p*	*span*	*Is the ontology still consistent?*
Container	Block	Atom	Yes
Container	Block	Inline	No: a container (syntax) cannot contain inline
Block	Block	Inline	No: a block (stanza) cannot contain at any level another block
Block	Inline	Inline	Yes
Inline	Inline	Inline	No: inlines (both stanza and syntax) cannot be the root of a hierarchy
Block	Inline	Inline for unit 1 and atom for unit 2	No: elements with the same general identifier (unit 1 and unit 2) must have the same pattern
Atom	Atom	Atom	No: atoms (both stanza and syntax) cannot contain other elements
Table	Atom	Atom	Yes
Record	Atom	Atom	No: records (stanza and syntax) can contain only elements with different general identifiers

Although this example is very simple (and focuses only on a few patterns), it should give readers the idea of how a reasoner and the ontological descriptions of a set of constraints can give validity results on rules expressed over an EARMARK document, and regardless of whether the document is in fact a single hierarchy (i.e., an XML document) or actually represents a multiplicity of hierarchies hard or impossible to express in XML. The fact that here we deal with patterns is just an example, and does not affect the

generality and applicability of this approach: additional classes and additional SWRL rules would allow us to test other properties of the same documents.

5 Related works

Three particular topics are relevant to what we examine in this work: the ontological declaration of markup on resources, the ontological representation of digital document properties such as schema validity and other non-ontological approaches for validating complex overlapping structures.

The first issue for ontological declaration of markup is to find a way to define locations and ranges on a particular text. A W3C activity (Iglesias and Squillace, 2009) proposes to address this issue. In this approach, the authors propose an ontology in which the various classes refer to different kinds of points that can be used to identify particular positions in a digital document, particularly XML and HTML structures. Through this ontology, we can define single points using one of several sub-classes of *SinglePoint* (e.g., *XPathPointer*, *ExpressionPointer*, *XPointerPointer*, etc.), and ranges using specific sub-classes of *CompoundPointer*.

After defining ranges of text content, the next step is to design a model that allows markup structures for the ranges. One of the best contributions in addressing that issue is presented in Tummarello et al. (2005). This study represents one of the first real inputs for our work. It offers an interesting way to encode text content into RDF resources – marking them up through a particular OWL ontology expressing the various parts of discourse – and allowing the simultaneous presence of multiple (and possibly overlapping) hierarchies. The result of this work is the development of a framework, called *RDF textual encoding framework* [or *RDFTEF* (http://rdftef.sourceforge.net)], that supports textual encodings for RDF and OWL, allowing expression of textual annotations in RDF/XML.

After generating a well-defined document structure mapped into ontological assertions, we would like to verify whether it satisfies some particular properties or not – for example, validation against a document schema. Document schemas can also be good starting points for developing or reengineering ontologies, as pointed out in Ferdinand et al. (2004). Proposing a translation mechanism from an XML document into a set of RDF assertions that describe it, the authors' main aim is to define a way to translate an XML schema document into an OWL ontology in order to obtain a complete TBox + ABox from a document with markup and its schema.

Similar to the previous work but following a different approach, (Rodrigues et al., 2006) introduces a framework for producing an ABox starting from an XML document, passing through an existing OWL ontology and the XML schema document used by the original document. The JXML2OWL (http://jxml2owl.projects.semwebcentral.org) framework thus generates such an Abox via some XSLT processing, made possible through handmade associations between the XML schema declarations and the OWL classes. The result of this process is a set of OWL instances related to the OWL ontology used for the transformation. Proving the consistency of these instances with respect to the ontology means verifying the validity of the original document against the referred schema.

Another study that works similarly is Yang et al. (2007). After introducing the main differences between XML schema and ontologies in general, the authors propose an

ontology-based approach for representing mappings between that document schema language and ontologies. The goals of this work are round-trip translations from an XML document into ontology instances and the realisation of an ontology-mediated transformation between different XML documents, in order to allow seamless data exchange between heterogeneous XML data source.

Considering non-ontological approaches, the *rabbit/duck grammars* mechanism (Sperberg-McQueen, 2006) is a good technique for the validation of documents with overlaps. The basic idea of this approach is simple: to define different document schemas whose intersection describes the markup language to be validated. In each schema, each markup element is associated to one out of four classes – *normal*, *milestones*, *transparent* and *opaque* – that define how elements have to be considered in the context of the schema taken into consideration. Given a complete set of rabbit/duck grammars, each element must belong to the class *normal* in at least one schema of the set. Then, a document is considered valid against the set of rabbit/duck grammars if and only if it is valid against each schema in the set.

6 Conclusions and future works

EARMARK is proving to be not just a mere exercise in substituting embedded markup with OWL assertions, but a tool to prove that any feature that was available in XML markup can still be kept with a semantic web mindset, yet without the limitations that embedding brings along, such as single hierarchy and forced containment. Being able to verify arbitrary rules on markup and content, and on multi-hierarchical documents seems a reasonably strong result so far.

Apart from that, a number of interesting applications can be understood better through EARMARK: from the problem, well-known in overlapping communities, of identifying enjambments (Di Iorio et al., 2009) in poems expressed in TEI (TEI Consortium, 2005), to a newer issue of correctly describing the versioning events of an office documents over which change tracking has been activated (Di Iorio et al., 2010), to the currently unexplored problem of purging malicious edits in a wiki page without affecting temporally subsequent, yet legitimate edits. In all these situations, the complex interweaving of annotations on content and annotations drawn from markup can be managed easily and fruitfully with the aid of the EARMARK ontology and model.

Of course, work on EARMARK is far from being finished. Embedding and disembedding tools need to be generated, so as to be able to convert XML documents back and forth to EARMARK, and simple mechanisms for turning schema documents into TBox ontologies for EARMARK are yet to be delivered. We will also explore more deeply all those languages, models and techniques that allow to formally specify, at the same time, different semantics for markup.

Acknowledgements

The authors wish to thank Michael Sperberg-McQueen for his deep and passionate remarks about EARMARK, and Federico Meschini for strongly supporting the ideas behind our project.

References

Adida, B., Birbeck, M., McCarron, S. and Pemberton, S. (2008) 'RDFa in XHTML: syntax and processing', W3C Recommendation, World Wide Web Consortium, 14 October, available at http://www.w3.org/TR/rdfa-syntax/ (accessed on 17 November 2010).

Alexander, C. (1979) *The Timeless Way of Building*, Oxford University Press, Oxford, UK.

Berglund, A., Boag, S., Chamberlin, D., Fernndez, M.F., Kay, M., Robie, J. and Siméon, J. (2007) 'XML path language (XPath) 2.0', W3C Recommendation, World Wide Web Consortium, 23 January, available at http://www.w3.org/TR/xpath20/ (accessed on 17 November 2010).

Bray, T., Paoli, J., Sperberg-McQueen, C.M., Maler, E., Yergeau, F. and Cowan, J. (2006) 'Extensible markup language (XML) 1.1 (Second Edition)', W3C Recommendation, World Wide Web Consortium, 16 August, available at http://www.w3.org/TR/xml11/ (accessed on 12 March 2011).

Ciccarese, P., Wu, E., Kinoshita, J., Wong, G., Ocana, M., Ruttenberg, A. and Clark, T. (2008) 'The SWAN biomedical discourse ontology', *Journal of Biomedical Informatics*, Vol. 41, No. 5, pp.739–751, Elsevier.

Clark, J. (2002) 'RELAX NG compact syntax', Committee Specification, Organization for the Advancement of Structured Information Standards, available at http://relaxng.org/compact-20021121.html (accessed on 17 November 2010).

Dattolo, A., Di Iorio, A., Duca, S., Feliziani, A.A. and Vitali, F. (2007) 'Structural patterns for descriptive documents', in *Proceedings of the 7th International Conference on Web Engineering 2007 (ICWE 2007)*, Como, Italy, Lecture Notes in Computer Science, 16–20 July, Vol. 4607, pp.421–426, Springer.

Di Iorio, A., Gubellini, D. and Vitali, F. (2005) 'Design patterns for document substructures', in *Proceedings the Extreme Markup Languages Conference 2005*, Montreal, Canada, 1–5 August, IDEAlliance, available at http://conferences.idealliance.org/extreme/html/2005/Vitali01/EML2005Vitali01.html (accessed on 17 November 2010).

Di Iorio, A., Peroni, S. and Vitali, F. (2009) 'Towards markup support for full GODDAGs and beyond: the EARMARK approach', in *Proceedings of Balisage: The Markup Conference 2009*, Montreal, Canada, 11–14 August, Balisage Series on Markup Technologies, Vol. 3, doi:10.4242/BalisageVol3.Peroni01.

Di Iorio, A., Peroni, S. and Vitali, F. (2010) 'Handling markup overlaps using OWL', in *Proceedings of the 17th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2010)*, Lisbon, Portugal, 11–15 October, Lecture Notes in Artificial Intelligence, Vol. 6317, pp.391–400, Springer.

Ferdinand, M., Zirpins, C. and Trastour, D. (2004) 'Lifting XML schema to OWL', in *Proceedings of the 4th International Conference on Web Engineering 2004 (ICWE 2004)*, Munich, Germany, 26–30 July, Lecture Notes in Computer Science, Vol. 3140, pp.776–777, Springer.

Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1994) *Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, Boston, Massachusetts, USA.

Goldfarb, C.F. (1990) *The SGML Handbook*, Oxford University Press, Oxford, UK.

Horridge, M. and Patel-Schneider, P. (2009) 'OWL 2 web ontology language: Manchester syntax', W3C Working Group Note, World Wide Web Consortium, 27 October, available at http://www.w3.org/TR/owl2-manchester-syntax/ (accessed on 17 November 2010).

Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B. and Dean, M. (2004) 'SWRL: a semantic web rule language combining OWL and RuleML', W3C Member Submission, World Wide Web Consortium, 21 May, available at http://www.w3.org/Submission/SWRL/ (accessed on 17 November 2010).

Huitfeldt, C. and Sperberg-McQueen, C.M. (2003) 'TexMECS: an experimental markup meta-language for complex documents', Working paper of the project Markup Languages for Complex Documents (MLCD), University of Bergen, January 2001, rev. October 2003, available at http://decentius.aksis.uib.no/mlcd/2003/Papers/texmecs.html (accessed on 17 November 2010).

Iglesias, C. and Squillace, M. (2009) 'Pointer methods in RDF 1.0', W3C Working Draft, World Wide Web Consortium, 29 October, available at http://www.w3.org/TR/Pointersin-RDF10/ (accessed on 17 November 2010).

JTC1/SC34 WG 4 (2008) 'ISO/IEC 29500-1:2008 – information technology – document description and processing languages – Office Open XML file formats – part 1: fundamentals and markup language reference'.

JTC1/SC34 WG 6 (2006) 'ISO/IEC 26300/2006 – information technology – open document format for Office applications (OpenDocument) v1.0'.

Nelson, T. (1980) *Literary Machines: The Report on, and of, Project Xanadu Concerning Word Processing, Electronic Publishing, Hypertext, Thinkertoys, Tomorrow's Intellectual... including Knowledge, Education and Freedom*, Mindful Press, Sausalito, California, USA.

Peroni, S. and Vitali, F. (2009) 'Annotations with EARMARK for arbitrary, overlapping and out-of order markup', in *Proceedings of the 2009 ACM Symposium on Document Engineering (DocEng 2009)*, Munich, Germany, 21–24 September 2010, pp.171–180, ACM.

Presutti, V. and Gangemi, A. (2008) 'Content ontology design patterns as practical building blocks for web ontologies', in *Proceedings of the 27th International Conference on Conceptual Modeling (ER 2008)*, Barcelona, Spain, 20–24 October, Lecture Notes in Computer Science, Vol. 5231, pp.128–141, Springer.

Renear, A., Dubin, D. and Sperberg-McQueen, C.M. (2002) 'Towards a semantics for XML markup', in *Proceedings of the 2002 ACM Symposium on Document Engineering (DocEng 2002)*, McLean (VA), USA, 8–9 November, pp.119–126, ACM.

Rodrigues, T., Rosa, P. and Cardoso, J. (2006) 'Mapping XML to existing OWL ontologies', in *Proceedings of the IADIS International Conference on WWW/Internet 2006*, Murcia, Spain, 5–8 October, pp.72–77, IADIS.

Sperberg-McQueen, C.M. (2006) 'Rabbit/duck grammars: a validation method for overlapping structures', in *Proceedings of Extreme Markup Languages Conference 2006*, Montreal, Canada, 7–11 August, IDEAlliance, available at http://conferences.idealliance.org/extreme/html/2006/SperbergMcQueen01/EML2006SperbergMcQueen01.html (accessed on 12 March 2011).

Sperberg-McQueen, C.M. and Huitfeldt, C. (2004) 'GODDAG: a data structure for overlapping hierarchies', in *Proceeding of the 5th International Workshop on the Principles of Digital Document Processing (PODDP 2000)*, Munich, Germany, 13–15 September 2000, Lecture Notes in Computer Science, Vol. 2023, pp.139–160, Springer.

TEI Consortium (2005) 'TEI P5: guidelines for electronic text encoding and interchange', TEI Consortium, available at http://www.tei-c.org/Guidelines/P5 (accessed on 17 November 2010).

Tennison, J. and Piez, W. (2002) 'The layered markup and annotation language (LMNL)', *Presented during the Extreme Markup Languages Conference 2002*, Montreal, Canada, 4–9 August, available at http://xml.coverpages.org/LMNLAbstract.html (accessed on 17 November 2010).

Tummarello, G., Morbidni, C. and Pierazzo, E. (2005) 'Toward textual encoding based on RDF', in *Proceedings of the 9th ICCC Conference on Electronic Publishing (ELPUB 2005)*, Leuven, Belgium, 8–10 June, Peeters Publishing, available at http://elpub.scix.net/data/works/att/206elpub2005.content.pdf (accessed on 17November 2010).

Yang, K., Steele, R. and Lo, A. (2007) 'An ontology for XML schema ontology mapping representation', in *Proceedings of the 9th International Conference on Information Integration and Web-based Applications & Services (iiWAS 2007)*, Jakarta, Indonesia, 3–5 December, Band, Vol. 229, pp.101–111, Austrian Computer Society.

Notes

1 More formally, for XML-based languages, a content model of a markup element is "a simple grammar governing the allowed types of the child elements and the order in which they are allowed to appear" (Bray et al., 2006).

2 The following code snippets are written using the Manchester syntax (Horridge and Patel-Schneider, 2009) where the prefixes rdfs and xsd refer respectively to RDF schema and XML schema namespaces, while the empty prefix refers to the EARMARK ontology URI plus '#'. Moreover, we use the prefix c to indicate entities taken from an imported ontology made for the SWAN project (Ciccarese et al., 2008), available at http://swan.mindinformatics.org/spec/1.2/collections.html.

3 This class (and its name) is based on the concept introduced by Ted Nelson in his Xanadu Project (Nelson, 1980) to refer to the collection of text fragments that can be interconnected to each other and transcluded into new documents.

4 In EARMARK, the RDF constructs rdf:Bag and rdf:Seq are intentionally avoided, preferring the use of the SWAN collection entities. This is needed to keep EARMARK an OWL2 DL ontology and, consequently, to successfully use OWL2 DL tools, such as reasoners.

5 The model is available at http://www.essepuntato.it/2011/03/schemaexample.

6 All tests are available at http://www.essepuntato.it/2011/03/schemaexample/test.

7 In verse, an enjambement is the continuation of a sentence without a pause beyond the end of a line, couplet, or stanza.