Handling markup overlaps using OWL

Angelo Di Iorio, Silvio Peroni, and Fabio Vitali

Department of Computer Science, University of Bologna
diiorio@cs.unibo.it, speroni@cs.unibo.it, fabio@cs.unibo.it

Abstract. A lot of applications handle XML documents where multiple overlapping hierarchies are necessary and make use of a number of workarounds to force overlaps into the single hierarchy of an XML format. Although these workarounds are transparent to the users, they are very difficult to handle by applications reading into these formats. This paper proposes an approach to document markup based on Semantic Web technologies. Our model allows the same expressiveness as XML and any other hierarchical meta-markup language, and, rather than requiring complex workarounds, allows the explicit expression of overlapping structures in such a way that search and manipulation of these structures does not require any specific tool or language. By simply using mainstream technologies such as OWL and SPARQL, our model – called EARMARK (Extremely Annotational RDF Markup) – can perform rather sophisticated tasks with no special tricks.

Keywords: EARMARK, OWL, change tracking, overlapping markup

1 Introduction

In the past, overlapping markup has received ambivalent, almost schizoid considerations in the field of markup languages. On the one hand, overlaps were the hallmarks of bad HTML coders and naive HTML page editors, taking advantage of an unjustified benevolence in web browsers that would display basically any HTML regardless of proper nesting. On the other hand, overlaps have been a fringe, almost esoteric discipline of scholars in the humanities, competently used for arcane specifications of linguistic annotations and literary analysis.

But although the first type of usage was approached with scorn and the second with awe, they both fundamentally represent a situation that is more common than thought, and the scholars were only more aware, and not more justified, about the need to represent overlaps.

Overlaps are needed whenever multiple markup elements need to be applied over the same content, and these elements are independent of each other. In some (rather frequent) situations, this independence means that the content referred to by some elements is partially but not completely the same as the content referred to by other elements.

Yet of course SGML, and now XML, grammatically impose and require a strict hierarchy of containment generating a single mathematical tree of the doc-

ument where no overlap is allowed. Thus, naively ignoring this requirement, carefully creating workarounds or inventing completely new markup languages, document authors have coped with this problem. But while new markup languages such as TexMecs [4] and LMNL [11] have but a small number of adepts and applications, workarounds such as segmentation, milestones or standoff markup [5] are frequently used and ubiquitous.

All workarounds manage to constrain secondary structural information so as not to break or obfuscate the main hierarchy that is expressed in the visible XML structure. But although this allows to manage multiple structures over the same content, this comes at a price: structures specified through workarounds are more difficult to find, identify and act upon than the structures in the main XML hierarchy.

In this paper we propose an alternative approach to overlapping that allows very simple tools to be used on all markup to generate sophisticated functionalities. Furthermore, rather than creating a completely new language requiring completely new tools and competencies, we propose to use Semantic Web technologies and tools to obtain much of the same results we would obtain with traditional XML tools.

Our proposal, EARMARK (Extremely Annotational RDF Markup), defines markup vocabularies by means of OWL ontologies [13], through which arbitrary markup over the same content can be expressed. Since each individual markup item is an independent assertion over some content or other assertions, overlaps stop being a problem, and similarly all other issues connected to physical embedding and containments, such as contiguity and document order.

Thus, while in previous works [6] [2] we concentrated on showing how XML documents could be converted in EARMARK and vice versa, and how to use workarounds when converting overlapping EARMARK structures into an XML document, in this paper we concentrate on identifying workarounds existing in real XML documents, and disentangle them into independent, direct EARMARK assertions on which sophisticated functionalities can then be generated.

The paper is structured as follows: in Section 2 we provide a brief overview of existing approaches to handle overlap using workarounds or brand new languages, and in Section 3 we provide a significative example of situations where overlaps exist today and sometimes in pretty mainstream situations. In Section 4 we introduce EARMARK, and in Section 5 we discuss an use case that should demonstrate the superiority of an EARMARK approach to a traditional XML one when overlaps come into question, and in Section 6 we draw some conclusions.

2 Existing Approaches To Overlapping

The need for multiple overlapping structures over documents using markup syntaxes such as XML and SGML is an age-old issue: much research has been carried out about techniques, languages and tools that allow users to create

multiple hierarchies over the same content. A good review of them can be found in [1].

Some of such research proposes to use plain hierarchical markup (i.e., SGML and XML) and employ specially tailored elements or attributes to express the semantics of overlapping in an implicit way. For instance, the TEI Guidelines [9] present a number of different techniques that use SGML/XML constructs to force multiple hierarchies into a single one, including:

- *milestones* (the overlapping structures are expressed through empty elements to mark the boundaries of the "content"),
- *fragmentation* (the overlapping structures are split into individual non-overlapping elements that may even be linked through id-idref pairs) and
- *standoff markup* (the overlapping structures are placed elsewhere and indirectly refer to their would-be locations through pointers, locators and/or id/idref pairs).

Given the large number of techniques to deal with overlapping structures in XML, in [5] we presented a number of algorithms to convert XML documents with overlapping structures from and to the most common approaches.

Other research actually proposes to get rid of the theory of trees at the base of XML/SGML altogether, and use different underlying models and newly invented XML-like languages that allow the expression of overlaps through some kind of syntactical flourishing.

For instance, *GODDAG* [10] is a Direct Acyclic Graph whose nodes represent markup elements and text. Arcs are used to explicit represent containment and father-child relations. Since multiple arcs can be directed to the same node, overlapping structures can be straightforwardly represented in GODDAG. *Restricted GODDAGs*, a subset thereof, can be and has been linearized into TexMecs [4], a multi-hierarchical markup language that also allows full GODDAGs through appropriate non-embedding workarounds, such as standoff markup.

LMNL [11] is a general data model based on the idea of *layered text fragments and ranges*, where multiple types of overlap can be modelled using concepts drawn from the mathematical theory of intervals. Multiple serializations of LMNL exist, such as CLIX and LMNL-syntax.

The *variant graph* approach [8] is also based on graph theory. Developed to deal with textual variations – that generate multiple versions of the same document with multiple overlapping hierarchies – this theory proposes a new data model to represent literary documents and a graph linearisation (based on lists) that scales well even with a large number of versions. In the same conference, [7] presented another detailed survey about overlapping approaches, also discussing the MultiX^2 data model – that uses W3C standard languages such as XInclude to link and fetch text fragments within overlapping structures – and a prototype editor for the creation of multi-structured documents.

3 More Frequent Than One May Think

So often overlapping structures have been considered as needed only in highly specific contexts (such as linguistics) and basically for scholars: solutions were complex since they were considered grounded in the intrinsic complexity of the topics themselves. Yet, overlapping structures can be found in many more fields than these, and even mainstream applications generate and use markup with overlapping structures. While the complexity of overlapping is hidden to the final user, application that consume such data may very well find it rather difficult to handle such information. In the following we discuss a significative context where overlapping already exist and fairly relevant information is encoded in multiple independent structures, leaving to special code the task of managing the complexity: word processors.

Word processors provide users with powerful tools for tracking changes, allowing each individual modification by individual authors to be identified, highlighted, and acted upon (e.g. by accepting or discarding them). The intuitiveness of the relevant interfaces actually hides the complexity of the data format and of the algorithms necessary to handle such information.

For instance, the standard ODT format used by Open Office – and similarly the DOCX format used by Microsoft Word – when saving change tracking information relies on two specific constructs for insertions and deletions that may overlap with the structural markup. While adding a few words within a paragraph does not imply the breaking of the fundamental structural hierarchy, any change that affects the structure itself (e.g. the split of one paragraph into two by the insertion of a return character, or vice versa the join of two paragraphs by the elimination of the intermediate return character) requires annotations to be associated to the end of a paragraph and the beginning of the next, in an unavoidably overlapping pattern. ODT uses milestones and standoff markup for insertions and deletions respectively, and also relies on standoff markup for annotations about the authorship and date of the change.

For instance, the insertion of a return character and a few characters in a paragraph creates a structure as follows:

```
<text:changed-region text:id="S1">
  <text:insertion>
    <office:change-info>
      <dc:creator>John Smith</dc:creator>
      <dc:date>2009-10-27T18:45:00</dc:date>
    </office:change-info>
  </text:insertion>
</text:changed-region>
<text:p>The beginning and
  <text:change-start text:change-id="S1"/></text:p>
<text:p>also<text:change-end text:change-id="S1"/>
  the end.</text:p>
```

The empty elements `<text:change-start/>` and `<text:change-end/>` are *milestones* marking respectively the beginning and the end of the range that

constituted the insertion, while the element `<text:insertion>`, before the beginning of the document content, is *standoff markup* for the metadata about the change (author and date information). The deletion operation shows a similar behaviour.

4 EARMARK And Its Support For Overlapping Features

This section discusses a different approach to metamarkup, called EARMARK (Extremely Annotational RDF Markup) based on ontologies and Semantic Web technologies. The basic idea is to model EARMARK documents as collections of addressable text fragments, and to associate such text content with OWL [13] assertions that describe structural features as well as semantic properties of (parts of) that content. The overall approach is similar but more general, robust and extensible than [12]. As a result EARMARK allows not only documents with single hierarchies (as with XML) but also multiple overlapping hierarchies where the textual content within the markup items belongs to some hierarchies but not to others. Moreover EAMARK makes it possible to add semantic annotations to the content though assertions that may overlap with existing ones.

One of the advantages of using EARMARK is the capability to access and query documents by using well-known and widely supported tools for Semantic Web. In fact, EARMARK assertions are simply RDF assertions, while EARMARK documents are modelled through OWL ontologies. The consequence is that query languages (such as SPARQL) and actual existing tools (such as OWLAPI or Pellet) can be directly used to deal with even incredibly complicated overlapping structures.

The EARMARK model itself is defined through an OWL document (available at `http://www.essepuntato.it/2008/12/earmark`) specifying classes and relationships. The model introduces three separate base concepts: *docuverses*, *ranges* and *markup items*. Each of them is represented with different, disjoint OWL classes.

The textual content of an EARMARK document is conceptually separated from the annotations, and is referred to by means of assertions on the specific class *Docuverse*. This class refer to the collection of text fragments that can be interconnected to each other or transcluded into new documents.

The individuals of this class represent the object of discourse, i.e. all the text containers related to a particular EARMARK document. The following code snippets are written using the Manchester Syntax [3]; the prefixes *rdfs*, *owl* and *xsd* refer respectively to RDF Schema, OWL and XML Schema, while the empty prefix refers to the EARMARK own namespace.

```
Class: Docuverse
DataProperty: hasContent
  Characteristics: FunctionalProperty
  Domain: Docuverse
  Range: rdfs:Literal
```

Any individual of the *Docuverse* class can specify the actual content of the document using the property *hasContent*.

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

```
Class: Range
  HasKey: begins , ends , refersTo
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: they define the way a range must be read. Note that, by using the *hasKey* OWL property, we assert that if there exist two ranges referring to the same docuverse and having the same begin and end locations, then they are the same range.

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

```
Class: MarkupItem
DataProperty: hasGeneralIdentifier
  Characteristics: FunctionalProperty
  Domain: MarkupItem
  Range: xsd:string
DataProperty: hasNamespace
  Characteristics: FunctionalProperty
  Domain: MarkupItem
  Range: xsd:anyURI
```

A *markupitem* individual is a collection of individuals belonging to the classes *MarkupItem* and *Range*. It might have a name, specified in the functional property *hasGeneralIdentifier* (the term comes from the SGML term to refer to the name of elements), and a namespace, specified by using the functional property *hasNamespace*.

5 Using EARMARK

There are multiple applications for the EARMARK approach. The most interesting for this work is the capability of dealing with overlapping structures in an

elegant and straightforward manner. Under EARMARK such structures do not need to be specified through complex workarounds as with XML, but they are explicit and can be easily rebuilt and accessed.

In fact, we do not expect all documents to be natively encoded with EARMARK. Thus, we developed an approach that takes as input an XML file and produces the corresponding EARMARK document in a finite amount of steps. Details of such a process are out of the scope of this paper. Our goal here is to show that complex overlapping data structures can be disentangled into EARMARK assertions that can be processed and queried in a very simple way. Towards this goal, we now go into details of the scenario presented earlier: ODT change-tracking.

The ODT format uses complex data structures to store overlap generated by change-tracking facilities, as discussed in Section 3. These structures make it very difficult to search and manipulate the content by directly using XML languages and tools. Let us discuss a very simple example:

```
<text:changed-region text:id="S2">
  <text:deletion>
    <office:change-info>
      <dc:creator>Silvio Peroni</dc:creator>
      <dc:date>2009-10-27T18:45:00</dc:date>
    </office:change-info>
    <text:p>.</text:p>
  </text:deletion>
  <text:insertion>
    <office:change-info office:chg-author="Angelo Di Iorio"
      office:chg-date-time="2009-10-27T18:42:00"/>
  </text:insertion>
</text:changed-region>
<text:changed-region text:id="A2">
  <text:insertion>
    <office:change-info>
      <dc:creator>Angelo Di Iorio</dc:creator>
      <dc:date>2009-10-27T18:42:00</dc:date>
    </office:change-info>
  </text:insertion>
</text:changed-region>
<text:p>This is one paragraph<text:change-start text:change-
    id="S1"/>;
  actually, it was!<text:change-end text:change-id="S1"/>
  <text:change text:change-id="S2"/>
  <text:change-start text:change-id="A2"/></text:p>
<text:p><text:change-end text:change-id="A2"/>
  <text:change text:change-id="A3"/>
  <text:change-start text:change-id="A4"/>S
  <text:change-end text:change-id="A4"/>plit in two.</text:p>
```

The document was originally composed by a single paragraph: "*This is one paragraph that will be split in two.*". The user "Angelo Di Iorio" split the para-

graph in two (change identified as A2), removed the text "*that will be*" (change A3), added a symbol "." at the end of the first paragraph (change A1, hidden by the following change S2, by Silvio Peroni) and capitalized the letter "*S*" at the beginning of the word "*split*" (change A4). Later, user "Silvio Peroni" substituted the final "." of the first paragraph with the text "*; actually, it was!*" (changes S1 and S2). Note that, for the sake of clarity, we did not show each element *text:changed-region* . It should not be difficult for the reader to picture them, following the discussion of Section 3.

Apart from difficulties in rebuilding the underlying overlapping structures from this complex (XML) syntax, applications have to face another issue: the fact that even simple queries become very complex. The complexity does not lie in the complexity of the information itself, but rather in the data structure over which the operation has to work.

For instance, assume that a reader is looking for "all text fragments inserted by Angelo Di Iorio". This very simple query ends up being surprisingly entangled when written in XPath:

```
for $id in //@text:id[../text:insertion//(dc:creator[. = '
    Angelo Di Iorio'] | @office:chg-author[. = 'Angelo Di
    Iorio'])] return //text:p//text()[(preceding-sibling::
    text:change-start[1][@text:change-id = $id] and following
    -sibling::text:change-end[1][@text:change-id = $id]) or
    ancestor::text:changed-region/@text:id = $id]
```

The EARMARK approach solves this issue is a very disciplined way. The point is that EARMARK stores overlapping data in a direct and elegant manner, that does not require tools to rebuild such information from twisted tree-based XML structures. The information is already available and expressed through RDF and OWL statements.

Fig. 1 shows how the example could be modelled with EARMARK. All information about all three versions of the document are stored in a single EARMARK structure. In fact, there are three EARMARK elements, with general identifier set to "text", representing the three versions. Each of them is an ordered collection of other EARMARK elements, while the leaf items of each hierarchy are ranges. The arrows indicate a containment relationship, specified through the properties of the imported collection ontology, and model the tree structure of each version. Note that the EARMARK element with general identifier "text" of version 1 (looking the picture, the bottom one) contains only one paragraph – where paragraphs are all the elements with general identifier "p" – while the elements of versions 2 (central) and 3 (top), with the same general identifier "text", contain two paragraphs. In turn, those paragraphs contain fragments that compose the textual content of each version. The crucial aspect is that each text fragment is also annotated with (RDF) statements that indicates the type, the author and the date of each modification.

Fig. 1 is a graphic representation of the OWL ontology describing our example. Translating that content into an actual OWL files makes it possible to access and query it with Semantic Web tools. Powerful searches can be then performed

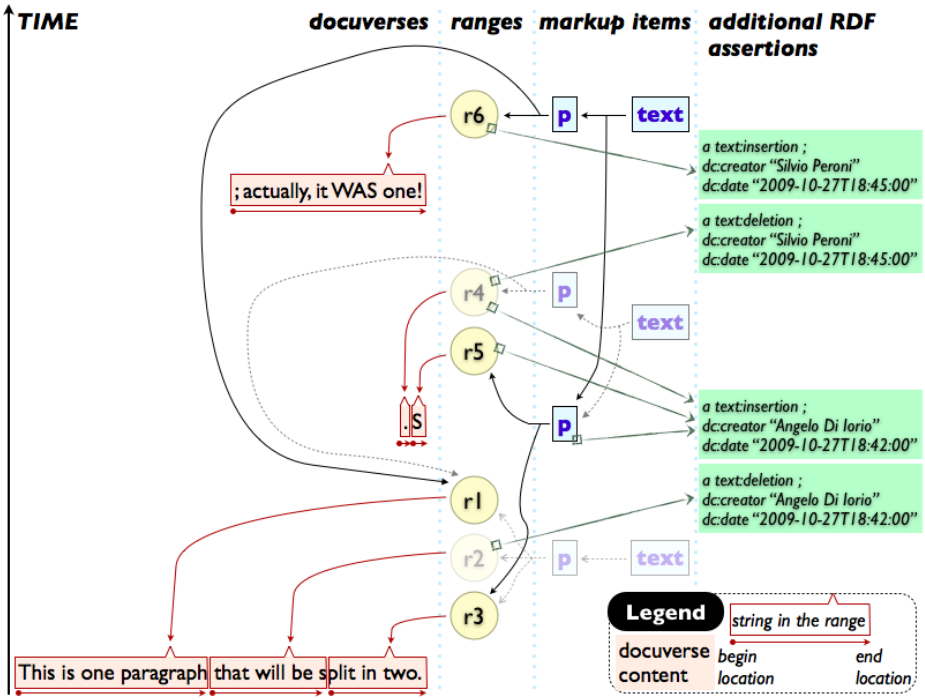

Fig. 1. Encoding the change-tracking use case with EARMARK data structures.

without using niche tools or complex XPath expressions but simply through mainstream technologies such as SPARQL. Thus, the following SPARQL query:

```
SELECT ?r WHERE {
  ?r a text:insertion ; dc:creator "Angelo Di Iorio" }
```

represents the above-mentioned query ("*all text fragments inserted by Angelo Di Iorio*") running on the EARMARK document described so far.

6 Conclusions

The importance of XML lead researchers to design languages for overlapping markup that fit into the XML tree-based model. Multiple hierarchies are in fact forced into a single one that can be directly processed by languages such as XPath and XQuery. This paper discussed how using these languages for even simple needs is actually very difficult, even when the overlap is encoded applying the most common and robust strategies found in the literature.

The EARMARK approach is radically different and consists of specifying both markup structures and semantic annotations through an ontological approach. EARMARK associates OWL assertions to pieces of content and makes

it possible to add RDF statements to the same content (or part of it). Multiple hierarchies can overlap without any limitation and without requiring the use of twisted syntactical workarounds. What makes our approach particularly interesting is that EARMARK documents are basically OWL ontologies. As a consequence, Semantic Web technologies – such as SPARQL – can be used *straightforwardly* to perform operations on their content.

Improving queries is not the only application of EARMARK. Validation is another interesting field we plan to investigate. The same ontological framework, in fact, can be used to prove properties concerning a document such as validity against a schema, compliance to co-constraint specifications or adherence to structural patterns.

References

1. DeRose, S. (2004). Markup Overlap: A Review and a Horse. In the Proceedings of the Extreme Markup Languages 2004 Conference. Montral, Canada.
2. Di Iorio, A., Peroni, S., Vitali, F. (2009). Towards markup support for full GODDAGs and beyond: the EARMARK approach. In the Proceedings of Balisage: The Markup Conference 2009. Montreal, Canada.
3. Horridge, M., Patel-Schneider, P. (2009). OWL 2 Web Ontology Language: Manchester Syntax. W3C Working Group Note. World Wide Web Consortium. http://www.w3.org/TR/owl2-manchester-syntax/.
4. Huitfeldt, C., Sperberg-McQueen, C. M. (2001). TexMECS: An experimental markup meta-language for complex documents. Working paper of the project Markup Languages for Complex Documents (MLCD), University of Bergen.
5. Marinelli, P., Vitali, F., Zacchiroli, S. (2008). Towards the unification of formats for overlapping markup. In New Review of Hypermedia and Multimedia, v. 14(1), pp. 57-94. Taylor and Francis, ISSN 1361-4568.
6. Peroni, S., Vitali, F. (2009). Annotations with EARMARK for arbitrary, overlapping and out-of order markup. In the Proceedings of the Document Engineering 2009 conference. Munich, Germany.
7. Portier, P., Calabretto, S. (2009). Methodology for the construction of multi-structured documents. In the Proceedings of Balisage: The Markup Conference 2009. Montral, Canada.
8. Schmidt, D., Colomb, R. (2009). A data structure for representing multi-version texts online. In International Journal of Human-Computer Studies, v. 67(6), pp. 497-514.
9. Sperberg-McQueen, C. M., Burnard, L. (2005). TEI P5 Guidelines for Electronic Text Encoding and Interchange. The Association for Computers and the Humanities.
10. Sperberg-McQueen, C.M., Huitfeldt, C. (2004). GODDAG: A Data Structure for Overlapping Hierarchies. In Lecture Notes In Computer Science. Springer.
11. Tennison, J., Piez, W. (2002). The Layered Markup and Annotation Language. Paper presented at the Late breaking at Extreme Markup. Montreal, Canada.
12. Tummarello, G., Morbidni, C., Pierazzo, E. (2005). Toward textual encoding based on RDF. 9th ICCC Conference on Electronic Publishing. Leuven, Belgium.
13. W3C OWL Working Group (2009). OWL 2 Web Ontology Language Document Overview. W3C Recommendation. World Wide Web Consortium. http://www.w3.org/TR/2009/REC-owl2-overview-20091027.