Tracking changes through EARMARK: a theoretical perspective and an implementation

Silvio Peroni
Department of Computer Science and Engineering
University of Bologna
Bologna (Italy)
essepuntato@cs.unibo.it

Francesco Poggi
Department of Computer Science and Engineering
University of Bologna
Bologna (Italy)
fpoggi@cs.unibo.it

Fabio Vitali
Department of Computer Science and Engineering
University of Bologna
Bologna (Italy)
fabio@cs.unibo.it

ABSTRACT

The Extremely Annotational RDF Markup, a.k.a. EARMARK, is an OWL 2 DL ontology that defines document meta-markup. It is an ontologically precise definition of markup that instantiates the structure of a text document as an independent OWL document outside of the text string it annotates, and through appropriate OWL and SWRL characterizations it can define organizations such as trees or graphs and can be used to generate validity constraints. In this paper we present an extension of EARMARK that allows us to describe how markup documents evolve in time, which complies with concepts expressed in the Functional Requirements for Bibliographic Records (FRBR).

Keywords

EARMARK, FRBR, OWL ontologies, change tracking, overlapping markup, provenance data

1. INTRODUCTION

Articles, novels, ancient poems, news, and many other kinds of documents change in time. Any creative act of such a text starts from a particular draft made by someone at a certain time, that is then modified through consecutive revisions, may end up being forked into different variants, is then possibly submitted for publishing where additional editorial activities may take place (including typo-fixing, shortening, restructuring, etc.), creating different strings (and, therefore, versions of the same document).

Keeping track of changes introduced in consecutive versions of the same document is an important task for even rather different disciplines. In Computer Science, it enables programmers to show how programming code or computational models evolve throughout the natural lifecycle of software development; in Philology, it provides mechanisms for scholars to tell the way in which variant copies of a same book overlap in time and content, and provides tools for hypotheses about the reconstruction of the original version of the text. In Scientific Publishing, it may help show the modifications provided by the authors following the peer-review, allowing editors to understand the entity and quality of the modifications and driving the final acceptance or rejection of the whole work. And the list may go on.

Our specific interest in this topic concerns what changes to keep track of among those that occur in markup structures such as XML documents, especially in the correct identification and follow-through of those markup elements that are a) directly affected, b) hierarchically affected, and c) completely unaffected by edits and document changes. In particular we are interested in providing a point of view about the following research questions:

- When a markup element E_1 within a document version V_1 changes in some way, e.g. by adding something to the text it contains, thereby generating document version V_2, are the two instances of E_1 in V_1 and E_2 in V_2 to be considered actually the same element?

- In the case the aforementioned instances of E are to be considered different, is the difference meant to be propagated also to their ancestor elements?

Intuitively, the concept of identity of elements throughout versions, whatever definition one may give of it, from complete continuity of the existence in time ("E_1 in V_1 is E_2 in V_2") to the effect of the application of a time-grounded function of transformation (" E_1 in V_1 becomes E_2 in V_2"), has enormous importance in the very definition of change tracking and in the possibility of following through of its content.

Herein we try to provide our answer to the questions above from a theoretical perspective, through the application of the our understanding of the *Functional Requirements for Bibliographic Records* (*FRBR*) [13], which is "a general model, proposed by the International Federation of Library Associations and Institutions (IFLA) for the description of documents and their evolution" [12]. After showing how to use FRBR to model document changes as multiple versions of markup hierarchies that overlap on the same content, we provide an implementation of it using EARMARK [8] [7], a meta-markup language based on Semantic Web technologies that enables in a very straightforward way the description of a hierarchy or even multiple hierarchies of markup elements upon some textual content, without adopting any particular XML workaround [6] [17] for handling the overlaps.

DChanges 2013, September 10th, 2013, Florence, Italy.
ceur-ws.org Volume 1008, http://ceur-ws.org/Vol-1008/paper6.pdf .

The rest of the paper is organised as follows. In Section 2 we show what kinds of markup structures are actually interested in document changes, introducing a simple running example. In particular, we show how changes can be modelled through FRBR and we discuss about the provenance data to add to comprehend the edits done version by version. In Section 3 we present EARMARK as a model to keep track of all the changes of a markup document through using a juxtaposition of following markup hierarchies, each overlapping upon each other. Finally, after discussing some related works on this topic in Section 4, in Section 5 we conclude the paper, sketching out some future works.

2. WHAT IS A CHANGE, REALLY?

In [22], Renear *et al.* claim they were baffled by a triad of sentences, each of which individually evident, that once taken together were thought to be inconsistent: (a) documents are strings; (b) strings cannot be modified; (c) documents can be modified.

Amused and frankly worried that this could be a real conundrum for philosophers, we provided in [12] our humble and hopefully definitive point of view on the subject, specifying that while in fact strings are immutable objects and documents are in fact mutable (or, in our nomenclature, *Timed Abstract Objects*, a.k.a. *TAOs*), the improbable simplification were in the identity equation of strings and documents, whereby for instance Alice's scribbling "Hello world" on a piece of paper creates a different document from Bruce's writing "Hello world" on a MS Word file on a PC, even if the string is clearly the same. The document is therefore something **more** than the string it is composed of, and this object could (and in fact does) refer to different strings in time, i.e. it creates a history of changes that are more or less pointers to different strings associated to the same document through a function grounded on time instants.

This idea of continuity of the identity of a document through different contents in time closely resembled to us the model that *FRBR* introduced in 1999 [13]: document is an overloaded word, and is better substituted by four different concepts called respectively Work, Expression, Manifestation and Items each of which provides a specific and unambiguous aspect of the overloaded *document* concept, and can replace it to provide a clearer view of the identity, evolution in time, specification of form (and format) and creation of copies that routinely baffle whoever insist in using documents for all situations.

Our idea of the string of characters constituting the content of a document is therefore associated to the FRBR concept of Expression – i.e. "the intellectual or artistic realization of a work in the form of alpha-numeric, musical, or choreographic notation, sound, image, object, movement, etc., or any combination of such forms" [13] –, which never changes in time, but is associated to a Work – i.e. "A distinct intellectual or artistic creation" [13] – through the *realization* function. A Work is therefore associated to a number of different Expressions, some of which are related to each other by time functions (e.g., they are subsequent versions of one another) but also related to each other by other types of functions (e.g., different variants for different audiences, or translations in different languages, etc.).

To understand how to use FRBR Expressions and their subcomponents in a change tracking scenario involving markup elements (where the containment between descendant components is modelled by means of the FRBR function *has part* between Expressions), let us introduce a change tracking history of a document, created by Alice, which is revised two times by Bob and Charles, as shown in Fig. 1. In particular, the original version made by Alice (in blue rectangles with solid border) was first revised by Bob (in yellow rectangles with dashed border), by inserting the text "very" within the emphasis. Then, Bob's version was again revised by Charles (in green rectangles with dotted border), by removing the second paragraph.

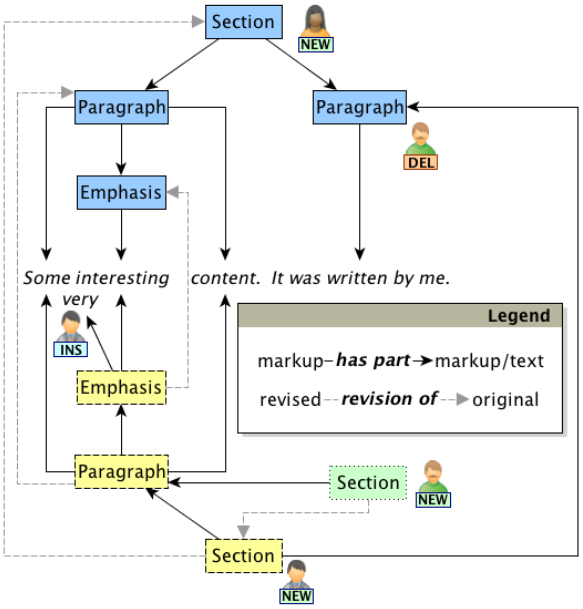

Figure 1: Three different versions of the same document, where each markup element is actually a particular FRBR Expression which contains (i.e. *has part* according to FRBR) other markup and/or textual content (i.e. other FRBR Expressions).

Through a specific FRBR function, i.e. *revision of*, is then possible to track consecutive revisions of a document (and of its markup elements) in a straightforward manner, creating overlapping hierarchies of document structures upon the same content. In addition to that, an important part of change tracking operation involves keeping track of provenance information (shown in Fig. 2) – how the document has been modified, who made the modification and when.

Thus, considering again the two research questions introduced in Section 1, we can claim that, according to FRBR, (a) the two instances of E_1 in V_1 and E_2 in V_2 are two different markup elements (i.e. FRBR Expressions) that are realisations of the same FRBR Work, and that (b) this difference is propagated also to *E*s' ancestor elements.

We implemented this theoretical representation of change tracking information by means of EARMARK, as we describe in the following sections.

3. TRACKING CHANGES IN PRACTICE: THE EARMARK APPROACH

In this section we introduce how to use EARMARK, i.e. a meta-markup language we developed as an OWL ontol-

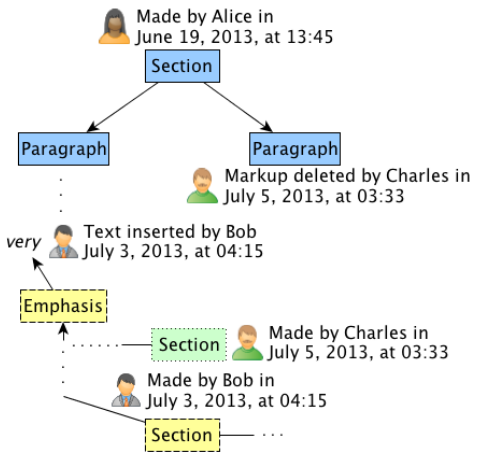

Figure 2: Provenance information added to the root of each version and to the particular items that have been inserted and deleted.

ogy to model complex markup hierarchies including graphs, to model multiple versions of a document within the same EARMARK document. In particular, after briefly introducing the language (Section 3.1), we show how to use it to track changes of markup documents (Section 3.2) and we discuss some of its advantages and drawbacks (Section 3.3 and Section 3.4).

3.1 The EARMARK ontology

EARMARK (the *Extremely Annotational RDF Markup*) [8] [7] is a different approach to meta-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 assertions that describe structural features as well as semantic properties of (parts of) that content. As a result, EARMARK allows one to create 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, EARMARK makes it possible to add semantic annotations to the content that may overlap with existing ones.

Our Java-based implementation[1] strictly follows what is defined in the EARMARK ontology [19] that specifies classes and properties. The core classes of our model describe three disjoint base concepts: *docuverses*, *ranges* and *markup items*.

The textual content of an EARMARK document is conceptually separated from its annotations, and is referred to through the *earmark:Docuverse* class. The individuals of this class represent the objects of discourse, i.e. all the containers of text from an EARMARK document. Any individual of *earmark:Docuverse* – commonly called a *docuverse* (lowercase to distinguish it from the class) – specifies its actual content through the property *earmark:hasContent*.

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

The class *earmark:MarkupItem* is the superclass defining artefacts to be interpreted as markup (such as elements and attributes). A *markupitem* individual is a collection[2] (*co:Set*, *co:Bag* and *co:List*, where the latter is a subclass of the second one and all of them are subclasses of *co:Collection*) of individuals belonging to the classes *earmark:MarkupItem* and *earmark: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 *co:element*;
- to define a markup item as a bag of items (defined by individuals belonging to the class *co:Item*), each of them containing a markup item or a range, by using the properties *co:item* and *co:itemContent* respectively;
- to define a markup item as a list of items (defined by individuals belonging to the class *co: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 *co:nextItem*.

A *markupitem* might also have a name, specified in the functional property *earmark:hasGeneralIdentifier*[3], and a namespace specified using the functional property *earmark:hasNamespace*.

All the three core classes are specialized in other subclasses for giving more specific information about EARMARK instances. First of all, the class *earmark:Docuverse* is specialized into either a *earmark:StringDocuverse* (the content specified through *earmark:hasContent* is a string) or into an *earmark:URIDocuverse* (the actual content is located at the URL specified in *earmark:hasContent*), that are disjoint. Specialized subclasses of *earmark:Range* (*earmark:PointerRange* and *earmark:XPathRange*) are defined to cope with plain-text and XML docuverses with different addressing schemes. *earmark:MarkupItem* is specialized in three disjointed sub-classes: *earmark:Element*, *earmark:Attribute* and *earmark:Comment*.

In order to understand how EARMARK is used to describe markup hierarchies, we show how Alice's version (the blue one in Fig. 1) is expressed in EARMARK. In particular, the following XML-based version of Alice's version:

```
<section>
  <p>Some <em>interesting</em> content.</p>
  <p>It was written by me.</p>
</section>
```

is rendered in EARMARK as follows (in Turtle [21])[4]:

```
@prefix : <http://www.essepuntato.it/2013/
    dchanges/> .

# Any EARMARK document is an ontology
<http://www.essepuntato.it/2013/dchanges>
```

[1] https://github.com/essepuntato/EarmarkDataStructure.

[2] In the following descriptions the prefix *co* to indicate entities taken from an imported ontology, the Collections Ontology [5], used for handling collections. Its latest version is available at http://purl.org/co.

[3] *General identifier* was the SGML term for the local name of the markup item, e.g., "p" for markup element " ... ".

[4] The whole EARMARK document (in Turtle format) containing all the four versions described in Section 3.2 is available online at http://www.essepuntato.it/2013/dchanges.

```
  a owl:Ontology .

# Textual content of the document
:content a earmark:StringDocuverse ;
  earmark:hasContent
    "Some interesting content.
    It was written by me."^^xsd:string .

:r1 a earmark:PointerRange ; # String 'Some '
  earmark:refersTo :content ;
  earmark:begins "0"^^xsd:nonNegativeInteger ;
  earmark:ends "5"^^xsd:nonNegativeInteger .

# String 'interesting'
:r2 a earmark:PointerRange ...

# String ' content.'
:r3 a earmark:PointerRange ...

# String 'It was written by me.'
:r4 a earmark:PointerRange ...

# Element 'section'
:section a earmark:Element ;
  earmark:hasGeneralIdentifier
    "section"^^xsd:string ;
  co:firstItem [ a co:ListItem ;
    co:itemContent :p1 ;
  co:nextItem [ a co:ListItem ;
    co:itemContent :p2 ] ] .

:p1 a earmark:Element ; # First element 'p'
  earmark:hasGeneralIdentifier
    "p"^^xsd:string ;
  co:firstItem [ a co:ListItem ;
    co:itemContent :r1 ;
  co:nextItem [ a co:ListItem ;
    co:itemContent :em ] ;
  co:nextItem [ a co:ListItem ;
    co:itemContent :r3 ] ] .

:em a earmark:Element ... # Element 'em'
:p2 a earmark:Element ... # Second element 'p'
```

3.2 Modelling changes with EARMARK

EARMARK enables the creation of multiple overlapping hierarchies upon the same content, which is one of the main feature of the language and a strict requirement to handle change tracking information as presented in Section 2.

In order to provide a full model to address what we introduced in Section 2, we developed the *EARMARK Changes Ontology* (*EChO*) [18]. EChO extends the EARMARK ontology presented in Section 3.1 and includes the OWL 2 DL implementation of FRBR [4] and the Provenance Ontology [16], so as to keep track of all the changes and provenance data related to different versions of the same document. In particular, we use:

- the EARMARK items (docuverses, ranges and markup items) to model the structure of the different document versions and to store them all within a single EARMARK document;
- the object property *frbr:revisionOf* to indicate that a markup item is a revision of another;
- the object property *prov:wasDerivedFrom* to indicate that a range is actually derived from another one defined in a previous version of the document;
- the object properties *prov:wasGeneratedBy* (coupled with instances of the classes *echo:VersionCreation* and *echo:ItemInsertion*) and *prov:generatedAtTime* to indicate that a particular markup item, a range or a whole document version has been created at a certain time;
- the object properties *prov:wasInvalidatedBy* (coupled with instances of the classes *echo:VersionRemoval* and *echo:ItemDeletion*) and *prov:invalidatedAtTime* to indicate that a particular markup item, a range or a whole document version has been deleted at a certain time;
- the object property *prov:wasAssociatedWith* to indicate the agent involved in the activity of generation/invalidation of a certain item.

Any EARMARK item is dereferenceable since it is actually identified by a particular URL. This enables us to easily add the aforementioned data to any EARMARK item we define. For instance, we can say that Alice created her document version (implicitly identified by the document element *section*) on June 19, 2013 at 13:45 (as shown in Fig. 2) in the following way:

```
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix echo: <http://www.essepuntato.it
    /2013/07/echo/> .

:section
  # Provenance information
  prov:wasGeneratedBy :creation-by-alice ;
  prov:generatedAtTime
    "2013-06-19T13:45:00Z"^^xsd:dateTime .

# Activity of creation of a new version
:creation-by-alice a echo:VersionCreation ;
  prov:wasAssociatedWith :alice .
```

Note that all the activities referenced by the properties *prov:wasGeneratedBy* and *prov:wasInvalidatedBy* can define the precise time interval when they have been performed – by using the properties *prov:startedAtTime* and *prov:ended AtTime*.

In the following subsections we show how to use EARMARK to describe all the versions of our exemplar document introduced in Fig. 2 by means of two tracking operations, i.e. insertions and deletions. In addition we also introduce another revision of Alice's version introduced in Section 3.1 that specifies both insertions and deletions within the same version.

3.2.1 Insertions

Bob's revision (shown in Fig. 1) of Alice's version was made on July 3, 2013, at 04:15. Bob's edits concern only the insertion of the string "very " as first textual node of the element *em*. So as to model this version, we extend the EARMARK document containing Alice's version so as to include all the modifications and provenance data related to Bob's version. The following excerpt shows the RDF statements we added to describe that scenario:

```
# New content of the document
:content-by-bob a earmark:StringDocuverse ;
  earmark:hasContent "very "^^xsd:string .

# New string 'very '
:r5 a earmark:PointerRange ;
```

```
  earmark:refersTo :content-by-bob ;
  earmark:begins "0"^^xsd:nonNegativeInteger ;
  earmark:ends "5"^^xsd:nonNegativeInteger ;
  # provenance information
  prov:wasGeneratedBy :insertion-by-bob ;
  prov:generatedAtTime
    "2013-07-03T04:15:00Z"^^xsd:dateTime .

:insertion-by-bob a echo:ItemInsertion ;
  prov:wasAssociatedWith :bob .

# Element 'section' by Bob
:section-by-bob a earmark:Element ;
  earmark:hasGeneralIdentifier
    "section"^^xsd:string ;
  co:firstItem [ a co:ListItem ;
    co:itemContent :p1-by-bob ;
  co:nextItem [ a co:ListItem ;
    co:itemContent :p2 ] ] ;
  # relation with previous version
  frbr:revisionOf :section ;
  # provenance information
  prov:wasGeneratedBy :creation-by-bob ;
  prov:generatedAtTime
    "2013-07-03T04:15:00Z"^^xsd:dateTime .

:creation-by-bob a echo:VersionCreation ;
  prov:wasAssociatedWith :bob .

# First element 'p' by Bob
:p1-by-bob a earmark:Element ...

# Element 'em' by Bob
:em-by-bob a earmark:Element ...
```

As shown in the above excerpt, so as to enable the insertion of new content, we created a new docuverse, i.e. *:content-by-bob*, containing all the new text added to Alice's version. In addition, as suggested in Fig. 1, we also created new instances of the elements *section*, *p* and *em* as revisions of those defined in Alice's version. Alice's second paragraph, i.e. *:p2*, is reused as it is (since it did not change at all in Bob's version).

3.2.2 Deletions

Charles' revision (shown in Fig. 1) of Bob's version was made on July 5, 2013, at 03:33. Charles' edits delete the Bob's second paragraph of the section. So as to model this version, we extend the EARMARK document containing both Alice's and Bob's versions so as to include all the modifications and provenance data of Charles' version. The following excerpt shows the RDF statements we added to describe that scenario:

```
# Element 'section' by Charles
:section-by-charles a earmark:Element ...
  prov:wasGeneratedBy :creation-by-charles ;
  prov:generatedAtTime
    "2013-07-05T03:33:00Z"^^xsd:dateTime .

:creation-by-charles a echo:VersionCreation ;
  prov:wasAssociatedWith :charles .

# Provenance information for the deletion of
# Second element 'p' of Bob's 'section'
:p2
  prov:wasInvalidatedBy :deletion-by-charles ;
  prov:invalidatedAtTime
    "2013-07-05T03:33:00Z"^^xsd:dateTime .

:deletion-by-charles a echo:ItemDeletion ;
  prov:wasAssociatedWith :charles .
```

As shown in the above excerpt, we did not need to add a new docuverse since we just had to delete only Alice's and Bob's second paragraph, i.e. *:p2*. Thus, we created only the new instance of the element *section* as revisions of Bob's section, and we added provenance information about the deletion of the Alice's and Bob's second paragraph.

3.2.3 Splitting ranges up

In Section 3.2.1 and Section 3.2.2 we showed how to create an EARMARK document that keeps track of consecutive revisions (made by different people) of the same markup document. We chose to describe versions specifying either one insertion or one deletion so as to make it easier to understand how to define change tracking information of such operations through EARMARK.

In this section we focus on another revision (Daniel's one) of the Alice's version (which actually defines a different branch of the tracking history of such a document, since it contrasts with Bob's revision), where Daniel decided to substitute the string "me" in the second paragraph with its name (i.e. the string "Daniel"), as shown in Fig. 3.

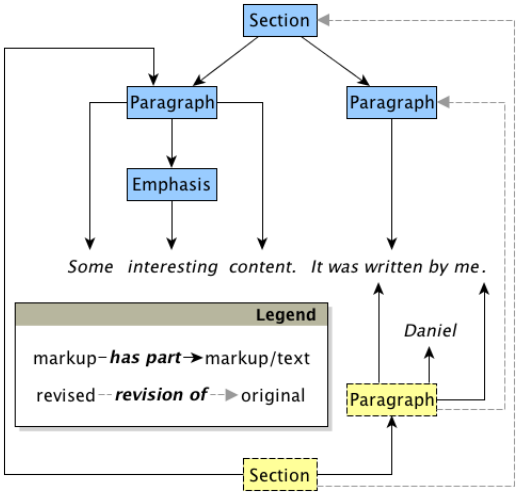

Figure 3: Daniel's revision of Alice's version, where the string "me" in the second paragraph was deleted and substituted with the string "Daniel".

In EARMARK, this string substitution (i.e. a deletion plus an insertion) can be described as shown in Fig. 4. Since the deletion happens in the middle of a range defined in Alice's previous version, we need to define four new ranges. Three of them implicitly represent a split-up of the range *:r4* (tracked through the property *prov:wasDerivedFrom*), and refer to the strings "It was written by ", "me" (to delete in Daniel's version) and "." respectively. The other range refers to the new content, i.e. the string "Daniel", added in Daniel's version. Thus we can add revision information and provenance data, as shown in Section 3.2.1 and Section 3.2.2, to these four ranges and to the related new elements *p* and *section* defining Daniel's revisions of the Alice's elements.

Note that the *prov:wasDerivedFrom* statements between ranges actually describe (at an abstract level) a more complex scenario of overlapping markup between Alice's second

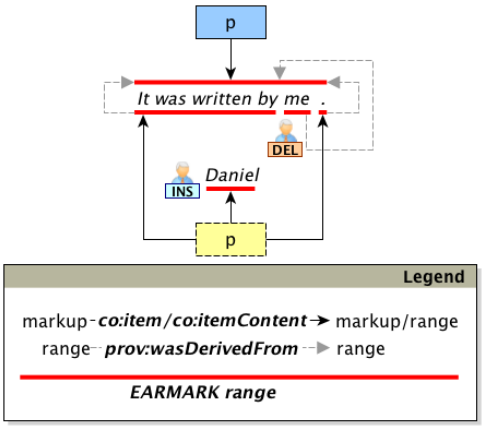

Figure 4: A graphical representation of the use of EARMARK to describe Daniel's revision as introduced in Fig. 3.

paragraph and its Daniel's revision[5]. The overlapping items between items of Alice and Bob's versions and between Bob and Charles' versions, introduced in Section 3.2.1, are explicit – e.g. *:section* overlaps with *:section-by-bob* since both contain the same second paragraph. Even comparing Alice's and Daniel's versions, some overlaps are still explicit – e.g. those between the Alice and Daniel's elements *section* that share the same first paragraph. However, according to the EARMARK representation of Daniel's version, it seems that Alice's second paragraph does not overlap with Daniel's one, since they do not have any range in common.

So as to recognise the actual overlap between *:r4* and the three ranges derived from it, we must perform some arithmetic operations on ranges, as described in [7] and modelled in a precise way by the *EARMARK Overlapping Ontology* [20]. In particular, the application of such an ontology on our exemplar EARMARK document allows us to infer automatically that each of the split-up ranges is actually overlapping with *:r4* and, thus, to infer that Alice and Bob's second paragraphs are actually *overlapping by range*[6]. Herein, we use the property *prov:wasDerivedFrom* so as to simplify the description of such a complex overlapping scenario.

3.3 Querying the changes history

Important advantages derive from using EARMARK as a model to describe consecutive versions of the same document. In particular, since EARMARK is defined by means of Semantic Web technologies, we can use already implemented standards such as SPARQL 1.1 [11] to query over the change tracking history of a certain EARMARK document. For instance, the following small SPARQL query returns a new EARMARK document (as a set of RDF statements) that contains only Bob's version, without taking care of other ones:

```
CONSTRUCT { ?other ?p ?o . ?version ?pv ?ov .
  ?docuverse ?pd ?od } WHERE {
  { SELECT DISTINCT
    ?other ?version ?pv ?ov WHERE {
    { SELECT DISTINCT ?version WHERE {
      ?version a earmark:Element ;
        prov:wasGeneratedBy ?activity .
      ?activity a echo:VersionCreation ;
        prov:wasAssociatedWith :bob } }
      ?other (^co:itemContent?/^co:item)+
          ?version } }
  ?version ?pv ?ov . ?other ?p ?o .
  OPTIONAL { ?other a earmark:PointerRange ;
    earmark:refersTo ?docuverse .
    ?docuverse ?pd ?od } }
```

Note that other XML-based formats for tracking document changes such as [15] and [14] cannot represent explicitly multiple document versions as alternative overlapping hierarchies within the same file since they strictly depend on the tree-based structure of XML. Thus, in these cases, we should reconstruct every time Bob's version when needed by analysing each overlapping trick that has been used to embed change tracking information within the XML document in consideration [7]. Contrarily, in EARMARK we have only to identify the document element of Bob's version and all the related descendant items, since all the hierarchies of the several versions stored are explicitly represented within the EARMARK document.

In addition, answering to very simple query like "select the textual content of all paragraphs removed by Charles" ends up rather entangled if we used directly XPath to query on documents containing change tracking data defined according to [15] and [14], as demonstrated in [7]. On the other hand, the SPARQL query to retrieve the same information from an EARMARK document is quite straightforward:

```
SELECT DISTINCT ?range WHERE {
  ?p a earmark:Element ;
    earmark:hasGeneralIdentifier
        "p"^^xsd:string ;
    co:item/co:itemContent ?range ;
    prov:wasInvalidatedBy ?activity .
  ?range a earmark:PointerRange .
  ?activity a echo:ItemDeletion ;
    prov:wasAssociatedWith :charles }
```

3.4 Do bytes matter?

One of the criticisms when proposing a new approach to solving a well-known problem is that the new solution may simplify the difficulties of the specific problem, but usually brings with it disadvantages and hidden costs that compensate the benefits, such as the growth of size of the data structures, or compatibility and conversion restrictions, etc.

In this section we focus on the problem of data size, which is one of the main disadvantages of the solutions that are based on Semantic Web technologies. In [7] we have already addressed the cost functions of EARMARK vs. XML-based formats. While XML is a linearisation format immediately expressible in actual bytes, EARMARK documents are defined as OWL ontologies, which may have several alternative linearisation formats (including XML itself) with corresponding huge differences in the final byte counts. For these reasons, in [7] we decided to bypass the test based on byte-lengths and to focus on comparing XML and EARMARK

[5]Two elements overlap when (a) one is neither descendant nor ancestor of the other and (b) they share an element as descendant

[6]An *overlapping by range* between EARMARK markup items happens when a markup item A contains a range that overlaps with another range contained by a markup item B. We recommend to read [7] for a more detailed discussion about possible overlapping scenarios in EARMARK documents.

in terms of basic constituents of each format, i.e. the number of nodes of XML documents vs. the number of RDF statements in EARMARK documents. The results of such analysis showed that EARMARK and the OpenDocument format [15] (used by OpenOffice) increase in a very similar way according to consecutive versions of a document, while OpenXML [14] (used by Microsoft Word) is much more verbose and grows faster than the other ones.

Herein, we have taken into consideration the same two documents used for the evaluation in [7] and, as a preliminary evaluation, we compared the size in bytes of consecutive versions of such documents according to OpenDocument and OpenXML formats, and to EARMARK linearised in six different formats: Turtle [21], RDF/XML [2], OWL/XML [26], N-Triples [27], HDT [10] and Manchester Syntax [28][7].

Fig. 5 shows the results of the comparison on the first document composed of seven different versions, named after the "Seven Dwarfs" for recognizability and obtained by applying very common edits (e.g. the insertion of few words, the deletion of some sentences, the split of paragraphs, etc.). Fig. 6 shows the results of a similar comparison on a different document and edits. We collected seven versions named after the weekdays and created by seven different authors when editing a very simple document.

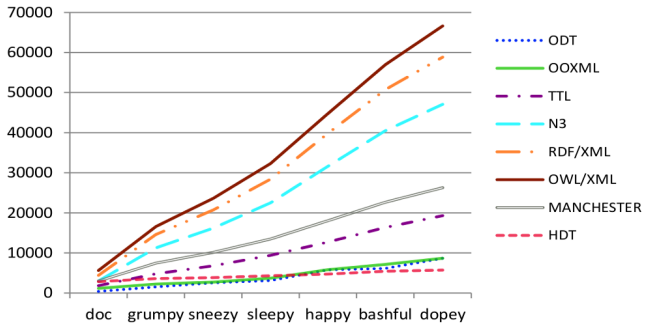

Figure 5: A graph summarizing the results of the first experiment. The file sizes are expressed in bytes.

The overall trend is interesting: comparing the OOXML and the OpenDocument formats we noted that the former was always more concise than the latter (label *ODT*). Focusing on EARMARK we noticed, as expected, a wide variety of results depending on the size of linearisation used. While RDF/XML, OWL/XML and N-triples grew very quickly, Turtle (label *TTL*) and Manchester Syntax did not deviate much from OpenDocument. The most important point concerns HDT, which was similar to OOXML and was even slightly below when the number of changes increased.

4. RELATED WORKS

The meaning expressed by the changes of markup documents is closely connected to the basic operations used to describe the changes themselves. Word processors such as Microsoft Word and Open Office Writer 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

[7]The full details about each version and each format are available at http://fpoggi.web.cs.unibo.it/DCHANGES.

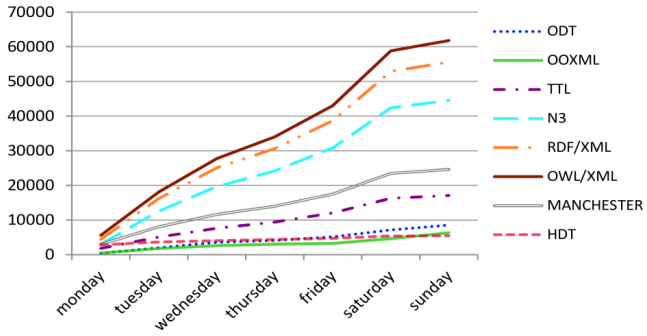

Figure 6: A graph summarizing the results of the second experiment. The file sizes are expressed in bytes.

them). The intuitiveness of the relevant interfaces actually hides the complexity of the data format and of the algorithms necessary to handle such information.

The standard OpenDocument Format [15] used by Open Office is an open XML-based format that tracks changes using three primitive operations: *insertion*, *deletion*, and *formatting change*. The modified regions of the document are identified by special empty elements tracking the location where changes happened. All the metadata related to changes are stored in a list, where each item references to the particular change it concerns through *id-idref* attributes.

The Office Open XML [14] is the XML-based format used by Microsoft Office. The basic operations used to track changes are: *insertion*, *deletion*, *move* and *attribute/property modification*. OOXML fragments change-tracking data across the elements involved in the modifications. All metadata about these editing operations are contained within the attributes of these elements.

Similarly, in [25] the authors introduces an XML-based model to associate a document instance with its history (specifying *insertion* and *deletion* operations) inside a standalone and consistent document, thus gaining strong potential for current or future interoperability.

The aforementioned operations are not the only possible strategies users use to edits documents, even if they are implemented in well-known word processors such as Miscrosoft Word and Open Office Writer. Other, more complex, edit operations are also possible, as those described in recent works about *diff* algorithms. For instance in [3], the authors describe four edit operations used, which include *update* and move in addition to the classic *insertion* and *deletion*.

In [23] and [24], the authors presents a standalone versioning model for XML documents based on three operations on nodes and sequences of nodes: *addition*, *delete* and *update*). Moreover, the model also defines a fingerprinting mechanism to help resolving conflicts during the merging process and an XML patch format. Its main limitation is that it is very focused on elements, making the other XML nodes "second class citizens" – e.g. the additions of attributes are identified as updates of the containing element.

JNDiff [9] is another diff algorithm that has been developed taking into account the notion of "naturalness", i.e. the capability of an algorithm to identify those changes which would be understood and identified by a manual approach. The set of natural operations used by the algorithm are

seven: *insertion*, *deletion*, *move*, *downgrade*, *upgrade*, *update* and *refactoring*.

5. CONCLUSIONS

In this paper we presented a theoretical approach to track document changes based on FRBR and provenance data, and its implementation through EARMARK, a Semantic Web-aware meta-markup language that enables the definition of multiple overlapping markup hierarchies representing different versions of the same document. In particular, we showed how to describe such a multiple versions document as an EARMARK document and how to add provenance data to its markup items and textual content. In addition, we highlighted the main advantages and drawbacks in terms of querying and storing such EARMARK documents. In the future we plan to extend EChO [18] so as to enable the description of additional change tracking operations such as those highlighted in Section 4 (e.g. [9]). We also plan to experiment the effective use of translation mechanisms – e.g. *Fretta* [1] – to convert EARMARK documents with change tracking information into XML formats, e.g. [15] and [14].

6. REFERENCES

[1] Barabucci, G., Peroni, S., Poggi, F., Vitali, F. (2012). Embedding semantic annotations within texts: the FRETTA approach. In Proceedings of SAC 2012: 658-663. DOI: 10.1145/2245276.2245403

[2] Beckett, D. (2004). RDF/XML Syntax Specification (Revised). W3C Recommendation 10 Feb 2004.

[3] Chawathe, S. S., Rajaraman, A., Garcia-Molina, H., Widom, J. (1996). Change detection in hierarchically structured information. In ACM SIGMOD Record, 25 (2): 493-504. DOI: 10.1145/235968.233366

[4] Ciccarese, P., Peroni, S. (2011). Essential FRBR in OWL2 DL. Version 1.0, June 29, 2011. http://purl.org/spar/frbr

[5] Ciccarese, P., Peroni, S. (2013). The Collections Ontology: creating and handling collections in OWL 2 DL frameworks. To appear in Semantic Web – Interoperability, Usability, Applicability.

[6] DeRose, S. (2007). Markup Overlap: A Review and a Horse. In Proceedings of the Extreme Markup Conference 2004.

[7] Di Iorio, A., Peroni, S., Vitali, F. (2011). A Semantic Web Approach To Everyday Overlapping Markup. In Journal of the American Society for Information Science and Technology, 62 (9): 1696-1716. DOI: 10.1002/asi.21591

[8] Di Iorio, A., Peroni, S., Vitali, F. (2011). Using Semantic Web technologies for analysis and validation of structural markup. In International Journal of Web Engineering and Technologies, 6 (4): 375-398. DOI: 10.1504/IJWET.2011.043439

[9] Di Iorio, A., Schirinzi, M., Vitali, F., Marchetti, C. (2009). A natural and multi-layered approach to detect changes in tree-based textual documents. In Proceedings of ICEIS 2009: 90-101. DOI: 10.1007/978-3-642-01347-8_8

[10] Fernández, J. D., Martínez-Prieto, M. A., Gutierrez, C. (2010). Compact Representation of Large RDF Data Sets for Publishing and Exchange. In Proceedings of ISWC 2010: 193-208. DOI: 10.1007/978-3-642-17746-0_13

[11] Harris, S., Seaborne, A. (2013). SPARQL 1.1 Query Language. W3C Recommendation 21 Mar 2013.

[12] Huitfeldt, C., Vitali, F., Peroni, S. (2012). Documents as timed abstract objects. In Proceedings of Balisage 2012. DOI: 10.4242/BalisageVol8.Huitfeldt01

[13] IFLA Study Group on the FRBR (2009). Functional Requirements for Bibliographic Records Final Report. IFLA.

[14] 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. ISO.

[15] JTC1/SC34 WG 6. (2012). ISO/IEC 26300:2006 – Information technology – Open Document Format for Office Applications (OpenDocument) v1.2. ISO.

[16] Lebo, T., Sahoo, S., McGuinness, D. (2013). PROV-O: The PROV Ontology. W3C Recommendation 30 Apr 2013.

[17] Marinelli, P., Vitali, F., Zacchiroli, S. (2008). Towards the unification of formats for overlapping markup. In New Review of Hypermedia and Multimedia, 14 (1): 57-94. DOI:10.1080/13614560802316145

[18] Peroni, S. (2013). EARMARK Changes Ontology. Version 1.0, July 3, 2013. http://www.essepuntato.it/2013/07/echo

[19] Peroni, S., Di Iorio, A., Vitali, F. (2011). EARMARK Ontology. Version 1.8.1, February 24, 2011. http://www.essepuntato.it/2008/12/earmark

[20] Peroni, S., Di Iorio, A., Vitali, F. (2013). The EARMARK Overlapping Ontology. Version 1.1, August 2, 2013. http://www.essepuntato.it/2011/05/overlapping

[21] Prud'hommeaux, E., Carothers, G. (2013). Turtle - Terse RDF Triple Language. W3C Candidate Recommendation 19 Feb 2013.

[22] Renear, A. H., Wickett, K. M. (2010). There are No Documents. In Proceedings of Balisage 2010. DOI: 10.4242/BalisageVol5.Renear01

[23] Ronnau, S., Borghoff, U. M. (2009). Versioning XML-based office documents. In Multimedia Tools and Applications, 43 (3): 253-274. DOI: 10.1007/s11042-009-0271-2

[24] Ronnau, S., Philipp, G., Borghoff, U. M. (2009). Efficient change control of XML documents. In Proceedings of DocEng 2009: 3-12. DOI: 10.1145/1600193.1600197

[25] Vion-Dury, J. (2010). Stand-alone Encoding of Document History (or One Step Beyond XML Diff). In Proceedings of Balisage 2010. DOI:10.4242/BalisageVol5.Vion-Dury01

[26] Motik, B., Parsia, B., Patel-Schneider, P. F. (2012). OWL 2 Web Ontology Language XML serialization (Second Edition). W3C Recommendation 11 Dec 2012.

[27] Beckett, D. (2013). N-Triples: A line-based syntax for an RDF graph. W3C Working Group Note 9 Apr 2013.

[28] Horridge, M., Patel-Schneider, P. F. (2009). OWL 2 Web Ontology Language: Manchester Syntax. W3C Working Group Note 11 Dec 2012.