Embedding semantic annotations within texts: the FRETTA approach

Gioele Barabucci, Silvio Peroni, Francesco Poggi, Fabio Vitali
Department of Computer Science, University of Bologna, Bologna, Italy
{barabucc, speroni, fpoggi, fabio}@cs.unibo.it

ABSTRACT

In order to make semantic assertions about the text content of a document we need a mechanism to identify and organize the text structures of the document itself. Such mechanism would closely resemble a document-oriented markup language and would be free of the classical constraints of an embedded markup language, having no limitations given by sequentiality, containment, or contiguity of text fragments. In the past years we developed EARMARK, our OWL proposal for expressing arbitrary semantic annotations about the structure and the text content of a document. In this paper we describe FRETTA, our mechanism for rendering arbitrary EARMARK annotations (including non-sequential, non-hierarchical and non-contiguous ones) in XML, bringing into a unifying framework a half dozen of syntactic tricks used in literature to handle overlapping structures in a strictly hierarchical language.

Categories and Subject Descriptors

I.7.2 [**Document And Text Processing**]: Document Preparation—*Markup languages*

General Terms

Languages

Keywords

EARMARK, embedded markup, overlapping markup

1. INTRODUCTION

The main purpose of markup is to state something about the text content it contains. This is both true when markup aims mainly at describing the *structural characteristics* of the text content of a document (e.g., HTML [6] and TEI [12]) as well as when it is used as the linearisation format for *semantic models* (e.g., RDF/XML [2] and RDFa [1]). As it has been said, "markup is not part of the text or content of the expression [of a document] but tells us something about it" [3]. Thus, under this point of view, there is no difference between "embedded" markup (e.g., XML vocabularies when used as document-oriented languages) and "semantic" markup (i.e., embedding semantic statements using a linearisation suitable to the format of the host document, such as RDFa for HTML) [9]. In fact, they are both following the same principles:

- they define a syntax to express relations between the annotations and the text – for XML, this means bracketed elements, wrapping around text fragments and other elements, while for RDF this means "*subject - predicate - object*" statements;
- they use (implicit or explicit) document-specific semantics for characterising what the markup refers to – informal human-readable definitions of the purpose of each element and attribute in XML (e.g., what purpose is the element *p* used for in HTML*)* and formally defined logic formulas in RDF (e.g., in order to ground the RDFS class *foaf:Person*[1]).

A major difference in these approaches is the syntactical flexibility of their meta-level models, usually forcing documents to respect specific syntactic constraints, such as a mandatory tree structure for XML documents or the impossibility of using literal values as subjects in RDF statements.

In order to go beyond the limitations of the tree structure for XML vocabularies, a large amount of scientific communities developed "workarounds", or "tricks", for expressing directed graphs in a tree-based syntax. In practice, this means allowing overlapping hierarchies in XML, and it is done in a number of ways by embedding special elements within the main structure of the document [4]. Surprisingly, to date there is a large amount of applications (e.g., Wikis, Microsoft Word, Open Office) that require directed graphs, and that ended up using such workarounds in their storing formats [5].

EARMARK [5] is our proposal for expressing arbitrary and non-embedded semantic annotations about the structure and the text content of a document using OWL. This meta-markup language overcomes the limitations given by the workarounds based on XML (causing much ambiguity of the interpretation of multiple embedded hierarchies) providing advanced features such as multiple overlapping hierarchies over the same text fragments, and subject to no constraints by sequentiality, containment, or contiguity of text fragments.

[1] http://xmlns.com/foaf/0.1/Person

SAC'12 March 25-29, 2012, Riva del Garda, Italy.

In this paper we introduce *Fretta* (**Fr**om **E**ARMARK **T**o **Ta**g), a general and extensible mechanism for expressing EARMARK annotations in an embedded syntax such as XML. Fretta allows the conversion of EARMARK documents into formats that use XML plus some syntactic tricks for representing element overlaps. To do so, users tell Fretta which tricks are used by which format. With this information, Fretta can serialise an EARMARK document into an XML document compliant with the destination format and the tricks it uses.

The rest of the paper is structured as follows: in Section 2 we introduce previous works related to the field of overlapping markup. In Section 3 and Section 4 we come to the main discussion of the paper, presenting EARMARK and, then, Fretta. In Section 5 we evaluate, by means of a comparison with a *golden standard*, the capabilities of Fretta in terms of quality of the returned output. Finally, in Section 6 we conclude the paper, drawing on-going and future research directions.

2. RELATED WORKS

Even before markup languages, the need to associate different annotations to a single piece of text was widespread, as were techniques that assisted document creators in this task. One example in which these problems are often founds are plays in verses. In plays, authors specify more than the dialogue: the script also states which character is saying each line, how different characters interrupt or interact with each other, how the scene changes while the actors are speaking, etc. When a play is in verses, each verse is often composed of parts of dialogues by different characters, and each fragment must be said with the correct rhythm, so as to make the fundamental beat of the verse perceivable by the audience below the individual voices of the different actors. Consider for instance the following extract from *La Mort d'Agrippine* by Cyrano de Bergerac:

```
TIBERE
Poursuivez...

AGRIPPINE
    Quoi, Seigneur?

TIBERE
        Le propos detestable
Ou je vous ai surprise.

AGRIPPINE
    Ah! Ce propos damnable d'une
si grande horreur tous mes sens travailla
```

This fragment shows two overlapping hierarchies over the same content: the first hierarchy describes which character is playing which dialogue lines, the second hierarchy groups spoken words into poetical lines[2] under a specific meter (see the overall structure in Fig. 1).

When written as a TEI document, the first hierarchy would look like the following snippet of XML:

```
<body>
  <sp><speaker>Tibere</speaker>
    Poursuivez...</sp>
  <sp><speaker>Agrippine</speaker>
    Quoi, Seigneur?</sp>
  <sp><speaker>Tibere</speaker>
    Le propos detestable ou je
    vous ai surprise.</sp>
  <sp><speaker>Agrippine</speaker>
    Ah! Ce propos damnable d'une si grande
    horreur tous mes sens travailla</sp></body>
```

The second hierarchy, while containing basically the same content, groups it in a rather different way:

```
<body>
  <l>Poursuivez... Quoi, Seigneur? Le propos
    detestable</l>
  <l>Ou je vous ai surprise. Ah! Ce propos
    damnable d'une</l>
  <l>si grande horreur tous mes sens
    travailla</l></body>
```

To represent these two hierarchies in a single XML document, various techniques have been developed in the past. The TEI guidelines [12] describe a few methods that are used to express multiple hierarchies over the same content as a single XML hierarchy.

The easiest of these techniques uses *milestones*, i.e. overlapping structures are expressed through a pair of empty elements to mark the boundaries of the "content". Using milestones for expressing elements *l*, the two hierarchies of the previous example could be fused in a single hierarchy such as:

```
<sp><speaker>Tibere</speaker>
  <l sID="l1" />Poursuivez...</sp>
<sp><speaker>Agrippine</speaker>
    Quoi, Seigneur?</sp>
<sp><speaker>Tibere</speaker>
      Le propos detestable<l eID="l1" />
  <l sID="l2" />Ou je vous ai surprise.</sp>
<sp> ... <l eId="l2" /></sp> ...
```

Another technique commonly used is *fragmentation*: overlapping structures are split into individual, non-overlapping elements that are linked through *id–idref* pairs. Our example rewritten using the fragmentation technique would look like the following snippet:

```
<sp><speaker>Tibere</speaker>
  <l xml:id='v1p1' next='#v1p2'>
  Poursuivez...</l></sp>
<sp><speaker>Agrippine</speaker>
  <l xml:id='v1p2' next='#v1p3'>Quoi,
  Seigneur?</l></sp>
<sp><speaker>Tibere</speaker>
  <l xml:id='v1p3'>Le propos detestable</l> ...
```

None of these techniques is right or wrong on its own. Depending on the context in which a document has been created, the constraints imposed by the software that must process it and the stylistic preferences present in the community that will deal with it, there may be techniques that are more acceptable than others.

Unfortunately, this approach to the overlap problem poses serious interoperability issues when documents must be exchanged between different communities or pieces of software. An example of such issues can be seen in the way the ODT format and the OOXML format deal with versioning annotations. Automatic conversion between these two formats is made even more complicated by the fact that ODT and OOXML rely on different techniques: ODT uses milestones, while OOXML uses fragmentation [5].

[2]The term "line" in English refers to both the individual utterance of a speaking character in a play and the basic unit of a poetical text, so as to make "the lines of the characters overlap the lines of the verse" correct but obscure.

All those aforementioned (and few other) techniques are fully described from theoretical and applicative points of view in [8] and [4], where a number of algorithms to convert XML documents with overlapping structures from and to the most common approaches are presented.

Beside the introduced XML-based approaches to overlap, other solutions use different underlying models and newly invented XML-like languages that allow the expression of overlaps through some kind of syntactical flourishing. For instance, TexMecs [7] is a markup language that allows multiple hierarchies through appropriate workarounds, such as standoff markup. LMNL [13] 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. XConcur [11] is a similar solution based on the representation of multiple hierarchies within the same document through layers.

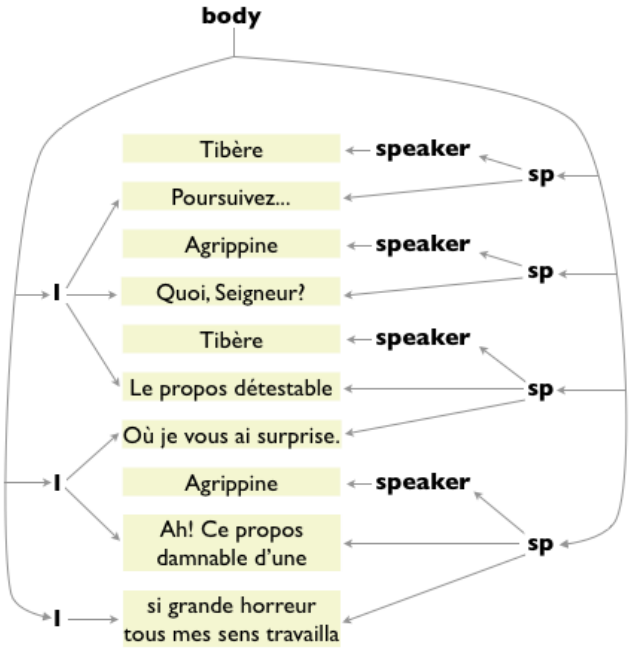

Figure 1: Two alternative TEI markup hierarchies built upon the same text content.

3. EXPRESSING MULTIPLE MARKUP HIERARCHIES USING EARMARK

EARMARK (**E**xtremely **A**nnotational **R**DF **Mark**up) [5] 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 multiple overlapping hierarchies where the textual content within the markup items belongs to some hierarchies but not to others.

In this section we introduce the main capabilities of EARMARK[3] using step by step examples (written in Turtle [10]) where we are going to implement (part of) the example shown in Fig. 1.

In EARMARK, all the textual content of a document is defined through the class *Docuverse*. In the following excerpt, an individual belonging to a Docuverse subclass (i.e., *StringDocuverse*) is defined as string container of the all the document text[4]:

```
:textual-content a earmark:StringDocuverse
  ; earmark:hasContent "Tibere Poursuivez...
      Agrippine Quoi, Seigneur? Tibere Le
      propos detestable Ou je vous ai surprise.
       Agrippine Ah! Ce propos damnable d'une
      si grande horreur tous mes sens travailla
      "^^xsd:string .
```

In order to compose the first line of the example (i.e., the first element *l*) we need to create three different text nodes in correspondence of "Poursuivez...", "Quoi, Seigneur" and "Le propos détestable". In EARMARK, text nodes are definable through the class *Range*. For creating the three text nodes we need, three different individuals of *PointerRange* (i.e., a subclass of *Range*) are defined as pairs of starting and ending locations on the docuverse content:

```
:poursuivez a earmark:PointerRange
  ; earmark:refersTo :textual-content
  ; earmark:begins "7"^^xsd:nonNegativeInteger
  ; earmark:ends "21"^^xsd:nonNegativeInteger .

:quoi-seigneur a earmark:PointerRange
  ; earmark:refersTo :textual-content
  ; earmark:begins "32"^^xsd:nonNegativeInteger
  ; earmark:ends "47"^^xsd:nonNegativeInteger .

:le-propos-detestable a earmark:PointerRange
  ; earmark:refersTo :textual-content
  ; earmark:begins "55"^^xsd:nonNegativeInteger
  ; earmark:ends "75"^^xsd:nonNegativeInteger .
```

Finally, the element *l* is created through the class *Element*, that is a subclass of *MarkupItem*. So as to define order among items contained by the element, it is used an external ontology[5] that was built to define ordered and unordered collections of OWL entities, such as lists, sets and bags. Therefore, we can create the new element *l* as follows:

```
:first-element-l a earmark:Element , c:List
  ; earmark:hasGeneralIdentifier "l"
  ; c:firstItem [ c:itemContent :poursuivez
  ; c:nextItem [ c:itemContent :quoi-seigneur
  ; c:nextItem [ c:itemContent
    :le-propos-detestable ] ] ] .
```

Since both markup items and ranges are clearly and univocally identifiable through IRIs, EARMARK allows to define overlapping elements, such as the first element *l* and the first element *sp*, both that include the same text node "Poursuivez..." as shown as follows:

```
:first-element-sp a earmark:Element , c:List
  ; earmark:hasGeneralIdentifier "sp"
  ; c:firstItem [ c:itemContent
    :first-element-speaker
  ; c:nextItem [ c:itemContent :poursuivez ]].

:first-element-speaker a earmark:Element ...
```

[3]An introduction to EARMARK can be found in [5].

[4]The prefixes *rdfs* and *xsd* refer respectively to RDF Schema and to XML Schema.

[5]The Collection Ontology, available at http://swan.mindinformatics.org/ontologies/1.2/collections.owl. The prefix *c* refers to its ontological lentities.

Of course EARMARK, whose core ontological classes and properties are illustrated in Fig. 2, allows to express more than what we introduced, such as multiple repeatability of items, ordered and non-ordered markup items, subclasses of *Range* able to express starting and ending locations as XPath expressions on XML strings, subclasses of *Docuverse* that refers indirectly to the content of a document through its URI, and generic RDF statements involving any items defined through the EARMARK ontology.

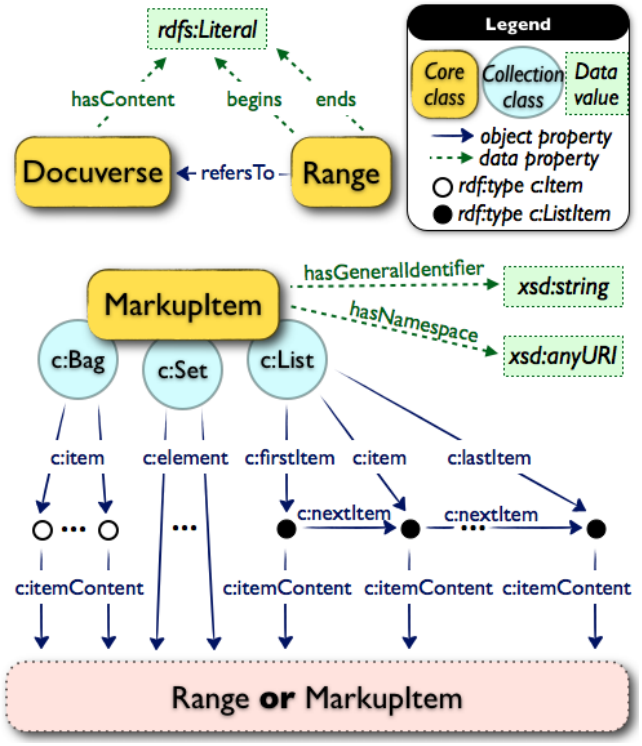

Figure 2: A graphical representation of the EARMARK ontology.

4. APPROACHING THE XML OVERLAPPING MARKUP USING FRETTA

The issue of studying and sketching algorithms for the automatic and semi-automatic conversion of multi-hierarchical markup documents into simple XML tree structure has been studied before from a pure theoretical perspective [8]. In fact, no implementations has been developed in the past years, in order to make this conversion process effective and used by interested research communities and practitioners.

Trying to address this issue, we developed *Fretta* (**Fr**om **E**ARMARK **T**o **Ta**g), a general and extensible Java framework for expressing EARMARK annotations in an embedded XML syntax. Users that want to convert from EARMARK into XML document formats (that use different tricks to store overlapping hierarchies) must indicate which trick are used in a certain format and how EARMARK elements can be encoded in that format. Once Fretta has been supplied with this information, it can perform the requested conversion, passing through four different and consecutive steps that will be illustrated in the following sections.

4.1 Tricks specification

In this step, Fretta adds semantic assertions to all the markup items indicated by the user in a particular XML configuration file. By means of this file, the user can specify rules (elements "trick") to define which trick, among all the available ones – i.e., milestones, fragmentation, stand-off, dominance embedding of an element, "copy of" and "same as" – has to be used for encoding either a particular element (element "refID"), all the elements having a particular name (element "refGID") or all the elements having a particular namespace (element "refNS"). For instance, if we would encode "http://www.example.com/el1" using a milestone, all the elements l through fragmentation, and all the elements having namespace "http://www.namespace.com" with dominance embedding, we would write as follows:

```
<!ENTITY trick "http://fpoggi.web.cs.unibo.it/
    fretta/overlap/overlaptrick.owl#">
<trickspec>
  <trick>
    <refID>http://www.example.com/el1</refID>
    <type>&trick;Milestone</type></trick>
  <trick>
    <refGID>sp</refGID>
    <type>&trick;Fragmentation</type></trick>
  <trick>
    <refNS>http://www.namespace.com</refNS>
    <type>&trick;Dominance</type></trick>
</trickspec>
```

All the URIs within the element type must refer to individuals of the class Trick as defined in the ontology[6] characterising overlapping tricks that we developed[7].

Starting from the configuration file, Fretta adds automatically, to the input EARMARK document, assertions that relates, through the OWL property toEncodeWith defined in the tricks ontology, each element indicated to the related trick, as shown in the following excerpt (taking into account again the EARMARK document introduced in Section 3):

```
:first-element-l trick:toEncodeWith
  trick:Fragmentation .
:second-element-l trick:toEncodeWith
  trick:Fragmentation .
:third-element-l trick:toEncodeWith
  trick:Fragmentation .
```

4.2 Structural conversion

Taking into account all the assertions added in the previous step, the framework runs a pure-structural conversion of the EARMARK document. The output of this step is a new EARMARK document in which some elements – i.e., those specified by the user through the configuration file – are transformed opportunely by using the associated trick.

For instance, if we consider the second rule (i.e., the second element "trick") of the above configuration file, and we ask Fretta to apply it on the EARMARK implementation of the document introduced partially in Section 3, we will

[6]Available at http://fpoggi.web.cs.unibo.it/fretta/overlap/overlaptrick.owl (the prefix *trick* it is used for referring to its entities).

[7]Besides what is already implemented, it is possible to extend the set of tricks managed by Fretta in two steps. First, to develop a JAR package that contains a Java class handling the conversion into XML according to the rationale of the new trick. Second, to extend the tricks ontology with the information about the new trick.

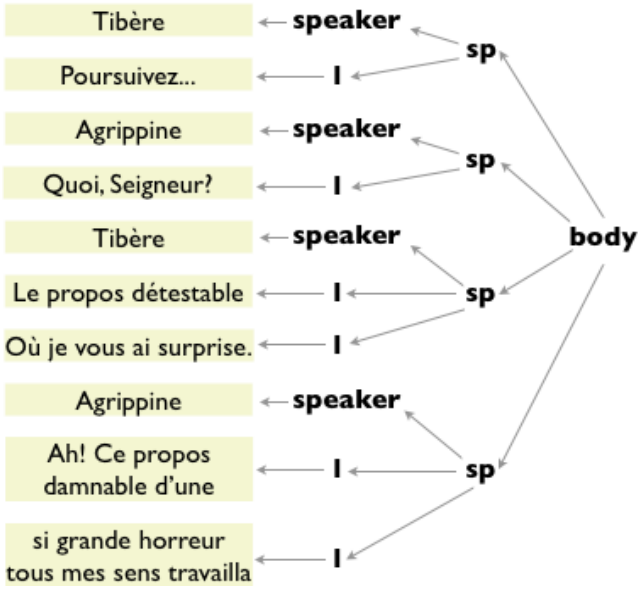

Figure 3: The result of the application of the fragmentation trick on all the element *l* of the document firstly introduced in Fig. 1.

obtain a new EARMARK document having all the elements *l* fragmented and embedded within all the element sp of the document, as shown in Fig. 3 and (partially) as follows:

```
:aFragmentation a trick:Fragmentation
  ; c:firstItem [ c:itemContent
    :fragment1-first-element-l
  ; c:nextItem [ c:itemContent
    :fragment2-first-element-l
  ; c:nextItem [ c:itemContent
    :fragment3-first-element-l ] ] ] .

:first-element-sp a earmark:Element , c:List
  ; earmark:hasGeneralIdentifier "sp"
  ; c:firstItem [ c:itemContent
    :first-element-speaker
  ; c:nextItem [ c:itemContent
    :fragment1-first-element-l ] ] .

:fragment1-first-element-l
  a earmark:Element , c:List
  ; earmark:hasGeneralIdentifier "l"
  ; c:firstItem [c:itemContent :poursuivez]...
```

4.3 Semantic conversion

Of course, each XML format has different way of encoding overlapping tricks. For example, each milestone pair of elements in TEI and ODT adopts the same general structural organization – i.e., two empty elements where the former refers to the latter through a *idref-id* pair of attributes – but may differ in element and attribute names – in fact, TEI use the attributes "sID" and "eID" for identifying those elements, while ODT uses the attribute "text:change-id".

For that reason, and strictly depending on the final format the user wants to obtain, Fretta allows to use (and be extended easily with) different XML markup trick semantics, such as TEI. Of course, this semantic conversion may change (and, usually, it does change) the current structure of the EARMARK document, returning a new EARMARK document containing new elements and attributes. The following excerpt shows how the fragmented elements, introduced in the previous section, are returned by Fretta after this step, considering the TEI tricks formalisation:

```
:fragment1-first-element-l a :Element , c:List
  ; earmark:hasGeneralIdentifier "l"
  ; c:firstItem [ c:itemContent :id-attr ]
  ; c:nextItem [ c:itemContent :next-attr ]
  ; c:nextItem [ c:itemContent :poursuivez ]]].

:id-attr a :Attribute , c:Set
  ; earmark:hasGeneralIdentifier "id"
  ; earmark:hasNamespace
    "http://www.w3.org/xml/ns" ...

:next-attr a :Attribute , c:Set
  ; earmark:hasGeneralIdentifier "next" ...
```

4.4 Linearisation

The final step of the process takes the EARMARK document, generated as a result of the previous step, and tries to linearise it as an XML tree. If everything goes well, Fretta returns an XML document that embeds overlapping elements by means of the tricks specified by the user in the first step, otherwise an exception is returned. An exemplar excerpt returned by the framework is:

```
<body>
  <sp><speaker>Tibere</speaker>
    <l xml:id="id1" next="id2">
      Poursuivez...</l></sp>
  <sp><speaker>Agrippine</speaker>
    <l xml:id="id2" next="id3">
      Quoi, Seigneur?</l></sp>...</body>
```

5. EVALUATION

In order to verify the quality of the XML documents produced by Fretta, we performed an evaluation where we aimed at comparing, according to particular principles, Fretta's outputs against a set of twelve TEI documents[8] written by experts, that represents our *golden standard*. We were interested in stressing the frameworks when handling milestones, fragmentation and dominance embedding tricks. The evaluation took into account four different principles:

- *well-formedness (WF)* , i.e., whether the framework returns well-formed XML documents;
- *validity (V)*, i.e., whether the framework returns valid XML documents according to the particular target XML vocabulary;
- *naturalness (N)*, i.e., how much the XML documents returned by the framework are structurally similar according to the *golden standard*;
- *minimality (M)*, i.e., how much the amount of nodes (i.e., elements and attributes) in the XML documents returned by the framework varies from the *golden standard*.

[8]All the documents considered in the evaluation are contained in the MOC platform, available at http://mlcd.blackmesatech.com/mlcd/2011/W/moc-poc/.

As input for our framework, we used seven EARMARK documents, obtained translating by hand the TEI documents within our golden standard, and we developed twelve configuration files where we specified which tricks should be used when converting the input documents. Finally, we ran Fretta on these input documents using the associated trick-specification documents (i.e., configuration files), and finally evaluated the results according to the four principles introduced above, following this rule: each output document is evaluated according to each principle and gains 1 or 0 depending on whether the principle is fully satisfied. All the results are summarised in Table 1[9]. and are completely available online and are completely available online

Table 1: Adherence to the evaluation principles of the outcomes of Fretta – WF and V are not indicated because they are always passed in our tests.

document	XML tricks	N	M
agrippine	fragmentation	yes	yes
agrippine	milestones	yes	yes
drivemycar	fragmentation	no	no
johnlovesmary	fragmentation	yes	yes
johnlovesmary	milestones	yes	yes
peergynt	fragmentation	yes	yes
peergynt	milestones	yes	yes
peterpaulhammer	milestones	yes	yes
thoughtalice	fragmentation	yes	yes
titwillow	fragmentation	no	yes
titwillow	fragmentation	no	no
titwillow	milestones	no	yes

These results shows how all the output documents are always fine for what concerns correctness and validity. Moreover, the main part of them (83%) continues to be minimal against the documents in the golden standard, sometimes even when the naturalness principle is not respected (33%).

6. CONCLUSIONS

In this work we introduced Fretta, a framework for converting any EARMARK document (documents that allow multiple overlapping hierarchies at the same time) into one or more embedded XML markup structures. The resulting multiple markup hierarchies are generated using well-known workarounds widely studied in literature. Moreover, we evaluated Fretta according to a golden standard defined as a set TEI documents made by experts.

We plan to expand our work in two different ways. On the one hand, we want to extend the current implementation of Fretta to handle other tricks, and implement alternative kinds of semantics, at least one according to the ODT format and another to the OOXML format. On the other hand, we want to integrate Fretta in a much broader ongoing framework for the semi-automatic and round-trip conversion from any supported XML format (embedding overlapping markup) into another, that will use EARMARK as intermediate markup format.

[9]A webpage with extended results and explanations is available at http://fpoggi.web.cs.unibo.it/cgi-bin/testData.php, with a link for downloading Fretta, the documents of the golden standard and the related EARMARK documents.

7. REFERENCES

[1] Adida, B., Birbeck, M., McCarron, S., Pemberton, S. (2008). RDFa in XHTML: Syntax and processing. W3C Recommendation October 14 2008, World Wide Web Consortium. http://www.w3.org/TR/rdfa-syntax/ (last visited 31 October 2011).

[2] Beckett, D. (2004). RDF/XML syntax specification (Rev.). W3C Recommendation February 10 2004, World Wide Web Consortium. http://www.w3.org/TR/REC-rdf-syntax/ (last visited 31 October 2011).

[3] Coombs, J. H., Renear, A. H., DeRose, S. J. (1987). Markup Systems and the Future of Scholarly Text Processing. In Communication of the ACM 30 (11): 933-947. DOI: 10.1145/32206.32209

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

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

[6] Hickson, I. (2011). HTML5 - A vocabulary and associated APIs for HTML and XHTML. W3C Working Draft 25 May 2011, World Wide Web Consortium. http://www.w3.org/TR/html5/ (last visited 31 October 2011).

[7] Huitfeldt, C., Sperberg-McQueen, C. M. (2003). TexMECS: An experimental markup meta-language for complex documents. http://decentius.aksis.uib.no/mlcd/2003/Papers/texmecs.html (last visited 31 October 2011).

[8] Marinelli, P., Vitali, F., Zacchiroli, S. (2008). Towards the unification of formats for overlapping markup. New Review of Hypermedia and Multimedia, 14 (1), 57-94.

[9] Peroni, S., Gangemi, A., Vitali, F. (2011). Dealing with Markup Semantics. In Proceedings of the 7th International Conference on Semantic Systems (i-Semantics 2011). Graz, Austria.

[10] Prud'hommeaux, E., Carothers, G. (2011). Turtle - Terse RDF Triple Language. W3C Working Draft 09 August 2011, World Wide Web Consortium. http://www.w3.org/TR/turtle/ (last visited 31 October 2011).

[11] Schonefeld, O., Witt, A. (2006). Towards validation of concurrent markup. In Proceedings of the Extreme Markup Languages 2006. Montreal, Canada.

[12] TEI Consortium. (2005). TEI P5: Guidelines for electronic text encoding and interchange. http://www.tei-c.org/Guidelines/P5 (last visited 31 October 2011).

[13] Tennison, J., Piez, W. (2002, August). The Layered Markup and Annotation Language (LMNL). Presented at the Extreme Markup Languages Conference 2002, Montreal, Canada.