A novel Textual Encoding paradigm based on Semantic Web tools and semantics

G.Tummarello[1] , C.Morbidoni[1], F. Kepler[2], F. Piazza[1], P.Puliti[1]

[1]DEIT, Università Politecnica delle Marche, Via Brecce Bianche, 60100 Ancona, Italy
giovanni@wup.it, christian@deit.univpm.it

[2]Institute of Mathematics and Statistics, University of Sao Paulo, Brazil
f.kepler@gmail.com

Abstract

In this paper we perform a preliminary evaluation on how Semantic Web technologies such as RDF and OWL can be used to perform textual encoding. Among the potential advantages, we notice how RDF, given its conceptual graph structure, appears naturally suited to deal with overlapping hierarchies of annotations, something notoriously problematic using classic XML based markup. To conclude, we show how complex querying can be performed using slight modifications of already existing Semantic Web query tools.

1. Introduction

In Academic and Educational communities in the field of Humanities, "textual encoding" refers to the techniques used to attach machine readable structural information and metadata to textual content. In particular, a textual encoding methodology specifies a set of markers (or tags) which are added to the electronic representation of the text, usually among words, in order to delimit textual features of interest. Such description can be performed usually down to fine granularities such as single words or syllables as well as across a number of conceptual concerns typically including grammar, syntax, history and revisions, typographical etc. Thanks to these machine processable descriptions it is possible for computer algorithms to locate important features and therefore perform useful tasks of Academic and Educational value. Among these, one might list advanced searching, supporting requests such as retrieving "*the 10 most used adjectives in a book*" or "*the average number of complete sentences in a page*" and advanced filtering and formatting, such as presenting or highlighting specific aspects of the text. Inserting explicit markers for features in the text that would otherwise be implicit is often referred to as 'markup', 'encoding' or 'tagging', and with encoding scheme or markup language one refers to the set of specifications that define the use of markup tags. Agreeing on an encoding scheme is an obvious step for interoperability and various approaches have been proposed to the community, and a consistent works of standardization has been performed. Most works have been carried within the Textual Encoding Initiative (TEI [1]), which provides a set of specifications for XML based textual markup. As educational and academic interest into machine aided literature analysis and processing raises, pure XML textual encoding formats prove however to be limiting under several point of view. A number of successive enhancement proposals has shown that given the limitations imposed by the XML, for advanced encoding complex solutions are needed sometimes abandoning XML conformance (e.g as in the MECS [2], JITT [3]). Even if XML is not completely abandoned, XML structure is conceptually far from the needs of advanced markup so a complex and idiosyncratic set of explanations and rules, resulting in complex implementations are needed in order to accomplish the task (see [4] for an excellent overviw). This in turns hinders diffusion and use of these methodologies outside the original niches.

In this paper we investigate the feasibility of performing markup using the tools, syntax and semantics developed by W3C in the Semantic Web Initiative. In particular we will show how the markup problem can be see as a knowledge representation issue where elements such as, e.g. words, sentences, pages, are instances of a appropriate encoding classes and are interconnected in a Semantic Network. To express such network we will use the syntax and follow the semantics of the Resource Description Framework (RDF) [5], while the ontologies will be expressed using the Ontology Web Language (OWL) [6]. Special interest will be devoted to the "overlapping markup" problem, a classical XML markup issue which is naturally solved in the proposed encoding scheme.

We first observe that at least in theory RDF is suitable to fulfil all the task that have been traditionally done in XML (see the 2 way mappings that have proposed in [7] or in similar works). The evaluation of this novel methodology will therefore not be concerned about what aspect or kind of metadata can or cannot be encoded, as they all basically can. Rather, we will concentrate on showing how the standard tools and concepts of Semantic Web are much more fit for the task than XML and represent a next logical step, possibly resulting in novel and unexpected forms of interoperability and reuse of the encoded material.

After describing the theory, we will conclude this paper presenting the RDF Textual Encoding Framework (RDFTef), a demonstrative open source API allowing markup based on the above principles.

2. The tools of the Semantic Web: an overview

The term "Semantic Web" (SW) refers to a vision of machine readable metadata annotations that can be retrieved over the Internet much like HTML is retrieved today. Currently, by SW some indicate the tools that the semantic web community has created or is using to enable such vision, mostly within the W3C initiative [8]. As most of these conceptual tools are not in widespread use today, we here provide a brief introduction to the main ones.

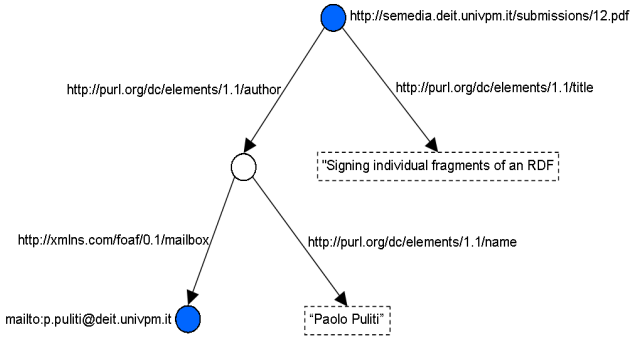
2.1 URIs – Uniform Resource Identifiers

The term “resource” refers to anything that can be somehow “identified”, that is, given a proper name. In particular, the identifiers used in both the SW and the MPEG-7 initiatives are known as Uniform Resource Identifiers (URI) and classically divided in URL (those that had a network locating mechanism, e.g. HTTP) and URN for those that didn't. URIs are divided in “schemes” (e.g. Http:, ftp:) and form “namespaces” (e.g. the set of URI of URN in the form "urn:isbn:n-nn-nnnnnn-n").

2.2 RDF

RDF data model forms the basis of the SW as defined by the W3C initiative. This model, sometime referred to as *language*, defines a method to connect resources (generally identified by URI’s) and literal (data values) using labeled arcs, thus creating semantic networks. RDF's main strength is simplicity and rigour: it is simply a network of nodes connected by directed arcs whose labels are mandated to be well defined resources (URIs) rather than simple textual tags. The use of URI as labels is a fundamental guarantee for interoperability as it gives a non ambiguous meanings and fosters reuse. Arcs are used to express properties of resources. The following illustration depicts an example annotation (or model or graph, we will use terms equivalently) where the address of a staff member is expressed in RDF:

In this graph we see how resources (blue circle nodes stating a URI) are connected to Literal nodes (dotted rectangles), and possibly via “blank” nodes (white circles). Blank nodes do not have an assigned URI and are therefore meant to be used only inside the the current model, for example to glue together statements into a logical group (in this case “the editor”).



At basic level, RDF models are uniquely defined by the set of their "triples". Triples are the individual graph arcs that connect *subjects* (URI or a Blank Node) via *predicates* (URI) to *objects* (URI, Blank Nodes or Literal). Rigorous semantics is provided in the RDF specifications to perform graph merging and to express a variety of useful structures like “bags” or “sequences”.

2.3 The SW ontology tools: RDFS/OWL

RDFS and OWL are tools that allow a formal definitions of Ontologies on the Semantic Web. RDFS provides the basic description tools, such as inheritance of classes and properties. Some form of property restrictions is also supported by RDF. Building on and extending RDFS, OWL enables more accurate and descriptive domain ontologies by providing tools such as cardinality constraints on properties, (e.g., that a Person has exactly one biological father), property transitivity, unique identifier (or key) for instances of a particular class etc. Just recently, OWL has reached the W3C recommendation status [6].

2.4 Higher level tools: Querying and Rules

Various query languages have been proposed and implemented by the Semantic Web community, specifically designed to operate on the RDF/S syntax and semantics. Among these SeRQL[9], and the newer SPARQL[10] provide powerful constructs to operate on graph structures in a "schema aware" way, that is, by taking into consideration the class hierarchies. In chapter 3 and 4 we will see usage examples of these languages applied to our textual encoding methodology.

Other initiatives have studied “rule systems” to be applied to RDF models. These are useful for applications to display automated behaviours and for graph transformations. RuleML [11] is an undergoing effort for standardizing a language to describe rules that operate on RDF/S/OWL datasets. Started in 2000, it has seen an interest shift from XML to RDF, and it's working its way to a 1.0 specification including concepts from a number of different logic paradigms. A very recent proposal partially overlapping with the RuleML specifications is SWRL [12]. Based on the extension of the OWL ontology language with metalog rules, SWRL seems at the same time simple and powerful, although it has proven to be, in its full form, indecidable.

3. Semantic Web Textual Encoding: resource centric (RDF) vs. text metadata (XML)

With the above overview in mind it is clear that RDF, although often seen in its XML serialization, is fundamentally different from XML. Rather than just providing a hierarchical structure of metadata to a text (XML), RDF deals with generic description of the relationships between resources. XML shortcomings have been evident and been object of study by the encoding communities in a problem referred to as "overlapping markup".

3.1 Existing frameworks supporting advanced (overlapping) markup

Almost all textual encoding tasks in the Humanities potentially imply the use of overlapping hierarchies of annotations. At basic level one might think of the encoding of the book structure (pages etc.) which is separated usually from the text structure encoding (chapters etc.). As a more advanced example, in any kind of text, an embedded text (e.g. a play within a play, or a song) may be interrupted by other matters; the encoder may wish to establish explicitly the logical unity of the embedded material (e.g. to identify the song as a single song, and to mark its internal formal structure).

While queries very often need to deal with overlapping hierarchies, in certain advanced use case there might even be the need for textual annotations that do so. We call these “cross concern annotations” (CCA). Lets consider the case of a book or manuscript which has been subject to

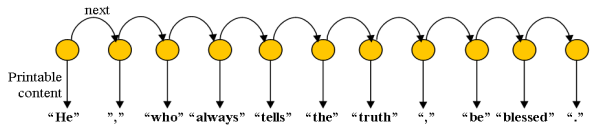


Figure 1: The raw symbol chain directly created from the textual content. Every element (word or punctuation) corresponds to a RDF node that has a property named "printable content". The value of such property is will be

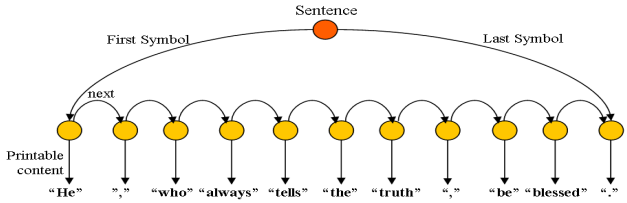


Figure 2: A sentence resource points at the first and last symbol of the underlying word chain.

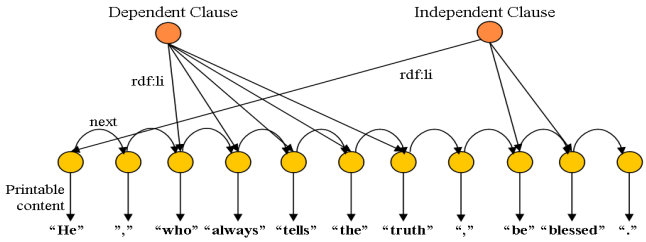


Figure 3: Clauses might not be contiguous group of symbols, so they are represented with a "bag" structure, where every symbol belonging to the clause is directly linked to it.

fire damage and successive restoration. A CCA might be used to annotate that at a specific time and due to a specific cause there were two events in time that rendered incomprehensible certain *pages*, *sentences* and individual *words*. A successive CCA might indicate that at a later time some of these were restored. The "restore" CCA would then point at the "fire" event CCA (to compensate which the recovery was needed) as well as pointers to the parts of the texts that became legible again.

Due to the XML specifications that requires a strict nesting of the elements, overlapping hierarchies and CCA are not directly supported by XML or XML tools.

This limitation has been subject of intensive study in the community, and a number of proposals have been made [3][13] (for a comprehensive overview see [4]). Although TEI specifies ways to represent overlapping markup (e.g. through the use of Joins and Milestones, as supported by our RDFTEF reference implementation), with time, the use of pure XML has been set apart in favour of richer ad hoc models, which then are filtered with pre or post processors to provide different XML compliant encoded files [3] or new markup languages all together [14]. Almost in all the proposed schema, self overlapping, schema validation and XML processing by standard tools is somehow limited or not possible.

One of the most interesting frameworks for advanced textual encoding is [15], which presents an ad hoc data structure, called GODDAG, and an ad hoc query language, EXPath, which extends the XML query language "XPath" with support for querying multiple hierarchical structures. In order to achieve output and compatibility with other formats, the GODDAG internal representation can be serialized into a series of parallel XML files.

These specific solutions seems to suitable to solve the immediate overlapping markup problem but are weak under an engineering point of view: they all propose a own set of idiosyncratic rules and semantics for the encoding. This in turns requires that developers not only need to understand new specifications but also cannot use existing tools (such as XML processors and APIs). Implementing the proposed extension to the XPath language also seems as a rather involved task, so while a JAVA version exists, it might be difficult to operate on this format with other languages.

3.2 Experiments with RDF based model for text encoding

The first step into using RDF for textual markup is making so that text and annotations themselves are "resources" identified by either blank nodes or URIs. Once these is performed, annotations will be simply states as relationships among these resources, thus naturally using RDF semantics and tools.

3.2.1 Encoding text into Semantic Web resources

One possible way of encoding a text into RDF resources is to create a "raw" RDF model where nodes represent every "printable element" in the analysed text document. These elements are usually words, punctuation or (if needed) lower level concepts such as characters, typographical signs, or, in case of encoding manuscripts, scratches and more. At this point, the natural ordering of the words can be encoded by connecting these elements in a linked list, assigning each the appropriate "next" element. Figure 1 illustrates this.

Building on top of such a printable symbol chain, higher level elements as, for example, sentences and

clauses are encoded as resources which point to the first and the last underlying raw symbol (Figure 2).

The same concept of "sequence of symbols" can then be applied also to annotation kind which have a natural ordering. "Page" annotations, for example, are resources which point to the underlying chain of "row" annotations.

For annotations that spawn non contiguous elements, such as "clauses", RDF Bags and Sequences collection tools come handy [5]. RDF Sequences and Bags allow ordered or unordered grouping of multiple elements. The Bag construct connects directly to itself all the contained elements using the RDF Syntax specified *rdf:li* property. shows its use to annotate the dependent and independent clause in a sentence.

Applying this techniques at different layered levels, a single RDF graph can then contain as many overlapping hierarchies as needed, as well as CCAs. ().

3.2.2 Toward an appropriate markup Ontology

It is important to note that the "interconnection" of resources as previously shown is in fact regulated, and validated, by a formal OWL ontology developed for this purpose. shows such such ontology, called RDF Textual Encoding Framework Ontology (RDFTef Ontology). The RDFTef Ontology is to be considered demonstrative only, as it was created just to show the basic proof of concept and support the need for basic markup and grammatical annotations for our chapter 4 use case, the Commedia of Dante Alighieri. Some classes, however, are of general use for the proposed technique such as the abstract "Symbol" class used to form the linked lists as previously shown. Symbols are the "raw" printable symbols (such as words or punctuation) but also logical symbols such as periods and propositions. The abstract class Grouping represents a generic annotations. These can be intervals, which rely on the underlying chain of symbols or bags, which form arbitrary sets. Multiple inheritance is a very useful feature of the RDF/S semantic; in this example both the class period and proposition are at the same time Symbols and Grouping, albeit supporting different properties. OWL constraints can be applied to some properties to ensure the correctness of symbol chains: the next of a page can only be a page, the first symbol of a sentence can only be a clause, and so on. OWL tools can be then used to validate an RDFTef encoded data file.

3.2.3 Querying the model: different paradigms

In this section we present two ways to query an RDFTef model: programmatically and by using different Semantic Web query languages.

Considering semantic web query languages, the first thing to note is that in their present form they all need a few adaptations, albeit simple. By design, all the considered query languages operate on fixed path length matching. This means that constructs like the linked lists (implemented by the "First_symbol" in Interval_group and "Next" property in the Symbol class) are not directly supported. Luckly, there are different solutions to this problem which are relatively straightforward.

We could use, instead of the "First_symbol" and "Last_symbol" properties, the proper RDF constructs for grouping such as rdf:Bag and rdf:Seq, as previously shown in the case of non sequential groups of symbols. In this case we could use any existing query language out of the box. We'll take as an example query "Which sentences are entirely or partially in page 2?" to be executed over a model built with the above overlapping hierarchy (Page->Rows->Words<-Sentences). This could be written, for example using the SeRQL language:

Example 1.

```
SELECT distinct SENTENCE where
            {SENTENCE} rdf:type {<ns1:Sentence>},
            {WORD} rdf:li {SENTENCE},
            {WORD} rdf:li {ROW},
            {ROW} rdf:li {<ns2:page_2>}.
```

Where the upper-case words are variables of the query, *ns1* is a sample namespace which contains the definition of classes (like *Sentence*), while *ns2* contains the instances (like *page2*), that is the RDF encoding of the text to be queried.

The SPARQL language provide facilities for defining custom operators which can be then invoked within a query using the FILTER construct. In the reference implementation, chapter 4, we provide to out SPARQL interpreter the appropriate FILTER functions to deal with this issue. In particular, we add a boolen 'BelongsTo' operator which provides the needed supports for the linked list model (first_Symbol and last_Symbol properties).

The following is the SPARQL syntax expressing the same query considered above, but can be performed on a linked list model, rather than on a Bag model:

```
SELECT ?sentence WHERE {
            ?sentence rdf:type ns1:Sentence.
            FILTER function:BelongsTo(?word, ?sentence).
            FILTER function:BelongsTo(?word, ?row).
            FILTER function:BelongsTo(?row, ns2:page_2).
            };
```

Another solution, which also enables much more powerful analysis is to explore the model programmatically, using the existing manipulation tools such as Jena[16] and Sesame[17]. While this in theory could be compared with using the DOM model exploration API available for XML, the higher expressiveness of the API and the perfect adherence of the RDF graph model with the annotation model make query by programming a much more realistic task. Given just a very basic wrapping API built on top of any RDF toolkit, such query could be performed by the following script in pseudo Java.

```
page=Model.getNode(Pagina2);
rowIterator=page.getSymbolIterator();
while (row=rowItarator.getNext) {
            wordIterator=row.getSymbolIterator();
            while (word=wordIterator.getNext()) {

period=word.getConnected("first_or_last",PeriodType);
                        if period!=null then
results.add(period); }}
```

The example above makes use of an imperative language and a graph exploration API. This is certainly not the ideal combination, albeit a very popular one.

However, once in the Semantic Web domain, a number of alternatives paradigms are more available and more suitable. Among the most powerful ones is the use of a Semantic Web aware Prolog interpreter such as SWI-Prolog [18]. Using one such language, it is possible to craft powerful rule like constructs that can be reused in building successive ones. For example, once the following

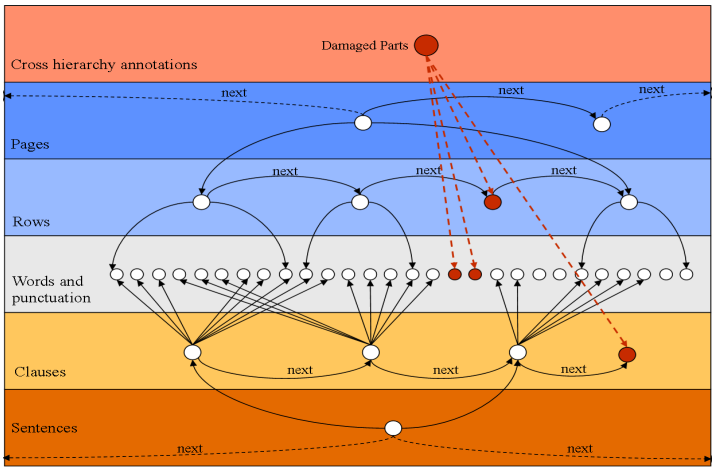


Figure 4: Different overlapping hierarchies and cross hierarchy (concern) annotations coexist and interrelate in the same RDFTef model.

general purpose constructs have been defined for handling the Interval Class:

```
// Checks if or lists the symbol follows another in the
chain
follows(X,X).
follows(X,Y):-next(X,Y).
follows(X,Y):-follows(X,Z),next(Z,Y).
// Checks if or lists the elements between 2 symbols
range_belongs(X,First,Last):-follows(First,X), not
follows(Last,X).
// Checks if or lists the symbols in a interval
belongs(Leaf,Root):-
first(Root,First),last(Root,Last),range_belongs(Leaf,Firs
t,Last).
// Checks if or lists (recursively) the leaves given a
root interval
sub_belongs(Leaf,Root):-belongs(Leaf,Root).
sub_belongs(Leaf,Root):-
belongs(NewRoot,Root),sub_belong(Leaf,NewRoot).
```

Formulating our test query simply means adding an explicit rule and firing it:

```
sentence_in_page(Sentence,Page):-

sub_belongs(Word,Page),belongs(Word,Sentence).
sentence_in_page(X,21);
```

to obtain the matching sentences. Given the nature of Prolog, the same rule, as well as those above, can immediately be used for the opposite purpose, e.g. to find, given a sentence, which page (or pages) it belongs to. Without resorting to the full power, and relative execution complexity, of Prolog, is also possible to use other reasoners such as those implemented in inside toolkits like Jena to encode similar query rules. Feasibility and limitations however will have to be evaluated with further studies and implementations.

3.3 Cooperative, incremental markup

Given that any conceptual entity (e.g. aspect of the encoding) is a resource in RDF, it might be very useful to give this entity a proper, globally addressable and "stable" URI. Once URIs are agreed upon, the SW semantics specifies the rules for document merging, immediately allowing an interesting scenario: annotations could be made cooperatively and incrementally in a distributed way. Independent documents, independently edited, could provide encodings of different aspects based on the same base URIs. At any time these could be merged into a unique document or simply taken into consideration, e.g. by a SW reasoner, as if they were one. As an example, it could be possible to link resources in the RDFTef version of Dante Alighieri's Commedia directly with the ontological concepts listed at www.semanticbible.org, possibly opening the way to interesting comparison between related text.

3.4 The role of encoding ontologies

While the graph structure provides the fundamentals of the encoding, it is the ontological aspects that probably could bring the most interesting possibilities. In general, instruments available for XML such as XPath and the relative enhancement for the overlapping annotations EXPath do not take ontological aspects into consideration. Even if XQuery supports concepts such as type extension, this is a way to provide more a validation syntax, rather than to create ontological concepts such as classes, properties and restrictions on these [19].

Using the proposed encoding scheme, the SW tool OWL becomes available. OWL, a W3C recommendation since 2004, is semantically coherent with the RDF model and enjoys ever increasing supported by APIs and understanding by the community.

OWL provides a number of extended functionalities which we will just hint at here. First of all it provides a solid base for model validation. For example, in our example annotation ontology, cardinality constraints are specified so that there has to be 1 and only 1 next for each basic symbol (except the last which points to a special "end of the document" symbol). Each interval must then have 1 and only 1 "first" and "last" and so on.

More advanced aspects of the ontology, which cannot be directly validated by the OWL description, can nevertheless be checked with simplicity if rule systems such as those previously defined are available.

Reasoners, however, go well beyond checking a model validity. When certain conditions are met or found in the model, new statements can be added or existing ones operated upon. Some of the advantages with respect to the simple XML schema are mentioned in [20]. As far as querying is concerned, the use of ontologies gives important capabilities. As a basic example, given an ontology for "manuscripts" including a taxonomy of "erasures" (e.g. from light pencil strikeout, to the word or a paragraph being ripped off the paper), one could encode the content using the most appropriate term for each case and be able to perform queries that contemplate all the sub

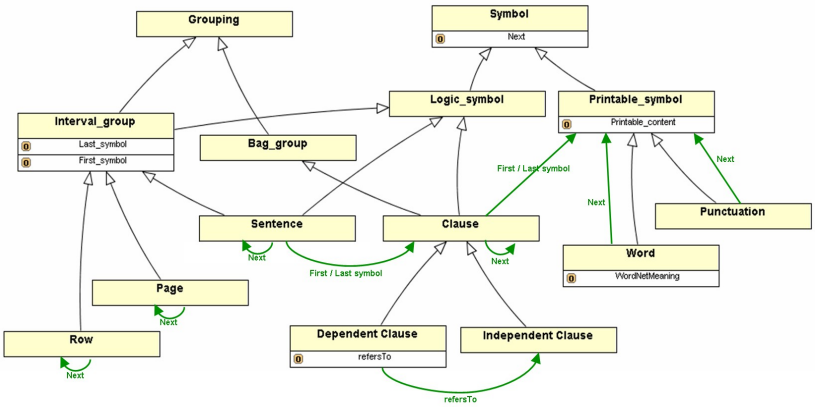


Figure 5: The experimental and demonstrative RDFTef Ontology

cases such as "*listing the erased adjectives*" in the manuscript.

4. Conclusions and future work

In this paper we presented a novel textual encoding methodology based on the languages of the Semantic Web, namely RDF, RDFS and OWL. We showed how text can be encoded into an RDF model and then queried by Semantic Web query languages as well as by using different programming paradigms (declarative or imperative). As RDF/OWL are specifically designed to allow the creation of networks of semantically rich annotations, it was clear that in theory they would be fit for the task. There was however the need to show this in practice, so we developed an open source java framework, called RDFTef, which is an implementation of the said methodology. RDFTEF currently supports importing and exporting of XML files in a subset of the TEI format, the query interface supports SPARQL Query Language.

The most important result, however, is represented by the simplicity of such implementation. In fact, as we were able to internally leverage the existing semantic APIs to a great extent, the implementation is particularly straightforward. This, we believe, shows the coherence between the task to be performed and the employed tools.

Furthermore, the proposed methodology inherits all the interoperability and support for distributed operations that the Semantic Web tools enjoy, thus opening the way to novel collaborative textual markup scenarios

As textual encoding is a complex and long debated discipline we're well aware that such a fundamentally innovative proposal faces a very long path to an actual widespread adoption. We believe however we succeeded with this study in showing that the idea is interesting and that further studies and evaluations are fully justified.

References

[1]Sperberg-McQueen C. M., Burnard "TEI P4: Guidelines for Electronic Text Encoding and Interchange." 2002

[2]C. M. Sperberg-McQueen, C. Huitfeldt "Concurrent Document Hierarchies in MECS and SGML" ALLC/ACH 1998

[3]P. Durusau, M.Brook O'Donnell "Just-In-Time-Trees (JITTs): Next Step in the Evolution of Markup?" Proceedings of 2002 Extreme Markup Languages Conference, Montreal, Canada 2002

[4]S. DeRose "Markup Overlap: A Review and a Horse" Proceedings Extreme Markup Languages 2004

[5]Editors: F. Manola, E. Miller RDF Primer2004http://www.w3.org/TR/rdf-primer/

[6]OWL Web Ontology Language Overview http://www.w3.org/2001/sw/WebOnt/

[7]S. Battle "Round-tripping between XML and RDF" Inernationa Semantic Web Conference 2004

[8]Semantic Web Activity http://www.w3.org/2001/sw/

[9]J. Broekstra, A. Kampman "SeRQL: A Second Generation RDF Query Language" 13-14 November 2003, Vrije Universiteit, Amsterdam 2003

[10]SPARQL Query Language for RDF2004http://www.w3.org/TR/2004/WD-rdf-sparql-query-20041012/

[11]D. Hirtle, H. Boley, B. Grosof, M. Kifer, M. Sintek, S. Tabet, G. Wagner "Schema Specification of RuleML" http://www.ruleml.org/0.89/ 2005

[12]I. Harrocks, P. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean "A Semantic Web Rule LanguageCombining OWL and RuleML. V0.7" http://www.daml.org/rules/proposal/ 2004

[13]P. Durusau "Implementing Concurrent Markup in XML" Extreme Markup Languages 2001, August 12-17, Montréal, Canada 2001

[14]J. Tennison, W. Piez "Layered Markup and Annotation Language" Extreme Markup Languages Conference 2002

[15]A. Dekhtyar, I. E. Iacob "A Framework For Management of Concurrent XML Markup" Data & Knowledge Engineering, Volume 52, Issue 2, February 2005 2005

[16]J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, K. Wilkinson "Jena: Implementing the Semantic Web Recommendations" HP Technical Report 2004

[17]J. Broekstra, A. Kampman and F. van Harmelen "Sesame: A Generic Architecture for Storing and Querying RDF" Proceedings of the First Internation Semantic Web Conference 2002

[18]J. Wielemaker "An Overview of the SWI-Prolog Programming Environment" WLPE'03, Mumbai, India 2003

[19]Y. Gil, V. Ratnakar "A Comparison of (Semantic) Markup Languages" Proceedings of the Fifteenth International Florida Artificial Intelligence Research Society Conference 2002

[20]M. Ferdinand, C. Zirpins, D. Trastour "Lifting XML Schema to OWL" Web Engineering - 4th International Conference, ICWE 2004, Munich, Germany, July 26-30 2004