Manuzio: An Object Language for Annotated Text Collections

[Extended Abstract] *

Marek Maurizio
Università "Ca Foscari" di Venezia
Via Torino 155
Venezia Mestre, Italy
marek@dsi.unive.it

Renzo Orsini
Università "Ca Foscari" di Venezia
Via Torino 155
Venezia Mestre, Italy
orsini@dsi.unive.it

1. INTRODUCTION

More and more large repositories of texts which must be automatically processed represent their content through the use of descriptive markup languages. This method has been diffused by the availability of widely adopted standards like SGML and, later, XML, which made possible the definition of specific formats for many kinds of text, from literary texts (TEI) to web pages (XHTML). The markup approach has, however, several noteworthy shortcomings. First, we can encode easily only texts with a hierarchical structure, then extra-textual information, like metadata, can be tied only to the same structure of the text and must be expressed as strings of the markup language. Third, queries and programs for the retrieval and processing of text must be expressed in terms of languages like XQuery [4]. In the XQuery data model, every document is represented as a tree of nodes; for this reason, in documents where parallel, overlapping structures exists, the complexity of XQuery programs becomes significantly higher.

Consider, for instance, a collection of classical lyrics, with two parallel hierarchies lyric > stanzas > verses > words, and lyric > sentences > words, with title and information about the author for each lyric, and where the text is annotated both with commentary made by different scholars, and with grammatical categories in form of tree-structured data. Such a collection, if represented with markup techniques, would be very complex to create, manage and use, even with sophisticated tools, requiring the development of complex ad-hoc software.

To overcome the above limitations due to the use of markup languages partial solutions exist (see for instance [3]), but at the expense of greatly increasing the complexity of the representation. Moreover, markup query languages need to be extended to take these solutions into consideration [1], making even more difficult to access and use such textual collections.

In the project "Musisque deoque II. Un archivio digitale dinamico di poesia latina, dalle origini al Rinascimento italiano", sponsored by the Italian MIUR, we have built a model and a language to represent repositories of literary texts with any kind of structure, with multiple and scalable annotations, not limited to textual data, and with a query component useful not only for the retrieval of information, but also for the construction of complex textual analysis applications. This approach fully departs from the markup principles, borrowing many ideas from the object-oriented models currently used in programming languages and database areas. A comprehensive description of the model, language, and system can be found in [5, 6]. The language (called Manuzio) has been developed to be used in a multi-user system to store persistently digital collections of texts over which queries and programs are evaluated. This abstract reports mainly the work done on the model and the language, since the system is still at its early stages of development with a prototypal implementation.

Appears in the Proceedings of the 1st Italian Information Retrieval Workshop (IIR'10), January 27–28, 2010, Padova, Italy.
http://ims.dei.unipd.it/websites/iir10/index.html

2. THE MANUZIO MODEL

The Manuzio model considers the textual information in a dual way: as a formatted sequence of characters, as well as a composition of logical structures called *textual objects*, similar to the content objects described in [2]. A *textual object* is a software entity with a state and a behavior. The state defines the precise portion of the text represented by the object, called the *underlying text*, and a set of *properties*, which are either *component* textual objects or *attributes* that can assume values of arbitrary complexity. The behavior is constituted by a collection of local procedures, called *methods*, which define computed properties or perform operations on the object. A textual object T is a *component* of a textual object T' if and only if the underlying text of T is a subtext of the underlying text of T'[1].

The Manuzio model can also represent aggregation of textual objects called *repeated textual objects*. Trough repeated textual objects it is possible to represent complex collections like "all the first words of each poem" or "all the first sentences of the abstracts of each article" in a simple and clean way. A *repeated textual object* is either a special object, called the *empty textual object*, or a set of textual objects of the same type, called its *elements*. Its underlying text is the composition of the underlying text of its elements.

Each textual object has a type, which represents a logical entity of the text, such as a word, a paragraph, a sentence, and so on. In the Manuzio model types are organized as a lattice where the greatest element represents the type of

[1] Differently from a substring, a subtext can comprise non-contiguous parts of a text.

the whole collection, and the least is the type of the most basic objects of the schema. Types can also be defined by inheritance, like in object-oriented languages. For instance, the types Novel and Poem are both subtypes of Work . An example of textual schema is given, by the means of a graphical notation, in Figure 1.

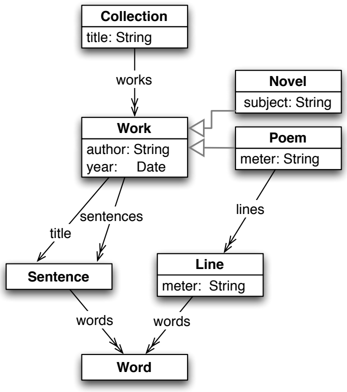

Figure 1: Example of Manuzio Model.

3. THE MANUZIO LANGUAGE

Manuzio is a functional, type-safe programming language with specific constructs to interact with persistently stored textual objects. The language has a type system with which to describe schemas as that illustrated in Figure 1, and a set of operators which can retrieve textual objects without using any external query language. A persistent collection of documents can be imported in a program and its root element can be referenced by a special variable `collection` of type `Collection`. From this value all the textual objects present in the collection can be retrieved through operators that exploit their type's structure: the *get* operator retrieve a specific component of an object, while the *all* operator retrieve recursively all the components and subcomponents of a certain type of an object. Other operators allow the creation of expressions similar to SQL or XQuery FLOWR expressions[2]. Since the queries are an integrated part of the language, they are subject to type-checking and can be used in conjunction with all the other language's features transparently.

The program in Source Code 1, for instance, assigns to a variable the first three sentences of each work. This portion of text can be subsequently refined or used in any retrieval context. In Source Code 2 a more complex example is shown, where an analysis of Shakespeare's plays extracts the top three "love speaking" characters in "A Midsummer Night's Dream". The results of such code are then reported in Source Code 3.

```
let most_relevant_sentences =
        select all SENTENCE 1..3 of works of c;
```

Source Code 1: Retrieve the most relevant sentences of each work.

[2]The full syntax and semantics of the Manuzio language can be found in [6].

```
let play =
    p in (get plays of collection)
    where p.title = "A Midsummer Night's Dream";
let loveSpeeches =
    s in (getall Speech of play)
    where some w in (getall Word of s)
    with (get stem of w) = "love";
let love_speech_count_by_speaker =
    select {speaker = s.speaker, n=(size of s.partition)}
    from s in (speeches groupby speaker);
output "The top 3 love spekaers are:";
output love_speech_count_by_speaker[1..3];
```

Source Code 2: Compute a new structure of the most love-speaking characters.

```
The top 3 love speakers are:
[{speaker="LYSANDER", n=17},
 {speaker="OBERON", n=13},
 {speaker="HERMIA", n=12}]
```

Source Code 3: Results of Source Code 2.

4. CONCLUSIONS

To evaluate the usefulness of our approach a first prototype of the Manuzio language has been developed by mapping the textual objects onto a relational database system. We are aware that a great deal of work on data representation and query optimization must yet be done to provide a satisfying performance for large collections of texts. However, we think that work on modeling and linguistics aspects of retrieval of texts and computations over them is very important, and prerequisite to enrich the solutions offered by research areas such as information retrieval and digital libraries. In particular, we believe that our language allows the user to take into account structural information in a simple way in queries, and this could improve the quality of their results (a term is certainly more significant when used in a title instead that in a footnote).

5. REFERENCES

[1] Alex Dekhtyar, Ionut E. Iacob, Kevin Kiernan, and Dorothy C. Porter. Extended xquery for digital libraries. In *JCDL '06: Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries*, pages 378–378, New York, NY, USA, 2006. ACM.

[2] S.J. DeRose, D.G. Durand, E. Mylonas, and A.H. Renear. What is text, really? *ACM SIGDOC Asterisk Journal of Computer Documentation*, 21(3):1–24, 1997.

[3] Steven J. DeRose. Markup overlap: A review and a horse. In *Extreme Markup Languages*, 2004.

[4] H. Katz and D.D. Chamberlin. *XQuery from the experts: a guide to the W3C XML query language.* Addison-Wesley Professional, 2004.

[5] Renzo Orsini Marek Maurizio. A model and query language for literary texts. Technical Report CS-2009-4, Dipartimento di Informatica, Università Ca' Foscari di Venezia, 2009.

[6] Marek Maurizio. *Manuzio: an Object Language for Annotated Text Collections.* PhD thesis, Dipartimento di Informatica, Università Ca' Foscari di Venezia, 2009.