Creation and Maintenance of Multi-Structured Documents

Pierre-Édouard Portier
Université de Lyon, CNRS
INSA-Lyon, LIRIS, UMR5205, F-69621, France
pierre-edouard.portier@insa-lyon.fr

Sylvie Calabretto
Université de Lyon, CNRS
INSA-Lyon, LIRIS, UMR5205, F-69621, France
sylvie.calabretto@insa-lyon.fr

ABSTRACT

In this article, we introduce a new problem: the construction of multi-structured documents. We first offer an overview of existing solutions to the representation of such documents. We then notice that none of them consider the problem of their construction. In this context, we use our experience with philosophers who are building a digital edition of the work of Jean-Toussaint Desanti, in order to present a methodology for the construction of multi-structured documents. This methodology is based on the MSDM model in order to represent such documents. Moreover each step of the methodology has been implemented in the Haskell functional programming language.

Categories and Subject Descriptors

H.3.7 [**Information Storage And Retrieval**]: Digital Libraries

General Terms

Human Factors, Algorithms

Keywords

Digital libraries, overlapping hierarchies, XML, Haskell

1. INTRODUCTION

This work introduces a new problem: the *construction* of multi-structured documents. The multiple uses of a same document has led to a proliferation of documentary structures (physical, logical, semantic, ...). Multi-structured documents [2] have to be analysed in their historical context where the most used formalisms for documents representation (first SGML then XML) implied tree structures. That is why this problem has so far been considered under the technical point of view of overlapping hierarchies [9].

By studying the construction of multi-structured documents we are close to the daily practices of users who are writing documents. Our work is based on experience gained working with philosophers who are building a digital edition of the handwritten archives of French philosopher Jean-Toussaint Desanti (1914-2002). Digital editing covers the whole editorial, scientific and critical process that leads to the publication of an electronic resource. In the case of manuscripts, editing mainly consists in the transcription and critical analysis of digital facsimiles, that is to say the creation of a textual document associated with the digitized images of a handwritten manuscript. We found that the problem of constructing multi-structured documents was at the heart of their work. Indeed, they need to let coexist a multiplicity of structures in order to be able to access a document according to many points of view. As we can see, our work does not consist in the conception of a model for the representation of multi-structured documents, but in the development of a methodology that promotes the emergence of multiple structures in a multi-users context.

In Section 2, we describe existing work that manage multi-structured documents. Then, in Section 3 we use one of them, the MSDM model, to present a methodology for the construction of multi-structured documents.

2. EXISTING SOLUTIONS

We studied existing solutions to the problem of representing multi-structured documents. The goal of this study is to show that there are no solutions to the problem of constructing multi-structured documents and also to determine if we could reuse an existing solution as a basis for our work, and finally to define more precisely what we mean by the "construction of a multi-structured document". Indeed the construction model of a document will depend on his representation model.

We divide the set of existing solutions to the problem of representation of multi-structured documents into four classes. First, a historical solution: CONCUR [7], then adhoc solutions as proposed by the TEI (Text Encoding Initiative) consortium [4], next models not compatible with the XML language ([8], [15], [13], [16]), finally those compatible with XML ([14], [10], [3], [12], [1]). Each solution is analysed according to six dimensions: expressiveness of the model determines if there is an explicitly defined model and if it responds to the problem of static representation of multi-structured documents ; genericity of the model determines, when a model exists, if we can modify it in order to manage problems outside of the initial scope of multi-structured documents representation ; quality of the implementation measures the care taken to develop an effective implementationa

DocEng'09, September 16–18, 2009, Munich, Germany.

; compatibility with XML tools determines if it is possible to integrate the solution with the numerous existing XML tools used to manage XML documents (especially typing tools such as XML Schemas, ...) ; query mechanisms for multi-structured documents ; change management in data or structures, analyses if the model is robust to change. We will not here analyse each solution but only the one we finally kept.

MSDM, MultiX.

MSDM [5] is a model used for the representation of multi-structured documents written by N.Chatti. An instance of this model, called MultiX, is expressed in the XML. It belongs to the category of stand-off markup solutions where content is isolated in a base structure, and documentary structures are built by references to the base structure.

In this model, a document is a graph $\mathcal{D}$ composed of:

- a set of nodes $\mathcal{BS}$ also called the base structure
- a family $(DS_j)_{j \in J}$ of trees also called documentary structures

Moreover, $\forall j \in J$, there is a relationship R_j that associates each node of DS_j with a subset of $\mathcal{BS}$; for each leaf of DS_j this subset must be non empty. Figure 1 illustrates each element of the model.

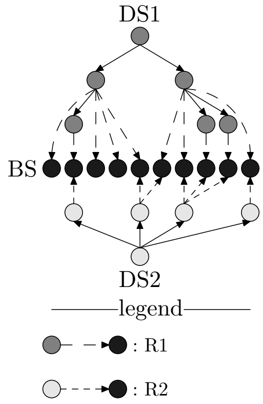


Figure 1: Illustration of the MultiX^2 model

Table 1 summarises the analysis by affecting, as objectively as possible, a score from 0 to 3 to each criterion (model expressivity, quality of implementation, use of standard XML tools, query mechanisms, management of changes in data and structures) and for each solution. For readability, maximum scores have been underlined. We chose MSDM as the representation model on which we built our methodology since, based on the stand-off markup technique, it was simple enough and yet well defined. In the next section we use the MSDM model in order to explain what is meant by the "construction of multi-structured documents".

Table 1: Rating of existing solutions to the multi-structured documents problem

model		expressivity	genericity	implementation	XML tools	query mechanisms	structure and data changes
TEI Guidelines	redundant encoding	0	0	1	1	0	0
	empty elements	0	0	1	0	0	0
	virtual elements	0	0	1	0	0	0
	stand-off markup	0	0	1	3	0	1
CONCUR		2	0	1	0	0	2
MuLaX		2	0	2	1	1	2
TexMECS		2	0	2	0	1	2
LMNL		3	0	2	0	0	2
Delay Nodes		2	1	2	3	2	0
Annotations Graphs		3	2	2	1	2	2
RDF (RDFTEF)		3	3	1	1	1	2
MonetDB		1	0	3	3	3	1
MCT		2	2	2	3	2	1
MSXD		3	2	2	3	3	0
GODDAG		3	3	2	2	3	2
MSDM/MultiX		3	3	2	3	3	2

3. CONSTRUCTION OF MULTI-STRUCTURED DOCUMENTS

We claim that the study of the construction of documentary structures is a way to approach the user interpretation of a document. For example, numerous critical edition projects begin with manuscripts images they then transcribe and annotate. During these operations, the documents will be manipulated by numerous users and under a multiplicity of perspectives that mostly depend on how the documents are used. We claim that most of these perspectives can be revealed by the distinction of structures: an operation that split an annotation vocabulary into sub-vocabularies, thus adding a new structure to the document. Thereby, the methodology we now present promotes the construction of a multiplicity of structures that should reflect the perspectives adopted by the users while accessing the documents. This methodology consists of three categories of methods :

- detection of needed restructuring and automatic differentiation of structures. As we will see, the overlapping hierarchies problem becomes an element of this category of methods.
- presentation to the user of the results of automatic restructuring.
- creation of a social network of documents authors in order to encourage argument about and sharing of annotation vocabularies

3.1 Restructuring stage

We analyse the conditions under which it is necessary to build a new documentary structure. For clarity, and since we know the field, we use an example taken from critical electronic edition of manuscripts. We suppose that for the transcription of a manuscript the researchers have the elements defined by the TEI. For example: pages, paragraphs and equations have been correctly tagged until a paragraph overlaps two pages (see dotted edges of Figure 2). It is then necessary to distinguish two structures so that pages and paragraphs do not share the same structure. The creation of a new structure is a purely formal operation (see Figure 3) consisting in the transformation of a graph into two trees. It should be noted that the meaning of the tags is not taken into account during this transformation: the pages could have been isolated inside a new structure while the equations and paragraphs would have been kept together.

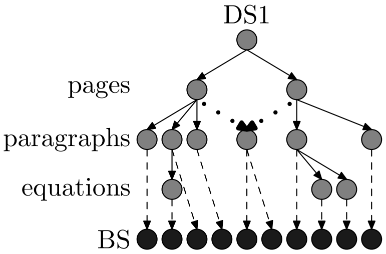


Figure 2: Restructuring is necessary (a paragraph overlaps two pages)

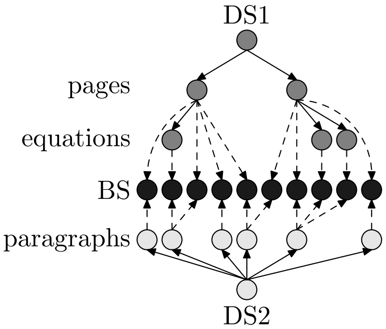


Figure 3: Automatic restructuring (two structures are distinguished)

We wrote an implementation of our methodology in the Haskell pure and statically typed functional programming language [11]. As a pure language, Haskell keeps side effects under the control of a class of type constructors called Monad, in practice this allows us to ensure that a valid document will always be transformed into a valid document. Since our methodology introduces numerous documents transformations, this property is very instersting. Moreover, as a statically typed language, the type signatures offer a good documentation for each of our functions. For example, the *addTag* function tries to add a tag to a structure, if the addition does not imply overlapping then the modified structure is returned, else a pair of structures is returned: the first structure is the original one except that every instances of the added tag have been transfered to the second structure:

```
addTag :: Taggee -> Structure ->
          Either (Structure,Structure) Structure
```

3.2 Integration of the user in the restructuring process

The automatic restructuring introduced above can be the occasion for a user to make modeling choices. For example, he can ask for the creation of a new mathematical structure for the equations and rename the structures (see Figure 4). *moveTag* is the main function offered to the user for reacting to the automatic restructuring, it allows him to move all the instances of a tag from one structure to another one. The function may fail if it introduces overlapping hierarchies (thus the Maybe type constructor).

```
moveTag :: TagId -> Structure -> Structure ->
           Maybe (Structure, Structure)
```

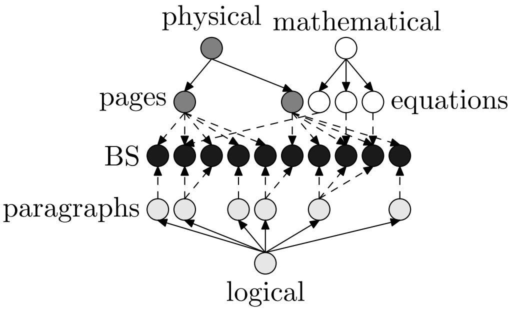


Figure 4: Intervention of a user (Three structures are named and distinguished)

3.3 Recommendation system for documents authors

We now try to involve even more the author of a document in the process of maintaining a coherent multiplicity of structures. This is why we promote the emergence of a social network of documents authors. The recommendation mechanism that helps to the construction of this network define two users as related, insofar as they are editing specific documents, if the implied tags trees of their structures are similar. We imagine three users, each creating documents containing mathematical notations. For each of them, a mathematical structure emerged from their annotation operations (as described in the previous sections). Users 1 and 2 have already decided to merge their tag hierarchies. The tag hierarchies are given below:

User 1 and 2:	User 3:
• theorem – statement – proof • lemma – statement – proof • cocycle • cobordism	• proposition – proof – operators • cohomology – cocycle

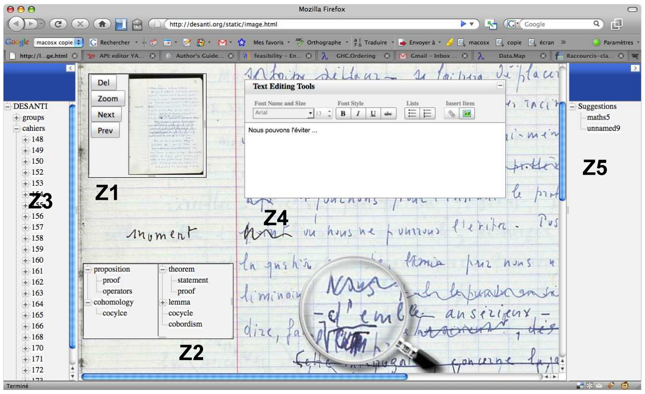
If these two hierarchies were detected as similar enough, each user would be proposed to ask the other users the authorization to merge their hierarchies. Thus, communities of users appear, centered on their annotation practices. In this previous example, users seem to work on the same kind of documents, but user 3 perspective may be formal logic whereas users 1 and 2 refer to a more traditional vocabulary for the description of proofs. Since the tips the users receive while annotating a document come from the hierarchy of tags associated with the current structure, once the merge is accepted, the users may align their annotation vocabularies or at least discuss their practices.

We have to compute the distance between every pair of implied tag structures. We choose a very straightforward editing distance equals to the number of ”add” and ”delete” operations needed to transform one set of tags into another. It does not take into account the structure of the tags and has for only purpose to guide the user towards other possibly related users and look at their documents!

4. A PROTOTYPE IMPLEMENTATION OF THE METHODOLOGY

Figure 5 is a screenshot of the client application written in javascript and running inside a Web browser: Z3 is a hierarchy of all the documents of the archive, it gives the researchers the synoptic view they need ; Z1 is a draggable navigator obtained by clicking on an element of the hierarchy Z3, it allows one to navigate among the images of the pages of a collection ; Z4 is an editor for the transcription ; Z5 is the set of recommendations for tag hierarchies similar to the one implied by the current documentary structure ; Z2 is the comparison frame obtained when the user click on one of the recommendations, it allows him to decide if he wants to merge his tags structure with the one suggested.

Moreover, we provide all the Haskell functions introduced above as a Web service that follows the REST [6] design pattern. We give as an example the HTTP operation used for tagging a new "equation" in the mathematical structure of a notebook. All we have to do is send a POST request with the required tag to the resource identified by the URL "http://desanti.org/cahiers/148/structures/math/taggees".



Nous pouvons l'éviter ...

Figure 5: Screenshot of a prototype implementation of the three stages of our methodology

5. CONCLUSIONS

We have identified a new problem: how to build multi-structured documents? This allowed us to take over the old issue of multi-structured documents as we pulled away from its technical formulation and bring ourselves closer to the uses of those building documents. We have shown that, although the enforcement of tree structures was for a long time considered as the crux of the problem, we could place it at the heart of a new solution where the emergence of overlapping hierarchies triggers the creation of a new structure that has to be validated by the user. Thus we managed to provide a methodology that addresses the needs of humanities researchers by promoting and maintaining a multiplicity of stuctures. Moreover, we developed a prototype implementation in the Haskell functional programming language of the algebraic operations described in the article. These operations are provided through a Web interface using the HTTP protocol in accordance with the REST design pattern.

6. REFERENCES

[1] W. Alink, R. A. F. Bhoedjang, A. P. de Vries, and P. A. Boncz. Efficient xquery support for stand-off annotation. In *XIME-P*, 2006.

[2] E. Bruno, S. Calabretto, and E. Murisasco. Documents textuels multi structurés : un état de l'art. *Revue i3*, 7(1), Mar. 2007.

[3] E. Bruno and E. Murisasco. Multistructured xml textual documents. *GESTS International Transactions on Computer Science and Engineering*, 34(1):200–211, november 2006.

[4] L. Burnard and S. Bauman. Tei p5: Guidelines for electronic text encoding and interchange. 2007.

[5] N. Chatti, S. Kaouk, S. Calabretto, and J.-M. Pinon. MultiX: an XML-based formalism to encode multi-structured documents. In *Proceedings of Extreme Markup Languages'2007, Montréal (Canada)*, Aug. 2007.

[6] R. T. Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, 2000.

[7] M. Hilbert, A. Witt, M. Québec, and O. Schonefeld. Making concur work. In *Extreme Markup Languages*, 2005.

[8] C. Huitfeldt and M. Sperberg-McQueen. Texmecs: An experimental markup meta-language for complex documents. 2003.

[9] I. E. Iacob and A. Dekhtyar. Processing xml documents with overlapping hierarchies. In *JCDL '05: Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*, pages 409–409, New York, NY, USA, 2005. ACM.

[10] H. V. Jagadish, L. V. S. Lakshmanan, M. Scannapieco, D. Srivastava, and N. Wiwatwattana. Colorful xml: one hierarchy isn't enough. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 251–262, New York, NY, USA, 2004. ACM.

[11] S. P. Jones, editor. *Haskell 98 Language and Libraries: The Revised Report*. http://haskell.org/, September 2002.

[12] J. Le Maitre. Describing multistructured xml documents by means of delay nodes. In *DocEng '06: Proceedings of the 2006 ACM symposium on Document engineering*, pages 155–164, New York, NY, USA, 2006. ACM.

[13] K. Maeda, S. Bird, X. Ma, and H. Lee. Creating annotation tools with the annotation graph toolkit. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, Apr 2002.

[14] C. M. Sperberg-McQueen and C. Huitfeldt. Goddag: A data structure for overlapping hierarchies. In *DDEP/PODDP*, pages 139–160, 2000.

[15] J. Tennison and W. Piez. The layered markup and annotation language (lmnl). In *Extreme Markup Languages*, 2002.

[16] G. Tummarello, C. Morbidoni, and E. Pierazzo. Toward textual encoding based on rdf. In *ELPUB*, pages 57–63, 2005.