Encoding and Querying Multi-Structured Documents

Noureddine Chatti, Sylvie Calabretto, Jean-Marie Pinon

LIRIS-INSA de Lyon
7 Av. Jean Capelle, F-69621 Villeurbanne Cedex, France
e-mail: {noureddine.chatti; sylvie.calabretto; jean-marie.pinon}@insa-lyon.fr

Abstract

This paper concerns the document multi-structuring issue. For various use objectives, many distinct structures may be defined simultaneously for the same original document. For example, a document may have a first structure for logical content organisation (logical structure), and a second structure to express a set of content formatting rules (physical structure). We have already proposed a generic model, called MSDM, for the multi-structured documents, in which several important features were established. In this paper, we address the encoding problem of this kind of documents. We present a new formalism, called MultiX, which allows encoding the multi-structured documents efficiently. This formalism is based on the MSDM model and uses XML syntax.

Keywords: multi-structured document; XML; MultiX; multi-structured document querying; XQuery

1 Introduction

The document structuring is used for different purposes. Document exchange, integration and information retrieval are examples of application domains where the document structuring is used. Moreover, the document is actually a vehicle of various media types. The multitude of purposes and the diversity of document content types led to different structuring needs. Several structure types, such as physical structure, logical structure and semantic structure [1], [1, 2], have been defined for several specific uses. There are also other structure types defined to characterise the document content. An example of these structure types may be the spatiotemporal structure attached to multimedia documents.

The same document may be used in several applications and purposes. Then, different structures may be defined simultaneously for the document content. In this case, these structures are called concurrent or parallel, since they share the same content. For example, humanities and more particularly the study of mediaeval manuscripts, imply concurrent hierarchies or structures. Indeed, we can consider two main structures on manuscripts that overlap: the manuscript book structure (a sequence of columns and lines) and the "syntactic" structure (a sequence of phrase and words). Another structure in this domain can be the "damaged" structure (a sequence of damaged elements). The TEI guidelines [4] provide various examples of possible multiple structures. Among them, we can mention: in verse drama, the structure of acts, scenes and speeches often conflicts with the metrical structure.

In a more general approach, we qualify by multi-structured a document having multiple structures linked between them through a shared content or other inter-structural relations. We have proposed in [5] a generic model called MSDM (Multi-Structured Document Model), in which we define the notion of multi-structured document. In this paper we deal with the multi-structured documents encoding issue. Often, parallel structures are encoded separately in distributed XML files. Encoding these structures in separate files has many disadvantages, such as content redundancy and interdependency breaking. This solution implies significant documents management problems.

To address this problem, it may be interesting to encode all structures in a same XML document (one files for all concurrent structures), by superposing them to avoid content redundancy. As a matter of fact, it is very difficult to encode concurrent structures in a single XML file. In fact, often, the result of the structures superposition cannot be a well formed XML document due to the structures interlacing (overlapping problem). XML is based on a tree model in which elements cannot overlap. The XML tree model is suitable only for a single hierarchy. We propose in this paper a new formalism called MultiX, which allows encoding the multi-structured documents efficiently. MultiX, which uses XML syntax, is based on a generic model called MSDM that we have already presented in [5].

After a survey of the main related works, we will present, in section 3, a short description of the MSDM model. In section 4, an example of document having multiple XML structures will be presented. This example will be used in section 5 to illustrate the MultiX formalism. Section 6 will be dedicated to the multi-structured documents querying issue.

2 Related Work

The problem of concurrent structures encoding has attracted many attentions. In this domain, several approaches have been proposed. The CONCUR option [6] is an SGML functionality which allows referencing of several parallel DTDs for the same content. In such SGML document, all structures cohabit in a single file. In this file, the first structure is encoded in a standard way, and for every added structure, a special prefix, denoting the reference to the corresponding DTD, is associated with each start tag. This solution is interesting but it was rarely implemented. For XML, that does not support multiple structuring, the problem is more persistent. In the XML version of the TEI guidelines, several methods have been proposed to allow encoding of multiple hierarchies [4]. These methods consist in fragmenting elements which do not nest within others. The TEI proposals cannot answer the general problem of the multiple structures encoding because they are not based on appropriate and clear models. To bridge the gaps of existing mark-up languages, some other works have been carried out in order to define new syntaxes. MECS (Multi-Element Code System) [7] was the first proposed language which allows overlapping between elements. TexMECS [8] is based on MECS language, but it is more complex. This language defines complex structures where elements can have multiple parents. LMNL (Layered Mark-up and aNnotation Language) [9] defines a specific syntax based on the notion of range, allowing the encoding of multiple structures where elements can overlap. Due to their complexity and incompatibility with XML syntax, these languages remained at experimental stages.

More recently, the use of the RDF formalism to overcome the overlapping problem has been studied in [10]. This method takes advantage of the RDF graph model, which may be exploited to encode complex structures with overlapped elements. Indeed, RDF allows expressing graph structures, but it is not appropriate for documents structuring. The use of RDF to encode concurrent structures is not intuitive and requires a lot of adaptation effort.

3 The Multi-Structured Document Model

To answer the problem of multiple structuring, we have proposed a specific model, called Multi-Structure Document Model (MSDM) [5]. In MSDM, the problem is approached in a more general way. In fact, we suppose that structures can share just some content fragments, and not necessarily exactly the same content. For our model, concurrent structures are a particular case of multi-structured documents. In this model, which is inspired by the model defined in [11], a multi-structured document is defined using the following notions:

- **Document Structure (DS)**: this is a description of a document content defined to a specific use. Such structure may be, for example, a physical structure defined for a presentation goal.
- **Base Structure (BS)**: this structure is visible only internally within the multi-structured document. It is defined strictly in order to organize the content in disjoint elementary fragments. These fragments serve to reconstitute, by composition, the original content associated initially to the document structure elements.
- **Correspondence**: a correspondence is a relation between two elements of two distinct structures. The source of a correspondence is always an element of a document structure. If the correspondence target is an element of the base structure the correspondence is noted $DS \rightarrow BS$. This kind of correspondence associates an element of a document structure to its content in the base structure. For example, in Figure 1 the first correspondence on the left associate the text content “a b” to the source element in the document structure. When the correspondence target belongs to a document structure the correspondence is noted $DS \rightarrow DS$. The correspondences $DS \rightarrow DS$ allow to make some hided relations between document structures explicit. Such correspondence may be used to express a synonymy relation between two elements for example.

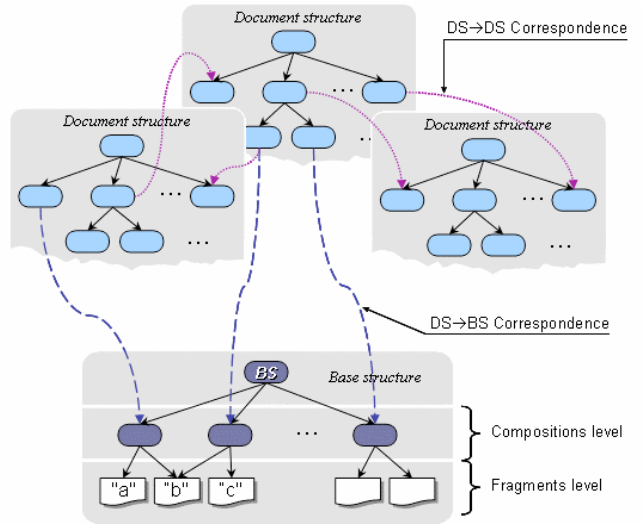

Figure 1: Illustration of the multi-structured document model

As shown in Figure 1 a multi-structured document is defined by a set of document structures, a base structure and a set of correspondences. In brief, a multi-structured document may be defined as the following triplet: $<BS, \{DS\}, \{DS \rightarrow BS, DS \rightarrow DS\}>$.

4 Example of a Document with Multiple Structures

We present here an example of a document having multiple structures, which will serve later to illustrate the MultiX formalism. We consider an image showing a fragment of an old manuscript presented in [12].

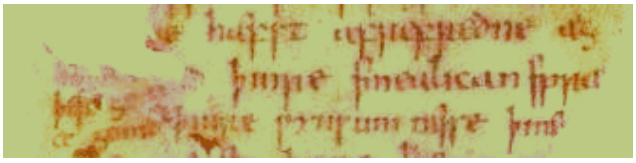

Figure 2: A fragment of an old manuscript

We consider four XML structures for this fragment. In the first structure, the original text of the manuscript fragment is transcribed by marking up the information about the lines breakdowns. This is the physical structure:

```
<lines>
        <line n="1">hu þu me hæfst afrefredne æg</line>
        <line n="2">þer ge mid þinre smealican spræ</line>
        <line n="3">ce, ge mid þinre wynsumnesse þines</line>
</lines>
```

The second structure marks all the words of the text, it is the lexical structure:

```
<words>
        <w>hu</w><w>þu</w><w>me</w><w>hæfst</w><w>afrefredne</w><w>ægþer</w>
        <w>ge</w><w>mid</w><w>þinre</w><w>smealican</w><w>spræce</w><w>ge</w>
        <w>mid</w><w>þinre</w><w>wynsumnesse</w><w>þines</w>
</words>
```

The third structure marks all the sequences of damaged characters. The <res> element contains damaged characters which have been restored from other manuscripts:

```
<damaged>
<res>þu m</res><dmg>er</dmg><dmg>mid</dmg><dmg>æ</dmg><dmg>g</dmg>
<dmg>þ</dmg><dmg>re</dmg><dmg>e</dmg><res>s</res>
</damaged>
```

The fourth structure describes particular image regions of the original manuscript fragment.

```
<text-regions>
<image src="manuscript.png">
        <region num="reg.1" description="first line">
                <zone x1="43" y1="50" x2="460" y2="94"/>
        </region>
        <region num="reg.2" description="second line">
                <zone x1="43" y1="108" x2="486" y2="152"/>
        </region>
        <region num="reg.3" description="third line">
                <zone x1="43" y1="166" x2="536" y2="210"/>
        </region>
        <region num="reg.4" description="word on two lines, 1 and 2">
                <zone x1="410" y1="50" x2="460" y2="94"/>
                <zone x1="43" y1="108" x2="93" y2="152"/>
        </region>
        <region num="reg.4" description="word on two lines, 2 and 3">
                <zone x1="'414" y1="108" x2="486" y2="152"/>
                <zone x1="43" y1="166" x2="72" y2="210"/>
        </region>
</image>
</text-regions>
```

In the next section, we will show how to encode with the MultiX formalism the multi-structured document created from these four structures. Within this document the first three structures will share text content fragments through the base structure (*BS*). *DS→DS* correspondences will be created between the "text regions" structure and the first three structures. In particular, these correspondences will allow to locate on image the transcribed lines, the words which are on two lines, and the damaged characters. Figure 3 shows a global view of the resulting multi-structured document.

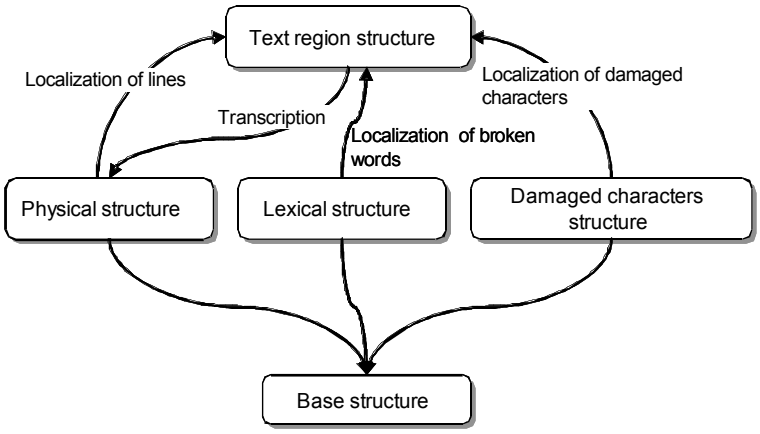

Figure 3: The multi-structured document created from the above structures

5 MULTIX

The MultiX formalism is an XML application based on the MSDM model. It allows to serialize a multi-structured document in a well formed XML document. A multi-structured document encoded with this formalism, that we call *MultiX document*, is composed mainly of three parts: the document structures (*DS*), the base structure (*BS*) and the correspondences. A MultiX document must have the following skeleton:

```
<msd:MSD name="manuscript" xmlns:msd="http://www.msdm.org/2006/MULTIX/">
<msd:DS name="name of the first document structure">
        <!-- here the encoding of the first document structure -->
</msd:DS>
<msd:DS name=" name of the second document structure ">
        <!-- here the encoding of the second document structure -->
```

```
</msd:DS>
<!-- others document structures -->
<msd:BS>
        <!-- the encoding of the base structure -->
</msd:BS>
<msd:correspondences>
        <!-- here the correspondences definitions -->
</msd:correspondences>
</msd:MSD>
```

5.1 Encoding of the Base Structure

Within the multi-structured document, the base structure serves to organize the contents in a set of disjoint fragments, from which several compositions may be created to rebuild original PCDATAs in document structures. Mainly, the base structure is defined to avoid the content redundancy, by fragmenting all shared content. However, the base structure may also contain unshared contents fragments (fragments belonging to only one document structure). When we use these fragmentations, document structures can share several contents fragments in different compositions. The contents fragments set of the base structure is obtained by superposing all PCDATAs, which have shared segments. For example, if we superpose the first line in the physical structure with the first six words in the lexical structure and the first element in the damaged structure, we obtain the contents fragments set: {“hu”, “þu”, “m”, ”e”, “hæfst”, “afrefredne”, “æg”}. This is the minimal set of disjoint content fragments which recover all superposed PCDATAs. For the multi-structured document in Figure 3, the contents fragments set in the base structure is encoded with MultiX formalism as follows:

```
<msd:BS>
        <msd:fragments>
                <msd:frag id="F1">hu</msd:frag><msd:frag id="F2">þu</msd:frag>
                <msd:frag id="F3">m</msd:frag><msd:frag id="F4">e</msd:frag>
                <msd:frag id="F5">Hæfst</msd:frag><msd:frag id="F6">afrefredne</msd:frag>
                <msd:frag id="F7">æg</msd:frag><msd:frag id="F8">þ</msd:frag>
                <msd:frag id="F9">er</msd:frag><msd:frag id="F10">ge</msd:frag>
                <msd:frag id="F11">mid</msd:frag><msd:frag id="F12">þinre</msd:frag>
                <msd:frag id="F13">smealican</msd:frag><msd:frag id="F14">spr</msd:frag>
                <msd:frag id="F15">æ</msd:frag><msd:frag id="F16">ce</msd:frag>
                <msd:frag id="F17">g</msd:frag><msd:frag id="F22">e</msd:frag>
                <msd:frag id="F18">mid</msd:frag><msd:frag id="F23">þ</msd:frag>
                <msd:frag id="F19">in</msd:frag><msd:frag id="F24">re</msd:frag>
                <msd:frag id="F20">wynsumnesse</msd:frag><msd:frag id="F21">þin</msd:frag>
                <msd:frag id="F25">e</msd:frag><msd:frag id="F26">s</msd:frag>
        </msd:fragments>
        </msd:compositions>
                < !-- compositions -->
        </msd:compositions>
</msd:BS>
```

As shown above, the base structure (msd:BS elements) contains two elements: msd:fragments and msd:compositions. The first element contains all contents fragments (msd:frag elements) which may be shared between document structures. The second element contains definitions of fragments compositions (msd:comp elements), which will be associated to document structures through *DS→BS* correspondences. For example, the composition allowing the reconstitution of the PCDTATA of the first line element in the physical structure is encoded in MultiX as follows:

```
<msd:comp id="C1" refs="F1 F2 F3=F4 F5 F6 F7"/>
```

The id attribute is used to identify the composition element. The refs attribute value is a sequence of fragments identifiers separated by whitespace or the symbol “=”. Each composition corresponds to a string obtained by concatenating contents fragments identified in the refs attribute. When two identifiers are separated by a whitespace the corresponding contents fragments are concatenated by inserting a whitespace between them. For example, the attribute value refs=“F1 F2” corresponds to the string “hu þu”. When the symbol “=” is used a simple concatenation is applied. For example, the attribute value refs=“F3=F4” allows to obtain the string “me”. The symbol “=” is useful for gluing together pieces of a fragmented word.

There is another useful symbol which can be used in the refs attribute at the beginning or/and at the end of the identifiers sequence. This symbol is "%" and it indicates the position (start or/and end of the composition) where a whitespace must be inserted. In our example the character "," contained initially in the PCDATA of the third line in the physical structure, is unshared. Then, it is not necessary to put it in the base structure. If we let the character "," at his original position in the physical structure, the full PCDATA may be obtained by concatenating three substrings. The first substring is "ce" obtained through a *DS→BS* correspondence to the following composition:

```
<msd:comp id="C3" idrefs="F16"/>
```

The second substring is the character ",". The third substring must be " ge mid þinre wynsumnesse þines" which may be obtained through a *DS→BS* correspondence to the following composition:

```
<msd:comp id="C4" idrefs="%F17=F22 F18 F23=F19=F24 F20 F21=F25=F26"/>
```

We have placed the symbol "%" at the beginning of the identifiers sequence to insert a whitespace between the character "," and the word "ge", what allows respecting the original disposition. The use of the symbol "%" is more relevant when a substring obtained from a composition is preceded or followed by a word. The msd:compositions element of the base structure must contain all needed compositions allowing the rebuilding of all fragmented PCDATAs.

5.2 Encoding of Document Structures and Correspondences

A MultiX document may have one or several document structures encoded within the msd:DS elements. A document structure in MultiX is similar to his representation in a standard XML format with a little difference. The fragmented PCDATAs are replaced by references allowing the linking through *DS→BS* correspondences to compositions in the base structure. Other MultiX elements are also inserted in document structures to create *DS→DS* correspondences.

5.2.1 THE *DS→BS* CORRESPONDENCES

In general, a correspondence has a source, a target and a label. For the *DS→BS* correspondences the target is always a composition in the base structure. The source of this correspondence type represents the position, in document structure, where will be placed the composition located by the correspondence target.

In MultiX, a correspondence may be defined internally within the document structure, or externally in the msd:correspondences element. For an *internal correspondence* the msd:clink element defining the correspondence is placed at the source position. The following *DS→BS* correspondence is defined internally and allows associating the composition identified by "C1" to the first line element in the physical structure.

```
<line n="1"><msd:clink target="BS" label="text content" to="C1"/></line>
```

The target attribute of the msd:clink element indicates the type of correspondence, "BS" for *DS→BS* correspondences, and "DS" for *DS→DS* correspondence.

For an *external correspondence* the source is located by the msd:anchor element which is inserted at the desired position in the document structure. For example, to rebuild the original PCDATA of the third line element in the physical structure, we can define two external correspondences by inserting anchors in appropriate positions:

```
<line n="3"><msd:anchor id="A.lne.3_1"/>,<msd:anchor id="A.lne.3_2"/> </line>
```

And then creating the msd:clink elements as follows:

```
<msd:correspondences>
        <msd:clink target="BS" from="A.lne.3_1" label="text content" to="C3"/>
        <msd:clink target="BS" from="A.lne.3_2" label="text content" to="C4"/>
        …
</msd:correspondences
```

The advantage of the external form of the correspondences is the reusability of the anchors. In fact, an anchor may serve as a source of a *DS→BS* correspondence; furthermore it can be used to locate source or target elements of the *DS→DS* correspondences. In addition, compared with XPath expressions, the use of anchor allows minimizing the correspondences updates after any modification in document structures.

5.2.2 THE *DS→DS* CORRESPONDENCES

Although the two correspondence types have two different roles, their syntax in MultiX is the same. For this correspondence type the source and the target are elements in two distinct document structures. If the correspondence is internal, the source is the parent of the msd:clink element. In the external correspondence case, the source is an element located by a msd:anchor child element. The target correspondence is located in the same way. For example, to locate the region elements in the text regions structure, we can insert msd:anchor elements as follows:

```
<msd:DS name="Text regions">
        <text-regions>
                <image src ="manuscrit.png">
                        <region num="reg.1" description="first line">
                                <zone x1="43" y1="50" x2="460" y2="94"/>
                                <msd:anchor id="A.reg.1"/>
                        </region>
                        <region num="reg.2" description="second line">
                                <zone x1="43" y1="108" x2="486" y2="152"/>
                                <msd:anchor id="A.reg.2"/>
                        </region>
                        <region num="reg.3" description="third line">
                                <zone x1="43" y1="166" x2="536" y2="210"/>
                                <msd:anchor id="A.reg.3"/>
                        </region>
                        <region num="reg.4" description=" word on lines 1 and 2">
                                <zone x1="410" y1="50" x2="460" y2="94"/>
                                <zone x1="43" y1="108" x2="93" y2="152"/>
                                <msd:anchor id="A.reg.4"/>
                        </region>
                        <region num="reg.5" description="word on lines 2 and 3">
                                <zone x1="'414" y1="108" x2="486" y2="152"/>
                                <zone x1="43" y1="166" x2="72" y2="210"/>
                                <msd:anchor id="A.reg.5"/>
                        </region>
                </image>
        </text-regions>
</msd:DS>
```

Supposing that the two broken words (words on two physical lines), in our example, are located in the lexical structure using anchors identified respectively by “A.wrd.1” and “A.wrd.2”. The following external correspondences may be used to localise from the lexical structures these words on the manuscript image.

```
<msd:clink target="DS" label="location" from="A.wrd.1" to="A.reg.4"/>
<msd:clink target="DS" label="location" from="A.wrd.2" to="A.reg.5"/>
```

External correspondences defined in msd:correspondences element can share several information. It is possible for several *DS→DS* correspondences to have the same label and/or the same source and/or the same target. Then, it may be interesting to factorize the shared attributes. The *grouped correspondences* (msd:clinks element) have been defined for this purpose. The shared attributes are placed in the msd:clinks element, and the unshared attributes are used with the msd:clink elements as shown in the following example:

```
<msd:clinks target="DS" label="location">
        <msd:clink from="A.lne.1" to="A.reg.1"/>
        <msd:clink from="A.lne.2" to="A.reg.2"/>
        <msd:clink from="A.lne.3" to="A.reg.3"/>
</msd:clinks>
```

This grouped form defines three correspondences from the physical structure to the text regions structure allowing the location of the regions occupied by physical text lines in manuscript image. The correspondences,

affects for each physical line in the manuscript image its transcription in the physical structure, may be viewed as the inverse of the above grouped correspondences. In this situation, it is possible to define these two correspondences groups at once by using the inverse attribute as follows:

```
<msd:clinks target="DS" label="location" inverse="transcription">
        <msd:clink from="A.lne.1" to="A.reg.1"/>
        <msd:clink from="A.lne.2" to="A.reg.2"/>
        <msd:clink from="A.lne.3" to="A.reg.3"/>
</msd:clinks>
```

For the *inverse correspondences* the sources are identified by the "from" attributes and the targets by the "to" attributes. As the other attributes, when it is used in a grouped correspondence, the inverse attribute may be factorized in the msd:clinks element or used with the msd:clink elements. It may also be used with a simple correspondence to define its inverse.

6 Querying MULTIX Documents

There are several XML query languages, which may be used for the MultiX documents querying. However, when using these languages the MultiX documents are viewed as pure XML documents. An important effort is then necessary to create queries taking in account the semantic of the MultiX formalism. To be queried easily the MultiX formalism needs an adapted query language. XQuery [13], which is a powerful formalism inspired from the SQL language, is the future standard of the XML documents querying. In order to adapt this language to the MultiX formalism, we have implemented an XQuery functions library allowing facilitating the exploitation of the MultiX semantic. We illustrate here some significant possibilities of this extension.

The following XQuery queries are applied on the MultiX document which encodes the multi-structured document in Figure 3.

Q1: *Find all damaged words; that contain damaged characters only.*

```
for $w in $doc//msd:DS[@name = "words"]//w,
    $d in $doc//msd:DS[@name = "damaged"]//dmg
where multix:include-fragments-of($w, $d) and multix:include-fragments-of($d, $w)
return
    multix:rebuild($w)
```

The function include-fragments-of of the multix library return true if the fragments set composing the content of the element in second parameter is a subset of the first parameter fragments set. Here, the fragments order in each composition is not important. The condition in the where clause allows to select words and damaged characters sequences, which are composed from the same contents fragments set. The rebuild function replaces all *DS→BS* correspondences sources by the text obtained from the target composition.

The result of the query Q1 is the word: <w>mid</w>

Q2: *Find all words broken on two distinct physical lines.*

```
let $doc := doc("manuscript.xml")
for $l in $doc//msd:DS[@name = "lines"]//line,
    $w in $doc//msd:DS[@name = "words"]//w
where multix:share-fragments($l, $w) and not(multix:include-content-of($l, $w))
return
     multix:rebuild($w)
```

In general, a word is included in a single line. A word which is on two lines shares only a part of its fragments with a line. It is this condition which is expressed in the where clause. The function share-fragments tests if both elements in parameters have or not shared contents fragments. The function include-content-of allows to verify if the content of an element is completely included in the content of another element. Here, the contents fragments order in each composition is taken into account.

The result of the query Q2 is composed of the two elements:

```
<w>ægþer</w>
<w>spræce</w>
```

Q3: *Find all words containing restored characters. Indicate for each found word the restored characters and the location of the corresponding line.*

```
let $doc := doc("manuscript.xml")
for $w in $doc//msd:DS[@name = "words"]//w,
    $r in $doc//msd:DS[@name = "damaged"]//res
where multix:share-fragments($r, $w)
return
    <mot-avec-carac-rest>
        {multix:rebuild($w),
        <carac-rest>{multix:get-shared-fragments($r, $w)}</carac-rest>,
        multix:rebuild(multix:get-linked-from($doc//$r, "localisation"))}
    </mot-avec-carac-rest>
```

After selecting the words and the restored characters sequences that share content fragments, we obtain for each word the restored characters using the get-shared-fragments function. To retrieve the zone within the manuscript image containing the line to which belongs the selected word, we use the *SD→SD* correspondences. The function multix:get-linked-from returns all target elements of the correspondences having a given source element and a given label.

The result of the query Q3 is displayed as follows:

```
<mot-avec-carac-rest>
  <w>þu</w>
  <carac-rest>þu</carac-rest>
  <region num="reg.1" description="ligne 1">
            <zone xtop="43" ytop="50" xdown="460" ydown="94"/>
  </region>
</mot-avec-carac-rest>
<mot-avec-carac-rest>
  <w>m e</w>
  <carac-rest>m</carac-rest>
  <region num="reg.1" description="ligne 1">
            <zone xtop="43" ytop="50" xdown="460" ydown="94"/>
  </region>
</mot-avec-carac-rest>
<mot-avec-carac-rest>
  <w>þin e s</w>
  <carac-rest>s</carac-rest>
  <region num="reg.3" description="ligne 3">
            <zone xtop="43" ytop="166" xdown="536" ydown="210"/>
  </region>
</mot-avec-carac-rest>
```

7 Conclusion

The MultiX formalism presented in this paper is based on a generic model defined for multi-structured documents. This new concept of multi-structured documents is independent of the encoding formats. This model allows to represent several kinds of relations between structures composing a multi-structured document. It is possible to invent a new query language for the multi-structured documents. Before that, it is necessary to verify the adaptability of existing languages. XML is widely used language and it is the basis of several other formalisms. MultiX is a XML application. So, it can benefit of several existing XML tools and formalisms. By using XML syntax, it has been possible to use XQuery for MultiX documents. We have developed a library of XQuery functions which allows to explore more easily the multi-structured documents and to make more elaborated queries.

References

[1] NANARD, M.; NANARD, J. et al. La métaphore du généraliste: acquisition et utilisation de la connaissance macroscopique sur une base de documents techniques. Acquisition et Ingénierie des Connaissances - Tendances actuelles. N. Aussenac-Gilles, P. Laublet, C. Reynaud. Toulouse : CEPADUES, 1996, p. 285-304.

[2] POULLET, L. Formaliser la sémantique des documents – Un modèle unificateur. *INFORSID'1997*. Toulouse, June 1997. p. 339-352.

[3] POULLET, L.; PIONON, J. M.; CALABRETTO, S. Semantic Structuring of Documents. *Proceedings of the Third Basque International Workshop on Information Technology, BIWIT'97, Biarritz*, *July 1997*. 1997. p. 118 – 124. ISBN 0-8186-8049-0.

[4] *The XML Version of the TEI Guidelines : Multiple Hierarchies*. URL http://www.tei-c.org/P4X/NH.html

[5] CHATTI, N.; CALABRETTO, S., PINON, J.M. *Vers un environnement de gestion de documents à structures multiples*. BDA 2004, Montpellier. October 2004, p. 47-64.

[6] *Standard Generalized Markup Language (SGML)*. International Organization for Standardization (ISO), Information Processing – Text and Office Systems – ISO 8879-1986

[7] HUITFELDT, C. *MECS - A Multi-Element Code System*. Working Papers from the Wittgenstein Archives at the University of Bergen, Version October 1998, no. 3.

[8] HUITFELDT, C.; SPERBERG-McQUEEN, C. M. *TexMECS: An experimental markup meta-language for complex documents*. Rev. 17 February 2001.

[9] TRNNISON, J.; PIEZ, W. The Layered Markup and Annotation Language (LMNL). **In** Extreme Markup Languages 2002. August 2002.

[10] TUMMARELLO, G.; MORBIDONI, C.; PIERAZZO, E. Toward textual encoding based on RDF. In *ELPUB'2005, Kath. Univ. Leuven, June 2005*.

[11] ABASCAL, R.; BEGBEDER,M.; BENEL, A.; CALABRETTO, S.; CHABBAT, B.; CHAMPIN, P. A.; CHATTI, N.; JOUVE, D.; PRIE, Y.; RUMPLER, B.; THIVANT, E. Modéliser la structuration multiple des documents. *H2PTM'03, Ed. Hermès, Paris , 24-26 septembre 2003*, p. 253-258.

[12] Dekhtyar A., Iacob E. A framework for management of concurrent XML markup. *Data Knowl. Eng*., 2005, vol. 52, no. 2, p. 185-208.

[13] *XQuery 1.0: An XML Query Language*. W3C Candidate Recommendation 3 November 2005. URL http://www.w3.org/TR/xquery/