Adaptation de XML et XQuery pour la représentation et l'interrogation des documents multi-structurés

Noureddine Chatti - Sylvie Calabretto

LIRIS UMR 5205 - INSA de Lyon
Bâtiment Blaise Pascal
7, avenue Jean Capelle
F-69621 Villeurbanne Cedex
{prénom.nom}@insa-lyon.fr

RÉSUMÉ. *Nous traitons dans cet article le problème de l'interrogation des documents à structures multiples, appelés aussi documents multi-structurés. Pour des besoins d'usages variés, plusieurs structurations différentes peuvent être associées à un même document initial. Par exemple, une première structure peut être définie pour organiser logiquement le contenu d'un document tandis qu'une deuxième explicitera les règles de sa mise en forme sur un support physique. Dans de précédents travaux, nous avons proposé une modélisation générique prenant en compte différents aspects de ce type de document. Partant de ce modèle nous avons également proposé un formalisme, basé sur XML, appelé MultiX permettant la sérialisation d'un document multi-structuré. Dans cet article, nous étudions l'exploitation de ces documents dans le contexte de la recherche d'information. Pour interroger efficacement les documents multi-structurés au format MultiX nous avons développé une extension du langage XQuery sous forme d'une bibliothèque de fonctions spécifiques permettant d'explorer les documents en prenant en compte la sémantique du format MultiX.*

ABSTRACT. *This paper deals with the interrogation of multi-structured documents. For various use aims, several distinct structures may be defined simultaneously for the same original document. For example, a document may have a first structure to express a set of content formatting rules (logical structure), and a second structure to express a set of content formatting rules (layout structure). In previous works, we have proposed a generic model for the multi-structured document and MultiX, an XML-based formalism to encode multi-structured documents. In this paper, we present one solution to query multi-structured documents. This solution is based on a library of XQuery functions.*

MOTS-CLÉS : *Document multi-structuré, XML, format d'encodage, recherche d'information, interrogation, XQuery.*

KEY WORDS.*Multi-structured document, XML, encoding format, information retrieval, querying, XQuery*

CORIA 2007, pp. 109–124
Saint-Étienne, 28–30 mars 2007

1 Introduction

Dans le contexte du numérique, et dans un monde de plus en plus communiquant, l'usage du document se diversifie, et des besoins nouveaux s'imposent. Pour des besoins d'usage différents, plusieurs structurations d'un même document peuvent être définies simultanément. Plusieurs types de structures, telles que la structure physique, la structure logique ou encore la structure sémantique ont été définis à des fins d'exploitation différentes. D'autres types de structures s'attachant plutôt à la nature du contenu ont également été définis. C'est le cas par exemple des structures spatiales, temporelles ou encore spatio-temporelles, associées aux documents multimédias [CHR 03]. Lorsque plusieurs structures partagent un contenu identique celles-ci sont appelées dans la littérature structures *concurrentes* ou encore *parallèles*. Nous appelons un document *multistructuré* plusieurs structures documentaires reliées entre elles via un contenu commun ou d'autres types de relations inter-structurelles. Les structures concurrentes sont alors des cas particuliers des documents multi-structurés. La structuration multiple des documents pose différents types de problématiques. Parmi celles-ci la représentation de ces documents ainsi que leur exploitation (l'interrogation conjointe de l'ensemble de ces structures) notamment dans un contexte de recherche d'information.

Pour illustrer la problématique de la multistructuralité des documents ainsi que notre proposition, nous avons choisi un exemple d'une image représentant un extrait d'un manuscrit ancien présenté dans [DEK 05]. A cette image (Figure 1) nous associons quatre structures différentes que nous avons encodées séparément au format XML.

Figure 1. Un fragment d'un manuscrit ancien

Dans la première structure (structure physique), le texte original de cet extrait du manuscrit est transcrit en marquant les lignes.

```
<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>
```

La seconde structure (structure lexicale) marque tous les mots du texte.

```
<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>
```

La troisième structure marque toutes les séquences de caractères endommagés. La balise *res* marque les caractères endommagés restaurés à partir d'autres manuscrits.

```
<damaged>
<res>þum</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>
```

La quatrième structure (structure des régions textuelles) permet de localiser et de décrire des zones particulières occupées par le texte manuscrit dans l'image.

```
<text-regions>
        <image src="manuscrit.png">
                <region num="reg.1" description="ligne 1">
                        <zone xtop="43" ytop="50" xdown="460" ydown="94"/>
                </region>
                <region num="reg.2" description="ligne 2">
                        <zone xtop="43" ytop="108" xdown="486" ydown="152"/>
                </region>
                <region num="reg.3" description="ligne 3">
                        <zone xtop="43" ytop="166" xdown="536" ydown="210"/>
                </region>
                <region num="reg.4" description="mot sur ligne 1 et ligne 2">
                        <zone xtop="410" ytop="50" xdown="460" ydown="94"/>
                        <zone xtop="43" ytop="108" xdown="93" ydown="152"/>
                </region>
                <region num="reg.4" description="mot sur ligne 2 et ligne 3">
                        <zone xtop="'414" ytop="108" xdown="486" ydown="152"/>
                        <zone xtop="43" ytop="166" xdown="72" ydown="210"/>
                </region>
        </image>
</text-regions>
```

Les requêtes sur les documents multi-structurés combinent plusieurs structures simultanément. Si nous nous intéressons à l'extrait de manuscrit, voici des exemples de requêtes que souhaitons pouvoir exprimer :
- Quels sont les mots coupés en fin de ligne ? (cette requête fait référence simultanément aux structures lexicale et physique) ;
- Quels sont les mots composés uniquement de caractères endommagés ? (référence à la structure lexicale et à la structure des caractères endommagés) ;

Nous avons proposé dans [CHA 04] un modèle spécifique pour les documents multi-structurés. Puis, dans [CHA 06], nous avons défini un formalisme basé sur ce modèle dédié à l'encodage de structures multiples de documents. Ce formalisme appelé MultiX est une application XML qui permet d'expliciter les caractéristiques particulières des documents multi-structurés et d'optimiser ainsi leurs exploitations. Dans cet article, nous présentons en détail les possibilités d'interrogation de documents au format MultiX. Après une présentation des principaux travaux connexes (sur les aspects modélisation et interrogation), nous ferons une description rapide de notre modèle de documents multi-structurés. La section 4 sera consacrée à la présentation du formalisme MultiX. Dans la section 5 nous développerons l'interrogation des documents multi-structurés encodés avec le formalisme MultiX et présenterons la bibliothèque de fonctions XQuery que nous avons développée.

2 Etat de l'art

Nous pouvons distinguer deux types d'approches pour modéliser les documents à structures multiples. Dans la première approche, les structures sont encodées séparément au format XML. Les principaux inconvénients de cette solution sont la redondance du contenu et la perte d'informations relatives aux relations entre les structures. La deuxième approche est fondée sur l'encodage de toutes les structures dans un même document XML sans répliquer le contenu, en les superposant les unes par rapport aux autres. Ainsi, le métalangage SGML [ISO 86] offre une fonctionnalité optionnelle CONCUR permettant de faire référence à plusieurs DTD. Grâce à cette option plusieurs types de balisage (structures) peuvent être utilisés au sein du même document SGML. Les balises sont identifiées par un préfixe indiquant la DTD où elles sont définies. Malgré son intérêt, cette option, n'a été implémentée que très rarement et ne proposait pas de solution pour interroger plusieurs structures simultanément. De plus SGML est aujourd'hui totalement remplacé par son successeur XML, avec une spécification moins complexe. En XML une seule DTD peut être référencée à la fois, et même si le mécanisme des espaces de noms permet de distinguer entre des grammaires différentes dans un même document, il ne peut pas remplacer la fonctionnalité CONCUR de SGML. La Text Encoding Initialive [TEI 02], a traité le problème d'encodage de hiérarchies multiples. Dans la version XML des directives de la TEI, un ensemble de techniques d'encodage de plusieurs hiérarchies dans un même fichier XML a été présenté. Ces techniques se basent sur le choix d'une structure qui sera encodée en premier, ensuite les éléments des autres structures sont insérés dans le document. Pour que le document XML reste bien formé, les éléments insérés sont cassés entre les éléments de la première structure et diverses techniques (milestone elements, fragmentation, virtrual joins, etc) sont proposées pour reconstruire virtuellement les structures. Bien que le document obtenu soit bien formé, il reste complexe et incompréhensible. De plus des mécanismes spécifiques doivent être implémentés pour recalculer automatiquement les structures d'origine.

Plus récemment, dans [TUM 05], il a été proposé d'utiliser le formalisme RDF (Resource Description Framework) pour encoder des structures de documents textuels. L'objectif de cette proposition est de tirer profit du modèle de graphe du langage RDF, pour pouvoir encoder des structures complexes comportant des entrelacements entre les éléments et de les interroger par une adaptation d'un langage de requête de RDF. L'utilisation du formalisme RDF pour l'encodage des documents multi-structurés est une tâche très complexe qui génère aussi des difficultés de gestion de l'évolution de ces documents.

D'autres travaux se sont intéressés à la définition de nouvelles syntaxes permettant d'améliorer les possibilités de structuration des langages existants. MECS (Multi-Element Code System) [HUI 98] a été le premier langage à supporter le chevauchement entre éléments. TexMECS [HUI 01] est un autre langage basé sur MECS qui permet de définir des structures complexes où un élément peut avoir plus d'un parent. Cependant, ces projets ne s'intéressent pas à la problématique de

l'interrogation de documents à structures multiples. LMNL (Layered Markup and aNnotation Language) [TEN 02] définit une syntaxe basée sur la notion d'intervalle permettant l'encodage de structures multiples dans lesquelles les éléments peuvent s'entrelacer. Le projet LMNL envisage de définir un langage de requêtes adapté s'appuyant sur XPath mais aucune spécification n'est actuellement disponible. Malgré l'intérêt des nouvelles possibilités de structuration introduites par ces langages et syntaxes, ceux-ci n'ont pas pu dépasser le stade d'expérimentation. La complexité de ces syntaxes et le manque de langages de requête et de logiciels permettant de les prendre en charge sont les principales raisons de leur insuccès.

Dans [DEK 05] les auteurs se basent sur le modèle GODDAG [SPE 00] pour créer une représentation interne de structures XML concurrentes. La structure de graphe obtenue permet de faire abstraction du problème de chevauchement. Pour interroger les documents sous cette représentation, les auteurs proposent une extension du langage XPath [W3C 99]. Cette extension est définie pour tenir compte des nouveaux types de relations entre les éléments de structures différentes. Il est par exemple possible d'atteindre tous les ancêtres d'un nœud dans toutes les hiérarchies (axe xancestor). De même, sont définis les axes xdescendant, xfollowing, xpreceding, et pour traiter les chevauchements, les axes overlapping, following-overlapping et preceding_overlapping. Ainsi, la requête exemple « quels sont les mots coupés en fin de ligne » s'exprime de la façon suivante : `/xdescendant ::lign e/overlapping ::m.` Ce modèle peut être appliqué pour les documents multi-structurés en général, mais il ne permet pas de représenter des relations génériques entre les structures. Les seuls types de relations qui peuvent être exploités dans ce modèle sont les relations de positionnement des éléments en fonction du contenu commun.

Une proposition très récente, MSXD (MultiStructured XML Documents), a été définie dans [BRU 06]. Il s'agit d'un modèle décrivant des structures XML indépendantes construites autour d'un contenu textuel identique. Les relations entre les éléments des structures sont modélisées par un ensemble de contraintes définissant un schéma du document multi-structuré. Le contenu textuel est répliqué dans chaque structure. Le schéma a pour rôle l'instanciation d'un ensemble de règles (les relations de Allen) permettant de décrire les positions relatives des fragments textuels dans les structures. Enfin, une extension de XQuery à la multistructure a été proposée. La requête exemple « quels sont les mots coupés en fin de ligne » s'exprime de la façon suivante en MSXD :

```
for $p in $doc//Phrase return
$doc//m[. is-overlapping $p]
```

L'inconvénient majeur de cette méthode est le problème de la redondance du contenu et les difficultés de gestion d'évolution qui en résultent.. De plus, cette méthode d'encodage nécessite la mise en œuvre de médiateurs assez complexes, pour l'interrogation des documents multi-structurés.

3 Le modèle de documents multi-structurés MSDM

Pour répondre de manière générique à la problématique de la structuration multiple des documents nous avons proposé un modèle appelé MSDM (Multi-Structured Document Model) [CHA 04], basé sur une première proposition définie dans [ABA 03]. Dans ce modèle les relations entre les structures à travers le contenu commun sont mises en évidence. Selon notre modélisation un document multi-structuré ne se résume pas seulement à un document ayant des structures partageant un contenu identique (cas des structures concurrentes, qui a été le seul cas étudié dans la littérature). Dans le modèle MSDM, le document multi-structuré est une entité dans laquelle sont mises en relation différentes structures. En plus des relations exprimant le partage du contenu, le modèle définit un autre niveau de relations entre les structures au sein du document multi-structuré, pour permettre d'expliciter des liens spéciaux entre les éléments de structures différentes. Selon le modèle MSDM un document multi-structuré est défini par la donnée de :

- Une structure de Base (SB) : cette structure joue un rôle unique et interne au sein du document multi-structuré. Elle organise le contenu partagé entre plusieurs structures en fragments élémentaires disjoints. Chaque partie de contenu initial fragmentée peut être ainsi reconstruite par composition à partir de fragments dans la structure de base.

- Un ensemble de structure documentaire (SD) : c'est une description du contenu, qui peut être définie pour un usage bien précis du document. Par exemple, pour les besoins de présentation nous pouvons définir une structure documentaire appelée structure physique décrivant les règles de formatage des différentes parties du contenu.

- Un ensemble de relations de correspondance entre les structures documentaires : une correspondance met en relation deux éléments (élément origine et élément extrémité) de deux structures distinctes du même document multi-structuré. Nous distinguons deux types de correspondances : les correspondances entre une structure documentaire est une structure de base notées SD→SB, et les correspondances entre deux structures documentaires notées SD→SD. Le premier type de correspondance permet d'associer une partie de contenu à un élément d'une structure documentaire. Le second type de correspondance permet de rendre explicite certaines relations utiles entre les structures documentaires. A titre d'exemple, une correspondance SD→SD peut être établie pour exprimer une relation de synonymie entre deux éléments de deux structures documentaires distinctes. Cette relation peut ensuite trouver son intérêt dans un processus de recherche d'information par exemple.

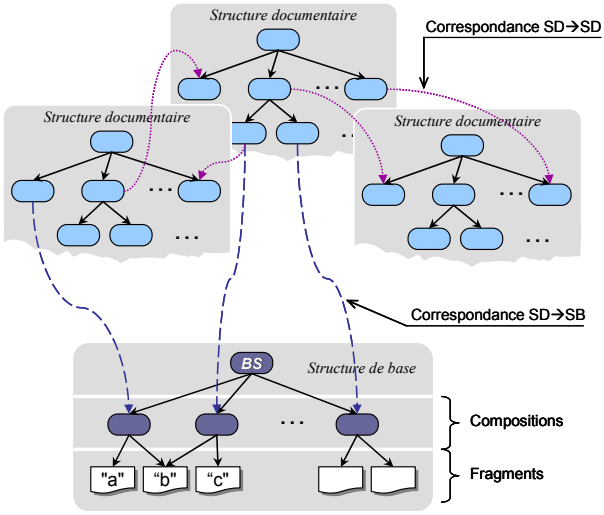

Figure 2. Illustration du modèle de document multi-structuré

Le modèle MSDM définit le document multi-structuré indépendamment de tout langage de structuration, ce qui permet de faire abstraction des problèmes de redondance et de chevauchement entre les structures.

4 Le formalisme de sérialisation MultiX

Le formalisme MultiX [CHA 06] permet de sérialiser un document multi-structuré, selon de modèle MSDM, sous la forme d'un document XML bien formé. Un document multi-structuré représenté dans ce formalisme, et que nous désignerons dans ce qui suit par document MultiX, se décompose en trois parties : la structure de base (SB), les structures documentaires (SD) et les correspondances.

4.1 La structure de base

La structure de base a pour rôle de réorganiser le contenu en le découpant en un ensemble de petites parties disjointes appelées fragments de base (ou fragments élémentaires), dans le but de faciliter son partage entre les structures documentaires. Cette réorganisation n'est pas arbitraire mais dépend de la disposition du contenu partagé, dans chaque structure documentaire. Dans le cas d'un contenu textuel, l'ensemble minimal de fragments élémentaires peut être explicité en procédant à la superposition de toutes les parties des structures documentaires structurant un contenu identique. En prenant comme repère commun la séquence des caractères composant le contenu de la structure physique du manuscrit exemple, nous obtenons

par superposition de tous les éléments contenant du texte dans les structures documentaires le découpage illustré dans la Figure 3.

```
<line n="1"><w>hu</w><w><res>þu</w><w>m</res>e</w><w>hæfst</w>
<w>afrefredne</w><w>æg</line><line n="2">þ<dmg>er</dmg></w>
<w>ge</w><w><dmg>mid</dmg></w><w>þinre</w><w>smealican</w>
<w>spr<dmg>æ</dmg></line><line n="3">ce</w>,<w><dmg>g</dmg>e</w>
<w>mid</w><w><dmg>þ</dmg>in<dmg>re</dmg></w><w>wynsumnesse</w>
<w>þin<dmg>e</dmg><res>s</res></w></line>
```

Figure 3. Les fragments élémentaires résultant de la superposition de trois structures

A partir de cet ensemble de fragments élémentaires, des compositions vont être créées dans la structure de base pour reconstituer les PCDATA des structures documentaires. Ainsi, la structure de base sera composée de deux parties : une première partie dans laquelle est défini l'ensemble des fragments disjoints, et une seconde partie définissant les compositions des fragments. L'ensemble des fragments disjoints doit être défini à l'intérieur d'un élément nommé fragments, et les compositions à l'intérieur de l'élément compositions de la structure de base. Pour l'exemple de document multi-structuré de la Figure 1, l'ensemble des fragments élémentaires et des compositions peut être sérialisé dans la structure de base de la manière suivante :

```
<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="F18">e</msd:frag>
                <msd:frag id="F19">mid</msd:frag>
                <msd:frag id="F20">þ</msd:frag>
                <msd:frag id="F21">in</msd:frag>
                <msd:frag id="F22">re</msd:frag>
                <msd:frag id="F23">wynsumnesse</msd:frag>
                <msd:frag id="F24">þin</msd:frag>
                <msd:frag id="F25">e</msd:frag>
                <msd:frag id="F26">s</msd:frag>
        </msd:fragments>
```

```
        <msd:compositions>
        <!-- Compositions pour la structure physique /-->
                <msd:comp id="CL1" idrefs="F1 F2 F3=F4 F5 F6 F7"/>
                <msd:comp id="CL2" idrefs="F8=F9 F10 F11 F12 F13 F14=F15"/>
                <msd:comp id="CL3_1" idrefs="F16"/>
                <msd:comp id="CL3_2" idrefs="%F17=F18 F19 F20=F21=F22 F23
F24=F25=F26"/>
        <!-- Compositions pour la structure lexicale /-->
                <msd:comp id="CW1" idrefs="F1"/>
                <msd:comp id="CW2" idrefs="F2"/>
                <msd:comp id="CW3" idrefs="F3=F4"/>
                <msd:comp id="CW4" idrefs="F5"/>
                <msd:comp id="CW5" idrefs="F6"/>
                <msd:comp id="CW6" idrefs="F7=F8=F9"/>
                <msd:comp id="CW7" idrefs="F10"/>
                <msd:comp id="CW8" idrefs="F11"/>
                <msd:comp id="CW9" idrefs="F12"/>
                <msd:comp id="CW10" idrefs="F13"/>
                <msd:comp id="CW11" idrefs="F14=F15=F16"/>
                <msd:comp id="CW12" idrefs="F17=F18"/>
                <msd:comp id="CW13" idrefs="F19"/>
                <msd:comp id="CW14" idrefs="F20=F21=F22"/>
                <msd:comp id="CW15" idrefs="F23"/>
                <msd:comp id="CW16" idrefs="F24=F25=F26"/>
                <msd:comp id="CR2" idrefs="F26"/>
        </msd:compositions>
</msd:BS>
```

Chaque élément *frag* définit un fragment élémentaire et lui associe un identificateur unique pour être référencé dans les compositions. La création d'une nouvelle composition se fait par l'insertion d'un élément *comp* à l'intérieur de l'élément *compositions*. Chaque composition a un identificateur unique (attribut id), et un attribut spécifique nommé *idrefs* dont la valeur est une séquence d'identificateur de fragments élémentaires. La valeur de l'attribut *idrefs* suit des règles de codage spécifiques permettant la reconstitution d'une nouvelle chaîne de caractères à partir de la séquence de fragments référencés par cet attribut. Si nous confondons l'identificateur d'un fragment avec la chaîne de caractères à laquelle il fait référence, et en appelant *concat* la fonction de concaténation de chaînes de caractères, nous pouvons résumer la signification des séparateurs par le Tableau 1.

MultiX	Signification	Résultat
idrefs = "F1 F2"	*concat*(F1, *concat*(" ", F2))	"hu þu"
idrefs = "F1=F2"	*concat*(F1, F2)	"huþu"
idrefs = "%F1=F2"	*concat*(" ", *concat*(F1, F2))	" huþu"
idrefs = "F1=F2%"	*concat*(*concat*(F1,F2), " ")	"huþu "

Tableau 1. Signification des symboles de gestion des espaces blancs dans les compositions

Si nous prenons l'exemple du premier élément line de la structure physique, la reconstitution du PCDATA que contient cet élément nécessite la composition, dans l'ordre, des fragments élémentaires identifiés par : F1, F2, F3, F4, F5, F6 et F7. Cette composition se traduit ici par des concaténations de chaînes de caractères, mais en gérant convenablement l'insertion des espaces blancs lorsque cela est nécessaire. Cette composition s'exprime alors en MultiX par l'élément vide suivant :

```
<msd:comp id="CL1" idrefs="F1 F2 F3=F4 F5 F6 F7"/>
```

4.2 Les structures documentaires et les relations de correspondance

Dans un document MultiX, une structure documentaire se définit à l'intérieur d'un élément DS (Document Structure, en anglais). En partant d'une structure encodée au format XML classique, la même structure au format MultiX s'obtient après une légère transformation. Le contenu commun à d'autres structures est remplacé par des relations de correspondances vers la structure de base. D'autres informations peuvent également être rajoutées pour créer des correspondances de type SD→SD. MultiX définit un mécanisme pour la création de relations de correspondance en prenant en compte l'évolution des structures, et en minimisant les modifications des structures initiales.

4.2.1 Les correspondances SD→SB

Le rôle d'une correspondance de ce type est d'associer à un emplacement spécifique (origine de la correspondance) d'une structure documentaire, une portion de contenu définie par une composition dans la structure de base. Dans un document MultiX les correspondances peuvent être déclarées soit à l'intérieur de la structure documentaire (correspondance interne), soit à l'extérieur de celle-ci (correspondance externe). Une correspondance interne se déclare dans une structure documentaire à l'emplacement qui représente son origine. Cet emplacement représente l'endroit où il faut placer le contenu reconstitué par l'intermédiaire de la correspondance. En MultiX, la déclaration d'une correspondance se fait en créant un élément *clink*. Dans notre exemple de document multi-structuré, la correspondance interne suivante permet d'associer la composition CL1, dans la structure de base, au premier élément line de la structure physique.

```
<line n="1">
        <msd:clink target="BS" label="text content" to="CL1"/>
</line>
```

La valeur de l'attribut *target* indique le type de la correspondance. Deux valeurs sont possibles pour cet attribut : BS pour le type SD→SB, et DS pour le type SD→SD. Dans le cas d'une correspondance vers la structure de base, l'attribut *to* indique l'identificateur de la composition cible. Une étiquette peut être associée à la correspondance via l'attribut optionnel label. La correspondance ci-dessus doit être interprétée par les applications comme une référence à la chaîne obtenue par la

composition CL1. Pour afficher la structure physique avec son contenu, il suffit de remplacer les correspondances SD→SB par les segments de texte calculés à l'aide des compositions cibles. Pour la structure des caractères endommagés il est plus simple de déclarer toutes les correspondances sous la forme interne. Cette structure documentaire peut donc être encodée de la manière suivante :

```
<msd:DS name="damaged">
        <damaged>
                <res><msd:clink target="BS" label="text content" to="CR1"/></res>
                <dmg><msd:clink target="BS" label="text content" to="CD1"/></dmg>
                <dmg><msd:clink target="BS" label="text content" to="CW8"/></dmg>
                <dmg><msd:clink target="BS" label="text content" to="CD2"/></dmg>
                <dmg><msd:clink target="BS" label="text content" to="CD3"/></dmg>
                <dmg><msd:clink target="BS" label="text content" to="CD4"/></dmg>
                <dmg><msd:clink target="BS" label="text content" to="CD5"/></dmg>
                <dmg><msd:clink target="BS" label="text content" to="CD6"/></dmg>
                <res><msd:clink target="BS" label="text content" to="CR2"/></res>
        </damaged>
</msd:DS>
```

Une correspondance externe se déclare, dans le document MultiX, à l'intérieur de l'élément *correspondences* du document multi-structuré. Un élément *anchor* est utilisé pour marquer l'emplacement de départ de la correspondance. A chaque élément *anchor* est attribué un identificateur unique permettant de localiser directement, et sans utilisation d'expressions XPath, un emplacement à l'intérieur d'une structure documentaire.

4.2.2 Les correspondances SD→SD

Dans le formalisme MultiX, le même mécanisme est utilisé pour déclarer les correspondances de types SD→SB et SD→SD. Cependant, chaque type de correspondance a une sémantique propre. Pour les correspondances SD→SD, il s'agit de représenter une relation entre deux positions dans deux structures documentaires distinctes. C'est l'élément de la structure documentaire qui contient l'une des deux balises clink, en cas d'une correspondance interne, ou anchor, en cas d'une correspondance externe, qui est considéré comme origine.

5 Recherche d'information dans les documents multi-structurés MultiX

En se basant sur le langage XML les documents MultiX peuvent profiter d'une large gamme d'outils et de formalismes génériques, tels que les langages d'interrogation et de transformation. L'interrogation est l'une des exploitations documentaires les plus importantes. Pour interroger les documents MultiX plusieurs choix sont a priori possibles. Comme il s'agit d'un document XML bien formé, un document MultiX peut être interrogé par les langages de requêtes existants conçus pour le format XML standard. Toutefois, l'utilisation de ces langages ne peut pas se faire sans un certain nombre de difficultés. En effet, pour ces langages le document MultiX est vu

comme n'importe quel autre document XML, c'est à dire une simple structure arborescente. Deux solutions s'offrent à nous pour interroger les documents MultiX. La première solution consiste à créer un nouveau langage d'interrogation spécifique intégrant la sémantique propre du formalisme MultiX. Cette solution nécessite beaucoup d'effort de conception et présente de nombreux inconvénients comme la nécessité d'implémenter des outils logiciels spécifiques. La deuxième solution, qui est la moins coûteuse, consiste à adapter un langage de requête existant. Nous retenons cette deuxième solution et nous choissions l'adaptation du langage XQuery. Avec ce langage il est possible de définir de nouvelles fonctions utilisateur qui peuvent être exploitées dans les requêtes de la même manière que les fonctions prédéfinies de XQuery et de XPath. Nous utilisons cette possibilité pour définir une extension de ce langage. Cette extension consiste alors en une bibliothèque de fonctions spécifiques permettant de simplifier l'élaboration de requêtes intégrant la sémantique du formalisme MultiX. Le but de ces fonctions est de cacher la complexité d'un document MultiX en évitant aux utilisateurs d'interpréter les éléments spécifiques se rapportant aux relations de correspondances. Dans la suite, nous commençons par lister les principales fonctions de notre bibliothèque MXQ qui constitue l'extension de XQuery pour le formalisme MultiX. Ces fonctions seront préfixées par *mxq* qui représente l'espace de noms dans lequel est définie notre bibliothèque.

5.1 Fonctions pour l'interrogation multistructurelle

L'un des principaux avantages d'un formalisme adapté pour la représentation de documents multi-structurés est la possibilité d'exploitation des interdépendances entre les structures. Le formalisme MultiX offre, à travers les correspondances de types SD→SB et SD→SD, un moyen efficace permettant la matérialisation de l'interdépendance entre les structures documentaires. Cette matérialisation permet alors d'offrir des possibilités d'exploration et d'interrogation étendues faisant intervenir plusieurs structures. Les fonctions de la bibliothèque MXQ contribuent à la valoriser du formalisme MultiX en permettant de simplifier l'exploration inter-structurelle et de tirer un maximum de profit de ce format d'encodage. Faute de place, nous ne pouvons pas faire une présentation détaillée de cette bibliothèque et de son utilisation. Nous nous contenterons donc de décrire quelques fonctions utiles:

rebuild($e as element()) as element() : cette fonction permet de reconstruire tout élément d'une structure documentaire en remplaçant tous les correspondances SD→SB qu'il contient par le contenu recomposé. Les correspondances de type SD→SD ne sont pas prises en compte.

share-content($e as element()) as xs:Boolean : Cette fonction permet de vérifier si un élément partage ou non une partie (éventuellement la totalité) de son contenu avec d'autres structures..

share-fragments($e1 as element(), $e2 as element()) as xs:Boolean : La fonction share-fragments teste si les deux éléments en arguments, qui appartiennent à deux structures documentaires différentes, ont ou pas des fragments de contenus élémentaires partagés.

get-shared-fragments($e1 as element(), $e2 as element()) as element(msd:frag)* : Cette fonction permet de retourner la séquence des fragments élémentaires partagés entre deux éléments de deux structures documentaires distinctes. Cette fonction est particulièrement utile pour rechercher des segments de contenu vérifiant deux critères différents.

include-fragments-of($e1 as element(), $e2 as element()) as xs:Boolean : Cette fonction retourne vrai si tous les fragments élémentaires associés au contenu de l'élément en deuxième argument sont inclus dans l'ensemble des fragments élémentaires associés au contenu de l'élément en premier argument.

same-fragments($e1 as element(), $e2 as element()) as xs:Boolean : Cette fonction vérifie l'égalité entre les ensembles de fragments élémentaires associés aux deux éléments en paramètres. L'ordonnancement des fragments n'est pas pris en compte.

include-content-of($e1 as element(), $e2 as element()) as xs:Boolean : Cette fonction vérifie l'inclusion totale d'un contenu d'un élément (deuxième argument) dans un contenu d'un deuxième élément (premier argument). Les deux éléments appartiennent à deux structures documentaires distinctes.

get-corr-target-from($e as element(), $label as xs:string) as element()* : cette fonction permet de sélectionner seulement les cibles des correspondances partant d'un élément donné et ayant une étiquette précisée.

5.2 Exemples de requêtes

L'objectif de cette section est l'illustration de l'utilisation de la bibliothèque MXQ à travers quelques exemples de requêtes multi-structurelles. Nous continuons à exploiter l'exemple de document multi-structuré associé à l'extrait du manuscrit ancien de la figure 1 (section 1). Les requêtes suivantes sont formulées pour être appliquées à la représentation de ce document en MultiX. Notons que les requêtes XQuery présentées ci-après peuvent ne pas représenter la façon la plus optimale pour exprimer les questions formulées en langage naturelle. Le but étant de montrer le potentiel du langage MultiX et de la bibliothèque de fonctions MXQ.

Première requête : *Trouver tous les mots endommagés, c'est-à-dire composés seulement de caractères endommagés.*

```
let $doc := doc("manuscript.xml")
for $w in $doc//msd:DS[@name = "words"]//w,
    $d in $doc//msd:DS[@name = "damaged"]//dmg
where mxq:same-fragments($w, $d)
return
    mxq:rebuild($w)
```

Dans cette requête une jointure est réalisée entre l'ensemble des mots dans la structure "words" et l'ensemble des caractères endommagés dans la structure "damaged". La condition de cette jointure consiste à vérifier l'égalité des ensembles de fragments composant les contenus des deux éléments. Dans le cas de cette requête il n'est pas nécessaire d'avoir le même ordre de fragments dans les deux ensembles. Les éléments w vérifiant la condition de jointure de cette requête ne sont pas renvoyés tel qu'ils ont été encodés dans le document MultiX. La fonction rebuild est appliquée à chacun de ces éléments afin de reconstituer son contenu.

Le résultat de cette requête est composé du seul mot :

```
<w>mid</w>
```

Deuxième requête : *Trouver tous les mots qui sont coupés en fin de ligne dans le texte original.*

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

Cette requête XQuery est également une jointure entre deux ensembles d'éléments appartenant à deux structures distinctes. Un mot qui se trouve sur deux lignes ne partage qu'une partie de ses fragments avec la ligne sur laquelle il débute. Le reste de son contenu est partagé avec la ligne suivante. C'est cette condition qui est exprimée dans la clause where de la requête. Pour chaque ligne la condition de jointure permet de sélectionner les mots partageant des fragments avec celle-ci mais qui n'y sont pas totalement inclus. Ainsi, cette condition est vérifiée pour chacune des deux lignes sur laquelle se trouve une partie d'un mot coupé. Le résultat de cette requête comportera donc des doublons.

Le résultat de cette requête, après élimination des doublons, est composé des deux mots suivants :

```
<w>ægþer</w>
<w>spræce</w>
```

Troisième requête : *Trouver les mots qui contiennent des caractères restaurés. Indiquer pour chaque mot trouvé les caractères restaurés qu'il contient et la localisation de la ligne contenant ce mot sur l'image du manuscrit.*

```
let $doc := doc("manuscript.xml")
for $w in $doc//msd:DS[@name = "words"]//w,
    $r in $doc//msd:DS[@name = "damaged"]//res,
    $l in $doc//msd:DS[@name = "lines"]//line
where mxq:share-fragments($r, $w) and mxq:include-content-of($l, $w)
return
    <mot-avec-carac-rest>
```

```
        {mxq:rebuild($w),
        <carac-rest>{mxq:get-shared-fragments($r, $w)/text()}</carac-rest>,
        mxq:rebuild(mxq:get-corr-target-from($doc//$l, "localisation"))}
    </mot-avec-carac-rest>
```

Dans cette requête nous avons exploité les deux types de correspondances qui existent dans le document « manuscript.xml », SD→SB et SD→SD. Le premier type de correspondances est exploité pour rechercher les mots contenant des caractères restaurés. Le deuxième type est exploité dans la requête pour retrouver les localisations spatiales des lignes sur lesquels se trouvent les mots sélectionnés.

Le résultat de cette requête s'affiche de la manière suivante :

```
<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>
```

6 Conclusion et perspectives

Le concept nouveau de document multi-structuré selon le modèle MSDM se distingue par sa généricité et son indépendance par rapport aux formats d'encodage. L'utilisation du langage XML nous a permis d'exploiter XQuery pour interroger les documents multi-structurés au format MultiX. La bibliothèque de fonctions MXQ facilite grandement la création de requêtes exploitant la sémantique du formalisme MultiX. Grâce à ce formalisme et à l'extension de XQuery, nous pouvons parler de Recherche d'Information Multi-Structurée (RIMS). Cependant, cette solution peut, dans certaines conditions, ne pas être suffisante. L'une des limitations de cette méthode est la lenteur du temps de réponse des requêtes, due à l'interprétation des fonctions. Ce problème est plus persistant lorsque les fonctions utilisées sont complexes et lorsque plusieurs fonctions sont appelées dans une même requête. L'utilisation de langages de requêtes XML peut aussi représenter une limitation, car leur sémantique n'est pas bien adaptée à la structuration multiple. La définition d'un langage de requête spécifique aux documents multi-structurés, offrant une syntaxe et une sémantique appropriées, est une future direction de recherche. Une autre direction porte sur l'extension de nos propositions aux documents multimédias.

7 Références

[ABA 03] R. Abascal, M. Beigbeder, A. Benel, S. Calabretto, B. Chabbat, P.A. Champin, N. Chatti, D. Jouve, Y. Prie, B. Rumpler, E. *Modéliser la structuration multiple des documents*. H2PTM'03, Ed. Hermès, Paris, 24-26 septembre 2003, pp. 253-258

[BRU 06] E. Bruno, E. Murisasco. *Describing and querying hierarchical structures defined over the same textual data*. In Proceedings of the 2006 ACM Symposium on Document Engineering (DocEng 2006), pp. 147-154, Amsterdam, The Netherlands, October 2006

[CHA 04] N. Chatti, S. Calabretto, J.M. Pinon. *Vers un environnement de gestion de documents à structures multiples.* 20ème JOURNEES BDA 2004, Montpellier. 19-22 October 2004, pp. 47-64.

[CHA 06] N. Chatti, S. Calabretto, J.M. Pinon. *MultiX : un formalisme pour l'encodage des documents multi-structurés.* Actes du XXIVème Congrès INFORSID, Hammamet, Tunisie, 31 mai-4 juin, 2006. pp. 975-990.

[CHR 02] C. Chrisment, F. Sedes. *Media annotations: toward a unified representation.* Chapitre du livre Multimedia mining, octobre 2002. Kluwer Academic Publisher.

[DEK 05] Alex Dekhtyar, Ionut Emil Iacob. *A framework for management of concurrent XML markup*. Data Knowl. Eng. 52(2): 185-208 (2005)

[DOM 03] World Wide Web Consortium (W3C*), Document Object Model (DOM),* DOM level 1, 2 and 3 specifications, 2003. http://www.w3c.org/DOM/

[HUI 98] C. Huitfeldt. *MECS - A Multi-Element Code System.* Working Papers from the Wittgenstein Archives at the University of Bergen, No 3, Version October 1998.

[HUI 01] C. Huitfeldt, C. M. Sperberg-McQueen*. TexMECS: An experimental markup meta-language for complex documents*. Rev. 17 February 2001.

[ISO 86*] Standard Generalized Markup Language (SGML).* International Organization for Standardization (ISO), Information Processing – Text and Office Systems – ISO 8879-1986

[RDF 04] World Wide Web Consortium (W3C), *Resource Description Framework, RDF* Specification Development, 2004. http://www.w3.org/RDF/

[SPE 00] C. M. Sperberg-McQueen, Claus Huitfeldt. *GODDAG: A Data Structure for Overlapping Hierarchies.* DDEP/PODDP 2000: 139-160.

[TEI 02] *The XML Version of the TEI Guidelines : Multiple Hierarchies*. http://www.tei-c.org/P4X/NH.html

[TEN 02] J. Tennison, W. Piez*. The Layered Markup and Annotation Language (LMNL)*. In Extreme Markup Languages 2002. August 2002.

[TUM 05] G.Tummarello, C.Morbidoni, E. Pierazzo. *Toward textual encoding based on RDF*. In ELPUB'2005, Kath. Univ. Leuven, June 2005.

[W3C 05] W3C. *XQuery 1.0: An XML Query Language*. W3C Candidate Recommendation 3 November 2005. http://www.w3.org/TR/xquery/

[W3C 99] W3C. *XML Path Language (XPath)* Version 1.0. W3C Recommendation 16 November 1999. http://www.w3.org/TR/xpath