Adaptation aux différents modes de lecture

Cécile Roisin

L'apparition de nouvelles possibilités de communication et de traitement de l'information sur Internet induit le développement de nouvelles solutions pour la définition des formats de contenu, leur structuration et leur restitution auprès des lecteurs. Ainsi, on peut constater que l'usage du web évolue fortement, d'une part en raison de la nature des contenus qui se diversifient et intègrent des données dynamiques et multimédias, d'autre part parce que l'accès au web n'est plus uniquement effectué depuis un PC, mais depuis des terminaux variés comme les ordinateurs portables, les PDA et les téléphones mobiles.

Dans ce contexte, la publication de documents doit répondre à deux catégories de besoins qui peuvent paraître *a priori* contradictoires : d'une part, de nombreuses applications veulent pouvoir obtenir des présentations homogènes pour des documents de même classe et, d'autre part, la multiplicité des terminaux et des contextes de présentation rend nécessaire la production de plusieurs types de présentation à partir d'un même ensemble de sources d'information.

Pour prendre en compte ces évolutions, les acteurs du domaine, qui se retrouvent dans les consortiums et organismes de standardisation, ont mis en place de nouveaux standards pour la définition des formats et des processus de publication. Par ailleurs, de nombreuses équipes de recherche s'intéressent au sujet et proposent de nouvelles pistes, par exemple pour améliorer la qualité de restitution (protocoles de *streaming*, techniques de préchargement), faciliter l'adaptation automatique (architecture de négociation, modèles de transformation, modèles de description sémantique) ou encore offrir de nouveaux environnements d'édition.

En amont d'une réflexion générale sur l'évolution du processus de publication sur le web [Chartron *et al.*, 2004], nous étudions ici quelques formats et

techniques nécessaires pour assurer la mise en place d'une diffusion des contenus adaptée au contexte du lecteur. Ce chapitre est ainsi constitué de deux parties : la première concerne les formats de description de l'information accessible par Internet, tandis que, dans la seconde partie, nous présentons les principes et quelques techniques utilisés pour adapter les contenus aux différents supports de restitution.

◆ 1. Formats de documents pour l'Internet

Nous présenterons tout d'abord quelques définitions pour cadrer notre propos. Nous suivrons ensuite une démarche ascendante dans la description des formats du web : comment sont définis les contenus (en particulier le codage des caractères), quels sont les principes qui ont conduit à la définition des modèles de documents, quels sont les standards du web, et enfin nous décrirons quelques systèmes de présentation de documents.

1.1 Définitions

De nombreux termes sont utilisés pour désigner l'information échangée sur le web, par exemple « média », « objet », « élément de contenu », « document », « présentation », « application multimédia ». Ils se recoupent plus ou moins et, suivant les contextes, ne signifient pas la même chose. Par exemple, le terme « médium/média » signifie « vecteur de transmission » quand il identifie le support papier, radio ou télévisuel mais, dans le contexte de l'informatique, il a perdu son lien avec le support d'origine et est directement associé au format des données véhiculées ; il désigne alors un élément de contenu, indépendant du support de restitution, qui peut être intégré dans une présentation.

La notion de « document numérique » fait l'objet actuellement d'une vaste réflexion pluridisciplinaire dans le cadre du groupe de travail RTPdoc. Un premier texte, signé R. T. Pédauque (Pédauque, 2003), présente une synthèse de ces travaux sur l'étude des conséquences de l'évolution du document papier vers le document électronique[1]. Ce travail se poursuit actuellement sous forme d'un ensemble de publications dans la revue *I3* et d'une nouvelle itération des contributions des membres du groupe [Pédauque, 2004]. R. T. Pédauque a identifié trois axes pour appréhender ce qu'est un document :
- la forme : l'objet matériel (ou immatériel) ;
- le signe : le sens de l'objet ;
- et le médium : le vecteur de transmission.

1 Voir aussi, dans le chapitre 1, pages 14-16.

Dans le cadre qui nous intéresse ici, nous allons nous focaliser sur le premier axe puisque nous considérons le problème de la représentation des contenus et de leur composition au sein de l'objet composite qu'est le « document ». Plus précisément, un auteur crée un document par le regroupement d'un ensemble d'éléments de contenu. Le terme « multimédia » ajoute à cette définition l'intégration de contenus de différentes natures, discrètes comme le texte ou l'image, continues comme le son ou la vidéo. La conséquence est l'introduction de la dimension « temps » puisque la synchronisation entre les éléments de contenu fait partie de la définition de la forme du document.

Les constituants élémentaires d'un document, qui sont appelés « éléments de contenu », « média », peuvent non seulement être de nature variée comme on l'a vu, mais également d'une granularité variable en fonction des besoins de composition : on n'obtient pas la même précision d'alignement spatial ou temporel si, pour le cas du contenu textuel, l'unité élémentaire est le caractère, le paragraphe, la section ou la page web ou si, pour le cas du contenu audiovisuel, l'unité élémentaire est l'image, le plan ou la vidéo entière.

Enfin, les termes de « présentation » et « applications multimédias » identifient l'étape de restitution du document auprès d'un utilisateur à l'aide d'un terminal. La nature plus ou moins dynamique des composants d'un document ainsi que les possibilités d'interaction associées induisent une variété de solutions de systèmes de présentation. Dans le cas de documents statiques, il s'agit d'afficher dans la fenêtre les différents éléments de contenu selon un placement spatial obtenu par le processus de formatage. Dans le cas de documents hypermédias adaptatifs, l'entité de restitution est une véritable application qui nécessite la coopération de plusieurs entités (côté client, côté serveur, au niveau de *proxy*) pour assurer la présentation et l'enchaînement temporel des composants tels qu'ils sont spécifiés dans le scénario de l'auteur.

1.2 Formats des contenus

Sans chercher à être exhaustif dans ce domaine, nous citons ci-dessous un certain nombre de formats qui contribuent à l'émergence de solutions permettant l'adaptation des documents aux modes de lecture. Ainsi, dans le codage des caractères, Unicode constitue une étape majeure pour permettre l'utilisation du web dans toutes les langues en prenant en compte les spécificités des systèmes d'écriture. De même, les formats de description structurés des éléments de contenus complexes, comme les graphiques vectoriels (format SVG) ou les formules mathématiques (MathML), permettent de conserver jusqu'à l'étape de formatage le maximum d'informations utiles pour leur présentation sans figer *a priori* la dimension de leurs constituants. Enfin,

les formats binaires en couches (JPEG2000, pour les images) ou avec des codages différentiels (MPEG2, pour les vidéos) permettent l'adaptation de la qualité de restitution en fonction des ressources (cpu, bande passante) disponibles par le lecteur. Pour aller plus loin dans la mise en place de solutions d'adaptation des contenus, il est nécessaire d'utiliser des descriptions externes au contenu. On parle alors d'annotations de contenu ou de métadonnées qui sont exploitées pour sélectionner et filtrer des fragments à partir desquels de nouvelles structures de documents sont générées. MPEG7, qui est un standard de définition des descripteurs de contenus audiovisuels, s'appuie lui-même sur les langages de description du web, comme XML Schema[2].

1.2.1 Codage du texte : Unicode

Avant de décrire Unicode, il est nécessaire de rappeler quelques définitions à la base du codage informatique des textes (ces définitions sont en grande partie issues du glossaire Unicode[3]) :

- système d'écriture : règles régissant l'utilisation d'une ou de plusieurs écritures pour transcrire une langue particulière.
- graphème : plus petite unité distinctive d'une langue écrite, telle que l'être humain la perçoit. Un graphème est une lettre dans les langues alphabétiques, tandis qu'il peut être un concept dans les écritures idéographiques.
- caractère : unité d'information abstraite utilisée pour représenter des données textuelles. La correspondance entre graphème et caractère n'est pas biunivoque (un graphème peut correspondre à plusieurs caractères, cela dépend du système utilisé, par exemple « à » ou « a+' »). On parle de « caractère codé » lorsqu'on associe une valeur numérique à un caractère.
- jeu de caractères (codés) : ensemble des caractères auxquels on a attribué un code.
- format de codage : représentation en mémoire des caractères.
- glyphe : forme visuelle utilisée pour présenter graphiquement un caractère en fonction du contexte courant (style utilisé, caractères voisins, etc.). Le glyphe ne fait pas l'objet d'une normalisation puisqu'il peut en exister une infinité pour un même caractère. Une ligature est un glyphe qui représente plusieurs caractères.
- police de caractères, fonte : regroupe un répertoire de glyphes pour un ensemble de caractères. Les fontes informatiques sont des bases de données qui associent des fonctions de tracés aux caractères. Normalement, une fonte recouvre tout le jeu de caractères considéré, mais il est fréquent que ce ne soit pas le cas !

2 Voir pages 137 et suivantes.
3 http://iquebec.ifrance.com/hapax/glossaire.htm

1.2.2 Quelques éléments introductifs à Unicode

L'objectif d'Unicode[4] (iso/cei-10646) [André *et al.*, 2002] est de permettre la représentation de la quasi-totalité des écritures du monde (et non des langues), qu'elles soient alphabétiques (grecques, arabes, cyrilliques) ou non alphabétiques (idéographiques comme en Chine) et quel que soit leur mode de représentation (gauche/droite, entrelacé, bidirectionnel, etc.). Unicode propose ainsi un mécanisme universel de codage des caractères qui permet de prendre en charge plusieurs écritures à la fois, facilite l'échange de données textuelles et permet de mettre en œuvre des applications indépendamment des changements de jeux de caractères (par exemple, pour les algorithmes de tri ou de rendu).

Unicode attribue à chacun de ses caractères un nom, un numéro, un ensemble de propriétés et un glyphe représentatif. C'est un modèle en couches qui permet de faire la distinction entre le jeu de caractères et la représentation physique (les glyphes donnés ne le sont qu'à titre d'illustration).

• Définition du jeu de caractères par l'attribution de noms officiels réunis au sein de blocs regroupant les caractères appartenant à un même jeu d'écriture ou à une même famille. Près de 100 000 caractères sont normalisés ainsi. Par exemple, « LATIN CAPITAL LETTER E » (ou en français « Lettre majuscule latine E ») appartient au bloc « Basic Latin » (latin de base). Les noms de caractères Unicode comportent des informations sémantiques sous forme de propriétés (casse, directionnalité, combinabilité, contrôle de restitution, hyphénation, etc.) permettant de les décrire de façon non ambiguë. Ces informations sont stockées dans la base de données UCD (Unicode Character Database).

Le tableau 1 donne quelques exemples de définition de caractères Unicode.

• Attribution d'un numéro appelé « valeur scalaire Unicode » ou « point de code » et noté avec 4 ou 6 chiffres hexadécimaux précédés de « U+ » (ex. : U+0041 pour le caractère « A »).
On peut remarquer que pour attribuer un code aux 95 221 caractères actuellement définis, il est nécessaire de disposer d'un codage d'au moins 17 bits ($2^{16} < 95\,221 < 2^{17}$). Comme le souci de regroupement en blocs cohérents implique un codage à trous, le codage complet est défini sur 32 bits. Le seizet de poids fort identifie un « plan » de 64 000 cellules, chacune pouvant être associée à un caractère. En fait, sept plans seulement sont définis et trois plans sont principalement utilisés :

4 La dernière version est Unicode 4.0.1, mars 2004 : www.unicode.org/

1. le plan multilingue de base (PMB), identifié par le seizet de poids fort à 00_{16}, pour les écritures alphabétiques, syllabiques et idéographiques (Unihan) ainsi que pour les chiffres et les symboles ;
2. le plan multilingue complémentaire (PMC), identifié par le seizet de poids fort à 01_{16}, pour les écritures non idéographiques anciennes ;
3. le plan idéographique complémentaire, identifié par le seizet de poids fort à 02_{16}, pour compléter le jeu des écritures idéographiques.

Tableau 1 - EXEMPLES DE CARACTÈRES UNICODE
(« = » pour équivalence, « => » pour renvoi)

Glyphe représentatif	Description (anglais, français : http://iquebec.ifrance.com/hapax/ListeDesNoms.htm)	Renseignements complémentaires
E	LATIN CAPITAL LETTER E LETTRE MAJUSCULE LATINE E	=> 2107 euler constant => 2130 script capital
à	LATIN SMALL LETTER A WITH GRAVE LETTRE MINUSCULE LATINE A ACCENT GRAVE	= 0061 a 0300
/	SOLIDUS BARRE OBLIQUE	= slash => 2044 / fraction slash => 2215 / division slash …
Ż	LATIN CAPITAL LETTER Z WITH DOT ABOVE LETTRE MAJUSCULE LATINE Z POINT EN CHEF	= 005A Z 0307

• Définition de formats de codage sous forme d'une suite d'entiers d'une largeur fixe ou variable pour stocker les caractères en mémoire :

1. UTF2-8[5] définit l'encodage des caractères sous forme d'une suite variable (de 1 à 4) d'octets. Il permet ainsi d'assurer la compatibilité avec le codage ASCII (sur 7 bits) pour les points de code de valeur 0 à 127 ;
2. UTF-16 permet de définir un encodage fixe sur 16 bits pour tous les points de code du PMB : la valeur d'encodage est alors directement celle du point de code. UTF-16 est également utilisé pour les autres plans, mais deux seizets par point de code sont alors nécessaires, ce qui est possible grâce à un mécanisme d'échappement prévu dans la table Unicode (points de code de U+D800 à U+DFFF) ;
3. UTF-32 offre un encodage à largeur fixe quel que soit le point de code, mais il occupe plus de place en mémoire.

5 UTF pour Unicode Transformation Format

Le tableau 2 illustre les deux premiers formats d'encodage.

De nombreux autres mécanismes sont définis dans ce standard, en particulier le moyen de combiner des caractères de base et diacritiques pour constituer des formes accentuées, ce qui fait qu'il peut exister plusieurs formes Unicode pour un même graphème.

Tableau 2 - EXEMPLES DE POINTS DE CODE ET DE FORMATS DE CODAGE

Glyphe représentatif	Point de code (hexa)	Encodage UTF-16 (hexa, binaire)	Encodage UTF-8 (hexa, binaire)
E	U+0041	0041 : 00000000 01000001	41 : 01000001
à	U+00E0	00E0 : 00000**000 11100000**	C3 : 110**00011** A0 : 10**100000**
Ż	U+017B	017B : 00000**001 01111011**	C5 : 110**00101** BB : 10**111011**
(idéogramme chinois, caractère « cœur »	U+5FC3	5FC3 : **01011111 11000011**	E5 : 1110**0101** BF : 10**111111** 83 : 10**000011**

1.2.3 Utilisation d'Unicode

Comme tout standard, le déploiement d'Unicode dépend de la disponibilité d'outils, en particulier de polices conformes à Unicode (mises en œuvre à partir de tables, dont en tout premier lieu la table de transposition « n° de point de code -> n° de glyphe ») et de logiciels de rendus qui prennent en compte les spécificités de langues pour obtenir le rendu souhaité.

L'annonce du support d'Unicode par un système ou un logiciel peut en fait recouvrir une réalité bien différente selon les cas. Par exemple, est-il possible de déterminer l'encodage d'un fichier ? Comment sont effectuées les comparaisons de chaînes, les tris ?

Le jeu de caractères utilisé par XML est Unicode avec comme encodage par défaut UTF-8, ce qui permet d'assurer facilement la compatibilité des fichiers ASCII.

1.3 Structuration des documents

Jusqu'à la fin des années quatre-vingt, le déploiement de techniques documentaires s'est effectué principalement dans des secteurs comme la

documentation technique ou les métiers du livre dans lesquels les volumes d'information sont importants et pour lesquels l'échange de documents sous une forme électronique exploitable est nécessaire. Ainsi, des solutions à base de formats standards comme SGML ou HyTime ont émergé. Elles reposent sur une structuration et un typage fort des données documentaires [Furuta *et al.*, 1988]. Un exemple caractéristique est l'aéronautique, qui utilise des techniques fondées sur SGML dans ses chaînes documentaires, depuis la création et la gestion de fonds documentaires jusqu'à la diffusion des documents techniques aux clients grâce à la définition de DTD communes, comme celles de l'ATA (Air Transport Association). Ainsi, l'industrie documentaire a obtenu une certaine maîtrise de la chaîne de traitement des documents grâce à l'utilisation de modèles de documents SGML/XML, tandis que les aspects dynamiques et hypermédias sont traités dans les applications multimédias le plus souvent sous forme de programmes spécifiques. Les formats et techniques développés dans le contexte du web tirent parti de ces différentes solutions, qui ne sont malgré tout pas suffisantes puisque les besoins d'échange à grande échelle amènent à considérer des critères comme la portabilité, l'adaptabilité ou l'accessibilité des informations.

Plus concrètement, de nombreux langages de description de documents sont utilisés dans les applications documentaires. Ils permettent de définir de façon plus ou moins effective les caractéristiques suivantes :
- le contenu : textuel, graphique, sonore, réactif, etc.
- l'organisation hiérarchique logique, qui exprime le regroupement des éléments de contenu selon un ensemble de règles constituant le modèle ou schéma du document (en XML, ce modèle est défini par une DTD ou *Document Type Definition* ou un schéma XML).
- l'organisation spatiale, c'est-à-dire la description de l'apparence physique des contenus, qui comprend les propriétés typographiques (taille des caractères, couleur, etc.) et des propriétés de placement spatial. Ces informations sont décrites à l'aide de langages de style (comme les CSS, Cascading Style Sheets).
- l'organisation temporelle, nécessaire dès que l'on veut intégrer des éléments dynamiques. La durée des objets, leur placement temporel et leur synchronisation constituent alors le *scénario temporel*.
- les possibilités de navigation, à travers des liens hypertextes, des objets d'interaction. La structure de navigation est décrite par une structure de graphe.
- l'organisation sémantique, qui peut être exprimée en partie à travers les structures logiques et hypertextes et complétée par des métadonnées.

Ainsi, la définition d'un *format de représentation* des documents structurés (et éventuellement multimédias) doit permettre d'exprimer les caractéristiques

relatives à chacune des différentes dimensions de ces documents : dimensions logique, spatiale, hypermédia et temporelle. S'il est parfois utile de décrire séparément ces différents aspects relatifs à un document (d'où le terme de « dimension » fréquemment employé), il est clair qu'ils ne sont pas indépendants. Par exemple, pour spécifier le déplacement d'un objet sur l'écran, il est nécessaire d'exprimer une information spatiale pour la trajectoire, comme les positions initiale et finale correspondant au déplacement, ainsi qu'une information temporelle, comme la durée du déplacement et sa synchronisation avec d'autres objets.

1.4 Standards de description de documents web

Les principes décrits ci-dessus ont été largement appliqués pour la définition des formats d'échanges du web : formats textuels et non binaires, approche déclarative, langage de description de structures hiérarchiques, possibilité de définir des classes de documents, séparation des structures de contenu des structures de présentation. Dans ce qui suit, nous résumons le rôle d'un certain nombre de standards du web, puis nous présentons une illustration de la description de documents XML pris dans le domaine de la gestion des publications : la représentation des ressources bibliographiques et des numéros ISBN en XML[6]. Enfin, nous donnons quelques éléments de syntaxe du langage SMIL qui permet d'intégrer des contenus dans l'espace et le temps[7].

Remarquons tout d'abord, puisque la base des formats de description et d'échange de l'information sur le web est effectuée sous forme de suites de caractères, qu'il est fondamental que le jeu de caractères utilisé soit un standard permettant de couvrir toutes les écritures. Il est donc naturel qu'Unicode soit le standard choisi[8].

Les formats de représentation standards s'appuient tous sur la syntaxe XML comme format de structuration et de description.

1. Les éléments source peuvent être organisés selon des structures et un vocabulaire « métier ». La plupart des grands domaines d'activité ont fait l'objet d'un travail de définition de leurs vocabulaires sous forme de DTD ou de schémas XML. Le site XML Applications and Initiatives [Cover, 2004] référence ainsi plus de 700 définitions ; par exemple, TransportationXML (tXML), Election Markup Language (EML), Trading Partner Agreement Markup Language (tpaML) ou encore XML System for Textual and Archaeological Research (XSTAR). Les DTD et les schémas XML sont utiles pour homogénéi-

6 Voir pages 139-142.
7 Voir pages 142-143.
8 Voir pages 131-135 et pages 146-148.

ser les formats de documents d'une même organisation, valider par traitement informatique les documents émis ou reçus, simplifier la saisie de documents complexes à l'aide d'une interface adéquate.

2. XML est utilisé pour décrire les contenus selon différents niveaux de granularité en fonction des applications visées. Par exemple, MPEG7 permet d'annoter les contenus audiovisuels à l'aide des descripteurs de différents niveaux structurels ou sémantiques. Les contenus structurés comme les mathématiques ou le graphique vectoriel peuvent être décrits en XML avec MathML et SVG. Un des intérêts d'utiliser ces formats de description plutôt que d'insérer des formats binaires (sous forme d'images) pour ces contenus est la possibilité d'appliquer les mêmes traitements sur leur contenu, comme la correction orthographique d'un texte dans un graphique.

Figure 1 - FORMULE MATHÉMATIQUE EN MATHML

```
</math><math xmlns="http://www.w3.org/1998/Math/MathML">
   <mfrac>
      <mn>1</mn>
      <mrow>
         <msqrt>
            <mrow>
               <mn>6</mn>
               <mi>x</mi>
            </mrow>
         </msqrt>
         <mrow>
            <mo>+</mo>
            <mn>1</mn>
         </mrow>
      </mrow>
   </mfrac>
</math>
```

3. La définition de la présentation : XML est également utilisé pour définir des structures de présentation, que ce soit pour les présentations statiques, XHTML étant l'exemple le plus répandu, ou dynamiques, comme le langage d'intégration de média SMIL. D'autres langages permettent de définir l'apparence des documents, comme les langages de style CSS ou XSL-FO, et de liens Xlink. Notons que la syntaxe utilisée par CSS n'est pas XML tandis que celle d'XSL-FO l'est.

4. La spécification des transformations : dès lors qu'on décrit l'information avec plusieurs langages XML, le langage XSLT est l'outil nécessaire pour spécifier la transformation des structures d'un modèle à l'autre. En particulier, un schéma de production largement répandu consiste à définir et structurer

les contenus selon un schéma « métier » et à les transformer vers un ou plusieurs langages de présentation cibles (XHTML, XSL-FO, SMIL) à l'aide de feuilles de style écrites en XSLT.

1.5 Exemples de modèles et d'instances de documents

Voici un exemple de document comprenant des ressources bibliographiques. Ces ressources (très simplifiées !) comprennent le numéro ISBN[9].

Figure 2 - INSTANCE DE DOCUMENT BOOKCATALOGUE

```
<?xml version="1.0"?>
<BookCatalogue xmlns="http://www.publishing.org"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.publishing.org
        BookCatalogue.xsd">
   <Book>
       <Title>The First and Last Freedom</Title>
       <Author>J. Krishnamurti</Author>
       <Date>1954</Date>
       <ISBN>0-06-064831-7</ISBN>
       <Publisher>Harper &amp;amp; Row</Publisher>
   </Book>
   <Book>
       <Title>Illusions The Adventures of a Reluctant
       Messiah</Title>
       <Author>Richard Bach</Author>
       <Date>1977</Date>
       <ISBN>0-440-34319-4</ISBN>
       <Publisher>Dell Publishing Co.</Publisher>
   </Book>
   <Book>
       <Title>Document numérique 3-4 (2002) Unicode, écriture du
       monde?</Title>
       <Author>J. André & H Hudrisier</Author>
       <Date>2002</Date>
       <ISBN>2-7462-0594-7</ISBN>
       <Publisher>Hermès Science Publications</Publisher>
   </Book>
</BookCatalogue>
```

Pour comprendre l'intérêt d'utiliser un schéma XML plutôt qu'une DTD XML, nous donnons le modèle de ces ressources selon les deux langages. L'instance de document de la figure 2 est donc conforme aux deux modèles suivants, l'un défini selon la syntaxe des DTD (fig. 3), l'autre selon celle de XML Schema (fig. 4).

9 Adapté de www.xfront.com/isbn.html

Figure 3 - MODÈLE POUR LES ÉLÉMENTS BOOKCATALOGUE SELON LA SYNTAXE DES DTD

```
<!DOCTYPE BookCatalogue [
<!ELEMENT BookCatalogue  (book* )>
<!ELEMENT book  (title, author, date, isbn, publisher)>
<!ELEMENT title      (#PCDATA )>
<!ELEMENT author     (#PCDATA )>
<!ELEMENT isbn       (#PCDATA )>
<!ELEMENT date       (#PCDATA )>
<!ELEMENT publisher (#PCDATA )>
]>
```

Le schéma XML ci-dessous définit le type d'élément *BookCatalogue* dans l'espace de noms *www.publishing.org* et y inclut le schéma *isbn.xsd*.

Figure 4 - MODÈLE POUR LES ÉLÉMENTS BOOKCATALOGUE SELON LA SYNTAXE DE XML SCHEMA

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNa-
mespace="http://www.publishing.org"
xmlns="http://www.publishing.org" elementFormDefault="qualified">
   <xsd:include schemaLocation="isbn.xsd"/>
   <xsd:element name="BookCatalogue">
       <xsd:complexType>
            <xsd:sequence>
                 <xsd:element name="Book" minOccurs="1"
                 maxOccurs="unbounded">
                      <xsd:complexType>
                           <xsd:sequence>
                                <xsd:element name="Title"
                                type="xsd:string"/>
                                <xsd:element name="Author"
                                type="xsd:string"/>
                                <xsd:element name="Date"
                                type="xsd:string"/>
                                <xsd:element name="ISBN"
                                type="ISBN-type"/>
                                <xsd:element name="Publisher"
                                type="xsd:string"/>
                           </xsd:sequence>
                      </xsd:complexType>
                 </xsd:element>
            </xsd:sequence>
       </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

Enfin, le schéma XML *isbn.xsd* permet de spécifier que le numéro ISBN est une chaîne de caractères qui respecte un certain nombre de contraintes comme « c'est une séquence d'exactement 10 chiffres comprenant 4 champs – pays, éditeur, numéro du titre, vérification – séparés par une espace ou par un tiret « - », etc. ». Ainsi, le type *isbn* est défini comme une *restriction* du type de base *string* dont on peut énumérer la liste des syntaxes possibles en fonction du pays et du nombre de titres alloués pour l'éditeur. Nous donnons en figure 5 un extrait de ce schéma (correspondant aux trois exemples ci-dessus).

Figure 5 - SCHÉMA XML POUR LE TYPE ISBN-TYPE

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:simpleType name="ISBN-type">
      <xsd:restriction base="xsd:string">
           <xsd:pattern value="0-[0-1][0-9]-\d{6}-[0-9x]">
                <xsd:annotation>
                     <!-- ISBN definition for English-speaking
                     Countries -->
                     <xsd:documentation>
                          group/country ID = 0 (hyphen after the
                          1st digit)
                          Publisher ID = 00…19 (hyphen after the
                          3rd digit)
                          Block size = 1,000,000 (requires 6
                          digits)
                          check digit is 0-9 or 'x'
                     </xsd:documentation>
                </xsd:annotation>
           </xsd:pattern>
           <xsd:pattern value="0-[2-6][0-9]{2}-\d{5}-[0-9x]">
                <xsd:annotation>
                     <!-- ISBN definition for English-speaking
                     Countries -->
                     <xsd:documentation>
                          group/country ID = 0 (hyphen after the
                          1st digit)
                          Publisher ID = 200…699 (hyphen after
                          the 4th digit)
                          Block size = 100,000 (requires 5
                          digits)
                          check digit is 0-9 or 'x'
                     </xsd:documentation>
                </xsd:annotation>
           </xsd:pattern>
           <xsd:pattern value="2-\d([0-9]|-){7}\d-[0-9x]">
                <xsd:annotation>
                     <!-- ISBN definition for French-speaking
```

```
                    Countries -->
                    <xsd:documentation>
                         group/country ID = 2 (hyphen after the
                         1st digit)
                         Country = France, Belgium, Canada,
                         Switzerland
                         check digit is 0-9 or 'x'
                    </xsd:documentation>
               </xsd:annotation>
          </xsd:pattern>
          ...
          ...
     </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```

On peut constater que, même s'il n'est pas possible de spécifier toutes les contraintes souhaitées (dans l'exemple ci-dessus, une feuille de transformation XSLT[10] a été définie pour affiner les contraintes), la structure des numéros ISBN en XML Schema n'est pas une simple chaîne de caractères comme elle est définie dans la DTD. Par ailleurs, XML Schema offre quelques mécanismes pour définir des types par héritage (restriction, extension) et permet de créer ses schémas de façon modulaire grâce à une meilleure intégration avec les espaces de noms. Enfin les schémas XML sont eux-mêmes des documents XML, ce qui permet de les traiter avec les mêmes outils que les instances.

1.6 Description des documents multimédias avec le langage SMIL

Le langage SMIL (*Synchronized Multimedia Integration Language*) est un langage d'*intégration* de média (au sens « d'élément de contenu ») pour réaliser des présentations audiovisuelles interactives accessibles sur le web. Le terme « intégration » concerne non seulement le placement spatial des objets dans l'écran, mais surtout leur agencement temporel défini sous forme d'un scénario qui exprime leurs synchronisations et enchaînements. À ces deux fonctions de base, placement spatial et placement temporel, est associé un ensemble d'éléments du langage qui couvrent toute une palette de besoins complémentaires : spécification de transitions, de liens hypermédias, d'animations et de contrôle fin du temps. SMIL est un langage XML qui vise à permettre aux auteurs d'utiliser une approche déclarative pour décrire leurs documents multimédias.

10 Voir pages 148-152.

Le modèle de synchronisation comporte trois types d'éléments syntaxiques :
1. des opérateurs temporels : par, seq et excl.
2. des structures de contrôle : switch et repeat.
3. des attributs de synchronisation : begin, end, dur et endsync.

Le modèle de placement spatial utilise deux types d'éléments syntaxiques :
1. des régions hiérarchiques : root-layout, region.
2. des attributs de placement : top, width, etc.

La structure d'un document SMIL (voir fig. 6) reflète ce besoin de prendre en compte deux structures indépendantes :
- la structure spatiale, par la hiérarchie d'éléments « region » dans la partie « layout » ;
- et la structure temporelle, par la hiérarchie d'éléments « seq », « par », « excl » dans la partie « body ».

Cependant, comme l'accent est mis sur la spécification des synchronisations entre éléments, c'est la structure temporelle qui est la structure maître : ses feuilles contiennent ou référencent les éléments de contenu de type « text », « img », « video », « audio », « a » (ancre), etc., qui eux-mêmes référencent la région d'affichage dans laquelle ils doivent être placés (attribut « region »).

Figure 6 - EXTRAIT D'UN SOURCE SMIL

```
<smil>
 <head>
  <layout>
   <root-layout width="250" height="500"
         background-color="blue" />
   <region id="logo" left="25" top="30" width="32"… />
   <region id="pres" left="50" top="70" width="100"… >
        <region id="sous-part"…/>
   </region>
   …
  </layout>
 </head>
 <body>
  <par>
   <seq>
       <img src="jojo.gif" region="logo" dur="6s" />
       <img src="lolo.gif" region="logo" dur="6s" />
   </seq>
   <video src="…" region="pres" dur="20s" />
  </par>
<body>
</smil>
```

1.7 Systèmes de production de contenus web

Il est impossible de lister tous les outils à disposition des développeurs de contenus et de schémas web[11]. Ces outils comprennent les environnements de conception de DTD et de schémas XML, de production de contenus structurés, de définition de feuilles de style et de transformation. Par exemple, près de 120 outils d'édition sont répertoriés sur la page des éditeurs XML de xml.com[12] et les outils de production, publication et gestion de contenus sont tout aussi nombreux[13].

1.7.1 Le cas du langage SMIL

Dans le domaine multimédia, la palette d'outils n'est pas aussi riche mais elle est surtout plus complexe (voir le tableau 3 ci-dessous listant les principaux logiciels de présentation ou *players SMIL*). En effet, les outils de visualisation SMIL n'offrent pas tous le même service en termes de couverture du standard et d'interopérabilité. Ces logiciels prennent en compte de façon homogène tout ou partie des éléments de structuration de SMIL (placement dans les régions spatiales, synchronisation temporelle), mais offrent une grande variété de traitements des contenus (format des éléments et attributs de style). En particulier, les éléments textuels sont traités de façon différente selon les logiciels : soit ce sont des éléments externes au source SMIL (au format standard html ou propriétaire comme RealText), soit ils peuvent être insérés dans le source SMIL.

Si l'on considère à présent les outils utilisés pour éditer des documents SMIL, on constate que les auteurs disposent d'une grande diversité de méthodes, mais qui cache en fait le manque d'outils vraiment complets et satisfaisants. Le premier niveau d'outils disponibles offre le moyen d'éditer le code source SMIL avec plus ou moins de services avancés : contrôle de structure, visualisation améliorée de la structure et des attributs, formulaires d'édition. L'environnement peut assurer simplement la validité XML (document bien formé), la validité vis-à-vis de la DTD SMIL (cas des éditeurs XML, comme Xerlin ou XmlSpy) ou peut s'appuyer sur la connaissance de la DTD pour offrir à l'auteur des menus et formulaires contextuels ; par exemple, SMIL Gen de Real Networks. Certains outils généralistes tels que GoLive CS d'Adobe ou Macromedia Flash intègrent des fonctions d'export vers des formats multimédias standards MMS, SMIL ou MPEG4.

11 Voir aussi, à ce sujet, le chapitre 3, pages 105 et suivantes.
12 www.xml.com/pub/pt/3
13 www.xml.com/pub/pt/1

Tableau 3 - LOGICIELS DE PRÉSENTATION DU LANGAGE SMIL

Nom du logiciel	Version/profil SMIL	Lien	Commentaires
Ambulant	SMIL2.0, SMIL 2.0Basic	http://www.cwi.nl/projects/Ambulant/distPlayer.html	Effort pour assurer une large couverture de formats
GriNS	SMIL2.0	http://www.oratrix.com	Éditeur et « player ». Le système n'est plus développé depuis 2002
Helix	SMIL 2.0	https://www.helixcommunity.org	Navigateur de Real Networks en code ouvert
Internet Explorer	xhtml+time (non standard)	http://www.microsoft.com/windows/ie/default.asp	Navigateur XHTML + SMIL
NetFront SMIL Player	MMS	http://www.access.co.jp/english/products/	Navigateur MMS pour les téléphones UMTS
PocketSMIL	SMIL 2.0Basic	http://wam.inrialpes.fr/software/pocketsmil/	Prototype de navigateur pour les systèmes embarqués
QuickTime	SMIL1.0	http://www.apple.com/quicktime	
RealOne	SMIL2.0	http://www.real.com/realone/index.html	Logiciel le plus utilisé Limitation dans les formats de contenu acceptés
RealPlayer	SMIL1.0	http://www.real.com	Premier player SMIL
SMIL Scenario Creator	MMS	http://avs.kddlabs.co.jp/sc/indexe.html	Éditeur MMS
Xsmiles	SMIL.2.0	http://www.x-smiles.org/	Navigateur intégrant SMIL à plusieurs standards (SVG, XForms, XHTML)

Les deux environnements qui ont vraiment cherché à couvrir les besoins d'édition de SMIL sont GriNS Editor d'Oratrix et LimSee de l'INRIA [Weck, 2004]. Tous deux proposent des services d'édition riches à partir d'un ensemble de vues, dont une vue temporelle. Dans le cas de LimSee, ces vues offrent également des services complémentaires, comme l'aspiration de présentations SMIL existantes avec un mode d'ouverture incrémentale des fichiers mal formés, l'importation de documents PowerPoint pour générer des « *slideshow* » en SMIL, et bien sûr, le contrôle de validité des documents en cours d'édition.

◆ 2. Techniques d'adaptation

Dans cette section, nous abordons quelques moyens mis en place pour assurer l'accès à l'information du web. Comme évoqué précédemment, il ne s'agit pas seulement de mettre en place des techniques de restitution (accès aux médias, formatage, synchronisation spatiale et temporelle, navigation, etc.), car la variété des terminaux implique de reconsidérer toute la chaîne de présentation des documents depuis la source jusqu'à la publication.

Dans la première partie sont résumés les travaux des trois groupes de travail du W3C qui oeuvrent à l'accès universel du web. Puis nous présenterons les principes de publication multi-cibles qui s'appuient sur des techniques de transformation. Nous terminerons ce chapitre par les possibilités qu'offre le langage SMIL pour assurer l'adaptation dynamique des présentations multimédias.

2.1 Les travaux du W3C

Comme son nom l'indique, le web a pour objectif l'universalité. Il faut donc que les technologies mises en œuvre dans le web permettent l'accès à tous (critère d'accessibilité), pour tous, quelle que soit la langue (critère d'internationalisation) et quel que soit le cadre d'accès (critère d'indépendance aux terminaux).

2.1.1 Accessibilité

Le premier objectif est pris en compte dans le groupe de travail WAI du W3C (Web Accessibility Initiative). Ce groupe produit des *guidelines* non seulement pour les créateurs de contenu « Web Content Accessibility Guidelines », mais aussi pour les logiciels d'édition « Authoring Tool Accessibility Guidelines ».

Ainsi, les règles de conduite de création de contenu web, proposées par le WAI [Caldwell *et al.*, 2004], visent à permettre l'accès à tous au web, indépendamment des handicaps. Ce contenu doit pouvoir être *perceptible* même lorsque l'utilisateur ne peut lire, entendre, utiliser la souris, etc. Les principes de séparation contenu/présentation[14] sont à la base des solutions d'accessibilité, puisqu'ils permettent de tirer parti de la structure en fonction du contexte d'accès en attachant des feuilles de style différentes. Ils permettent également d'aider à la compréhension du contenu grâce à la perception de la structure (énumérer oralement une liste de sections, par exemple). Pour cela, le groupe WAI préconise d'offrir différentes formes à un même contenu par

14 Voir pages 135-137.

un mécanisme d'alternatives (affichage avec ou sans couleur, transcodage texte <-> audio, légendes textuelles synchronisées aux vidéos et audios, etc.). Il est également nécessaire d'offrir plusieurs modes de navigation, non seulement avec la souris ou le clavier, mais avec la parole.

2.1.2 Internationalisation

Le W3C a constitué le groupe de travail « W3C Internationalization Activity » (ou I18N) pour coordonner ses activités internes et externes relatives au déploiement de solutions permettant l'utilisation du web dans différentes langues, écritures et cultures. Les solutions s'appuient sur Unicode qui seul permet de traiter toutes les écritures du monde, mais il est nécessaire d'aller plus loin pour expliciter, dans les documents et les applications, les encodages utilisés au niveau des caractères et les langues utilisées. Ces informations sont indispensables pour la mise en place d'applications de traitement des contenus, comme l'affichage, le transcodage, la correction, la traduction, etc. Ainsi, sont en cours de définition :

1. le standard « Character Model for the World Wide Web 1.0: Fundamentals », *working draft* du 25 février 2004, qui indique comment rendre interopérables les manipulations de textes dans les applications du web. Outre la définition précise des termes employés dans le contexte de l'utilisation du jeu de caractères Unicode[15], ce texte indique les règles à suivre pour la normalisation des caractères, la définition de caractères d'échappement ou encore pour l'indexation dans les chaînes de caractères ;

2. l'« Internationalized Resource Identifiers » (IRIs), IETF *draft* du 9 mai 2004, qui vise à permettre d'utiliser UCS[16], le jeu de caractères Unicode, pour identifier les ressources du web. En effet les URI (Uniform Resource Identifiers) sont contraints à être définis avec le jeu de caractères ASCII, ce qui empêche les utilisateurs non anglophones de donner des noms significatifs à leurs ressources web.

2.1.3 Indépendance par rapport aux terminaux

Ce groupe de travail W3C a pour objectif de développer et promouvoir des solutions pour rendre le web accessible à travers une grande variété d'appareils. Là encore, ces solutions s'appuient fortement sur le principe d'indépendance entre contenu et présentation puisqu'il permet de définir des formats indépendants des dispositifs de présentation. Certains formats offrent également le moyen de spécifier des alternatives en fonction du contexte[17],

15 Voir ci-dessus, pages 131-134.
16 UCS : Universal Character Set (jeu de caractères universel).
17 Voir le cas de SMIL, pages 152-153.

conduisant à la création de documents adaptables. Mais ce n'est pas suffisant, car la variété des appareils de restitution et de leur contexte d'utilisation (bande passante) nécessite de disposer d'un véritable langage de description de profils d'appareils et d'utilisateurs tel qu'il est défini par le standard CC/PP (Composite Capability/Preference Profiles). L'adaptation peut alors être mise en œuvre par une combinaison de techniques comme la négociation préalable entre le client et le serveur, la sélection ou le transcodage des contenus, ou encore la transformation complète des documents en fonction des profils.

2.2 L'adaptation par transformation de structures

De façon générale, le processus de présentation prend en entrée un document XML quelconque et produit un document formaté [Roisin *et al.* 2000]. Le document source appartient à une classe de document, c'est-à-dire qu'il est valide par rapport à une DTD ou un schéma XML particulier. La présentation du document s'effectue alors en deux étapes.

• La première est une étape de *transformation* qui, à partir du document source, permet de sélectionner et de (re)structurer les informations de contenu selon les capacités et les besoins du système de restitution. Elle permet aussi de positionner (dimension spatiale) les objets médias, de les lier (dimension hypermédia) et si besoin de les synchroniser (dimension temporelle). Le résultat de cette transformation est une spécification du document décrite dans un langage de formatage (XHTML, XSL-FO, SMIL) qui permet d'encoder les propriétés spatiales, temporelles et hypermédias du document.

• Lors de la seconde étape, le document multimédia est *formaté* de telle façon que le résultat puisse être directement interprété par le système de présentation. Cette étape peut faire appel à différentes techniques de formatage comme la résolution de contraintes ou l'interprétation de règles d'héritage pour produire le document formaté.

C'est la première étape que nous allons plus précisément détailler ici puisque c'est elle que l'utilisateur devra maîtriser pour atteindre ses objectifs d'adaptation multi-cibles. Grâce aux travaux de standardisation, les techniques et les outils de transformation sont largement stabilisés et disponibles. Le langage standard de transformation de structures XML est XSL-T. C'est un langage de transformation à base de règles dont nous résumons dans la suite la syntaxe à travers un exemple de transformation pour la génération de deux types de présentation pour des ressources bibliographiques de la section 1.5 (voir fig. 7 et 8).

2.2.1 Éléments de syntaxe XSL-T

Un fichier XSL-T est constitué d'un élément racine : *stylesheet* et d'une suite d'éléments fils de la racine : *template*. Chaque élément *template* définit une règle qui comprend deux parties.

1. Une partie déclaration ou sélecteur, définie par l'attribut *match* dont la valeur est une expression Xpath, qui permet de spécifier les nœuds source pour lesquels la règle doit être appliquée. Xpath permet de sélectionner des nœuds en fonction de leur type, de leur contexte ou encore de la valeur de leurs attributs. Voici quelques exemples de requêtes :
- *child::ISBN* sélectionne tous les fils du nœud courant qui sont des éléments de type *ISBN* ;
- *child::Book[position()=1]* sélectionne le premier fils du nœud courant qui est un élément de type *Book*.

2. Une partie exécution (ou corps de la règle) qui est le contenu de l'élément *template*. Elle est constituée d'une suite d'instructions XSL (espace de nommage XSL) qui permettent :
- de contrôler le processus de transformation (instructions),
- de générer des balises et des attributs dans l'arbre cible,
- de copier des éléments source vers l'arbre cible.

L'application d'une règle s'apparente à un appel de méthode pour les langages de type impératif. La différence réside dans le fait que la règle à appeler n'est pas nommée directement, il est nécessaire de la rechercher en fonction du nœud source courant. À l'état initial, le nœud courant est le nœud racine. Une règle est recherchée et, si elle existe, alors le corps de la règle est exécuté.

Les éléments/instructions de transformation permettent de spécifier l'enchaînement des règles à appliquer ainsi que les contenus à générer. Voici quelques instructions XSL avec la description de leur sémantique :
- *apply-templates* : lorsque cet élément est exécuté, une liste ordonnée de nœuds source est sélectionnée à partir de la requête spécifiée avec l'élément sous forme d'attribut (attribut *select*). Pour chacun de ces nœuds source, une règle est recherchée et appliquée si elle est trouvée ;
- *value-of* : cette instruction génère un contenu à partir du contenu source sélectionné. Par exemple *<value-of select="Title"/>* génère le contenu de l'élément *Title* et *<value-of select="@day"/>* génère la valeur de l'attribut *day* de l'élément source courant ;
- *if* : cette instruction a un attribut *test* qui spécifie une expression booléenne. Si l'évaluation de cette expression est vraie, alors le contenu de l'instruction *if* est instancié, sinon rien n'est créé ;
- *include/import* : ces instructions permettent respectivement d'inclure ou

d'importer un ensemble de règles spécifié dans un fichier externe. L'inclusion des règles de transformation est un simple ajout de règles qui complète la feuille de transformation. L'importation de règles définit une hiérarchie de feuilles qui permet de compléter et surcharger les règles de la feuille importée. Grâce à ces instructions, il est possible de rendre modulaires les feuilles de transformation.

2.2.2 Exemples de transformation XSL-T

Le premier exemple contient deux règles de transformation spécifiées grâce à l'élément *xsl:template*. La première règle, dont l'attribut est *match="/"*, sera appliquée pour la racine du source XML, tandis que la seconde règle, d'attribut *match="Book"*, sera appliquée pour tous les éléments de type *Book*. L'instruction *<xsl:value-of select="Title"/>* permet de copier le contenu de l'élément *Title* de l'arbre source à l'emplacement courant de l'arbre cible (ici un élément HTML *li*).

Figure 7 - FEUILLE DE TRANSFORMATION XSL-T QUI EXTRAIT LES TITRES DE LIVRES D'UN DOCUMENT BOOKCATALOGUE DANS UN DOCUMENT HTML

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html"/>
<xsl:template match="/">
      <html>
      <body>
      <h1>Liste des livres</h1>
      <ul>
      <xsl:apply-templates/>
      </ul>
      </body>
      </html>
</xsl:template>
<xsl:template match="Book">
      <li><xsl:value-of select="Title"/></li>
</xsl:template>
</xsl:stylesheet>
```

Le deuxième exemple, adapté de [Bonhomme, 2002], permet de sélectionner les titres et les numéros ISBN et de générer un document cible dans le langage XSL-FO. On peut constater que le langage de formatage XSL-FO permet de définir de façon très fine les modèles de page souhaités et permet d'exprimer comment les blocs de contenus sont organisés dans la page. Le langage de transformation XSL-T permet d'exprimer le passage d'une structure logique (ici le schéma XML BookCatalogue) vers une structure de

Figure 8 - FEUILLE DE TRANSFORMATION XSL-T QUI EXTRAIT LES TITRES ET LES NUMÉROS ISBN D'UN DOCUMENT BOOKCATALOGUE ET CRÉE UN DOCUMENT XSL-FO

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:fo="http://www.w3.org/1999/XSL/Format"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:template match="/">
      <fo:root>
      <fo:layout-master-set >
      <fo:simple-page-master
        master-name="PageCatalogue"
        margin-top="75pt"
        margin-bottom="70pt"
        margin-left="75pt"
        margin-right="55pt">
        <fo:region-before extent="10mm" margin-top="20pt"/>
        <fo:region-body margin-top="50pt" margin-bottom="50pt"/>
        <fo:region-after extent="10mm" margin-bottom="20pt"/>
      </fo:simple-page-master>
      </fo:layout-master-set>
<fo:page-sequence master-reference="PageCatalogue">
      <fo:static-content flow-name="xsl-region-before">
        <fo:block>en tête</fo:block>
      </fo:static-content>
      <fo:static-content flow-name="xsl-region-after">
        <fo:block font-size="6pt">
            <xsl:text>Page </xsl:text>
            <fo:page-number font-size="6pt"/>
        </fo:block>
      </fo:static-content>
      <fo:flow flow-name="xsl-region-body">
        <xsl:for-each select="BookCatalogue/Book">
        <fo:block font-size="8pt"
          text-align="center" space-after="4pt"
          space-before="7pt">
          <xsl:value-of select="Title" />
        </fo:block>
        <fo:block font-size="6pt"
          text-align="center" space-after="4pt">
          N° ISBN: <xsl:value-of select="ISBN" />
        </fo:block>
        </xsl:for-each>
      </fo:flow>
      </fo:page-sequence>
      </fo:root>
</xsl:template>
</xsl:stylesheet>
```

formatage XSL-FO dans laquelle la hiérarchie correspond à celle de mise en page exprimée en termes de pages, régions et blocs. De même, à ces structures de formatage sont associés des attributs de style (marges, espacements, taille de fonte, etc.).

2.3 L'adaptation de documents multimédias

Les « documents structurés multimédias » se trouvent au cœur de nombreuses classes d'applications, comme la réalisation de cédéroms interactifs, de sites web multimédias, ou comme les systèmes de télévision interactive et de visioconférences. Comme on l'a vu, ces applications nécessitent toutes (à des degrés divers) de prendre en compte les problèmes d'intégration d'éléments multimédias avec leur synchronisation temporelle (ordonnancement temporel), de stockage de données volumineuses nécessitant des techniques de compression et d'adaptation de flux en fonction des contraintes d'accès. Par exemple, la même présentation doit pouvoir être jouée de façon satisfaisante, même lorsque les conditions du réseau changent (les synchronisations doivent être maintenues, le format des contenus peut être changé sans nuire à la sémantique du document). De même, une présentation doit pouvoir être restituée dans différentes langues sans nécessiter de dupliquer les contenus et la structure qui ne dépendent pas de la langue.

Le langage SMIL offre quelques éléments de réponse pour couvrir cet objectif d'adaptibilité :

1. le standard ayant été conçu de façon modulaire, différents profils ont été définis, chacun correspondant à un contexte de restitution différent : le profil SMIL2.0 Basic est adapté aux terminaux de faibles capacités comme les PDA tandis que le profil SMIL2.0 complet vise les PC ;

2. de façon plus fine, il est possible d'utiliser des éléments syntaxiques SMIL pour permettre l'adaptation du document au moment de sa présentation. Le constructeur « *switch* » associé aux attributs de sélection comme « systemScreenSize » ou « systemBitrate » permet de combiner une structure et/ou un contenu partagé(s) avec des fragments spécifiques d'un contexte. Le contexte peut être défini à l'aide d'attributs prédéfinis comme la taille de l'écran ou la langue, mais il peut être également défini de façon *ad hoc* à l'application ;

3. un attribut de préchargement permet de demander l'envoi en avance du contenu pour assurer la bonne synchronisation entre éléments (comme le démarrage simultané d'une vidéo et d'un commentaire audio associé).

Ainsi, Nicoll [Nicoll *et al.*, 2003] et la SMILthèque de l'INRIA [INRIA, 2004] utilisent SMIL pour diffuser des séminaires comprenant la synchronisation

des transparents avec le film vidéo et la navigation dans ce flux à l'aide d'une table des matières.

Pour aller plus loin et réaliser des applications dont les contenus sont adaptés aux utilisateurs (exemple typique, la réalisation d'applications pédagogiques), il faut compléter ces processus de production avec des mécanismes d'adaptation (par filtrage et/ou restructuration) et enrichir les modalités d'interaction. Pour être exploités de façon dynamique et flexible, ces mécanismes doivent être utilisés dans le contexte d'une architecture plus complexe que celle d'un simple client/serveur classique. Une étape de négociation entre le client et le serveur doit précéder l'envoi des éléments du document (source SMIL et éléments de contenu). Cette négociation, qui s'appuie sur des structures d'identification du contexte comme définies dans CC/PP, peut être mise en œuvre au niveau du serveur ou par un *proxy* [Lemlouma *et al.*, 2004].

L'auteur tient à remercier Irène Vatton pour son partage d'expérience sur Unicode ainsi que tous les membres des projets Opéra, puis Wam, qui ont contribué à développer les travaux de l'équipe sur les documents structurés multimédias.

◆ Bibliographie

André J., Hudrisier H., éditeurs. *Document numérique*, numéro spécial « Unicode, écriture du monde ? », 2002, vol. 6, n° 3-4.

Bonhomme S. Notes de cours en Licence professionnelle, 2002, www.exselt.com/

Bortzmeyer S. Unicode : traiter toutes les écritures du monde, JRES 2003, p. 529-542.

Chartron G., Rebillard F. Modèles de publication sur le web, Rapport d'activités de l'Action Spécifique CNRS 103, RTP 33 : Document et contenu : création, indexation, navigation, 2004. http://rtp-doc.enssib.fr/archiveas.html

Cover Pages, Contents Listing for XML Applications and Industry Initiatives. http://xml.coverpages.org/xmlApplications.html.

Furuta R., Quint V., André J. Interactively Editing Structured Documents, Electronic Publishing. *Origination, Dissemination and Design*, avril 1988, vol. 1, n° 1, p. 19-44.

Inria. Communication scientifique multimédia. 2004. www.inria.fr/multimedia/

Lemlouma T., Layaïda N. Context-Aware Adaptation for Mobile Devices, IEEE International Conference on Mobile Data Management, Berkeley, 19-22 janvier 2004, p. 106-11.

Nicoll S., Pirot V., Bonaventure O. eConf: a pragmatic tool to produce and distribute multimedia e-learning content, SMIL Europe 2003 Conference, Paris, 12-14 février 2003.

The MPEG Home Page. www.chiariglione.org/mpeg

Pédauque R.T. Document : forme, signe et médium, les re-formulations du numérique. 8 juillet 2003. http://archivesic.ccsd.cnrs.fr/sic_00000511.html.

Pédauque R.T. Permanence et transformations, version du 15-06-2004. http://archivesic.ccsd.cnrs.fr/documents/archives0/00/00/10/03/index_fr.html.

Roisin C., Villard L. Transformation de documents dans les présentations multimédia, École thématique « Documents&Évolution » du GDR I3 : Le document multimédia en sciences du traitement de l'information, Marseille, 4-8 septembre 2000, p. 23-41.*CÉPADUÈS-Éditions*

Rowe L. A., Jain R. ACM SIGMM Retreat Report on Future Directions in Multimedia Research. 4 mars 2004. www.acm.org/sigmm/main/events/sigmm_retreat/sigmm-retreat03-final.pdf

Tran Thuong T., Roisin C., A Multimedia Model Based on Structured Media and Sub-elements for Complex Multimedia Authoring And Presentation, *Int'l Journal of Software Engineering and Knowledge Engineering*, Special Issue on Image and Video Coding and Indexing, 2002.

Weck D. LimSee2, A cross-platform SMIL2.0 authoring tool. 2004. http://wam.inrialpes.fr/software/limsee2/