CHAPITRE XX

Transformation de documents dans les présentations multimédia

Cécile Roisin[*,†] et Lionel Villard[*]

[*] Projet Opéra, INRIA Rhône-Alpes, http://www.inrialpes.fr/opera
[†] Université Pierre Mendès-France (Grenoble)
(Cecile.Roisin, Lionel.Villard)@inrialpes.fr

INTRODUCTION

Contexte

Le domaine des outils de traitement des documents a beaucoup évolué en quelques années pour passer du traitement d'informations purement textuelles (des suites de caractères et de paragraphes) à des informations structurées, hypermédia, dynamiques ou encore animées.

Deux mots-clés résument les deux axes (non-disjoints) dans lesquels l'activité sur les documents électroniques a pris récemment un essor très important : le *web* et le *multimédia*.

- L'essor formidable du web a entraîné (et parfois écrasé!) dans son sillage de nombreux domaines liés aux documents comme l'hypertexte, les documents structurés ou l'indexation et la recherche d'information. Les groupes de travail du consortium W3C constituent des lieux privilégiés où sont discutés, spécifiés et expérimentés les standards qui sont et seront les références du domaine. Ainsi en est-il du standard XML qui peut être considéré comme le standard pivot du web à partir duquel et autour duquel les autres standards de représentation des informations du web sont définis. Ces travaux constituent l'infrastructure de base pour la définition de nouvelles applications réparties qui s'appuient sur le concept de document pour présenter les données et interagir avec le (ou les) utilisateur(s).

- Le multimédia prend une importance de plus en plus grande dans notre société, notamment par le fait que la communication se fait de plus en plus en utilisant des supports qui intègrent du son, de la vidéo, du texte, etc. Les cédéroms, les sites web, les bornes interactives ou la télévision interactive ne sont plus maintenant considérés comme des technologies futuristes mais entrent dans le quotidien des entreprises et du grand public. L'émergence à grande échelle de ces applications multimédia implique d'apporter des solutions pour la création et pour la présentation de ces applications sur des terminaux aussi divers que des PC, des screenphones, des PDA (Portable Digital Assistant) ou même des téléphones cellulaires. La définition de formats standard est bien sûr au coeur de ce domaine puisqu'ils doivent répondre aux besoins de portabilité, d'échange, de pouvoir d'expression et de faisabilité. Les travaux actuels menés dans le cadre du consortium W3C sont essentiels pour l'essor de ce domaine (c.f. SMIL [19]).

Jusqu'à très récemment, les résultats des travaux de recherche sur les documents ont principalement suscité l'intérêt 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 standard comme SGML ou HyTime ont-elles émergé. Elles reposent sur une structuration et un typage fort des données documentaires. 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 à leurs clients grâce à la définition de DTD communes comme celles de l'ATA (*Air Transport Association*). Cependant, si des solutions à base d'hypermédia commencent à être employées pour permettre la mise en ligne de ces documents, ces secteurs ne sont pas encore prêts à introduire massivement le multimédia pour des raisons liées au mode de fonctionnement de ce marché qui s'appuie sur des standards et surtout au manque d'outils disponibles actuellement qui soient capables de traiter les importants volumes d'informations de ce domaine [4].

En parallèle à l'ingénierie documentaire classique, on peut constater l'émergence des techniques multimédia dans de nombreux secteurs d'activité: les jeux électroniques, le tourisme, l'enseignement assisté par ordinateur (EAO) sont actuellement le cadre des plus nombreuses applications multimédia car l'apport de ces techniques est (potentiellement) important. Ainsi, l'EAO peut tirer parti des caractéristiques de différents média pour réaliser des supports

pédagogiques qui soient d'une part plus attractifs grâce aux images, aux animations et au son, et qui soient d'autre part plus interactifs et adaptables aux élèves grâce aux fonctions de navigation hypermédia. D'autres domaines s'intéressent aux documents structurés multimédia et notamment la médecine. Les données issues des plateaux techniques d'imagerie médicale, comme les images par rayons X, par résonance magnétique ou par échographie, peuvent être exploitées sous forme numérique et intégrées à des données textuelles (par exemple les informations relatives au patient), et sonores (les commentaires du médecin), pour former de véritables documents multimédia médicaux qui peuvent être consultés à distance par les médecins.

Ainsi, de façon schématique, 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 hypermédia sont traités dans les applications multimédia le plus souvent sous forme de programmes spécifiques. De nouvelles approches sont donc nécessaires pour couvrir de façon plus globale les besoins de traitements de documents multimédia. L'objectif de cet article est de montrer que les techniques à base de transformation sont au cœur des systèmes de traitement documentaire.

Les besoins

Dans cet article nous allons uniquement considérer le problème de la conception et de la présentation d'informations multimédia à partir de sources (plus ou moins) structurées. Les aspects liés au problème de la recherche d'informations (modèles sémantiques pour l'indexation et l'accès par des requêtes sur des bases de documents) de même que ceux liés à la génération automatique ne seront pas développés ici.

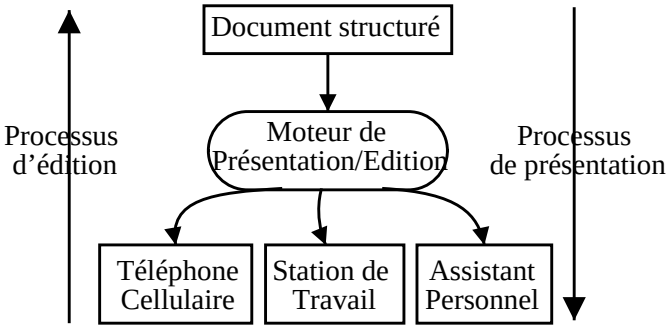

FIG. 1 – *Architecture de présentation multimédia*

Cependant, même dans ce cadre restreint, nous pouvons identifier trois classes de besoins (cf. figure 1) :

- Les besoins d'expression. L'apport du multimédia vient non seulement de la dynamique et de l'interactivité, mais aussi de l'augmentation de la qualité de l'information présentée. La variété des types de média utilisés apportent de nouvelles possibilités aux créateurs en leur permettant de jouer sur un plus grand nombre de modes de perception. Ainsi, il est nécessaire d'offrir les moyens de spécifier cette richesse que ce soit en termes de média de base que d'organisation spatiale et temporelle de ces média [14][13].
- Les besoins liés à la présentation. Outre les services nécessaires à la restitution des informations multimédia (accès aux média, synchronisation spatiale et temporelle, navigation temporelle, etc.), de nombreuses applications veulent pouvoir obtenir des présentations homogènes pour des documents de même classe (d'où l'intérêt des feuilles de styles attachées à une DTD). De façon inverse, la multiplicité des terminaux et des contextes de présentation rend nécessaire l'émergence de solutions qui permettent la production de plusieurs types de présentation à partir d'un même ensemble de sources d'information [12][2].
- Et les besoins liés à la conception. Les tâches de conception d'applications multimédia sont complexes du fait de la nature même des objets média et des différents modes de composition de ces objets (espace, temps, lien, etc.). Les auteurs doivent pouvoir disposer de réels moyens de perception de ces différentes caractéristiques ainsi que d'outils d'édition vraiment conviviaux. Enfin, un environnement

auteur doit permettre un processus "naturel" d'édition par une approche incrémentale, modulaire et par la réutilisation de parties de documents [6].

La définition de formats standard est bien sûr au cœur des solutions par leur capacité à répondre aux besoins de portabilité, d'échange, de pouvoir d'expression et de faisabilité.

Organisation du document

Dans la suite de cet article, nous allons décrire une architecture de présentation de documents multimédia qui combine les techniques issues des approches génériques avec les techniques de présentation multimédia (synchronisation, navigation, etc.). Cette approche permet en outre de répondre aux besoins d'adaptation au contexte de présentation.

Nous décrivons tout d'abord dans la section suivante un processus général de traitement de documents qui permet de produire une (ou plusieurs) présentation multimédia à partir de sources d'informations structurées. Ce processus s'appuie d'une part sur un *langage de présentation multimédia* qui contient les informations nécessaires au formatage du document et d'autre part sur un *langage de transformation* qui permet d'exprimer les règles pour passer d'un format source à un format de présentation. La description et l'illustration de ces deux langages font l'objet des deux sections principales de ce document.

PROCESSUS DE PRÉSENTATION

Dans notre contexte, le processus de présentation (c.f. figure 2) utilisé prend en entrée un document XML quelconque et produit un document multimédia. Le document source appartient à une classe de document, c'est-à-dire qu'il est valide par rapport à une DTD particulière. La présentation du document multimédia s'effectue en deux étapes :

- premièrement, l'étape de transformation permet de créer les objets multimédia à partir du document source. Elle permet aussi de les synchroniser (dimension temporelle), de les positionner (dimension spatiale) et de les lier (dimension hypermedia). Le résultat de cette transformation est une spécification du document multimédia décrite dans un langage de formatage (SMIL, Madeus) qui permet d'encoder les propriétes spatiales, temporelles et hypermédia du document.

- lors de la seconde étape, le document multimédia est formaté de telle façon que le résultat soit directement interprétable par le système de présentation. Cette étape peut faire appel à différentes techniques de formatage comme la résolution de contraintes, l'interprétation de règles d'héritage, etc. pour produire le document formaté.

FIG. 2- *Processus général de présentation*

Dans la suite nous présentons dans un premier temps un langage de formatage nommé *Madeus.* Nous verrons ensuite comment produire de tels documents un utilisant une mécanique de transformation.

LANGAGE DE PRESENTATION MULTIMEDIA

Dans notre contexte, nous utilisons un langage de formatage pour spécifier des documents multimédia. Leur description est organisée autour de quatre familles de propriétés : celles correspondant au « style », les propriétés spatiales, temporelles et hypermédia. Dans cette section, nous présentons ce langage pour chacune des familles et montrons comment les combiner.

La syntaxe utilisée pour la description de notre langage est XML. Nous donnons dans la suite seulement quelques fragments d'instance de document, la description complète de la DTD pouvant être trouvée ici [15]. La structure générale d'une instance est décomposée en quatre parties principales :

- l'élément *Content* défini le contenu brut de chaque média pouvant être utilisé dans le document ;
- l'élément *Media* « instancie » une description de contenu en y associant des informations de contexte d'utilisation comme le style ;
- l'élément *Temporal* décrit la synchronisation entre les media ;
- l'élément *Spatial* décrit l'organisation spatiale des média ;

Du fait de sa nature intrinsèque, l'information hypermédia est représentée dans les parties *Media* et *Temporal*. Un document Madeus est donc de la forme suivante :

```
<Madeus>
    <Content> ... </Content>
    <Media> ... </Media>
    <Temporal> ... </Temporal>
    <Spatial> ... </Spatial>
</Madeus>
```

Modèle du contenu

Une présentation multimédia est composée d'un ensemble de média (image, son, animation 3D, etc.) caractérisé par un contenu et un ensemble de propriétés. Dans le but de réutiliser un même contenu, celui-ci est spécifié séparément de son contexte d'utilisation. L'élément *Content* contient donc les informations de base d'un média, par exemple un ensemble de pixels pour une image, une chaîne de caractères pour un texte, etc. Il contient aussi les propriétés intrinsèques au média, telles que la durée d'une vidéo ou sa taille.

La description du contenu peut être de complexité variée et être encodée de plusieurs manières. Dans le cas de description spécifiée dans un format autre que XML, celle-ci est généralement contenu dans un fichier externe. Le lien vers ce fichier est réalisé en utilisant l'attribut *href* défini dans *Xlink* [17]. Dans le cas d'une description XML, celle-ci peut être inclue directement dans la partie *Content* grâce au mécanisme des *Namespace* [18] . Comme exemple de contenu, on peut citer par exemple *SVG* [5] qui permet de spécifier des dessins vectoriels, MathML qui permet de spécifier une expression mathématique, ou encore une description structurée de vidéos en scènes, plans, occurrences etc. [11].

La spécification du contenu s'effectue de façon hiérarchique. L'élément *C-Group* permet de regrouper un ensemble de contenus sous un même noeud. **Sa sémantique dépend du type de document**.

```
<Content>
  <C-Group>
    <C-Group ID="Text" MIMEType="text/plain">
      <DefContent ID="SlideTitle1">
        Introduction
      </DefContent>
      <DefContent ID="SlideTitle2">
        Multimedia systems
      </DefContent>
```

```
    </C-Group>
    <C-Group ID="Video" MIMEType="video/mpg">
      <DefContent ID="Film"
                xlink:href="http://.../Film.mpg">

        <Scene ID="Sc1" StartFrame="0"
               EndFrame="20">
        <!-- Slide 1 presentation -->
        <Shot ID="Sh1" StartFrame="0"
             EndFrame="5"/>
        <!-- Slide 2 presentation -->
        <Shot ID="Sh2" StartFrame="5"
             EndFrame="20"/>
      </Scene>
    </DefContent>
  </C-Group>
  <C-Group ID="Fond" MIMEType="image/svg-xml"
         xmlns:svg="…/SVG/SVG-19991203.dtd">
    <DefContent ID="Background">
      <svg:svg>
        <svg:rectangle fill="color:red"/>
      </svg:svg>
    </DefContent>
  </C-Group>
</Content>
```

Modèle des média

L'élément Média permet d'instancier la description d'un contenu en définissant ses attributs de style. L'instanciation peut se faire à différents niveaux de la description, par exemple pour présenter un plan d'une vidéo, ou une sous-partie d'un dessin vectoriel.

```
<Media>
 <M-Group>
  <DefMedia ID="M-Film" Content="Film"
          BorderWidth="1" BorderColor="black"/>
   <M-Group ID="TOC_entries" FontColor="black">
     <DefMedia ID="M-title1_toc"
            Content="SlideTitle1" FontSize="12"/>
     <DefMedia ID="M-title2_toc"
            Content="SlideTitle2" FontSize="12"/>
   </M-Group>
   <M-Group ID="Slide1" FontColor="blue">
     <DefMedia ID="M-title1"
            Content="SlideTitle1" FontSize="32"/>
     <M-Group ID="M-Body1"> ... </M-Group>
```

```
   </M-Group>
   <M-Group ID="M-Slide2"> ... </M-Group>
 </M-Group>
</Media>
```

Modèle temporel

La partie temporelle permet d'organiser les média à travers le temps. Cette organisation repose sur des travaux sur Madeus sur la définition d'un langage de synchronisation [7]. Le modèle sous-jacent à ce langage est un modèle à base d'intervalles. À chaque objet correspond un intervalle de temps caractérisé par les attributs *begin*, *duration* et *end*. Chacun de ces attributs est défini selon un intervalle de valeurs [min, pref, max] (de zéro jusqu'à l'infini).

Pour chaque média déclaré dans la partie *Media* est associé un intervalle de temps. Les attributs portés par l'intervalle surchargent les attributs temporels intrinsèques au média. Si la durée de l'intervalle est plus grande que la durée intrinsèque, l'attribut *Fill* est ajouté afin de préciser le comportement désiré : *Fill="repeat"* (le média est rejoué dans l'intervalle), *Fill="Freeze"* (la dernière image est affichée jusqu'à la fin de l'intervalle), *Fill="cut"* (le média devient invisible). Certaines parties du scénario peuvent devenir actives seulement quand une interaction extérieure est effectuée (par exemple lorsque l'utilisateur active un hyperlien, c.f section 2.5). De tels liens sont définis grâce à l'attribut *xlink:href* attaché à l'intervalle source.

Par exemple, l'association de l'intervalle T-Slide1_toc avec le média M-Slide1_toc se spécifie comme suit :

```
<Interval ID="T-Slide1_toc" Media="M-Slide1_toc"
    Duration="min:50s pref:60s max:300s"
    Fill="freeze" xlink:href="@T-Slide1.begin"/>
```

Dans le langage Madeus, la synchronisation est spécifiée à la fois par regroupement de d'intervalles au sein de nœuds composites (eux-mêmes des intervalles) et par le placement de relations temporelles entre intervalles.

Un nœud composite (l'élément *T-Group*) permet de regrouper temporellement des intervalles. Une contrainte *during* est placée entre chacun des fils et leur nœud père.

Cette synchronisation de base peut être raffinée en plaçant des relations temporelles entre les descendants d'un nœud composite. Par exemple,

spécifier que deux transparents se jouent en séquence peut se faire en plaçant une relation *meets* entre eux.

La désignation d'un intervalle dans une relation peut s'effectuer soit en utilisant l'identifiant de l'intervalle (l'attribut ID), soit en utilisant une désignation relative au nœud composite : *prev*, *all* et *next*. Ce dernier type de désignation est nécessaire lorsque l'identifiant est *a priori* inconnu, notamment durant la transformation d'un document (c.f. Section 3). De plus, la désignation relative permet, dans certains cas, de simplifier la spécification, par exemple lorsqu'une relation temporelle s'applique à tous les fils d'un nœud. Par exemple, la spécification d'une séquence de transparents peut s'effectuer comme suit :

```
<T-Group>
  <Interval ID="T-Slide1"
    Duration="min=15 pref=20s max=25s"/>
  <Interval ID="T-Slide2"
    Duration="min=15s pref=20s max=25s"/>
  <T-Relations>
    <T-Relation Name="Meets" Intervals="all"/>
  </T-Relations>
</T-Group>
```

Modèle spatial

La partie spatiale d'un document Madeus est similaire à la partie temporelle. Les deux principales différences sont d'une part l'utilisation d'un vocabulaire de relations spatiales (left_align, bottom_spacing, region, etc.) et d'autre part l'extension aux deux dimensions spatiales. De plus, un attribut spatial ne peut pas avoir une valeur infinie.

Plus précisément, la partie spatiale organise l'espace en une hiérarchie de boîtes 2D. Un nœud composite (l'élément *S-Group*) permet de grouper un ensemble de régions 2D (l'élément *Shape*) à l'intérieur d'une boîte 2D.

Modèle hypermédia

L'objectif est de décrire les liens entre les éléments. La base de ce modèle est le standard Xlink [17] que nous avons étendu pour prendre en compte les dimensions temporelle et spatiale. Les propriétés des liens sont les suivantes :

Comportement lié à l'affichage : quand l'élément source est activé, l'élément cible peut soit être affiché dans une nouvelle fenêtre, soit

remplacer le document source, ou soit être inclu **à la suide de** l'élément source ;

Localisation de la cible : cet attribut indique la localisation de l'élément cible. Jusqu'à maintenant, sa valeur est une URI. Mais dans le contexte multimédia, cette valeur peut être étendue afin de désigner un positionnement temporel et spatial. La localisation temporelle peut être absolue (une date) ou relative par rapport au début d'une partie de la présentation du document. De même, la localisation spatiale peut être absolue ou relative par rapport à la position d'un élément.

Type d'activation : cette propriété définit comment le lien doit être activé. Elle est spécifiée à travers les attributs suivants : *href_dur* (la période de temps pendant laquelle le lien peut être activé), *href_number* (le nombre d'activations autorisé) et *href_activation* (le lien peut être activé automatiquement, où interactivement selon plusieurs sources d'interaction).

LANGAGE DE TRANSFORMATION

Dans cette partie nous présentons un langage de transformation à base de règles [16][3][17]. La syntaxe utilisée dans nos exemples est celle définie dans le standard XSL-T.

Dans ce langage, une règle comporte deux parties : la partie déclaration (la signature) et la partie exécution (le corps). La partie déclaration permet de spécifier les nœuds source pour lesquels la règle doit être appliquée (ou instanciée). L'application de la règle pour un nœud source donné consiste en l'exécution de sa partie corps.

Nous présentons dans la suite un exemple de classe de document qui va nous servir pour illustrer les concepts de la transformation pour la génération de présentations multimédia. Nous décrivons ensuite les caractéristiques des deux parties de règles.

Exemple

Le document XML suivant représente une présentation de type *slideshow*. Il contient une partie *header* permettant spécifier le titre de la présentation, les auteurs et la date de présentation. La partie *slides* permet de spécifier une liste ordonnée d'éléments *slide*. Un élément *slide* est caractérisé par un titre (élément *title*) et un corps (élément *body*).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<slideshow>
  <header>
    <title xml:lang="fr">
    Intégration de la video structurée dans un
outil d'édition multimédia
    </title>
    <authors>
      <author>Cécile Roisin</author>
      <author>Tien Tran_Thuong</author>
      <author>Lionel Villard</author>
    </authors>
    <date day="17" month="9" year="1999"/>
  </header>

  <slides>
    <slide>
      <title xml:lang="fr">Plan</title>
      <body>
        <!-- Corps du premier transparent -->
      </body>
    </slide>


    <slide>
      <title xml:lang="fr">Exemple</title>
      <body>
        <!-- Corps du second transparent -->
      </body>
    </slide>
  </slides>
</slideshow>
```

Déclaration d'une règle

La déclaration d'une règle permet de spécifier un fragment de document résultat de la transformation pour un ensemble de nœuds source. La sélection des nœuds source s'effectue en utilisant un langage de requête tel qu'une expression régulière, ou une expression de type XPATH [16]. Le langage XPATH permet de sélectionner des nœuds en fonction de leur type, de leur contexte, de la valeur de leurs attributs, etc. Voici quelques exemples de requêtes :

child::para sélectionne tous les fils du noeud courant qui sont des éléments de type *para*

child::para[position()=1] sélectionne le premier fils du noeud courant qui est un élément de type *para*

/child::doc/child::chapter[position()=5]/child::section[position()=2] sélectionne la seconde *section* du cinquième *chapter* de l'élément *doc*

Le corps de la règle est constitué d'éléments de transformation intercalés avec d'autres types d'éléments. Les éléments de transformation orientent le processus de transformation, tandis que les autres éléments sont les briques pour le document cible final. Dans notre contexte, ces autres éléments appartiennent au langage de présentation multimédia défini dans la section 2. Dans la suite nous parlerons d'éléments de formatage.

L'exemple suivant contient deux règles de transformation spécifiées grâce à l'élément *xsl:template*. La première règle sera appliquée pour tous les éléments de type *title* dont l'élément père est de type *header*, tandis que la seconde règle sera appliquée pour tous les éléments de type *slide*.

```
<xsl:template match="header/title">
  <!-- Partie Exécution -->
  <Text ID="M-ShowTitle" FontFamily="Times"
        FontSize="12">
    <xsl:value-of select="."/></Text>
  </Text>
</xsl:template>

<xsl:template match="slide">
  <!-- Partie Exécution -->
  <Text FontSize="32">
    <xsl:attribute name="ID">
      M-SlideTitle-<xsl:number level="any"
                               count="slide"/>
    </xsl:attribute>
    <xsl:value-of select="title"/>
  </Text>
</xsl:template>
```

Exécution d'une règle

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.

L'exécution d'une règle commence par l'instanciation de celle-ci. A l'état initial, le noeud courant est le nœud racine. Une règle est recherchée

et si elle existe, alors le corps de la règle est exécuté. Comme nous l'avons précédemment dit, deux classes d'éléments constituent la partie exécution : les instructions (ou éléments) du langage de transformation et les éléments de formatage.

Les éléments de formatage sont directement générés dans l'arbre cible par rapport à son contexte courant.

Les éléments de transformations ont une sémantique différente selon leur type. Voici quelques éléments XSL-T 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 spécifié 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. (c.f. figure 3) ;

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 ;

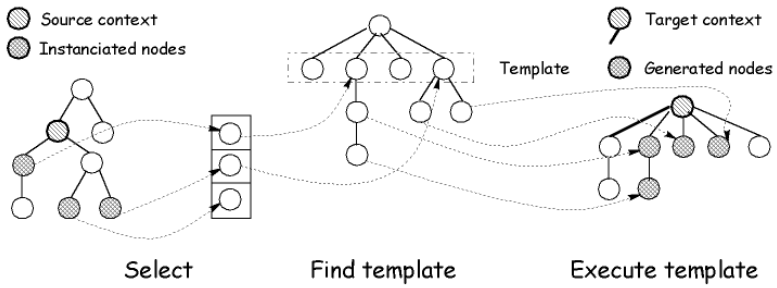

FIG. 3 - *Processus d'instanciation de règles de transformation*

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 modulariser

les feuilles de transformation. Par exemple, il est possible de séparer les différentes dimensions caractérisant un document multimédia. De plus, les règles s'appliquant à la DTD du document sont séparées de celles spécifiques à l'instance.

Application à la génération de présentations multimédia

Cette architecture est en cours d'expérimentation à travers le prototype appelé Madeus. Ce prototype, implanté en Java, est développé en utilisant la boîte à outils multimédia Kaomi [7] (que nous développons aussi). La transformation du document XML est effectué grâce à Xalan, une implémentation java d'un processeur xslt [1]. Ce prototype permet de répondre aux besoins énoncés dans la section 1.2. L'utilisation de feuilles de transformation pour spécifier la présentation du document XML permet à la fois d'homogénéïser les présentations d'une même classe de document et aussi de créer plusieurs présentations adaptées au contexte du lecteur.

Nous avons défini deux étapes dans le processus de production d'une présentation multimédia (c.f. figure 4) :

- l'étape de transformation grâce à l'utilisation du langage de transformation décrit précédemment. Cette étape offre un pouvoir d'expression maximal mais est complexe dans la génération du résultat ;
- l'étape de décoration grâce à l'utilisation de feuilles de décoration qui offrent la possibilité d'ajouter des éléments de style à une structure de document multimédia déjà existante. Cette étape permet d'ajuster une présentation à un moindre coût.

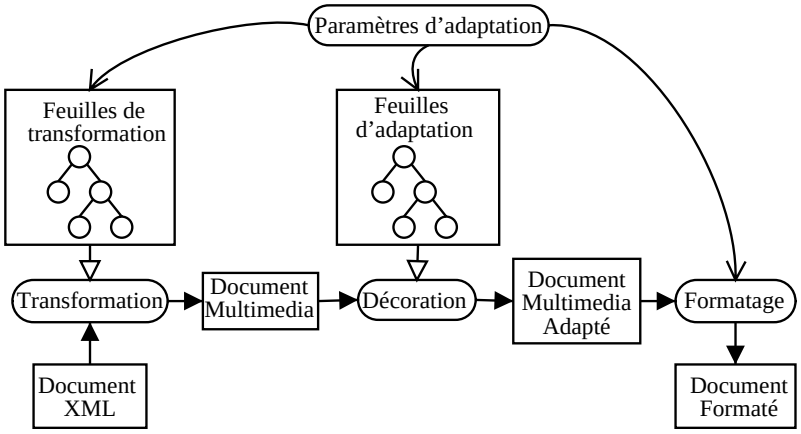

FIG. 4 – *Les deux étapes du processus de présentation dans Madeus*

Cette architecture a été expérimentée avec le document de type slideshow. La figure 5 représente la hiérarchie des feuilles de transformation définies. Trois présentations ont été produites : une présentation de type diaporama (show.xsl), une présentation de la table des matières (TOC.xsl) et une présentation de la table des matières adaptée à un assistant personnel (PDA_TOC.xsl).

CONCLUSION ET PERSPECTIVES

Dans cet article une architecture de présentation de documents multimédia a été présentée. Elle vise à couvrir les besoins de présentation multimédia homogènes d'informations structurées en XML et à en faciliter l'adaptation en fonction des conditions de restitution. Le système repose d'une part sur un langage de présentation couvrant les besoins d'expression et d'autre part sur l'utilisation de techniques de transformation permettant la génération des présentations à partir de l'expression modulaire des informations de présentation correspondant aux différents types de composition (spatial, temporel). Les fonctions de restitution ne sont pas décrites ici (voir [8]) mais il est intéressant de noter que le langage de présentation Madeus maintient l'information de présentation sous une forme riche (relations), ce qui permet aux processus de formatage et d'ordonnancement de mettre en œuvre des stratégies

adaptables de placement spatial et temporel ainsi que la gestion des désynchronisations et de la qualité de service [9].

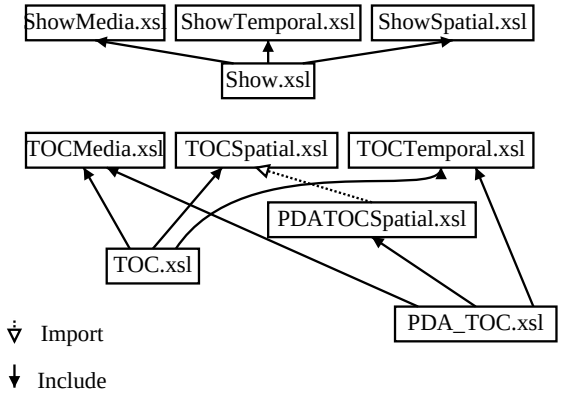

FIG.5 - *Feuilles de transformation pour une présentation slideshow*

Les axes de travail actuellement en cours de développement concernent d'une part la validation de cette architecture à une plus grande échelle et d'autre part l'étude des conséquences d'une telle approche pour la phase de conception : comment offrir un environnement convivial pour l'édition de documents résultats d'un ensemble d'informations structurées transformées par un ensemble de règles de style. En effet, lors d'une opération d'édition, l'auteur doit percevoir (et pouvoir décider) s'il agit sur les informations source ou sur une feuille de style. Dans une approche de feuilles de style modulaires, il doit en plus pouvoir sélectionner la feuille à éditer. Une nouvelle ère de conception d'outils auteur s'ouvre donc.

REFERENCES

[1] Apache XML Project, *Xalan*, http://xml.apache.org/xalan/index.html, 2000.

[2] Boll S., Wolfgang K., Wanden J., *A cross-Media Adaptation Strategy for Multimedia Presentation*, Proceedings of the ACM Multimedia'99, October 30 – November 5, 1999, Orlando, Florida, USA, 1999.

[3] Clark J., *XSL Transformation*, http://www.w3.org/TR/xslt, 1999.

[4] Duluc F., *Multimedia and aeronautical technical documentation : new challenge..., and new issues*, Markup Technologies'98 Conference proceedings, Chicago(IL), USA, pp 135-143, novembre 1998.

[5] Ferraiolo J., *Scalable Vector Graphics (SVG) 1.0 specification*, http://www.w3.org/TR/SVG, 2000.

[6] GriNS, http://WWW.cwi.nl/GriNS/.

[7] Jourdan M., Roisin C., Tardif L., *A Scalable Toolkit for Designing Multimedia Authoring Environments*, Special Number, "Multimedia Authoring and Presentation: Strategies, Tools, and Experiences" of Multimedia Tools and Application Journal, Kluwer Academics Publishers, 1999.

[8] N. Layaïda, C. Roisin, L. Sabry-Ismaïl, Support d'exécution de documents multimédia, vol. Chapitre 8, Hermès Science Publications, Paris, à paraître 2000.

[9] N. Layaïda, L. Sabry-Ismail, C. Roisin, *Dealing with uncertain durations in synchronized multimedia presentations*, Multimedia Tools and Applications Journal, Kluwer Academic Publishers, à paraître, 2000.

[10] Omnimark, *Guide to OmniMark 5*, http://www.omnimark.com/develop /om5/doc/, 2000.

[11] Roisin C., Tran_Thuong T., Villard L., *Integration of structured video in a multimedia authoring system*, Proc. of Eurographics Multimedia'99 Workshop, Springer Computer Science, pp.133-142, Septembre 1999.

[12] Rutledge L., Hardman l., J. v. Ossenbruggen, D. C.A. Bulterman, *Mix'n'Match : Exchangeable Modules of Hypermedia Style*, Proceedings of the ACM Hypertext '99, 1999.

[13] Junehwa Song, G. Ramalingam, and Raymond E. Miller, *Modeling Timed User interactions in Multimedia Documents*, IEEE International Conference on Multimedia Computing and Systems, Hiroshima, pp.407-416, June 1996.

[14] Michalis Vazirgiannis, *Interactive Multimedia Documents*, Lecture Notes in Computer Science n°1564, Springer, 1999.

[15] Villard L., *DTD Madeus*, http://www.inrialpes.fr/opera/ madeusmodel.dtd, 2000

[16] W3C, *Path Language (XPATH) 1.0*, http://www.w3.org/TR/xpath, 1999.

[17] W3C, *XML Linking Language (XLink)*, http://www.w3.org/TR/xlink, 2000.

[18] W3C, *Namespaces in XML*, http://www.w3.org/TR/REC-xml-names, 1999.

[19] W3C, *Synchronized Multimedia Integration Language (SMIL) Boston Specification Working Draft*, http://www.w3.org/TR/smil-boston/, 2000.