Adaptabilité : pistes d'étude pour la définition d'une infrastructure d'accès au contenu multimédia pour des machines hétérogènes[1]

Nabil Layaïda

Projet Opéra, INRIA Rhône-Alpes

[1] Dans la suite de ce rapport, le terme machines ou périphériques hétérogènes désigne les machines ayant des ressources, une utilisation (mobile ou non) et des capacités très variables.

1 Introduction

Pour définir une stratégie d'accès aux services multimédia au moyen de terminaux hétérogènes, il est nécessaire de prendre une décision sur la politique à adopter pour y parvenir. Faut-il développer des applications et du contenu séparés pour ce type de machines ? ou est-il possible de réutiliser des applications et du contenu déjà déployés sur le web? Cette question n'est pas encore résolue pour les documents web classiques, elle l'est encore moins pour les documents multimédia, c'est à dire intégrant texte, image, audio et vidéo.

2 Objectif du rapport

Ce rapport est destiné à mettre en évidence les tendances dans le domaine de l'adaptabilité de l'information disponible sur Internet. Par adaptabilité, on entend les moyens automatiques ou semi-automatiques qui permettent aux contenus circulant sur le web (documents html par exemple) d'être utilisables sur des terminaux ayant des caractéristiques et des ressources très variées. Nous reviendrons sur ce point plus tard dans le document pour évoquer de façon plus précise l'impact de cette variabilité de ressources sur la nature de l'adaptation à opérer.

En premier lieu, nous allons d'abord présenter les trois types d'approches qui sont possibles pour aborder le problèmes de l'adaptabilité :

- **Les méthodes automatiques :** ce sont les méthodes qui s'appliquent sur les données du web telles qu'elles sont disponibles aujourd'hui sur les réseaux. C'est, par exemple, un ensemble de pages HTML qui, à l'origine, ont été élaborées pour être affichées sur des terminaux de type PC et doivent maintenant être adaptées à des machines hétérogènes. Un traitement fondé sur XSLT fait partie des méthodes automatiques.
- **Les méthodes semi-automatiques :** ce sont des méthodes qui s'appliquent sur des données plus riches et plus structurées déjà préparées pour un processus de transformation (eg. CSS). C'est à la suite de cette transformation que les documents peuvent êtres présentés sur des terminaux de types différents. Une étape préliminaire consiste donc à structurer les données avant de les transformer.
- **Les méthodes manuelles :** ce sont les méthodes qui permettent de définir des langages spécifiques pour chaque type de périphérique ou pour des classes de périphériques. Les documents doivent être écrits dans chacun de ces langages pour être exploitables sur l'ensemble des périphériques. Dans ce cas, il n'y a pas d'opération d'adaptation au sens de processus automatique de transformation. Une méthode manuelle peut être réalisée par exemple au moyen de l'extension d'un éditeur pour la saisie de l'ensemble des versions du document qui doivent être produites.

Il faut noter que les transformations traitées dans ce document ont une porté plus ou moins grande en fonction des périphériques visés. Par exemple, afficher le contenu d'un document MHML (édité et présenté à l'origine pour un écran de PC) sur l'écran d'un téléphone cellulaire nécessite un degré de transformation nettement plus important que pour un écran haute résolution, même de plus petite taille.

Etant donné l'état de l'art très peu développé sur ces questions, nous donnons la priorité dans ce rapport à trois aspects qui représentent à nos yeux les points clés de la mise en œuvre d'une solution satisfaisante et extensible :

- L'architecture globale du système d'information multimédia adaptable.

- Les transformations qui doivent être intégrées à ce système pour qu'il soit adaptable aux différents périphériques.
- L'identification et l'intégration des standards et les protocoles du web au sein du système considéré.

Ce dernier point mérite une attention particulière. Nous pensons qu'il est, en effet, important de se positionner dès le début par rapport aux standards du web et à leurs évolutions. Compte tenu des évolutions rapides dans ce domaine, un système d'information multimédia adaptable sur internet ne peut être déployé avec succès que s'il intègre ses innombrables évolutions. Notons au passage que des standards comme XML, XSL, SMIL Boston et RDF, pour ne citer que ceux-là, concernent directement ce travail.

3 Caractéristiques des machines hétérogènes

Une des opérations importantes qui intervient dans le processus d'adaptation est la description des contraintes et les capacités des périphériques hétérogènes. Les capacités visées sont d'ordre matériel mais aussi logiciel. Dans le tableau ci-dessous, nous reprenons les caractéristiques fournies par le rapport UIR/C/99/0053 pour plusieurs types de machines.

Il faut noter que du point de vue de l'utilisateur, même dans le cas d'un accès à l'information depuis un PC, il pourrait avoir un certain nombre de préférences pour la visualisation de documents multimédia. Par exemple, il peut être amené à vouloir une fenêtre d'affichage plus réduite pour accéder aux documents multimédia. Dans ce cas, on parle de préférences étant donné que la contrainte ne vient pas des possibilités de la machine mais des souhaits de l'utilisateur. La description des caractéristiques des machines est ainsi étendue pour décrire les préférences.

	ScreenPhone	GSM	PC	Portable PC	PDA
Throughput of the connection	modem: 56 Kbps	9600 bps	modem: >28,8 Kbps	9600 bps up to 2 Mbps (UMTS)	>9600 bps
Screen					
Display	640x480	~90x53	800x600 up to 1600x1200		160x160 up to 240x320
Color	8 b	no	8, 16, 24, 32 b		4 grey scale
Black&white	yes	yes	yes		yes
Video	yes	no	yes	yes	
Image	.GIF and .JPEG	.BMP			
Audio	.AU and .WAV	no			
Supporting protocol	HTTP	HDTP	HTTP		
Supporting languages	HTML 3.2 P.JAVA applets	HDML	HTML JAVA	HTML JAVA	

RAM	16 Mb				

Dans la section suivante, nous étudions comment ces capacités peuvent être décrites au moyen du langage RDF développé au sein du consortium W3.

4 Description des ressources : la piste RDF

Dans cette section, nous décrivons la première étape dans tout processus d'adaptation : un service de profile qui permet de décrire les capacités et les préférences des applications multimédia ouvertes au web. Nous regroupons sous le terme (CC/PP) Composite Capability/Preference Profile une collection regroupant les capacités et les préférences associées à un utilisateur ainsi que l'ensemble des composants qui interviennent dans le processus permettant de visualiser du contenu multimédia sur le web. Ces composants incluent une description de la plateforme matérielle ainsi que les modules d'application utilisés. Les capacités d'un composant peuvent être perçus comme un ensemble de méta-données ou de propriétés et descriptions sur le matériel ainsi que le logiciel utilisé.

Une description des capacités et des préférences est nécessaire mais insuffisante afin de fournir une solution générale pour la négociation de contenu ou l'adaptation. Un schéma général pour la négociation de contenu nécessite les moyens permettant de décrire les méta-données ou attributs et préférences de l'utilisateur ainsi que de ses périphériques, les attributs liés au contenu ainsi que les règles pour l'adaptation du contenu pour les capacités des périphériques ainsi qu'aux préférences de l'utilisateur. Les mécanismes existants au sein des protocoles comme les entêtes accept de HTTP et les balises <alt> sont très limités. Les services offerts dans le futur doivent être beaucoup plus complets. Par exemple, le contenu peut être saisi en plusieurs langues et avec différents niveaux de lecture et de détails. Pour obtenir un schéma de négociation adapté, un ensemble de règles pour la sélection d'une version donnée du contenu est nécessaire.

La solution qui nous semble la plus prometteuse est celle qui s'appuie sur un schéma XML/RDF. RDF a été conçu pour spécifier des propriétés exploitables par des programmes pour décrire le contenu du web. L'approche consiste à utiliser RDF pour décrire les capacités des composants qui interviennent dans l'accès au contenu ainsi que les préférences associées à un utilisateur et celles qu'il souhaiterait appliquer sur les périphériques et le logiciel utilisé. Nous pensons que l'utilisation d'une technologie standardisée à base de RDF pour l'encodage de ce type de méta-données va non seulement simplifier leur utilisation dans le web mais aussi leur adoption par l'ensemble de la communauté des développeurs d'applications. Des outils puissants pour la manipulation de XML et de RDF, comme la localisation de logicielles emballés (*packages RPM*), sont de plus en plus disponibles.

Nous ne décrivons pas encore le processus de négociation entre le serveur de contenu et l'application client qui peut être relativement complexe. Nous espérons simplement à ce stade que RDF peut faciliter le développement de ce type de négociations, l'implémentation de règles de négociation adaptées n'est pas considérée dans la norme mais est laissée aux développeurs d'applications.

Par ailleurs, des méthodes alternatives pour la description des attributs ou méta-données sont à l'étude au sein de l'IETF (CONNEG Content Négociation Working Group, voir www.ietf.org).

La solution à base de RDF, bien que non compatible avec celle de CONNEG, permet d'utiliser plusieurs vocabulaires. Cela permet une inter-opérabilité au moins au niveau des noms d'attributs (identification des propriétés). The Group CONNEG est en train de développer une algèbre (media feature matching algebra) pour la sélection du contenu qui est complémentaire à ce que RDF peut fournir. Il est donc indispensable d'étudier comment cela s'intègre dans un schéma général de négociation de contenu. Les forums WAP Forum et ETSI sont aussi actifs dans ce domaine et un état de leurs études doit être établi.

Nous donnons ici, à titre d'exemple une description RDF d'une machine cliente ainsi que ses caractéristiques logicielles.

```
<?xml version="1.0"?>
<RDF
xmlns:RDF="http://www.w3C.org/TR/WD-rdf-syntax#"
xmlns:PRF="http://www.w3C.org/TR/WD-profile-vocabulary#">
<RDF:Bag>
   <RDF:Description about="HardwarePlatform">
     <PRF:Defaults>
      <Description
        PRF:Vendor="Alcatel"
        PRF:Model="2160"
        PRF:Type="PDA"
        PRF:ScreenSize="800x600x24"
        PRF:CPU="PPC"
        PRF:Keyboard="Yes"
        PRF:Memory="16mB"
        PRF:Bluetooth="YES"
        PRF:Speaker="Yes" />
     </PRF:Defaults>
     <PRF:Modifications>
      <Description
       PRF:Memory="32mB" />
    <PRF:Modifications>
    </RDF:Description>
  <RDF:Description about="SoftwarePlatform">
    <PRF:Defaults>
     <Description
      PRF:OS="EPOC1.0"
      PRF:HTMLVersion="4.0"
      PRF:JavaScriptVersion="4.0"
      PRF:WAPVersion="1.0"
      PRF:WMLScript="1.0" />
```

```
      </PRF:Defaults>
      <PRF:Modifications>
       <Description
        PRF:Sound="Off"
        PRF:Images="Off" />
      </PRF:Modifications>
    </RDF:Description>
</RDF:Bag>
</RDF>
```

5 Documents multimédia : au delà du contenu, la structure

Depuis de nombreuses années, les documents électroniques ont fait l'objet d'études qui ont conduit à l'identification de caractéristiques attachées aux documents et classées selon différentes dimensions [AFQ 1989], [Stern 1997], [ISO 1994]. Le résultat majeur de ces travaux est la définition de standards comme XML (Extended Markup Language) [W3C XML 1998] qui permettent de représenter la dimension logique des documents indépendamment de leur contenu et de leur dimension spatiale (forme), cette dernière faisant elle-même l'objet de standards comme DSSSL, CSS ou XSL. Cette décomposition de l'information portée par les documents a pour premier objectif de faciliter la portabilité des documents ainsi que leur traitement par des applications variées. Sur la base de cette approche, d'autres caractéristiques sont regroupées pour former de nouvelles dimensions : la dimension hypertexte, correspondant à l'ensemble des informations permettant de lier des documents (ou des fragments de documents) entre eux, et la dimension temporelle qui identifie le comportement dans le temps des documents (comme la durée des objets ou leur synchronisation).

Pour chacune de ces dimensions, le travail de modélisation consiste à identifier d'une part les entités de base, comme les éléments textuels (#PCDADTA dans la syntaxe XML), les boîtes pour le placement spatial, les intervalles temporels pour le temps, et d'autre part leur mode de composition qui doit permettre de couvrir des besoins d'expression comme :

- La composition logique : « un livre est constitué d'un titre, d'un auteur et d'une suite de chapitres, chacun d'eux étant formé d'un titre et d'une suite de paragraphes ».
- La composition spatiale, appelée également composition graphique : « le titre occupe une largeur de 80% de celle de la page et est centré ; chaque note doit être placée en bas de la page dans laquelle se trouve la première référence à cette note ».
- La composition temporelle :« la présentation de l'entreprise débute par l'affichage de son logo pendant 5 secondes, puis la photo du directeur est affichée en même temps que son discours est joué ; la fin de la présentation est composée d'une vidéo de 3 minutes ».
- La composition hypertexte : « un lien est placé entre chaque entrée bibliographique et chacune des références à cette entrée ».

La diversité des besoins en matière de représentation de documents a conduit à une approche modulaire de ces spécifications et à la notion de profils (sous-ensemble de modules de

spécifications pour un domaine applicatif donné). C'est une approche en cours de discussion dans le groupe de travail « XML Syntax » du W3C.

Notons également qu'un des objectifs des travaux de modélisation des documents est de faciliter la réutilisation, d'où la notion de modèles génériques que l'on retrouve pour les dimensions logique et spatiale (et dans une plus faible mesure pour l'organisation hypertexte).

5.1 XML

De nombreux secteurs professionnels définissent des DTD qui sont à la base des échanges de documents entre les acteurs de ces domaines : CALS pour la documentation technique des armements, ATA pour la documentation technique aéronautique, TEI (Text Encoding Initiative [TEI 1998]) pour les documents littéraires, ISO 12083 et MathML [W3C MathML 1998] pour les documents scientifiques. Des travaux sont en cours pour la définition de DTD pour les graphiques 2D [W3C SVG 1998].

Les avantages de l'utilisation de tels modèles ont été identifiés depuis longtemps [FQA 1988], mais, malgré la définition il y a plus de 10 ans de standards comme SGML [ISO 1986] ou ODA qui appliquent ces principes, les formats de représentation de documents utilisés par la majorité des outils actuels sont des formats propriétaires avec peu de structuration de l'information. Cette situation peut s'expliquer par l'état du marché et par le peu d'outils simples et peu coûteux permettant de traiter des documents structurés. La rigidité et la complexité induites par certains choix de base de SGML ont été également des facteurs de frein au déploiement d'outils adaptés à cette norme.

Le besoin d'échanger à travers le web des informations sémantiquement plus riches que ne permet de le faire HTML a conduit les acteurs concernés au sein du W3C à la définition d'un nouveau standard issu de SGML : XML [W3C XML 1998]. Pour répondre aux besoins spécifiques du web, XML ainsi que les standards associés offrent :

- des simplifications syntaxiques (par rapport à SGML),
- la définition de structures de lien et d'adressage plus riches (XLink [W3C XLink 1998] et XPointer [W3C XPointer 1998]),
- des possibilités de modularité grâce à la spécification d'espaces de noms [W3C Namespaces 1999],
- des possibilités d'association de contraintes syntaxiques ou sémantiques (XML Schéma [W3C XML Schema 1999]),
- la possibilité de traiter un document XML même si l'on ne dispose pas de la DTD correspondante (notion de document bien-formé).

Il est clair que nous sommes à une étape charnière dans ce domaine car la dynamique générée autour de XML concerne non seulement les navigateurs web (Netscape, Microsoft) ou les outils SGML (Softquad, Arbortext, Omnimark), mais aussi la bureautique (Microsoft, Lotus), les outils graphiques, les SGBD (Oracle, SAP) et les outils de développement (SUN, IBM).

Les prochaines années vont consacrer le passage du numérique au structuré. On le voit très bien actuellement au travers du phénomène XML, et cela ne touche pas seulement le texte, mais aussi

le multimédia (SMIL) et les données (avec leurs programmes) en général qui transitent sur le web (on parle déjà de RPC basé sur XML).

Pour faire court, on peut dire que XML, DOM (Document Object Model) et les langages de programmation tournés vers le réseau seront les briques de base des applications ouvertes sur Internet. On peut ainsi espérer passer d'une gestion de documents à une gestion massive de documents et de données structurés.

Les acquis de ce domaine, bien que très récents (sauf les expériences issues de SGML), sont nombreux et la tendance s'accélère avec le déploiement généralisé des documents et des données structurés sur le web. Après ce déploiement, on mesurera l'impact de cette évolution sur les systèmes d'information en général.

5.2 HTML

HTML a évolué de façon très importante ces dernières années. De nouvelles fonctionnalités ont été ajoutées alors que d'autres ont été progressivement abandonnées. Cette approche de l'évolution de HTML ne peut plus être utilisée dans l'avenir car les utilisateurs qu'on rencontre aujourd'hui sur le web ont des besoins très différents. Une communauté d'utilisateurs souhaite obtenir un sous-ensemble de HTML compact et bien défini qui peut être implanté facilement surtout sur des périphériques ayant de faibles ressources. D'autres utilisateurs voudraient, au contraire, ajouter de nouvelles fonctionnalités pour supporter les notations mathématiques ainsi que les dessins vectoriel. Le langage XML a été utilisé afin de répondre à ce problème. L'objectif recherché est de définir HTML comme un ensemble de modules d'application de XML.

Un problème particulier avec HTML est l'utilisation du langage de marquage pour définir des tables et donc de la présentation. Cette approche mélange la structure du document avec le formatage spatial et rend la restitution sur des périphériques mobiles très difficile.

5.3 WML

Wireless Markup Language (WML) est un langage de marquage restreint qui est défini à partir d'une DTD XML. En comparaison avec HTML, WML est plus optimisé pour l'utilisation sur des terminaux d'appareils mobiles en partie parce qu'il prend en compte les contraintes de ressources comme la faible bande passante, la faible puissance de calcul ainsi que l'espace d'affichage et de moyen d'interaction limités. En termes plus fonctionnels WML est conçu de la façon suivante :

- Un document est organisé sous forme d'un ensemble de cards et decks. Les cards spécifient les unités d'interaction dans un document (comme les entrées texte ou menu de choix). Un deck consiste en un ensemble de cards et représente l'unité de transmission de contenu et est identifié par une URL.
- Support de texte et d'images. WML inclut une variété de commandes de formatage et de positionnement géométrique. Par exemple, des fontes en gras peuvent être spécifiées.
- Navigation inter-cards et hyperliens. WML contient un support pour la gestion explicite de la navigation entre les cards et decks. Il fournit aussi un support pour les événements dansles machines hétérogènes qui peuvent être exploités pour la navigation et pour l'exécution de scripts. WML support aussi les liens à la HTML 4.0 .
- Paramétrage de chaînes de caractères et gestion d'états. Tous les decks peuvent être paramétrés en utilisant un modèle d'états finis. Des variables peuvent être employées à la

place de chaînes de caractères et sont substitués à l'exécution. Cette paramétrisation permet une utilisation plus efficace des ressources réseau (diminution de la bande passante). L'état d'un document WML représente ce qui est appelé un contexte. Un contexte est utilisé pour gérer ces variables, l'historique de navigation ainsi que d'autres types de paramètres laissés à l'implémentation (état des caches, etc.).

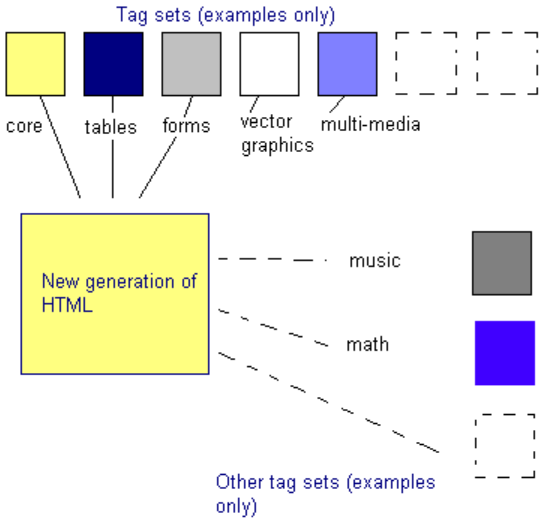
WML occupe une place très importante car il se rapproche le plus des documents HTML disponibles sur le web. Étant donnée que ce langage est décrit sous forme d'une DTD XML, sa génération automatique à partir des documents web peut être réalisé au moyen des transformations de structures logiques ainsi qu'au moyen du langage XSLT. Ces dernières techniques sont assez bien maîtrisées par un bon nombre d'outils et font donc de ce langage un candidat très important pour être le langage de choix pour les machines hétérogènes.

6 Modularisation et Profiles comme réponse à l'hétérogénéité

Une des approches à l'adaptabilité considérée au W3C est la modularisation. Afin de donner un example de cette modularisation, nous décrivons ce qui a été effectué dans le cas de HTML. Signalons qu'une approche similaire est en cours pour SMIL Boston.

La proposition XHTML est une réecriture de HTML 4.0 sous forme d'une application XML. Nous commençons d'abord par rappeler les motivations de cette réecriture afin de dégager le travail qui doit être accompli pour les langages multimédia en général. XHTML est organisé comme un ensemble de modules indépendants pour la description des titres, paragraphes, listes, hyperliens, images et autres construction utilisés dans les documents HTML. Les modules permettent de définir assez facilement des sous-ensembles ou des extensions de HTML pour les plates-formes émergentes. Le travail de modularisation, c'est à dire le découpage de HTML 4.0 en modules de plus en plus fin n'est pas encore achevé pour les formulaires qui sont des constructions très dépendantes des périphériques.

La figure suivante montre la modularisation de HTML 4.0, c'est à dire XHTML tel qu'il est perçu aujourd'hui.



Comment peut-on identifier quel sous-ensemble (extensions) de modules un type de terminal donné supporte ? La situation aujourd'hui est que HTML et CSS couvrent un large spectre de fonctionnalités. Aucun des browsers ne supporte la totalité de la spécification. Ceci a conduit à des problèmes d'inter-opérabilité comme l'apparition sous différentes formes graphiques d'un même document.

Afin de remédier à cette situation qui, dans le fond, est très similaire aux limitations de fonctionnalités par manque de ressources, le W3C est en train de développer la notion de **device profile**. Un device profile définit exactement quel est le sous-ensemble de modules (de XHTML pour l'instant) qu'un appareil donné supporte.

Le device profile est complété par le **document profile** qui fournit, entre autres, le niveau de support de XHTML et de feuille de style nécessaire pour visualiser le document. Le document profile est en cours de définition au sein du W3C.

7 Systèmes d'information multimédias "intelligents"

L'objectif au niveau des systèmes multimédia est de concevoir et de développer un environnement "intelligent" sur le web qui améliore le confort et la qualité de la restitution du contenu pour l'ensemble des appareils : on parle alors de qualité de service du système multimédia. Ces dernières années, la gestion répartie de la qualité de service a attiré de plus en plus d'attention (voir le journal *Multimedia Systems* de l'ACM). La difficulté réside dans la mise en œuvre de systèmes qui deviennent de plus en plus complexes, notamment à cause de l'hétérogénéité du matériel et du logiciel.

Il est donc important d'identifier les nouvelles caractéristiques de chaque composant du système (utilisateurs, application cliente, documents, appareil, réseaux et serveurs) qui intervient dans la qualité d'une présentation. L'objectif est de s'adapter aux conditions d'utilisation (ressources disponibles, mobilité, …) de façon à optimiser les performances du système tout entier. Cette adaptation peut être effectuée à plusieurs niveaux : au niveau de l'application sous forme d'un processus de transformation de documents comme évoqué dans les section précédentes (dégradation, sélection du contenu à partir de méta-données, etc.), ajustements au niveau du support système et réseau, mobilité du code, supervision de l'exécution répartie.

Le travail à accomplir dans ce domaine consiste à aborder le problème à travers une approche plus globale. Jusqu'à présent, la plupart des solutions proposées se limitaient à l'étude d'un composant particulier (gestion indépendante des transferts des différents médias, synchronisation du côté client, stockage des données sur des serveurs dédiés) sur une infrastructure relativement homogène. L'avantage d'une approche globale consiste à formuler le problème en partant des données structurées comme les documents multimédia contenu dans les applications. Ceci permet de considérer la qualité de service à l'échelle de toute une présentation (de bout en bout) car les structures d'un document, en particulier temporelle, permettent une meilleur évaluation des besoins en termes de ressources (bande passante, puissance CPU). Pour que cette démarche aboutisse, il faut s'appuyer sur trois axes complémentaires : l'enrichissement de la représentation des données et des documents multimédia (méta-données), l'élaboration d'algorithmes de supervision répartis qui exploitent cette représentation et enfin, l'amélioration des services système et réseau en tirant profit des structures des documents.

8 Adaptation au niveau des documents et de leur méta-données

Avant d'introduire la problématique de l'adaptation nous allons d'abord rappeler ses origines. Celles-ci se rapportent directement à l'hétérogénéité du réseau et de la puissance des points de calcul présentés plus haut.

Jusqu'à présent, les travaux de recherche sur les documents multimédia (et en particulier ceux menés dans le projet Opéra) ont surtout porté sur la définition des modèles de document qui tiennent compte de toutes ses dimensions (dimension logique, spatiale, temporelle et sémantique). Le résultat est largement entériné dans l'état de l'art et les efforts en cours en terme de standardisation vont dans le même sens. Il se trouve néanmoins que l'hétérogénéité au niveau de l'infrastructure informatique est encore peu prise en compte de façon complète.

Le problème est principalement lié à la nature des transformations qu'il faudrait appliquer pour pouvoir "exploiter" dans de bonnes conditions du contenu sur cette infrastructure.

- L'adaptation est effectuée en une fois, à savoir qu'un document spécifique est créé après analyse des différentes contraintes au moment de la demande. Plus aucune modification n'a lieu ensuite lors de la présentation.
- Les principes d'adaptation prennent en compte à la fois des alternatives prévues par le créateur de document et des techniques d'adaptation automatique.

Quatre facteurs d'adaptation se dégagent :

- le type de terminal,
- le profil utilisateur,
- le réseau (QoS),
- le contexte ou environnement (géolocalisation, environnement bruité, en voiture…).

Ces facteurs impliquent un certain nombre de contraintes :

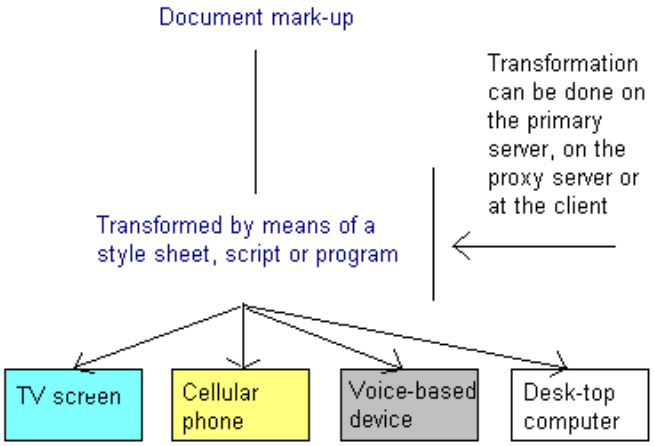
- Contrainte média : transformation d'un média (dégradation, résumé) ou substitution d'un média par un autre.
- Contrainte spatiale : position ou dimension des éléments.
- Contrainte temporelle : modification de la durée ou de l'enchaînement de la présentation.
- Contrainte de ressources : CPU, taille de fichiers, type de players disponibles.
- Contrainte sémantique : type d'utilisateur (expert, débutant), contexte (géolocalisation).

Dans la suite de cette section, nous allons décrire l'architecture d'un système multimédia adaptable. Nous allons ensuite présenter, à travers un exemple concret, le langage de transformation XSLT. Enfin, nous nous focalisons sur les adaptations portant sur chacune des dimensions des documents.

8.1 Architecture globale d'un système multimédia adaptable

Une méthode générale qui se dégage pour traiter les problèmes de l'adaptation consiste à développer des mécanismes de transformation pour affiner le langage de marquage ainsi que les feuilles de style pour les différents périphériques. Si le sous-ensemble de modules et donc les fonctionnalités supportées par un périphérique peuvent être décrites de façon plus précise alors des algorithmes de transformation peuvent être utilisés pour adapter (re-purpose) les documents d'une manière plus simple.

Les transformations peuvent être réalisées par les auteurs sur un site donné, sur des serveurs proxy ou encore dans le browser même (cf. figure suivante).



Pour le moment, nous pouvons déjà considérer que le rôle des feuilles de style va devenir de plus en plus important, en plus de la définition du rendu graphique comme la fonte, la couleur ainsi que la forme de diagrammes et autres constructions que peut inclure un document. Les feuilles de style vont prendre une partie des transformations du langage source pour adapter le contenu sur les différents appareils de restitution. Si on considère que dans le futur l'auteur insère le contenu une seule fois dans le serveur web pour qu'il soit accessible sur l'ensemble des appareils, les outils de transformation peuvent être réalisés sous trois formes :

- Des feuilles de style écrites dans un langage comme XSLT.
- Des scripts attachés aux documents.
- Des programmes déployés sur les serveurs proxy.

8.2 Langage de transformation : XSL et adaptation

Afin de simplifier la réalisation de bases documentaires il est possible de spécifier des documents à un niveau générique à l'aide de la notion de classe de document et de présenter ces documents selon différents formats de sortie à l'aide de différentes applications. Ainsi n'importe quelle instance XML appartenant à une classe de document donnée peut être présentée de façon automatique dans une forme pré-spécifiée. Comme a priori la structure de présentation peut être

totalement différente de la structure du document source, il est nécessaire d'utiliser un langage de transformation pour, par exemple, renommer les attributs, créer des éléments, réordonner des éléments (exemple : la table des matière) et associer des propriétés spatiales et temporelles.

La transformation ajoute au document des éléments de présentation propres à l'application cible. Avec une approche générique, on peut spécifier les *règles de transformation* qui doivent à partir de la structure logique du document définir des informations sur les dimensions spatiales et temporelles propres à l'application multimédia.

8.2.1 Exemple de transformation

Pour illustrer ces principes, nous nous sommes placés dans le cadre particulier de la classe de document de type *SlidesShow* définie ci-dessous. Le but est de transformer un document XML de type *SlidesShow* en un document multimédia (avec des propriétés temporelles qui définissent le déroulement du document).

Par exemple, voici un exemple un document de type *SlidesShow* :

<SlidesShow>

<Header >

<Title> font = "serif" *Transformation de documents structurés* </Title >

<Author> font *Nicolas Seytre* </Author >

<Date> *27 août 1999* </Date >

</Header >

<Slide >

  <Title> font = "serif" *Edition de documents multimédia* </Title >

 <Body> ... </Body >

</Slide >

<Slide>

 <Title font = "serif"> *Conclusion* </Title>

 <body> ... </body >

</Slide>

</SlidesShow>

Ce document est bien formé et est valide pour la DTD suivante :

<!ELEMENT SlidesShow (Header, Slide*)>

<!ELEMENT Header (Title, Author*, Date)>

<!ELEMENT Slide (Title, Body)>

<!ELEMENT Title (#PCDATA)>

<!ATTLIST Title font CDATA #REQUIRED>

<!ELEMENT Date (#PCDATA)>

<!ELEMENT Body (#PCDATA)>

Cette DTD exprime qu'un document *SlidesShow* est composé d'un élément de type *Header*, et de 0 ou n éléments de type *Slide*. L'élément *Header* contient des élément de type *Title*, *Dat*e et 0 ou n éléments de type *Author*. L'élément *Slide* contient des élément de type *Title*, *Date* et *Body*. Les éléments *Title*, *Date* et *Body* contiennent des éléments de type *text* (#PCDATA). De plus une liste d'attributs est définie pour l'élément *Title*. Elle contient l'attribut font qui est de type text et qui est requis.

On souhaite, à partir de ce document XML obtenir le Document Madeus suivant (pour que l'exemple soit plus lisible, aucun attribut de présentation n'est défini). Les informations de contenu qui proviennent du document source sont représentées en italique :

```
<Madeus Name="DocMadeus" Version="1.0">
<Composite Name="Présentation">
<Composite Name="EnTete"
<Text Name="TitrePresentation" Value="Transformation de document structuré"/>
<Text Name="HautGauche"
Value="Nicolas Seytre" />
<Text Name="HautDroit"
Value="27 août 1999"/>
</Composite>
<Composite Name="Transparent.1">
<Text Name="TitreTransparent"
Value="Edition de documents multimédia"/
<Text Name="Contenu" Value="..."
</Composite>
<Composite Name="Transparent.2">
<Text Name="TitreTransparent"
Value="Conclusion"/>
<Text Name="Contenu" Value="..."/>
</Composite>
<Relations>
<Temporal>
<Meets Interval1="Transparent.1" Interval2="Transparent.2"/
</Temporal>
</Relations>
</Composite>
```

</Madeus>

Le contenu du document transformé doit être conforme à celui du document source. Une dimension temporelle est ajoutée au document pour pouvoir jouer les transparents de façon séquentielle.

8.2.2 Transformations nécessaires

- produire un élément *Composite* de nom *présentation* contenant un élément *Composite* de nom *EnTête* et des éléments *Composite* de nom *Transparent ;*

- l'élément *Composite* de nom *entête* doit contenir les éléments *Text* de nom *TitrePrésentation*, *HautGauche* et *HautDroit ;*

- l'élément *TitrePrésentation* doit posséder un attribut *value* dont le contenu est celui de l'élément source *Slides/Header/Title* (Un élément est sélectionné en fonction de son contexte ascendant. Par conséquent nous le désignons par le chemin qui permet d'y accéder dans l'arbre qui définit la structure logique du document). L'élément *HautGauche* doit posséder un attribue *value* dont le contenu est celui de l'élément *source Slides/Header/Author*. De même l'élément *HautDroit* possède un attribut *value* dont le contenu est celui de l'élément *Slides/Header/Date ;*

- pour chaque *Slides/Slide* on veut produire un élément *Composite* de nom *Transparent* dont le nom défini par l'attribut *name* est indexé selon l'ordre de définition des éléments slide du document source. Chacun de ces éléments *Transparent* doit contenir un élément *Text* de nom *TitreTransparent* et un élément *Text* de nom *Contenu* ;

- chaque élément *TitreTransparent* doit posséder un attribut *value* dont le contenu est celui de l'élément *Slides/Slide/Title* correspondant dans le document source. De même chaque élément *Contenu* doit posséder un attribut *value* dont le contenu est celui de l'élément *Slides/Slide/Body* correspondant dans le document source ;

- enfin on souhaite ajouter une dimension temporelle à notre document. Pour cela on crée des relations temporelles comme défini par la DTD Madeus. Pour ce type de document on a choisi de placer une relation *Meets* entre chaque *Transparent*. Cette relation est définie par : l'élément référencé par l'attribut *interval2* est joué après l'*interval1*.

De plus, des attributs de présentation spécifiques à la DTD Madeus peuvent être ajoutés sur les éléments. Par exemple, on pourrait spécifier des attributs *Font* pour chaque élément *Text.* On pourrait également placer des relations spatiales entre les différents éléments de chaque transparent.

8.2.3 Processus de transformation avec XSL

- Spécification de la transformation :

  Pour spécifier la transformation nous pouvons utiliser le langage XSL (eXtensible StyleSheet Language). Ce choix est justifié par le fait que c'est un langage de définition de feuilles de styles pour les documents XML. Celui-ci comprend une partie qui permet de décrire des transformations d'arbre et une partie de formatage. Nous nous intéressons

uniquement à la partie transformation que nous appliquons à notre propre langage de formatage (à base de relations).

- Interprétation de la spécification de transformation :

  Il existe pour cela plusieurs processeurs XSL dont le principe est le suivant:

  Un processeur XSL est un programme qui prend en entrée un document XML bien formé (mais pas forcément valide pour une DTD) et applique à celui-ci les règles définies par une feuille de styles XSLT.

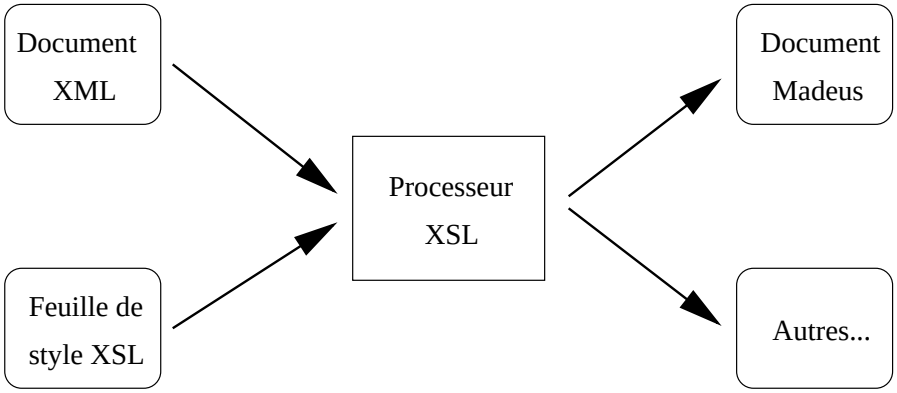


Figure 1 - Processeur XSL

  Les processeurs référencés par le W3C sont :

  - Koala XSL de Jérémy Calles
  - XT de James Clark
  - Lotus XSL

- Présentation du document transformé.

  Dans notre exemple, il s'agit de présenter notre document de type *Slide*sShow à l'aide de l'application Madeus. Nous avons donc dans un premier temps intégré le processeur XSL choisi dans l'application Madeus.

8.2.4 Application à l'exemple

Nous allons maintenant décrire une partie de la feuille de styles qui a été réalisée pour illustrer le problème de la transformation à l'aide du langage XSLT. L'exemple est basé sur le document de type *SlidesShow* défini en 8.2.1. L'objectif est d'obtenir le document Madeus de type *Transparent* décrit de façon informelle en 8.2.2.

Une feuille de styles décrivant la transformation du document XML de type *SlideShow* en un document Madeus est relativement complexe. C'est pourquoi nous donnons en exemple uniquement la partie de transformation qui permet d'obtenir la présentation d'une séquence des éléments *Slide* joués de façon séquentiel.

Le document source ne décrit que la structure logique du document. De cette structure nous devons extraire une interprétation pour présenter ce document à l'aide du langage défini dans Madeus. La principale difficulté est d'ajouter au document une dimension temporelle. La construction de contraintes temporelles est directement déduite de la structure du document source. En effet les transparents doivent être joués séquentiellement dans l'ordre où ils sont définis dans le document source (figure 3).

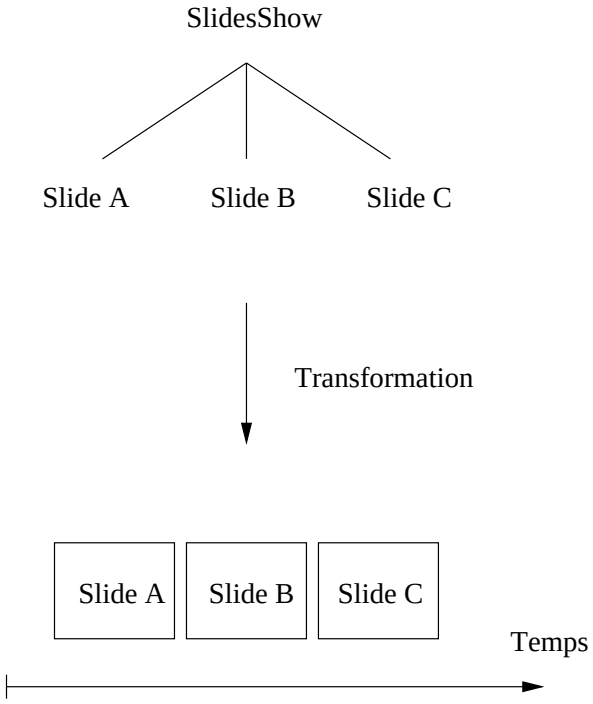


Figure 3 - *Ordonnancement temporel des éléments d'un document*

Pour pouvoir placer des relations entre les transparents il faut que les éléments Madeus qui les définissent aient des noms différents les uns des autres. Il est donc nécessaire d'indexer le nom de chaque élément qui défini un transparent. Nous allons donc produire des éléments *Composite* de nom *Transparent.i* avec i la position de l'élément *Slide*.

Une règle XSLT est définie par l'élément *xsl:template* dont l'attribut *match* définit le noeud du document source qui instancie la règle. Dans notre cas nous définissons une règle (*template*) qui sélectionne (*match*) le noeud *SlidesShow* (ligne 2).

Ensuite nous souhaitons réaliser un traitement sur tous les éléments de type *Slide.* On utilise dans ce cas l'élément *xsl:for-each* (ligne 3). Le traitement défini à l'intérieur de cette option est appliqué à tous les éléments sélectionnés par son attribut *select*.

Le traitement réalisé est le suivant:

- ligne 5 à 8 : définition d'un élément *xsl:variable* qui possède un attribut *name* dont la valeur sera le nom de cette variable. Le contenu de la variable est défini par un élément *xsl:text* dont le contenu est la chaîne "Transparent.". Cette chaîne est concaténée à un indice défini par

l'élément *xsl:number*. L'attribut *expr* de l'élément *xsl:number* est affecté avec la position de l'élément du document XML source, instancié par la règle XSL qui est appliquée ;

- ligne 10 à 13 : construction d'un élément Composite dont l'attribut *name* est affecté avec le résultat de la variable *idSlide*. La référence à cette variable se fait en préfixant son nom par un caractère "$". Les accolades signifient que le processeur doit évaluer le contenu de cette variable.

  L'attribut Value est affecté avec le contenu du noeud source *Title*. Les accolades signifient que le processeur doit évaluer le contenu du noeud source *Title*.

```
1. <!-- la règle est appliquée sur le noeud slides -->
2. <xsl:template match="slides">
3.   <xsl:for-each select="slide">
4.   <!-- on construit le nom du slide courant -->
5.     <xsl:variable name="idSlide">
6.       <xsl:text> Transparent.</xsl:text>
7.       <xsl:number expr = "position()" />
8.     </xsl:variable>
9.     <!-- production du Transparent -->
10.    <Composite Name="{$idSlide}">
11.     <Text Name = "TitreTransparent" Value="{title}"/>
12.     <Text Name="Contenu" Value="{body}"/>
13.    </Composite>
14.  </xsl:for-each>
15.</xsl:template>
```

Le traitement précédent permet d'obtenir des éléments *Composite* de nom différent pour chaque transparent. On peut maintenant placer des contraintes temporelles entre ceux-ci.

Nous exprimons les relations temporelles à l'aide de la relation *Meet* entre deux éléments. Il y a donc une relation de moins que d'éléments *Slide*. Il faut donc filtrer le premier élément *Slide*. Pour cela nous utilisons la possibilité d'exprimer des conditions à l'aide de l'élément *xsl:if*.

```
<!-- Construction de l'élément relation temporelle -->
<Relations>
<Temporal>
<xsl:for-each select="slide">
<!-- On place une relation Meet entre le premier transparent et une table des
matières qui pourrait être définie par ailleurs.
Pour cela on teste si l'on est sur le premier transparent -->
<xsl:if test="position()=1">
<Meets Interval1="tableOfContent" Interval2="Transparent.1"/>
```

```
</xsl:if>
<!-- Si l'on n'est pas sur le premier transparent on construit une variable
qui contient le nom de l'élément transparent précédent et une variable qui
contient le nom de l'élément transparent courant -->
<xsl:if tests="not(position()=1)">
<xsl:variable name="précédent">
<xsl:text>
Transparent.
</xsl:text>
<xsl:number expr = "position()-1"/>
</xsl:variable>
<xsl:variable name="courant">
<xsl:text>
Transparent.
</xsl:text>
<xsl:number expr = "position()"/>
</xsl:variable >
<!-- Enfin on construit une relation Meets entre le transparent courant et le
precedent -->
<Meets Interval1="{$précédent}" Interval2="{$courant}"/>
</xsl:if>
</xsl:for-each>
</Temporal>
</Relations>
```

Cette feuille de style permet donc de présenter dans Madeus toute instance de la classe de document *SlidesShow*.

8.3 Dimensions d'un document et adaptation

Le but des transformations des documents pour l'adaptation est de pouvoir passer d'une représentation d'un document, par exemple purement spatiale, à n'importe quelle autre représentation, par exemple avec une forte structure de navigation mais peu d'organisation spatiale, ou encore une organisation surtout temporelle. L'objectif est donc d'obtenir une richesse au niveau du modèle du document qui permet de s'abstraire des dimensions du document jusqu'à ce que leur génération à la demande puisse être réalisée. Il est bien entendu très tôt pour cerner les propriétés que doit fournir ce modèle (ie. on ne sait pas encore jusqu'à quel point le processus automatique de génération de ces dimensions peut-être automatisable). Ce travail de recherche représente donc une problématique ouverte.

8.3.1 Espace : Adaptation au moyen des contraintes spatiales

L'adaptation dans la dimension spatiale consiste à utiliser les techniques de spécification et de résolution de contraintes géométriques. L'adaptation pour un écran de plus petite taille que celui sur lequel un document à été initialement conçu consiste à s'assurer dans une première étape que

la description est conforme au langage de contraintes spatiales. Cela suppose que le positionnement est décrit en termes de relations plutôt que de tailles et de positions absolues. Il est possible que cette phase nécessite une analyse qui fait intervenir la structure logique du document afin de retrouver une expression à base de relations. A la suite de cette phase, il faudra appliquer une phase de résolution de contraintes pour adapter le nouveau rendu graphique aux nouvelles contraintes. Une étude sur l'intégration des résolveurs de contraintes spatiales est disponible dans [Car98].

8.3.2 Temps : Adaptation au moyen des contraintes temporelles

L'adaptation dans la dimension temporelle concerne plus précisément l'apport des contraintes pour ce problème. Une hypothèse de travail est de profiter de la flexibilité induite par la spécification par contraintes (qui décrit un espace de solutions et non une solution) pour définir des stratégies d'adaptation pertinentes. Peu de travaux sur l'adaptation considèrent les approches par contraintes. Seul le projet Tiempo [Wir97] envisage le problème sous cet angle.

8.3.3 Structure : Adaptation au moyen de la restructuration du document

Si l'information est décrite sous une forme suffisamment typée au travers de DTD XML, une part de l'adaptation peut être effectuée en utilisant des techniques de transformation de structures. Par exemple, il est possible de transformer un document constitué d'une longue liste de liens (difficile à formater sur un PDA) en un document comportant plusieurs niveaux de structures de listes. Le principe de base est fondamentalement le suivant :

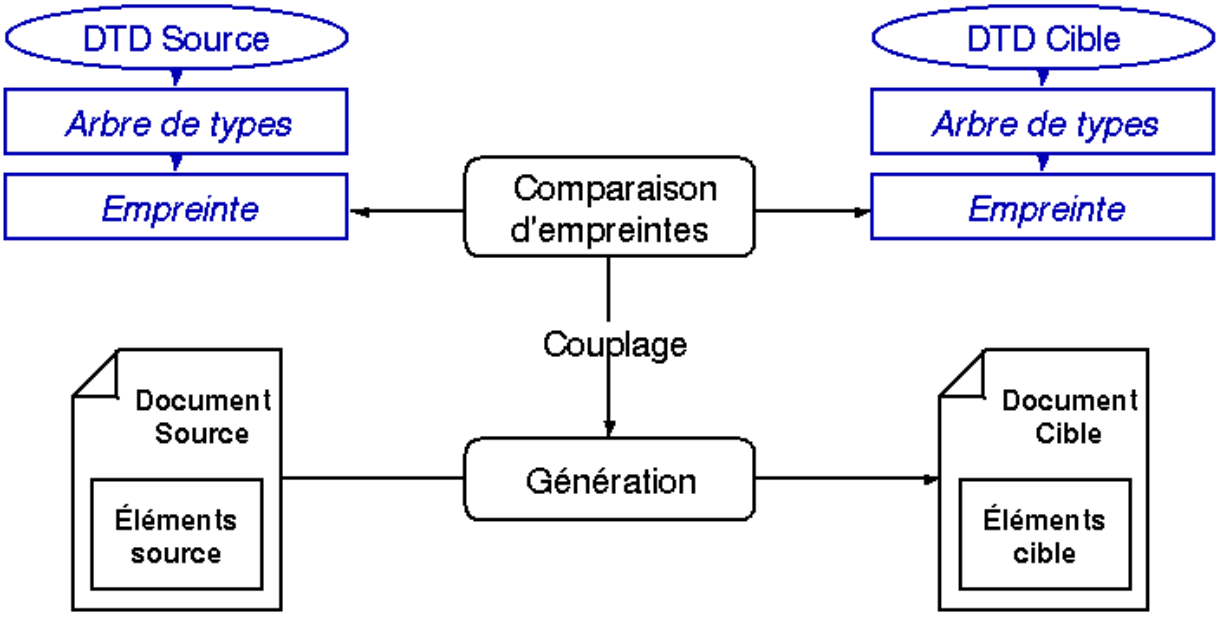
Etant donné un document appartenant à une classe, elle-même définie par une DTD, par quel procédé est-il possible de transformer l'intégralité ou une partie de ce document pour le rendre compatible avec une autre DTD ?

Les techniques de transformation de structures sont largement répandues dans les chaînes de traitement documentaire. On peut identifier trois familles de techniques de transformation fondées sur des principes différents : les filtres, les systèmes de transformation explicite et les systèmes de transformation automatique.

- **Les filtres** permettent la traduction de documents d'un format vers un autre, les deux formats étant définis à l'avance. Les filtres sont utilisés pour traduire des documents entiers et sont indépendants des autres applications. Ils permettent la transformation des documents appartenant à un unique format ou à une même DTD source et produisent un document d'un format préalablement défini. Ils sont en particulier utilisés pour l'importation et la sauvegarde de documents codés dans un format spécifique à une application. On peut citer les filtres de traduction de LaTeX vers HTML, de document formatés vers des documents SGML, conformes à des DTD particulières comme les DTD d'acquisition (TEI).

- **Les systèmes de transformation explicite** se basent sur un ensemble de règles guidant la transformation (voir figure 1). Ces règles permettent de paramétrer les transformations et leur permettent, à la différence des filtres, de s'appliquer à différents formats de documents. Cette technique peut être mise en œuvre par des applications dédiées aux transformations ou intégrées dans d'autres applications (éditeurs de documents, bases de données). Les langages de transformation utilisés dans ces systèmes peuvent être de nature déclarative, comme les langages d'IDM, de Chameleon ou de XSL, impérative comme le langage de Balise et de CoST 2 ou par requêtes comme SgmlQL et Scrimshaw [Bon98].

figure 1: Principe de la transformation explicite



- La comparaison de DTD permet une **transformation automatique** de documents selon le principe de la figure 2 proposé dans [Bon98]. Pour effectuer la comparaison entre une DTD source et une DTD cible, l'ensemble des règles constituant chaque DTD est représenté sous forme d'un arbre (appelé arbre de types). Pour des raisons d'efficacité, la comparaison est réalisée sur des formes linéaires des arbres de types appelées empreintes . Le résultat d'une comparaison est un ensemble de relations entre un type source et un type cible appelé couplage. La génération d'un document transformé peut alors être effectuée à l'aide de ce couplage.

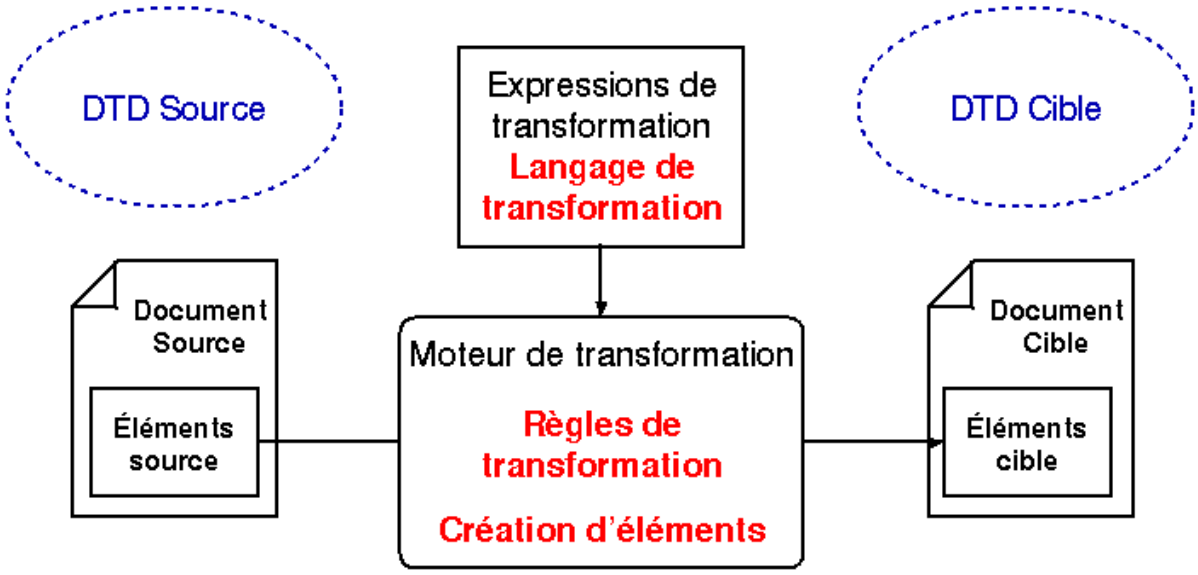


figure 2 Principe de la transformation automatique

Il est clair que ces méthodes (et les outils qui les utilisent) peuvent apporter des éléments de réponse au problème de l'adaptation. Leur atout principal vient de leur généricité et de leur intégration dans un processus qui prend en compte toutes les caractéristiques des documents (en particulier on peut associer des propriétés de style spatial et temporel aux structures transformées).

8.3.4 Navigation : Adaptation au moyen des structures de navigation

L'adaptation au moyen des structures de navigation consiste à doter un document d'une structure de navigation qui permet de faciliter l'accès a son contenu. Par exemple, pour les téléphones portables, les capacités d'interaction avec l'utilisateur et d'affichage étant très limitées, une des transformations pour améliorer l'accès au document consiste à le subdiviser en portions de petite taille et de le doter d'une structure de navigation efficace. Une telle structure peut être une représentation du document à plusieurs échelles de détails. Par exemple, à partir d'une structure extraite automatiquement et qui ressemble à une table des matières, l'accès à une portion précise d'un document pourra se trouver facilité. De plus, le contenu des ancres de navigation peut lui-même être généré pour satisfaire les contraintes d'affichage comme le nombre de caractères visibles à un instant donnée. L'activation de ces ancres peut aussi être adaptée aux capacités de saisie de la machine cible de l'adaptation. Au moment où nous menons cette étude peu de travaux portent sur cette problématique qui est un sujet supplémentaire d'étude.

9 Adaptation au niveau du support système et réseau

Les applications multimédia réparties manipulent des média continus dont le traitement doit être réalisé à des instants précis et dans des délais bornés ; en d'autres termes, elles présentent des caractéristiques temps réel. La vidéo en est un exemple : une image doit être présentée à l'utilisateur tous les 1/25 s. ; une tâche vidéo chargée du traitement et de l'affichage d'une image doit pouvoir accéder aux ressources CPU au moins une fois à chaque période. Elle doit aussi

disposer du flot de données, qui peut provenir d'une source distante. Le travail de recherche qui reste à accomplir dans ce domaine reste très vaste et encore peu exploré. Trois voies semblent très prometteuses :

- **Supervision répartie de la présentation :** l'objectif général au niveau de l'application multimédia est de concevoir et d'optimiser le traitement des documents hypermédia. Il est clair cependant qu'une réponse pertinente au problème de la présentation de documents multimédia ne pourra être correctement apportée que si l'on prend en compte les trois parties qui sont impliquées :

    1. L'environnement de présentation (machine cliente ou "point de calcul", localisation, profil utilisateur).

    2. L'infrastructure réseau (gestion des contraintes du réseau comme la bande passante, les latences d'accès.

    3. Les entrepôts de données multimédia (machine serveur de données pour les documents, la vidéo, l'audio, les images, le texte, etc.).

    Ainsi, les environnements lecteur (les clients) ont à la fois besoin d'une meilleure supervision de la présentation (adaptation de la dimension temporelle des documents présentés) ainsi que des moyens d'agir sur les serveurs pour réaliser de telles adaptations. Du côté du serveur, le mode de représentation des données (codage hiérarchique), de leur stockage et surtout de leur accès doit être en conséquence adaptatif. Dans une première étape, il est nécessaire de définir des critères pour exprimer les contraintes de ressources à l'échelle des documents multimédia interactifs, ensuite il faut concevoir un processus de supervision permettant, à partir de l'évolution de ces paramètres, de décider dynamiquement quand et comment les adaptations doivent être mises en œuvre. Dans les documents multimédia, les scénarios temporels sont déjà manipulés de façon explicite par le système de présentation. Il devient donc possible de mettre à profit cette structure pour améliorer la qualité d'une présentation par prédiction des besoins et interpolation des ressources disponibles.

- **Support d'exécution :** le support d'exécution des application multimédia actuellement considéré est principalement tourné vers l'application cliente. Les points de calcul intermédiaires sont peu ou pas considérés (nœuds de routage, serveurs,...). Cependant, on s'aperçoit, dès à présent que l'impact sur la qualité d'une présentation multimédia résulte du manque d'une solution de bout en bout. Le volume important de données à traiter, à transférer et à stocker induit une dérive de la présentation que les outils hypermédia doivent prendre en compte. Il faut donc que le schéma général de l'exécution d'une présentation soit très performant pour être en accord avec des traitements de type temps-réel. En outre, il doit être extensible de façon à intégrer de nouveaux schémas d'exécution comme ceux déclenchés par la communication initiée par le serveur (*server push*), les agents mobiles ainsi que la programmation structurée comme dans DOM, permettant une plus grande intégration au sein d'infrastructure.

- **Protocoles de communication :** Les applications multimédia ont des exigences en terme de latence de transmission et de support multipoint beaucoup plus importantes que les applications classiques comme le transfert de fichiers. Face à ces nouveaux besoins, de nouveaux protocoles de transport comme RSVP et ST-2 ont été proposés [Deg97].

Cependant, l'Internet ne permet aujourd'hui ni de faire de la réservation de ressources ni d'instaurer des priorités sur les flux transmis. Dans ces conditions, le rôle des protocoles de transport est assez limité [Haf98].

Cependant, la gestion des accès réseau a déjà fait l'objet d'un certain nombre de travaux. Dans le cadre des documents multimédia, il est possible d'explorer d'autres voies pour répondre à ces problèmes. Parmi celles-ci, une exploitation du scénario temporel permet, par exemple, d'anticiper le chargement d'objets multimédia par prédiction du futur proche. Il y a là, des compromis à trouver entre la capacité de stockage d'un système (éventuellement une base de données) et les garanties sur la qualité d'une présentation. D'autres questions sont soulevées, comme la possibilité de définir des requêtes et des mécanismes d'accès atomiques appliqués à un groupe d'objets multimédia (une portion de document).

10 Conclusion

Dans ce rapport exploratoire, nous avons identifié les études complémentaires qu'il faudra poursuivre pour obtenir une vision plus claire et surtout plus complète de ce que doit être un système d'information multimédia adaptable. Dans une première étape, nous avons décrit comment les évolutions du web qui tentent, sans apporter une réponse au problème de l'adaptabilité, d'anticiper une infrastructure qui facilite le déploiement de ce type de systèmes. Nous avons ensuite montré en quoi consiste la modularisation des standards du web et la notion de profils d'utilisateurs et de machines. De plus, nous avons décrit les transformations portant sur chacune des dimensions des documents et qu'il faudra sans doute combiner pour obtenir le niveau d'adaptation souhaité.

Malgré l'état de l'art très peu développé et extrêmement pauvre sur ces questions, nous avons tenté de donner la priorité dans ce rapport à trois aspects qui représentent à nos yeux les points clés de la mise en œuvre d'une solution satisfaisante et extensible :

- L'architecture globale du système d'information multimédia adaptable. En particulier, la place prédominante qu'occupe les mécanismes de négociation entre un client et un serveur de document multimédia adaptable.
- Les transformations qui doivent être intégrées à ce système pour qu'il soit adaptable aux différents périphériques. En particulier, les transformations qui doivent être appliquées à l'ensemble des dimensions du document.
- L'identification et l'intégration des standards et les protocoles du web au sein du système considéré.

Ce qui ressort de cette étude est la nécessité de définir un projet ambitieux et surtout intégré pour qu'un système adaptable opérationnel voit le jour. Par rapport à l'étude préliminaire menée chez Alcatel et dans le but de lui être complémentaire, nous avons donné la priorité à l'adaptation fondée sur des transformations de document plutôt que celle liée à la qualité de service au sens réseau du terme. En effet, c'est surtout à ce niveau que le contenu circulant sur le web peut être rendu accessible à des machines hétérogènes.

11 Références

[AFQ 1989] J. André, R. Furuta, V. Quint, *Structured documents,* Cambridge University Press, Cambridge, 1989.

[Bon98] Stéphane Bonhomme, Transformations de documents structurés : une combinaison des approches déclaratives et automatiques, Doctorat d'informatique, Université Joseph Fourier, décembre 1998. ftp://ftp.inrialpes.fr/pub/opera/theses/bonhomme.ps.gz

[Car97] Laurent Carcone, Gestion de contraintes spatiales dans Madeus : un système d'édition multimédia, Mémoire d'ingénieur Cnam, Grenoble, 1997. ftp://ftp.inrialpes.fr/pub/opera/rapports/MemoireCarcone97.ps.gz

[Deg 97] M. Degermark, M. Kohler, S. Pink and O. Schelen, "Adavance reservations for predictive service over the Internet", *ACM Journal on Multimedia Systems,* num. 5, pp. 177-186, May 1997.

[Han98] Hannelle Antikainen, Kaisa Kostiainen and Caj Södergard, "News content for mobile terminals", VTT Information Technology, 13 November 1998.

[Xml98] T. Bray, J. Paoli, C. M. Sperberg-McQueen,"Extensible Markup Language (XML) 1.0", 10 Feb 98. This is available at http://www.w3.org/TR/REC-xml.

[Haf98] A. Hafid, G. V. Bochmann et R Dssouli. Distributed Multimedia Application and Quality of Service: a Review. Electronic Journal on Network and Distributed Processing, février 1998.

[Html98] D. Raggett, A. Le Hors, I. Jacobs, "HTML 4.0 Specification", 24 Apr 98. This is available at http://www.w3.org/TR/REC-html40.

[SMIL1.0] "Synchronized Multimedia Integration Language (SMIL) 1.0 Specification W3C Recommendation 15-June-1998 ". Available at: http://www.w3.org/TR/REC-smil.

[SMIL-MOD] Patrick Schmitz, Ted Wugofski, Warner ten Kate, "Synchronized Multimedia Modules based upon SMIL 1.0". Available at http://www.w3.org/TR/NOTE-SYMM-modules.

[Tat99] Tatsuya Hagino, Mobile Access Activity Statement, The World Wide Web Consortium. Available at http://www.w3.org/Mobile/Activity.

[Oht99] Hidetaka Ohto and Johan Hjelm, W3C/Panasonic/Ericsson, "CC/PP exchange protocol based on HTTP Extension Framework", W3C Note 24 June 1999.

[Rey99] Franklin Reynolds (Nokia), Johan Hjelm (W3C/Ericsson), Spencer Dawkins (Nortel) and Sandeep Singhal (IBM). "Composite Capability/Preference Profiles (CC/PP): A user side framework for content negotiation". W3C Note 27 July 1999. Available at http://www.w3.org/TR/NOTE-CCPP/.

[Wir97] WIRAG S. Modeling of Adaptable Multimedia Documents Proceedings of European Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS'97) Darmstadt septembre 1997.

[W3C Namespaces 1999] W3C Recommendation, *Namespaces in XML,* http://www.w3.org/TR/1998/REC-xml-names, 14-January 1999.

[W3C SVG 1998] W3C Working Draft, *Scalable Vector Graphics Language (SVG) Version 1.0,* http://www.w3.org/TR/WD-svg, 1998.

[W3C XSL 1998] W3C Working Draft, *Extensible Specifiaction Language (XSL) Version 1.0,* http://www.w3.org/TR/WD-xsl, 16 décembre 1998.

[W3C XML 1998] W3C Recommendation, *Extensible Markup Language (XML) 1.0,* http://www.w3.org/TR/1998/REC-xml-19980210, 10-February 1998.

[W3C XML-QL 1998] W3C Submission, *XML-QL: A Query Language for XML,* http://www.w3.org/TR/1998/NOTE-xml-ql-19980819/, 19-August 1998.

[W3C XLink 1998] W3C Working Draft, *XML Linking Language (XLink),* http://www.w3.org/TR/WD-xlink, 3-March 1998.

[W3C XPointer 1998] W3C Working Draft, *XML Pointer Language (XPointer),* http://www.w3.org/TR/WD-xptr, 3-March 1998.

[W3C XML Schema 1999] W3C Notet, *XML Schema Requirements,* http://www.w3.org/TR/NOTE-xml-schema-req, 15-February 1999.

[W3C MathML 1998] W3C Recommendation, *Mathematical Markup Language (MathML) 1.0 Specification,* http://www.w3.org/TR/REC-MathML/, 07-April 1998.

[W3C HTML 1998] W3C Recommendation, *HTML 4.0 Specification,* http://www.w3.org/TR/REC-html40/, 24-April 1998.

[W3C RDF 1998] W3C Recommendation, *RDF Specification,* http://www.w3.org/TR/REC/, 1998.

[W3C CSS 1998] W3C Recommendation, *Cascading Style Sheets, level 2, CSS2 Specification,* http://www.w3.org/TR/REC-CSS2/, 12-May 1998.

[W3C SMIL 1998] W3C Recommendation, *Synchronized Multimedia Integration Language (SMIL) 1.0 Specification,* http://www.w3.org/TR/REC-smil, 15-June 1998.

[W3C WebDAV 1998] W3C Working Draft, *(WebDAV),* http://www.w3.org/TR/WD-WebDAV, 1998.

[TEI 1998] TEI, *Text Encoding Initiative,* University of Illinois at Chicago, 1940 W. Taylor St., Room 124 Chicago, IL 60612-7352, USA, 1998. http://www.uic.edu/orgs/tei/index.html.

[FQA 1988] R. Furuta, V. Quint, J. André, ``Interactively Editing Structured Documents", *Electronic Publishing,* vol. 1, num. 1, pp. 19-44, avril 1988.

[ISO 1986] International Standard ISO 8879, *Information Processing - Text and Office Systems - Standard Generalized Markup Language (SGML),* International Standard Organization, 1986 http://www.sil.org/sgml/sgml.html.

[WAP-WML 99] Wireless Application Protocol : Wireless Markup Language Specification Vesion 1.2. http://www.wapforum.org/