N° attribué par la bibliothèque

année: 2004

THÈSE

présentée à

L'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

pour obtenir le titre de

DOCTEUR DE L'INPG

Spécialité
Informatique : système et communication

soutenue par

Tayeb LEMLOUMA

le 09 juin 2004

Titre

Architecture de Négociation et d'Adaptation de Services Multimédia dans des Environnements Hétérogènes

Directeur de thèse : Cécile Roisin
Co-encadrant : Nabil Layaïda

Jury

M. :	Jacques	Chassin de Kergommeaux	Président
M. :	Harald	Kosch	Rapporteurs
	Michel	Riveill	
M. :	Jean-Claude	Dufourd	Examinateurs
	Jérôme	Gensel	

Les machines un jour pourront résoudre tous les problèmes,
mais jamais aucune d'entre elles ne pourra en poser un !

- Albert Einstein

Remerciements

Je tiens à remercier Vincent Quint et Cécile Roisin pour m'avoir accueilli au sein du projet WAM (ex projet OPERA) et de m'avoir donné la possibilité de mener à bien ma thèse. Je remercie également Nabil Layaïda pour sa disponibilité, son aide, ses conseils précieux et son chaleureux soutien tout au long de cette thèse.

Je remercie également les membres du jury :

Jacques Chassin de Kergommeaux Professeur à l'INPG et l'ENSIMAG de m'avoir fait l'honneur de présider mon jury de thèse,

Harald Kosch Professeur à l'université de Klagenfurt et Michel Riveill Professeur à l'Ecole Supérieure en Sciences Informatiques (ESSI) / Université de Nice d'avoir accepté de juger ce travail,

Jean-Claude Dufourd directeur d'études et professeur à l'ENST/Paris et Jérôme Gensel Maître de Conférences à l'UPMF d'avoir accepté de faire partie de mon jury.

Je remercie Cécile Roisin, Nabil Layaïda et Vincent Quint pour leurs relectures attentives du mémoire.

Je voudrai aussi remercier mes collègues du projet WAM (ex projet OPERA) et du groupe de travail *Device Independance* du W3C, anciens et nouveaux, qui par leurs conseil et encouragements ont contribué à l'aboutissement de ce travail. Je remercie Irène Vatton, Vincent Kober, Daniel Weck, Peter Hewat, Jean-Yves Vion-Dury, Pierre Genvès, Lionel Villard, Frédéric Bes, Tien Tran-Thuong, Rhys Lewis, Roger Gimson, Stephane Boyera, Cedric Ulmer, Luu Tran, Kazuhiro Kitagawa, Rotan Hanrahan, ...

Enfin et surtout, je remercie toute ma famille, mes parents, mes sœurs et mes frères qui sans eux je ne serais pas là. Je remercie également tous ceux qui m'ont soutenu durant ces trois dernières années, en particulier : Guylaine, Hamoudi, Jézabel, Lilia, *Mathilde*, Mounir, Mylène, Oussama, Sandrine, ...

Table des matières

Table des figures

Chapitre 1

Introduction

1.1 Introduction

Les progrès technologiques récents ont permis l'apparition d'une grande variété de nouveaux moyens permettant à un utilisateur d'accéder et d'utiliser l'information multimédia qui l'intéresse en tout lieu et à tout moment. L'accès au contenu ne s'effectue plus de la même façon ni par les mêmes appareils qu'il y a quelques années. Les appareils d'accès à l'information ont subi une véritable révolution. En effet, les utilisateurs peuvent accéder au contenu à travers des appareils très divers : ordinateurs portables, assistants personnels, téléviseurs, téléphones cellulaires, etc. Le nombre des utilisateurs de ces nouveaux appareils continue sa croissance exponentielle. Les moyens d'accès au contenu ont également évolué, avec de nouveaux réseaux tels que les réseaux sans fil WiFi, GPRS, UMTS, etc. Ces réseaux se sont développés et se sont intégrés à l'Internet. L'utilisation du World Wide Web ne ressemble donc plus à ce qu'elle était à l'origine, où l'utilisateur accèdait à l'information depuis son ordinateur personnel et à travers le réseau filaire.

L'hétérogénéité des moyens et des appareils d'accès s'est accompagné d'une évolution importante du côté du contenu de l'information disponible sur le réseau. Aujourd'hui, on trouve une multitude de formats complexes avec de nouvelles fonctionnalités, telles que la vidéo interactive, les animations 3D et le dessin vectoriel. Ces formats s'appuient sur de nouveaux modèles qui intègrent le monde multimédia (la structure logique, l'espace, le temps et la navigation).

Cette évolution des contenus ainsi que des appareils et des moyens d'accès a fait apparaître des *environnements hétérogènes* où des utilisateurs ayant des facultés différentes souhaitent d'accéder au même contenu à travers différents réseaux de communication. D'autre part, le contenu peut être trop complexe pour qu'un terminal ayant des capacités limitées puisse le traiter et le présenter correctement. Face à cette réalité il est nécessaire de trouver des mécanismes qui permettent l'accès et l'utilisation de l'information sous une forme qui corresponde aux contraintes imposées par l'environnement.

Le travail présenté dans cette thèse a pour objectif de contribuer à l'adaptation et à la négociation des contenus en considérant les limitations des terminaux et les contraintes de leur environnement.

1.2 Cadre de travail

Ce travail de thèse s'est déroulé au sein du projet WAM (Opéra jusqu'en décembre 2002) de l'Institut National de Recherche en Informatique et en Automatique INRIA Rhône-Alpes. Il a été mené durant les deux premières années dans le contexte d'une collaboration avec la société Alcatel-Bell.

Les centres d'intérêt de la société Alcatel-Bell sont divers. Ils incluent le monde des télécommunications et de la téléphonie, l'accès au multimédia en large bande, l'accès aux données par satellites, etc. La collaboration concernait les services de négociation et d'adaptation dans une architecture client-serveur. Cela comprend :

- la notion de *profiling*,
- l'édition unique de contenu et l'adaptation du même contenu pour plusieurs appareils,
- le support de la diversité des terminaux mobiles,
- l'indépendance du contenu par rapport aux terminaux,
- l'utilisation des technologies Web telle que XML et RDF.

L'objectif de la collaboration était de : a) résoudre les problèmes liés à l'adaptation de contenu afin de satisfaire les contraintes des utilisateurs en termes de limitations matérielles ou de préférences utilisateur, et b) proposer des stratégies de négociation de contenu où les caractéristiques des clients et les fonctionnalités du contenu sont variées.

Le projet WAM s'intéresse aux problèmes posés par les évolutions du Web. Il se focalise sur la transformation de documents considérée comme un type de traitement générique des documents du Web. Il considère plus particulièrement les documents multimédia qui intègrent étroitement des média statiques (texte, images, équations) et dynamiques (vidéo, son, animations). Il applique ses résultats à l'adaptation de documents multimédia et à l'indépendance des appareils d'accès au Web. Les travaux de cette thèse contribuent directement à cette dernière activité. Notre travail s'inscrit également dans le groupe de travail *Device Independence* du W3C.

Les actions de recherche du projet WAM trouvent leur application dans plusieurs logiciels et prototypes :

- LimSee2, un éditeur de structures temporelles et spatiales pour les documents multimédia spécifiés dans le langage SMIL ;
- Amaya, un outil d'édition/navigation pour le Web ;
- NAC, un prototype d'architecture pour la négociation et l'adaptation de contenu dans les environnements hétérogènes. L'architecture NAC est développée dans cette thèse.

1.3 Motivation et objectifs

La conception d'une architecture qui permet d'adapter les contenus multimédia pour n'importe quel terminal de l'environnement hétérogène du Web représente un vrai défi. Les solutions proposées actuellement ne s'attaquent pas au problème de l'adaptation avec des architectures complètes, mais essaient de fournir des solutions à des besoins très spécifiques tels que l'adaptation des images pour les mobiles, le transcodage de la vidéo, etc. Ces solutions sont souvent basées sur des moyens de

programmation pure, en suivant des approches fermées et peu flexibles. Il paraît donc important d'étudier comment rendre l'adaptation plus ouverte et flexible et ainsi couvrir plusieurs types d'adaptation au sein d'une même architecture.

Si les solutions proposées sont spécifiques et limitées à quelques aspects de l'environnement (la taille de l'écran pour le retaillage des images, le débit du réseau pour le transcodage de la vidéo, etc.), c'est parce qu'il n'y a pas de modèle qui prenne en compte toutes les dimensions du contexte. Le cadre de travail CC/PP[1] (Composite Capabilities/Preferences Profile [43]), défini par le W3C, représente un modèle pour la description des caractéristiques des terminaux et des préférences des utilisateurs pour guider les processus d'adaptation de contenu. A présent, ce cadre de travail reste incomplet et ne peut être utilisé dans une architecture d'adaptation et de négociation de contenu. Cela est dû à deux raisons fondamentales. La première raison est que CC/PP ne considère qu'un seul maillon (le terminal) de la chaîne d'adaptation qui lie le serveur de contenu avec le client en passant par les éventuels processus d'adaptation et de négociation. La deuxième raison est l'absence de stratégie de négociation et de protocole qui définissent la façon dont les profils doivent être échangés et traités.

On ne peut apporter une solution efficace au problème de l'adaptation qu'en considérant tous les composants du système qui interviennent depuis l'émission de la requête cliente jusqu'à la réponse finale. C'est pourquoi nous avons suivi une approche globale pour l'identification de tous les éléments de l'adaptation et de leurs caractéristiques en commençant par le client lui-même et le contenu qu'il peut demander. La connaissance des caractéristiques des différents composants du système : client, serveur, méthodes d'adaptation, réseau, versions du contenu, etc. permet d'affiner le processus d'adaptation sur la base des contraintes posées. Cette connaissance permet de négocier d'une manière efficace sur la base de ce que le système d'adaptation est capable d'offrir, des capacités du terminal et des préférences de l'utilisateur.

Un modèle de description des caractéristiques de l'environnement est nécessaire, mais il n'est pas suffisant pour répondre aux requêtes des applications clientes. En effet, le modèle de description doit être accompagné d'une stratégie d'analyse des caractéristiques afin de trouver une correspondance entre les différentes dimensions du contexte. Cette stratégie doit être la plus complète possible, en considérant tous les scénarios possibles. Elle doit, par exemple, éviter le recours à l'application de méthodes d'adaptation dans le cas où le serveur possède une version du contenu qui correspond au contexte du client cible. Dans un environnement hétérogène, une stratégie d'analyse de description de l'environnement ne peut pas être effectuée localement puisqu'elle met en jeu plusieurs entités du réseau (le client et le serveur sont les entités de base). Il faut donc communiquer et échanger de l'information entre ces entités. Cet échange n'est pas relatif au transport des données, tel qu'il est fait dans les protocoles de communication classiques. L'objectif ici est d'enrichir la connaissance des entités concernées par l'adaptation, afin de transmettre finalement un contenu adapté. Mis à part la modeste prise en charge de la négociation par le protocole HTTP, aucun travail ne traite le protocole de négociation qui est, à notre sens, fondamental. Notre vision est que la solution du problème de négociation et d'adaptation de contenu, dans un environnement hétérogène, doit définir un ensemble de composants qui collaborent afin de trouver un consensus sur la meilleure adaptation à effectuer. Ces composants doivent inclure une prise en charge complète des différentes dimensions du contexte et

[1] Working draft depuis l'année 2000 et recommandation W3C depuis janvier 2004

de leurs changements ainsi que des techniques d'adaptation applicables à la structure et aux objets média du contenu.

Ces motivations nous ont conduit à aborder cette thèse selon trois axes complémentaires :

- L'identification des différents acteurs de la chaîne d'adaptation du contenu et la prise en charge de la description de leurs caractéristiques dans un modèle déclaratif et flexible.

- La définition d'une stratégie de gestion et d'analyse du contexte et de mise en correspondance de ses dimensions ainsi que la définition d'un protocole de négociation de contenu.

- La spécification et la mise en œuvre d'un système de techniques d'adaptation au sein d'une architecture complète appelée NAC (Negotiation and Adaptation Core).

Tout au long de cette thèse, nous avons tenté de mener une activité équilibrée entre la théorie et l'application. Nous pensons en effet qu'il est fondamental de confronter les propositions théoriques avec un contexte applicatif concret. Deux types de résultats sont donc visés par ce travail :

1. Des résultats théoriques, portant sur des schémas de description de l'environnement hétérogène, différentes organisations d'architecture, la modélisation du processus d'adaptation sous forme de graphe, des algorithmes de création et de gestion de profil, la formulation de la stratégie de satisfaction du contexte et l'adaptation des ressources.

2. Des résultats pratiques, sous la forme du prototype d'architecture NAC pour la négociation et l'adaptation de contenu qui inclut plusieurs composants implémentés tels que le proxy et la prise en compte du protocole HTTP, le module UCM ainsi qu'une diversité de méthodes de transformation de contenu et d'adaptation de ressources médias.

1.4 Plan de la thèse

Ce mémoire est organisé en deux parties. La première partie (le chapitre 2) représente un état de l'art et la seconde partie décrit notre contribution théorique ainsi que les implantations qui en découlent. Nous détaillons ci-dessous le contenu des chapitres.

Chapitre 2

L'objectif de ce chapitre est de familiariser le lecteur avec les systèmes d'adaptation et de négociation de contenu dans les environnements hétérogènes. Ce chapitre présente les principales stratégies et approches pour transmettre un contenu adapté au contexte cible. Le chapitre commence par introduire les notions de base de la négociation et de l'adaptation, puis il discute les problèmes posés et propose une

classification des stratégies de négociation. Dans la suite, le chapitre couvre une étude des travaux existants relatifs à la négociation et à l'adaptation de contenu en termes de protocoles et de modèles. Cette étude est complétée par une classification des systèmes d'adaptation ainsi que par des exemples de techniques utilisées.

Chapitre 3

L'objectif de ce chapitre est de présenter l'architecture NAC et de détailler les principales fonctionnalités des différentes entités qui la composent. Le chapitre explique les organisations possibles de l'architecture et détaille comment les entités de l'architecture coopèrent entre elles afin d'effectuer une adaptation contextuelle fondée sur les contraintes de l'environnement hétérogène.

Chapitre 4

Ce chapitre présente le système de description et de gestion des contextes de l'architecture NAC. Ce système permet une prise en charge des différentes contraintes imposées par l'environnement, par les terminaux et leurs utilisateurs. Le système définit un modèle de description accompagné d'un cadre d'adaptation automatique qui s'appuie sur les différentes informations du contexte. Le cadre d'adaptation permet d'appliquer une stratégie d'adaptation et de négociation de contenu basée sur le contexte et de transmettre finalement un contenu qui satisfait le contexte de l'application cible.

Chapitre 5

Ce chapitre présente les processus d'adaptation de l'architecture NAC ainsi que les stratégies et protocoles de négociation adoptés. Le système d'adaptation défini permet de considérer les différents aspects et fonctionnalités d'un contenu. Quant au protocole de négociation, il permet de prendre la meilleure décision d'adaptation possible qui prenne en compte les caractéristiques du contexte de l'environnement.

Chapitre 6

Ce chapitre présente en détail différentes techniques d'adaptation implémentées dans l'architecture NAC. Le système d'adaptation considère la transformation structurale des documents ainsi que l'adaptation des différentes ressources média. Afin de prendre en compte le changement du contexte, deux approches d'adaptation dynamique sont détaillées. Le chapitre inclut une formulation de l'adaptation des ressources ainsi que des évaluations des techniques d'adaptation et de gestion des profils.

Chapitre 7

Dans ce chapitre de conclusion, nous résumons l'apport de cette thèse. Nous proposons ensuite plusieurs perspectives dans ce nouveau domaine de recherche qu'est l'adaptation et la négociation de contenu.

Chapitre 2

Systèmes de négociation et d'adaptation de contenu

Résumé

L'objectif de ce chapitre est de familiariser le lecteur avec les notions d'adaptation et de négociation de contenu dans les environnements hétérogènes. Nous discutons les principales stratégies et approches définies pour transmettre un contenu adapté au contexte cible.

Contenu

2.1 Introduction

Grâce aux progrès technologiques récents, les utilisateurs disposent de nos jours d'une grande diversité de nouveaux moyens et plates-formes qui permettent l'accès à l'information n'importe où et n'importe quand. Les progrès au niveau des moyens d'accès sont accompagnés par une évolution particulièrement marquante du contenu. Comme les moyens d'accès à l'information sont très hétérogènes, les contenus des serveurs ne peuvent pas être envoyés de la même manière pour tous les client. Il y a donc un besoin d'*adaptation* de contenu afin de satisfaire les caractéristiques de chaque client cible. Ces caractéristiques incluent les capacités matérielles et logicielles du terminal utilisé ainsi que les préférences de l'utilisateur.

Afin d'appliquer correctement l'adaptation du contenu, il est nécessaire de faire une correspondance entre les contraintes des clients et les fonctionnalités du contenu demandé. Il est nécessaire aussi de savoir si le serveur d'origine détient une version du contenu qui soit conforme aux contraintes du client et si le contenu peut être adapté dans le cas contraire. Cette mise en correspondance et analyse du contexte est assurée par la tâche de *négociation de contenu*.

L'objectif de ce chapitre est de familiariser le lecteur avec les systèmes d'adaptation et de négociation de contenu dans les environnements hétérogènes. Dans ce chapitre, nous discutons les principales stratégies et approches adoptées pour transmettre un contenu adapté au contexte cible. Le chapitre commence par introduire les notions de base de la négociation et de l'adaptation du contenu dans les environnements hétérogènes suivies par une discussion des problématiques posées et une classification des stratégies de négociation. Dans la suite, le chapitre étudie les travaux existants relatifs au problème de négociation et d'adaptation de contenu en termes de protocoles et de modèles qui considèrent quelques aspects du problème. Cette étude est complétée par une classification des systèmes d'adaptation ainsi que par des exemples de techniques utilisées.

2.2 Négociation et adaptation de contenu

2.2.1 Définitions

Le problème majeur des environnements hétérogènes est la présence de nombreuses contraintes relatives aux différentes entités du système : le client, les fonctionnalités du contenu, le réseau de communication, les proxy, les serveurs du contenu, etc. *L'adaptation de contenu* est définie généralement comme le processus qui transforme un contenu de son état initial vers un état final afin de satisfaire un ensemble de contraintes. Le contenu final peut provenir du même contenu source ou faire usage d'autres ressources de contenu. Satisfaire une contrainte donnée, revient à changer l'état du contenu d'origine afin que l'usage du contenu, dans son état final, ne pose pas de problème en la présence de la contrainte. Si un contenu donné ne peut pas satisfaire une contrainte de l'environnement, cela peut causer un problème d'utilisation du contenu dans cet environnement. Ce problème peut être d'ordre *sémantique*, c'est-à-dire le contenu reçu par le client ne correspond pas à la sémantique du contenu original tel qu'il a été créé par l'auteur ; ou d'ordre *présentation*, c'est-à-dire le contenu visualisé par le client ne correspond pas à la présentation du contenu original.

Un cadre de travail d'adaptation de contenu regroupe de nombreuses entités et tâches. Un cadre de travail basique regroupe au moins :

- L'application cliente, c'est-à-dire l'application utilisée par le client pour émettre la requête d'accès au contenu, recevoir et traiter le contenu reçu. Le traitement du contenu implique la visualisation des différentes ressources transmises par le serveur et le formatage du contenu selon le dispositif d'affichage du terminal.

- Un système de communication, qui assure le moyen de dialogue et d'échange de messages entre les entités du réseau et en particulier entre le client et le serveur. Un système de communication inclut un protocole prédéfini qui organise l'échange des messages et leur donne une sémantique. Plusieurs types de protocole existent pour couvrir des besoins différents.

- Un serveur de contenu, qui est l'entité du réseau dont la tâche principale est de satisfaire les requêtes clientes reçues. Généralement, satisfaire une requête revient à envoyer le contenu demandé dans la requête.

- Un processus de négociation, qui est le composant du système adaptatif responsable d'appliquer une stratégie permettant de prendre la meilleure décision vis-à-vis des contraintes du contexte courant. Le processus de négociation peut décider que le contenu source respecte les contraintes, et peu donc être transmis sans modification. Dans le cas contraire, le processus de négociation échange des messages de négociation avec le client, et éventuellement avec d'autres composants du système, afin d'effectuer le meilleur effort pour satisfaire le contexte courant. Une fois la décision prise, le processus de négociation lance le processus d'adaptation du système.

- Un processus d'adaptation, qui est le composant du système d'adaptation chargé d'appliquer réellement les différentes techniques d'adaptation. Le processus prend une ressource ou un contenu composite en entrée et génère un nouveau contenu en sortie. La génération du contenu peut être effectuée par transformation, choix de version, filtrage, etc.

- Un protocole de négociation et d'adaptation, c'est-à-dire un dialogue prédéfini et connu par toutes les entités participantes permettant d'atteindre l'objectif de l'adaptation. En effet, les participants du protocole d'adaptation, en particulier le client et le serveur, peuvent négocier en s'échangeant quelques messages d'information afin de guider le processus d'adaptation en effectuant une négociation sur la base des caractéristiques de l'environnement. Dans certains cas, les fonctionnalités du protocole de négociation et d'adaptation peuvent être incluses dans le protocole de communication, comme dans le cas dans HTTP [67] que nous allons étudié.

- Des techniques d'adaptation, matérialisées par des processus prêts à être appliqués sur un type de ressources média bien défini. Une technique d'adaptation peut être appliquée sur la structure du contenu, sur l'encodage d'une ressource, sur la transmission d'un flux etc.

Dans la suite de ce chapitre, après avoir exposé les problématiques liées à la

négociation et l'adaptation du contenu, nous discutons l'état de l'art concernant les stratégies et les architectures de négociation et d'adaptation du contenu. Nous verrons comment l'application de la négociation peut suivre plusieurs stratégies et comment l'adaptation du contenu peut être mise en œuvre en adoptant plusieurs façons d'organiser le système et les différents composants du réseau.

2.2.2 Problématiques

De nombreux efforts ont été faits dans le domaine des systèmes adaptatifs du contenu. Bien que ces efforts ne proposent pas de solutions complètes qui prennent en compte toutes les difficultés du problème, quelques travaux se sont focalisés sur certains aspects particuliers et ont essayé de proposer quelques techniques permettant d'améliorer les architectures existantes. Attaquer le problème de négociation et d'adaptation du contenu sur un seul aspect mène à des lacunes dans les solutions proposées. Souvent, ces solutions sont incompatibles et ne peuvent pas être intégrées dans un seul système, ce qui a pour conséquence que les nouvelles architectures restent hétérogènes et incomplètes.

Le problème de négociation et d'adaptation couvre plusieurs aspects. Le point de départ est que les systèmes multimédia qui existent actuellement sont de plus en plus hétérogènes et regroupent une grande diversité de terminaux, de serveurs, de méthodes d'accès, de formats et de fonctionnalités de contenu, etc. Ce problème est apparu avec l'évolution rapide qu'a connu l'informatique dans les technologies des supports physiques à partir desquels l'information est consultée. Désormais on peut accéder aux informations boursières sur son téléphone cellulaire, effectuer des transactions avec un assistant personnel, etc. En parallèle, le contenu de l'information a aussi évolué. Beaucoup de nouvelles fonctionnalités ont été intégrées avec de nouvelles techniques d'encodage avancé. On trouve aujourd'hui du contenu sous forme de dessins vectoriels, animations et vidéos complexes, scènes tridimensionnelles, etc.

La conception et l'implémentation d'un système de contenu qui assure le service de la négociation et de l'adaptation impliquent des efforts dans plusieurs domaines des systèmes informatiques. Les efforts de recherche et de l'implémentation dans le domaine multimédia se font généralement simultanément [98]. Nous listons ci-dessous certains aspects importants pour la conception d'une solution d'adaptation et de négociation du contenu :

1. La diversité des applications clientes : L'environnement hétérogène peut inclure des terminaux qui n'ont pas les mêmes caractéristiques. Cela veut dire que le contenu n'est pas traité de la même manière par tous les clients et les formats acceptés sont différents. En outre, même les méthodes d'accès peuvent être hétérogènes ce qui implique que le protocole de communication peut varier d'une application cliente à une autre. Les caractéristiques matérielles et logicielles des applications clientes imposent de nouvelles contraintes aux serveurs afin que leur contenu soit compréhensible et utilisable.

2. La diversité du contenu : Le contenu utilisée dans un système adaptatif peut être très hétérogène. Les serveurs de contenu peuvent transmettre une grande diversité de formats et des documents riches en fonctionnalités. Un contenu complexe peut être demandé par un terminal qui présente des limitations de traitements,

d'affichage, etc. Afin d'assurer une bonne qualité de service, le système d'adaptation doit être capable de transmettre et de transformer ce contenu complexe pour qu'il soit compatible avec les capacités et les préférences de l'utilisateur final.

3. La description de l'environnement : La définition d'un modèle de description du système est nécessaire pour décrire l'environnement hétérogène et le contexte courant des clients lors de la réception d'une requête. La description du contexte est importante car c'est elle qui guide les processus d'adaptation et de négociation du contenu. Le modèle de description peut être vu comme une forme de représentation des contraintes de l'environnement. A l'aide de cette forme de représentation, le système pourra appliquer les mécanismes nécessaires pour transmettre un contenu adapté aux contraintes présentes.

4. La gestion du contexte : La négociation du contenu pose le problème de gestion des descriptions des différents composants du système. Le problème est de gérer toutes ces descriptions des clients, du contenu, du réseau, etc. afin de prendre la meilleure décision qui permette à la fin de transmettre un contenu bien adapté aux conditions courantes de l'environnement. La difficulté du problème vient du fait qu'il n'y a pas une seule source de contraintes. En effet, chaque composant du réseau (client, réseau, serveur, etc.) peut poser ses propres contraintes et c'est au gestionnaire du contexte de trouver un compromis entre tous ces composants. La gestion du contexte inclut aussi la transmission et l'extraction du contexte. Si le protocole de communication utilisé pour transmettre le contenu ne dispose pas de directives avancées pour la gestion du contexte, le système doit définir son propre protocole d'extraction et d'échanges de messages afin d'assurer ce service. Il est possible que le système dédie un composant à part qui assure la transmission des contextes sans alourdir les composants de base du réseau, les clients et les serveurs (voir Chapitre 5).

5. La gestion des versions : Dans de nombreuses situations, les serveurs de contenu disposent de plusieurs variantes du même contenu. Ces variantes sont généralement créées par l'auteur du contenu afin de les transmettre dans plusieurs contextes différents. Les variantes (ou versions) d'une même ressource peuvent exister sur le même support sans aucune organisation explicite de la part de l'auteur. Le système de gestion de variantes doit transmettre la variante la plus appropriée dans un contexte donné. Cela implique que le système doit être en mesure de connaître les caractéristiques des différentes ressources et de leurs variantes afin de faire la correspondance nécessaire entre les contraintes du contexte et la base des variantes.

6. L'adaptation : Une solution d'adaptation et de négociation de contenu ne doit pas se satisfaire de la gestion des variantes. Il est possible que l'auteur ne fournisse qu'une version pour un contenu source, ou que les versions existantes ne satisfassent pas les contraintes d'un contexte donné. Il est donc important de résoudre le problème de l'adaptation du contenu source à un contexte imposé par l'environnement. Une fois le contenu adapté, le terminal cible pourra l'utiliser directement sans risque d'erreurs. Résoudre le problème de l'adaptation évite que l'architecture du système soit basée sur la multi-édition du contenu qui nécessite un effort énorme de la part des auteurs.

7. Les performances du système : L'ajout de nouveaux processus dans une architecture existante peuvent dégrader les performances du système. Les mécanismes et les composants d'adaptation et de négociation doivent être intégrés de telle sorte qu'ils assurent une bonne qualité de service. Les modèles de description du contexte ainsi que les protocoles d'échange de messages entres les composants du réseau ont un impact sur les performances. L'objectif est d'assurer la transmission d'un contenu adapté en évitant de dégrader les performances du système.

Pour résoudre les problèmes liés à l'adaptation et la négociation du contenu, il faut une architecture complète qui identifie le rôle de chaque composant et définit la manière dont ces composants coopérent.

2.2.3 Classification des stratégies de négociation

Suivant l'invocation du processus d'adaptation, nous distinguons deux types de négociaions :

1. **La négociation à la demande** : Dans ce type de négociation, le processus d'adaptation est invoqué lors de la réception de la requête de l'application cliente. Les résultats de l'adaptation sont directement transmis au client cible. Dans ce cas, l'ensemble des profils document qui peuvent exister au niveau serveur ne subissent pas de changement. On dit alors que l'ensemble des profils (noté D_P, voir Chapitre 5) est stable.
2. **La négociation fondée sur des caches** : au contraire du premier type de négociation, cette négociation se fonde sur l'utilisation des versions du contenu source préalablement sauvegardées. Cela veut dire qu'à la réception de la requête de l'application cliente, le serveur n'invoque pas directement de tâches d'adaptation. Le serveur (ou l'entité intermédiaire) cherche d'abord des versions existantes [1]. Si la version recherchée n'existe pas le serveur invoque le processus d'adaptation en utilisant la meilleure transformation disponible dans le système. Le contenu source est donc adapté et le contenu généré est stocké pour une utilisation ultérieure. Dans ce cas, l'ensemble des profils s'enrichit par les nouveaux profils qui correspondent au contenu stocké.

Le deuxième type de négociation est préférable si l'utilisation d'un même contenu est fréquente, par un seul client ou par un groupe particulier de clients[2]. En effet, dans ce cas, l'utilisation du cache évite de lancer les mêmes méthodes d'adaptation plusieurs fois, ce qui améliore les performances du système. Cependant, si le contexte du client change fréquemment, l'application du premier type de négociation est plus efficace car il est inutile de stocker des contenus, générés par adaptation, qui seront rarement utilisés.

Une stratégie de négociation qui combine les deux approches peut être définie en se basant sur :

- La trace et la fréquence des requêtes des clients, et
- Le changement du contexte client

[1] le contenu source est aussi considéré comme une version

[2] les statistiques d'utilisation du contenu utilisent généralement des compteurs d'accès [64]

D'autres techniques peuvent être exploitées dans une approche de négociation, telles que les délais de garde [132] pour la sauvegarde des profils, des techniques avancées de cache du contenu [136].

2.3 Stratégies de négociation et d'adaptation

Dans un système hétérogène, la stratégie de négociation de contenu représente l'outil avec lequel le client peut exprimer le type[3] d'information qu'il peut accepter, et le serveur décide de l'information à retourner. Par exemple, la négociation peut être utilisée pour sélectionner la langue (français, anglais, etc.) d'un document, ou pour choisir le format d'une ressource média (JPEG, WAVE, etc.) que l'application cliente peut visualiser.

Le cadre de travail d'une stratégie de négociation nécessite des moyens pour la description des méta-données, des attributs et des préférences de l'utilisateur et de ses applications clientes, des caractéristiques du contenu et les règles d'adaptation du contenu vis-à-vis des capacités et préférences de l'utilisateur.

Une stratégie de négociation doit principalement couvrir trois aspects : l'expression des capacités du serveur de contenu et des ressources à transmettre, l'expression des capacités du client (i.e. le récepteur du contenu), et le protocole avec lequel les capacités et les autres informations relatives à la négociation sont transmises. Ces différents aspects ont été adressés par plusieurs approches et protocoles. Dans ce qui suit, nous essayons de couvrir quelques stratégies existantes.

2.3.1 Négociation fondée sur le type Mime (Mime Type)

Le type MIME (Multipurpose Internet Mail Extensions) est un standard qui a été proposé par les laboratoires Bell Communications en 1991 afin d'étendre les possibilités du courrier électronique (Internet E-mail). En effet, à l'origine le courier électronique était définit uniquement pour l'Ascii par le RFC 822 [27]. Les RFC 2045 [37], 2046 [38], 2047 [85], 2048 [39] et 2049 [40] ont défini un ensemble d'extensions connu sous le nom MIME dans le but de pouvoir échanger des messages de formats hybrides. Cela permet donc d'insérer dans un message d'autres types de ressources (images, texte, audio, etc.). Lors de la reception du message les différentes ressources sont reconnues et traitées automatiquement. Le protocole SMTP (Simple Mail Transfer Protocol) [94], protocole de communication utilisé pour l'envoi de courrier, a donc été enrichi de nouvelles commandes qui permettent au client de spécifier par exemple le type, la longueur et l'encodage du contenu qu'il envoie. A l'aide de ces informations, le récepteur du contenu reconnaît les types des ressources et peut par conséquent les associer à des applications capables de les traiter.

Depuis 1991, le type MIME est exploité d'une part pour typer les ressources attachées à un courrier électronique mais aussi pour typer les ressources transférées sur le Web par le protocole HTTP. En effet, durant une session client/serveur, le serveur peut envoyer le type MIME de la ressource à l'application cliente afin que cette dernière puisse la traiter correctement. Les commandes MIME sont normalisées et font intégralement partie du protocole HTTP [7] [35].

[3] le type mais aussi d'autres aspects de l'information

Le principe de déclaration des types MIME s'appuie sur l'utilisation du champ '*Content-Type*'. Ce champ indique la nature des média transmis à l'application cliente. Le format général du champ est donné comme suit :

```
Content-Type   = "Content-Type" ":" media-type
```

Le champ media-type [95] fournit un cadre ouvert et extensible de typage des données et une négociation de types. Le format du champ media-type est donné comme suit :

```
media-type     = type "/" subtype *( ";" parameter )
type           = token
subtype        = token
```

Les paramètres peuvent suivre le format type/sous-type sous forme de paires attribue/valeur. La présence ou l'absence des paramètres peut être significative dans le traitement des types de média. Cela dépend de la sémantique des paramètres vis à vis de la définition du type de média. De nombreuses applications clientes HTTP ignorent le paramètre du type de média. La spécification du protocole HTTP 1.1 [35] souligne que les applications clientes doivent tenir compte de ces paramètres seulement s'ils sont nécessaires pour la définition du couple type/sous-type. Voici un exemple d'utilisation du champ Content-Type :

```
Content-Type: text/plain; charset=ISO-8859-15
```

Les valeurs des types de média ont été normalisées et associées à des identificateurs [100] en suivant la procédure de normalisation définie par [95].

Le type d'une ressource média est défini par l'association d'un type général (texte, image, audio, application, etc.) et d'un sous-type donnant plus de précision sur le format exacte de la ressource. Par exemple, si on considère une ressource média de type image, le sous-type indiquera le format de l'image (jpeg, gif, wbmp, etc.). Lorsqu'un fournisseur de contenu envoie une ressource image de type gif, le type MIME est déclaré automatiquement (en se basant sur la configuration du serveur). Dans ce cas, la déclaration est écrite comme suit :

```
Content-type: image/gif
```

Le Tableau 2.1 liste quelques exemples de types MIME associés à des ressources média. Un client et un serveur peuvent participer à une négociation de contenu basée sur les types MIME. Les types des médias servent à préciser ce qu'une application particulière peut supporter et ce qu'un serveur peut fournir. Une fois que les deux parties atteignent un consensus vis à vis du type de média à utiliser, la requête de l'application cliente est satisfaite (voir Figure 2.1). Malheureusement, peu de serveurs (exemple les serveurs Apache et W30) et d'applications clientes appliquent une négociation poussée qui considère complètement les types MIME.

GStreamer [46] représente un exemple de plate-forme qui applique une stratégie de négociation basée sur les types MIME. GStreamer est une infrastructure multimédia logicielle qui permet aux applications de partager les capacités communes de lecture, encodage, montage et modifications de tout genre d'audio et de vidéo.

La plate-forme utilise une gestion de types de média pour améliorer la négociation des caractéristiques des ressources utilisées par les différents composants du système.

FIG. 2.1 – Principe des types MIME

TAB. 2.1 – Exemples de types MIME

Type MIME	Type de la ressource
text/html	ressources HTML
application/x-latex	ressources LaTeX
audio/basic	audio basiques
audio/wav	audio de type wav
image/jpeg	images Jpeg
image/svg+xml	ressources svg
video/mpeg	vidéos MPEG
image/vnd.wap.wbmp	images Wbmp

Cette approche demande une organisation complexe des composants afin de supporter n'importe quel type de média dans le cas où le système dispose d'un composant qui peut le traiter. L'objectif est d'assurer une organisation dynamique sans la connaissance préalable de l'application.

L'infrastructure utilise deux types de composants : les composants *Elément* et les composants *Pad*. Un *élément* a une fonction unique et spécifique. Cette fonction peut être la lecture d'un fichier, le décodage d'une ressource audio ou la transmission d'un flux audio à une carte son. Un *Pad* représente la manière dont une ressource média est transmise en entrée ou en sortie. Un fichier source, par exemple, doit avoir un *Pad* à travers lequel ses données sont transmises à l'élément suivant liée à ce *Pad*.

Lorsque deux *Pads* sont connectés [4], le système applique une stratégie de négociation des types de données. Les capacités des *Pads* représentent les types de média qui peuvent être traités. Ces capacités sont déclarées à l'aide d'une structure appelée GstCaps (voir la Figure 2.2). La structure contient un type MIME et une liste arbitraire de paires clé/valeur. Les valeurs peuvent être uniques (une valeur entière par exemple) ou multiples (un intervalle, une liste de valeurs, etc.). Pour un *Pad* donné, ces capacités sont uniques et ne subissent pas de changement.

Plusieurs structures de capacités peuvent être enchaînées entre elles afin de former une description détaillée des capacités. Le système considère les capacités comme un ensemble de contraintes sous forme de types MIME et de propriétés posés sur le type de média. Par définition, une structure GstCaps nulle signifie l'absence de contraintes.

[4] le premier *Pad* doit être de type *source* et le deuxième de type *cible*

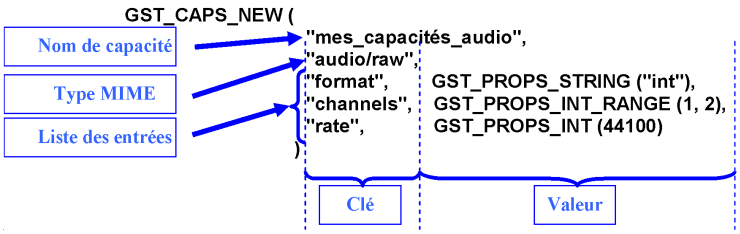

FIG. 2.2 – Déclaration des capacités d'un *Pad*

Les capacités d'un *Pad* définissent donc ce que le *Pad* est capable de faire. L'objectif du processus de négociation de type est de trouver une structure de capacités fixes qui définit un type de média spécifique.

Lors de la connexion des *Pads*, une première vérification est effectuée afin d'assurer la cohérence des types de données. Ceci est fait par une vérification des types MIME des *Pads* et la création d'un ensemble commun de paires clé/valeur. Si les types MIME sont différents ou s'il n'y a pas de paires clé/valeur communes, la connexion est impossible ; elle est donc refusée.

Après la première inspection, les deux *Pads* sont informés des capacités du couple déterminées par une *fonction de connexion* du système. Dans cette fonction, le processus de traitement d'un média inspecte les paires clé/valeur et se configure pour manipuler le média. La plupart des processus des médias ne se configurent pas pour un type bien déterminé s'ils ne reçoivent pas de capacités uniques. L'ensemble commun des paires clé/valeur est obtenu en calculant l'intersection des deux structures de capacités des deux Pads (voir la Figure 2.3).

Exemple : La Figure 2.3 montre un exemple de connexions de deux *Pads*. Les deux *Pads* à connecter possèdent chacun un ensemble de types média possibles. Pour le *Pad* source, les types sont A, B et C. Pour le *Pad* cible, les types sont B, F, A et G. Lors de l'opération de connexion, l'intersection entre les deux ensembles de capacités est calculée (voir la Figure 2.4).

La fonction de connexion des deux *Pads* est donc appelée. Cette fonction peut accepter, retarder ou refuser la connexion. L'intersection dans le cas de notre exemple est donc un ensemble de capacités. Dans ce cas là, le *Pad* source peut choisir entre les deux types MIME (A ou B) et appliquer une fonction prédéfinie du système pour mettre à jour ses capacités finales. Une fois le choix fait, les capacités sont mises à jour et tous les flux médias passants par ce *Pad* appartiendront au type choisi.

Après une connexion de *Pads*, il n'est pas nécessaire que les processus de traitement des médias décident le type MIME à traiter. Ils peuvent retarder la négociation jusqu'à ce qu'un flux média se présente en entrée du *Pad*. Par exemple un processus de décodage MP3 connecté à un composant qui permet de jouer de l'audio, ne peut pas informer le composant du type de données qui pourra être fourni, jusqu'à ce qu'il reçoive le premier échantillon MP3 et extrait quelques informations relatives au type du média (nombre de canaux audio, vitesse, etc.). La plate-forme fournit aussi des

***Pad* source**	***Pad* cible**
A	B
B	F
C	A
	G

FIG. 2.3 – Négociation pour la connexion de deux *Pads*

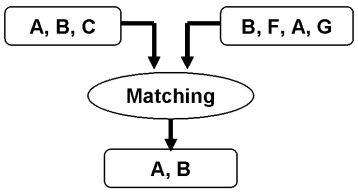

FIG. 2.4 – Négociation des types MIME

interfaces pour faire des requêtes sur les capacités des paires de *Pads*. Cela permet de choisir le meilleur type de média qui peut être traité par une paire du système. Par exemple, un processus de décodage de vidéo, qui peut fournir des images de différentes gammes de couleurs, peut d'abord effectuer des requêtes sur les capacités d'une paire de *Pads* afin de sélectionner le format le plus approprié, et ensuite informer la paire du type de données choisi.

Beaucoup de systèmes et de protocoles de communication et de transmission de données nécessitent d'assurer une stratégie de négociation vis à vis des ressources médias qu'ils manipulent. Les types MIME assurent une méthode normalisée pour l'expression de plusieurs activités de négociation. Cependant, les systèmes hétérogènes actuels comporte une large diversité ressources média qu'on ne peut pas toujours décrire à l'aide des types MIME définis actuellement.

2.3.2 Négociation du protocole HTTP 1.0

Le protocole HTTP (*Hypertext Transfer Protocol*) est un protocole de niveau applicatif destiné aux systèmes de donnés distribués, coopératifs et hypermédias. Le protocole HTTP/1.0 est issu de sa version originale 0.9 [7] qui est rarement utilisée. La version 0.9 définit une seule méthode de communication (la méthode GET) et pose des restrictions sur le type du contenu. En outre, aucune information sur le contenu transmis n'est envoyée et les applications clientes ne peuvent pas soumettre des informations aux serveurs. Le protocole HTTP/1.0 est défini dans le RFC 1945 [7] par le groupe de travail HTTP (HTTP-WG) de l'IETF (Internet Engineering Task Force). Dans [7], une description de l'usage commun du protocole est détaillée. Cependant, le document ne représente pas un standard formel. HTTP/1.0 fournit quelques mécanismes simples pour supporter la négociation de contenu. La Figure 2.5 montre un exemple de requête HTTP/1.0 d'un Pocket PC. Dans cette section, nous discutons la stratégie et le schéma de la négociation adoptés par ce protocole.

```
GET http://192.168.0.1/Negotiation/ContentNegotiationTest.html HTTP/1.0
Accept: */*
UA-OS: Windows CE (POCKET PC) - Version 3.0
UA-color: color16
UA-pixels: 240x320
UA-CPU: ARM SA1110
UA-Language: JavaScript
Accept-Encoding: gzip, deflate
If-Modified-Since: Thu, 10 Oct 2002 16:22:02 GMT
If-None-Match: "5523-1034266922000"
User-Agent: Mozilla/2.0 (compatible; MSIE 3.02; Windows CE;
240x320)
Host: 192.168.0.1
Proxy-Connection: Keep-Alive
```

FIG. 2.5 – Exemple de requête HTTP d'un Pocket PC

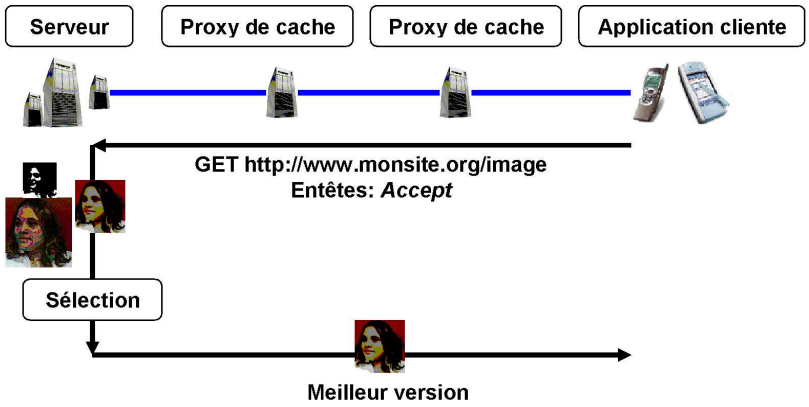

FIG. 2.6 – Modèle de négociation du protocole HTTP/1.0

2.3.2.1 Stratégie de négociation

Le protocole HTTP/1.0 suit le modèle de négociation de contenu présenté par la Figure 2.6. Lorsqu'une application cliente envoie une requête pour une ressource qui existe sur le serveur, les entêtes de type *Accept* (incluses dans la requête) expriment les capacités de l'application et les préférences de l'utilisateur. Le serveur utilise ces entêtes pour choisir la meilleure variante disponible du contenu source. La variante choisie est envoyée comme réponse à la requête.

Le champ d'une entête de type *Accept* est utilisé pour indiquer la liste des médias qui sont acceptables dans la réponse de la requête. Le caractère '*' est utilisé pour rassembler les types de média. Par exemple, l'expression '*/*' signifie que tous les types de ressources sont acceptés, l'expression 'type/*' signifie que tous les sous-types de 'type' sont acceptés. L'expression des types médias est toujours liée au contexte de la requête envoyée.

Exemples :

```
GET /paper HTTP/1.0 User-Agent: WuxtaWeb/1.0 (Linux/X11)
Accept-Language: en, fr Accept: text/html;q=1.0, text/plain;q=0.5,
*/*:q=0.001 audio/x-aiff;q=0, audio/x-wav;q=0.3,
video/quicktime;q=0.4, video/x-msvideo;q=0.5,
video/x-sgi-movie;q=0.7, image/gif;q=0.8, image/ief;q=0.9,
image/jpeg;q=0.9, image/tiff;q=0.6, image/x-cmu-raster;q=0.5,
image/x-portable-anymap;q=0.6, image/x-portable-bitmap;q=0.9,
image/x-portable-graymap;q=0.7, image/x-portable-pixmap;q=0.9,
image/x-rgb;q=0.8, image/x-xbitmap;q=0.5, image/x-xpixmap;q=0.4,
image/x-xwindowdump;q=0.5, message/external-body;q=0.9,
...
text/richtext;q=0.6, text/tab-separated-values;q=0.4,
text/x-setext;q=0.9, video/mpeg;q=0.5,
...
application/activemessage;q=0.5, application/andrew-inset;q=0.6,
application/applefile;q=0.7, application/atomicmail;q=0.7,
application/dca-rft;q=0.7, application/dec-dx;q=0.9,
application/mac-binhex40;q=0.9, application/macwriteii;q=0.6,
application/msword;q=0.5, application/news-message-id;q=0.5,
...
message/partial;q=0.7, message/rfc822;q=0.5,
multipart/alternative;q=0.8, multipart/appledouble;q=0.3,
multipart/digest;q=0.5, multipart/mixed;q=0.9,
```

FIG. 2.7 – Problème de la négociation dans HTTP/1.0

Accept-Charset Ce champ est utilisé pour indiquer la liste des jeux de caractère préférés. Par défaut la valeur est US-ASCII et ISO-8859-1. Ce champ permet aux clients qui sont capables de comprendre des ensembles particuliers de caractères de l'indiquer aux serveurs afin qu'ils représentent les documents avec ces ensembles de caractères.

Accept-Encoding Ce champ est utilisé pour restreindre les encodages du contenu selon les capacité de l'application cliente. L'encodage effectif est donné par l'entête *Content-Encoding* du protocole [7].

Accept-Language Ce champ est utilisé pour indiquer l'ensemble des langues naturelles qui sont préférées dans la réponse de la requête cliente.

Le problème majeur du modèle de négociation du protocole HTTP/1.0 est qu'il représente un modèle non flexible. En effet, l'expression de toutes les capacités et préférences effectuée dans les entêtes *Accept* peut être très volumineuse (voir la Figure 2.7). En outre, l'envoi de ces expressions volumineuses dans chaque requête pose un sérieux problème de performance, et n'importe aucun bénéfice si le serveur ne dispose pas de variantes du contenu source ou si le système d'adaptation s'appuie sur la transformation et la génération dynamique du contenu adapté.

2.3.2.2 Négociation et utilisation du cache

Un système de cache est un système qui consiste à sauvegarder les réponses aux requêtes (ou aux interactions de l'utilisateur) et à assurer la gestion des sauvegardes,

leur interrogation et leur suppression. Généralement, les systèmes de cache sont utilisés afin d'assurer une utilisation optimale des ressources de l'environnement, par exemple pour réduire le temps de réponse, la bande passante, etc. Dans un réseau informatique, n'importe quel entité (client, serveur ou autre) peut être amenée à appliquer une gestion de cache.

L'exploitation d'une mémoire de cache revient à trouver une information déjà sauvegardée en se basant sur une valeur de clé et en appliquant un algorithme de recherche. Le protocole HTTP/1.0 [7] utilise une gestion simple de cache. Il considère comme clé, l'URL de la requête de l'application cliente. Ce mécanisme va dans le sens contraire du principe de la négociation du contenu. En effet, la réponse à une requête peut dépendre de paramètres autres que l'URL, tels que les entêtes *Accept-Language*, *Accept-Charset*, etc.

La gestion du cache du protocole HTTP/1.0 n'est pas très avancée ; le client et le serveur ne disposent pas de moyens pour envoyer des instructions explicites au cache. La gestion du cache est ouverte à un ensemble d'heuristiques qui ne sont pas spécifiées dans la définition du protocole [7]. Par conséquent, deux problèmes se posent : un problème sémantique dû à l'application incorrecte du cache sur des réponses qui ne doivent pas être sauvegardées telles quelles ; et un problème de performance dû à la non sauvegarde des réponses qui doivent être cachées [67].

2.3.2.3 Optimisation des ressources réseau

Une des caractéristiques des environnements répartis est la variabilité de la bande passante du réseau. Souvent les terminaux mobiles disposent d'une bande passante limitée et non stable. Dans ce genre de situation, l'utilisation non optimale des ressources réseau augmente le temps de réponse des requêtes clientes et cause une latence de transmission. Le protocole HTTP/1.0 ne dispose pas de moyen efficace pour assurer une gestion fiable de la bande passante.

Par exemple, lorsque le client envoie une requête qui vise un contenu très large sachant qu'il ne s'intéresse qu'à une partie du contenu, le protocole ne fournit aucun moyen de transmettre des parties du contenu. D'un autre côté, le protocole peut causer une consommation inutile des ressources réseau, par exemple dans le cas où le serveur est incapable de traiter certains types de requêtes. Dans ce cas, le serveur ne transmet une réponse négative qu'après la réception de la requête, et des ressources réseaux sont utilisées inutilement surtout si la requête est volumineuse.

Cette utilisation non optimale est dûe à l'absence de négociation entre le client et le serveur sur le type de requêtes qui peuvent être traitées avant que ces dernières ne soient envoyées.

2.3.3 Négociation du protocole HTTP 1.1

Le protocole HTTP/1.1 a été proposé pour pallier les inconvénients du protocole HTTP/1.0 et faire face à ses limitations telles que l'organisation hiérarchique des proxy, la gestion du cache, les connexions persistantes, la négociation du contenu, etc. La version 1.1 du protocole HTTP résout également le problème de compatibilité des implémentations qui déclarent supporter HTTP/1.0. En effet, la spécification du protocole [35] exprime strictement les conditions pour que les implémentations soient

réellement conformes HTTP/1.1.

Le protocole HTTP/1.1 définit le processus de négociation de contenu, comme un mécanisme de sélection de la représentation la plus appropriée en réponse à une requête cliente. La représentation des ressources peut être négociée dans n'importe quelle réponse du serveur. Dans cette section nous discutons en détail le modèle de la négociation du protocole HTTP/1.1 et les différentes stratégies qu'il propose.

2.3.3.1 Stratégies de négociation

Le protocole HTTP/1.1 permet la gestion des différentes versions (ou variantes) du même contenu sous une unique URI. Le processus de négociation du contenu de HTTP/1.1 consiste à effectuer une sélection de la meilleure variante en fonction des caractéristiques du client cible. La sélection est basée sur la recherche de correspondances entre les propriétés des variantes disponibles et les capacités de l'application cliente et les préférences de l'utilisateur d'autre part. Ce principe est le même que dans l'ancienne version du protocole ; cependant HTTP/1.1 définit de nouvelles stratégies et considère la négociation du côté des entités de cache.

Comme nous avons déjà vu (voir Section 2.3.2.1), le modèle de négociation du protocole HTTP/1.0 peut causer des transmissions volumineuses et inutiles concernant l'expression des préférences et des capacités. HTTP/1.1 propose un modèle appelé *négociation transparente du contenu* [53]. Ce modèle évite l'envoi des entêtes de type *Accept* de grande taille qui sont inutiles dans le cas où le serveur ne dispose pas de versions multiples, ou lorsque le client est incapable de traiter les versions disponibles.

En utilisant le principe de la négociation transparente du contenu, le serveur envoie à l'application cliente une liste des variantes disponibles avec leurs propriétés (voir Figure 2.8). La description d'une variante consiste en un ensemble d'attributs et de valeurs. Chaque attribut peut apparaître une seule fois dans une description.

Exemple : Une liste de variantes concernant une ressource 'document' peut être donnée comme suit :

```
{"document.1" 0.7 {type text/html} {language de}},
{"document.2" 1.0 {type text/html} {language fr}},
{"document.3" 0.8 {type application/pdf} {language en}}
```

Lorsqu'une liste est reçue, l'application cliente peut choisir la meilleure variante et par la suite la récupérer. Le modèle de communication entre l'application cliente et le serveur de contenu est représenté par la Figure 2.8. La première réponse du serveur est donc sous forme d'une liste de versions. La seconde réponse est une réponse classique qui transporte la ressource sélectionnée. Cette seconde réponse ne comporte aucune information relative à la négociation du contenu.

Dans ce modèle de négociation, qui est donc différent du modèle HTTP/1.0, la description des capacités et des préférences est uniquement utilisée par l'application cliente elle-même. Par conséquent, l'envoi de plusieurs entêtes *Accept* est évité. Notons que l'envoi de quelques entêtes *Accept* peut aider à l'accélération du processus de

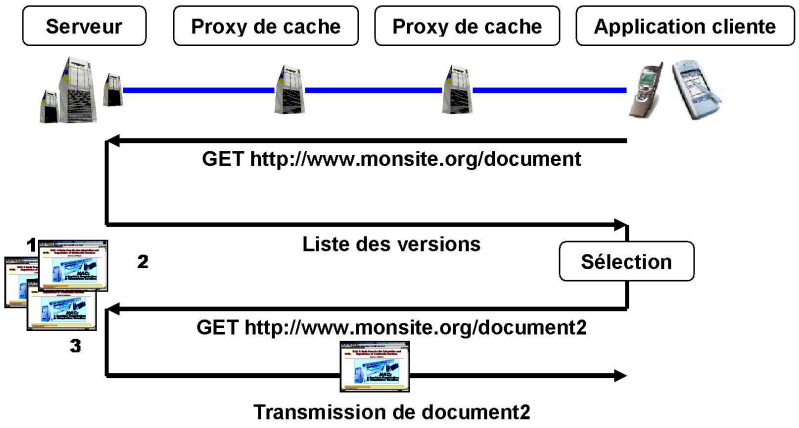

FIG. 2.8 – Modèle de la négociation transparente du contenu

négociation comme nous allons voir par la suite.

Un exemple de réponse HTTP/1.1 peut être donné comme suit. Cette réponse correspond à la liste des versions donnée plus haut :

```
HTTP/1.1 300 Multiple Choices
Date: Wed, 09 Jul 2003 00:00:01 GMT
TCN: list
Alternates: {"document.1" 0.7 {type text/html} {language de}},
            {"document.2" 1.0 {type text/html} {language fr}},
            {"document.3" 0.9 {type application/pdf} {language en}}
Vary: negotiate, accept, accept-language
ETag: "blah;1234"
Cache-control: max-age=99900
Content-Type: text/html
Content-Length: 175
<h2>Choix multiples :</h2>
<ul>
<li><a href=document.1>HTML - Version en Allemand</a>
<li><a href=document.2>HTML - Version en Francais</a>
<li><a href=document.3>PDF - Version en Anglais</a>
</ul>
```

L'entête *Alternates* dans la réponse du serveur comporte la liste des variantes. L'entête *Vary* est utilisée pour assurer un traitement correct en cas de l'utilisation d'un cache (voir Section 2.3.3.4). L'entête *ETag* permet de revalider la réponse par les autres caches, cette revalidation est contrôlée par l'entête *Cache-Control*. La portion HTML incluse dans la réponse permet à l'utilisateur de sélectionner la meilleure variante.

2.3.3.1.1 Optimisation du processus de négociation Comme nous avons vu, le modèle de négociation précédent implique deux transactions, pour la récupération de la liste des variantes et pour la récupération de la ressource sélectionnée. Dans la négociation du protocole HTTP/1.1, d'autres scénarios de communication sont possibles pour optimiser le scénario principal présenté par la Figure 2.8. Ces optimisations peuvent réduire l'utilisation des ressources réseau, telles que la bande passante,

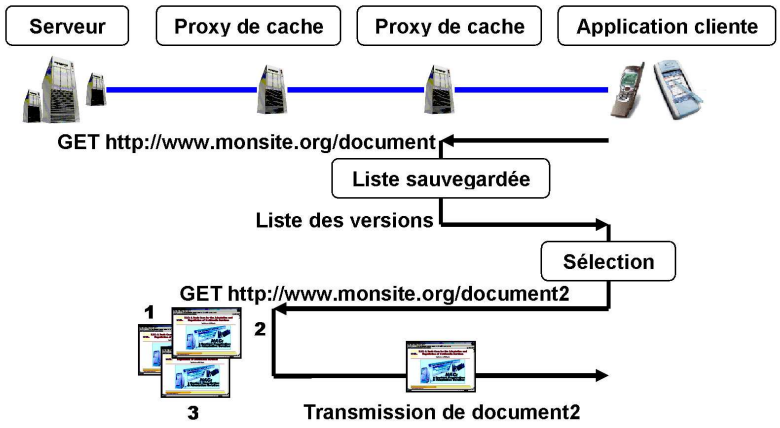

FIG. 2.9 – Premier cas d'optimisation de la négociation

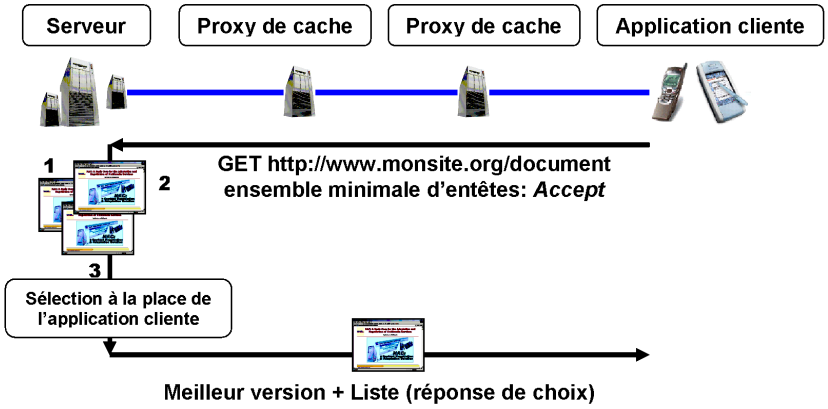

FIG. 2.10 – Deuxième cas d'optimisation de la négociation

minimiser le temps de réponse de la communication client/serveur et réduire la charge du serveur d'origine.

Trois cas d'optimisation ont été identifiés. Dans le premier cas, les proxy de cache peuvent sauvegarder les variantes et les listes des variantes. Cette sauvegarde peut réduire la surcharge de communication comme la montre l'exemple de la Figure 2.9.

Dans le deuxième cas d'optimisation, l'application cliente peut envoyer un petit ensemble d'entêtes de type *Accept*, suffisant pour que le serveur puisse sélectionner la meilleure variante et la renvoyer directement au client.

Ce type de sélection de variantes, basé sur un ensemble réduit d'entêtes *Accept*, est assuré à l'aide de l'utilisation d'un algorithme appelé *algorithme de sélection distante des variantes* (voir Figure 2.10). Ce genre d'algorithme prend en entrée la liste des variantes et les entêtes *Accept*. Il vérifie ensuite si les informations des entêtes données sont suffisantes pour pouvoir effectuer la sélection à la place de l'application cliente. Si c'est le cas, l'algorithme détermine la meilleure variante du contenu. Si la meilleure

variante est une variante voisine [5], elle peut être retournée à l'application cliente accompagnée de la liste des variantes dans une réponse de choix.

Dans le cas où l'application cliente supporte la négociation transparente du contenu, l'application de la sélection côté serveur ne pourra se faire que si le client autorise explicitement (dans sa requête de négociation) l'utilisation d'un algorithme de sélection distante. Les applications clientes capables d'appliquer des algorithmes avancés de sélection de variantes peuvent interdire une sélection distante. Elles peuvent aussi l'autoriser pour des types particuliers de ressources, ou quand l'algorithme local est limité [52].

L'exemple suivant donne une réponse du serveur qui peut correspondre au scénario de la Figure 2.10 :

```
HTTP/1.1 200 OK
Date: Wed, 09 Jul 2003 00:00:01 GMT
TCN: choice
Content-Type: text/html
Last-Modified: Tue, 08 Jul 2003 09:07:07 GMT
Content-Length: 6623
Cache-control: max-age=99900
Content-Location: document.2
Alternates: {"document.1" 0.7 {type text/html} {language de}},
            {"document.2" 1.0 {type text/html} {language fr}},
            {"document.3" 0.9 {type application/pdf} {language en}}
Etag: "cloooo;1234"
Vary: negotiate, accept, accept-language

<title> Document 1 ...
```

Le dernier cas d'optimisation consiste à combiner les deux situations précédentes. Si un proxy de cache possède la liste des variantes, il peut dans certains cas effectuer la sélection à la place de l'application cliente. Ce scénario est représenté par la Figure 2.11.

2.3.3.1.2 L'entête *Negotiate* L'entête *Negotiate* peut comporter des directives pour un processus de négociation quelconque initié par la requête. Les directives de négociation peuvent avoir l'une des valeurs suivantes : trans, vlist, guess-small, rvsa-version, '*' ou negotiate-extension. Exemples :

```
Negotiate: 1.0, 2.5
Negotiate: *
```

La sémantique des différentes directives est la suivante :

– '*trans*' : l'application cliente supporte la négociation transparente du contenu pour la requête courante.

[5] Une variante est dite voisine d'une autre ressource si la variante possède une URL HTTP telle que l'URL absolue de la variante délimitée par son dernier '/' soit égale à l'URL absolue de la ressource délimitée par son dernier '/' [35]

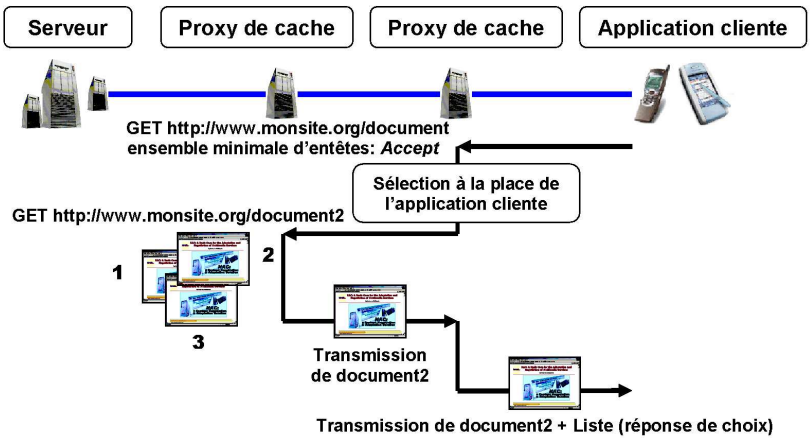

FIG. 2.11 – Troisième cas d'optimisation de la négociation

– '*vlist*' : l'application cliente demande que les réponses des ressources négociables pour la requête courante incluent une entête *Alternates* avec la liste des variantes.

– '*guess-small*' : l'application cliente autorise les serveurs d'origine à appliquer un algorithme de calcul de la meilleure variante pour la requête, et à retourner cette variante sous forme d'une réponse de choix.

– '*rvsa-version*' : l'application cliente autorise les serveurs (et les différents proxy) à appliquer un algorithme de sélection distante de variantes qui respecte les numéros de version donnés.

– '*' : l'application cliente autorise les serveurs et les proxy à appliquer n'importe quel algorithme de sélection distante de variantes.

Dans le cas où une directive de négociation est incompréhensible par le serveur, la directive est ignorée. Si l'application cliente autorise l'utilisation de plusieurs algorithmes de sélection, le serveur doit utiliser des heuristiques internes pour sélectionner l'algorithme le plus efficace.

2.3.3.1.3 L'entête *TCN* L'entête *TCN* (Négociation Transparente de Contenu) est utilisée par le serveur pour signaler qu'une ressource a été négociée en appliquant la stratégie de négociation transparente du contenu du protocole HTTP/1.1. Dans le cas où la ressource n'est pas négociée, l'entête TCN n'est incluse dans aucune réponse. L'entête TCN comprend trois champs : un champ type de réponse, un champ de directives et un champ d'extension. Le champ de directive est représenté par l'attribut *server-side-override-directive* qui peut prendre les valeurs 're-choose' (re-choisir) ou 'keep' (garder). Si la directive est 're-choose', le serveur inclut une entête *Alternate* avec la liste des variantes de la ressource originale. L'application cliente doit donc utiliser son propre algorithme pour la sélection et la récupération de la meilleure variante de la liste. Si la directive de négociation est 'keep', le client ne négocie pas la réponse, il l'utilise directement.

2.3.3.2 La dimension *fonctionnalité* de la négociation

La négociation transparente du contenu comporte une nouvelle dimension de négociation, appelée *fonctionnalité*, qui s'ajoute à celles qui ont été déjà définies et qui sont le type MIME (voir Section 2.3.1), le jeu de caractères et la langue naturelle. Cette nouvelle dimension a été définie pour les domaines de négociation non couverts par les dimensions classiques. Des exemples sur ces domaines non couverts sont : la négociation basée sur les extensions sur d'autres types de médias, les capacités du client vis-à-vis de l'affichage du contenu, la taille du dispositif d'affichage, etc. La dimension fonctionnalité rend la négociation transparente extensible. En effet, de nouvelles dimensions de la négociation peuvent être définies sans passer par de nouveaux efforts de standardisation, mais seulement par l'enregistrement du nouvel élément de fonctionnalité.

2.3.3.2.1 Eléments de fonctionnalité Un élément de fonctionnalité (*feature tag* ou *ftag*) identifie une entité ou une ressource qui peut être sujet de négociation, telle qu'une propriété de présentation, une capacité de terminal, une préférence particulière, etc. Les éléments de fonctionnalité sont utilisés dans les ensembles et les prédicats de fonctionnalité. Les prédicats sont utilisés par les attributs des éléments qui forment la description des variantes. Ces attributs spécifient comment la présence des éléments de fonctionnalité, pour une application cliente, affecte la qualité globale d'une variante.

Un élément de fonctionnalité peut avoir zéro valeur ou plus. Les valeurs donnent une sémantique à l'élément. Par exemple, un élément de fonctionnalité 'MmsEnabled' peut être associé aux valeurs 'yes' et 'no', pour indiquer si le terminal supporte ou non le format MMS.

Un ensemble de fonctionnalités d'une application cliente est une structure de données qui inclut les capacités du client et les préférences de l'utilisateur. Du point de vue description, la structure joue le même rôle que les profils CC/PP [43] et UPS [77], bien qu'elle se base uniquement sur le simple principe de sélection. Les ensembles de fonctionnalités sont utilisés par les algorithmes locaux de sélection des variantes. Une application cliente peut utiliser l'entête *Accept-Features* pour transmettre les valeurs de ses fonctionnalités aux algorithmes de sélection distante des variantes. La structure générale d'un ensemble de fonctionnalités est la suivante :

élément de fonctionnalité, ensemble de valeurs de l'élément

La signification de la présence d'une instance (élément de fonctionnalité, ensemble de valeurs de l'élément) dans l'ensemble, est que l'application cliente supporte la capacité correspondante ou que l'utilisateur a exprimé la préférence correspondante. Une instance peut ne pas supporter de valeurs, l'ensemble de valeurs est donc vide tout le temps, ou il peut supporter zéro valeur ou plus. La spécification [53] ne définit pas de notation normalisée pour la description des fonctionnalités. Un exemple de notation mathématique peut être donné comme suit :

```
{ ( "SMILContentControlModule" , { } ) ,
  ( "MmsEnabled"  , { "yes" } ),
  ( "Display"  , { "360x240", "240x360" } )
}
```

Le même exemple en notation XML :

```
<fonct>
  <SMILContentControlModule/>
  <MmsEnabled>yes</MmsEnabled>
  <Display>
      <val>360x240</val>
      <val>240x360</val>
  </Display>
</fonct>
```

Comme il n'est pas possible généralement que l'application cliente comprenne tous les éléments de fonctionnalité qui peuvent exister dans une description de variantes, les éléments incompréhensibles sont ignorés sans aucune implication pour les autres éléments. Les valeurs des éléments de fonctionnalité d'une application cliente peuvent changer selon le type de la requête ou de la ressource demandée. Elles peuvent aussi être mises à jour par l'application pour refléter les changements de capacités (logicielles par exemple, telle que l'installation d'un nouveau processeur de média) ou de préférences de l'utilisateur (telle que la taille de la fenêtre d'interface utilisateur).

2.3.3.2.2 Prédicats de fonctionnalités Les prédicats des fonctionnalité sont des prédicats posés sur le contenu des ensembles des fonctionnalités. Ils sont utilisés par les attributs des éléments qui forment la description des variantes. Les expressions des prédicats suivent une syntaxe simple :

```
fpred = [ "!" ] ftag
      | ftag ( "=" | "!=" ) tag-value
      | ftag "=" "[" numeric-range "]"

numeric-range = [ number ] "-" [ number ]
```

L'évaluation des expressions des prédicats peut être résumée comme suit :

- *ftag* (élément de fonctionnalité) : Vrai si la fonctionnalité est présente, Faux sinon.
- *!ftag* : Vrai si la fonctionnalité est absente, Faux sinon.
- *ftag*=V : Vrai si la fonctionnalité est présente avec la valeur V, Faux sinon.
- *ftag* !=V : Vrai si la fonctionnalité n'a pas la valeur V, Faux sinon.
- *ftag*=[N-M] : Vrai si la fonctionnalité est présente avec au moins une valeur numérique et la plus grande valeur présente de la fonctionnalité est incluse dans l'intervalle [N-M], Faux sinon. Si N n'est pas spécifié, la borne inférieure de l'intervalle est zéro par défaut. Si M n'est pas spécifié, la borne supérieure est l'infini.

Voici quelques exemples d'utilisation des fonctionnalités de négociation dans les listes des variantes transmises par les serveurs HTTP/1.1.

```
1- {"presentation.smil.basic"  0.6 },
   {"presentation.smi"  1.0  {features MediaAccessibility RepeatTiming
                                                    InlineTransitions}}
2- {"presentation.txt"  0.6},
   {"presentation.smi"  1.0  {features !textonly}}

3- {"presentation.normale" },
   {"presentation.laptop"  1.0  {features screenwidth=[600-1000]}},
   {"presentation.PC"  1.0  {features screenwidth=[1000-]}},
   {"presentation.pocketPC"  1.0  {features screenwidth=[-240]}},
```

2.3.3.3 L'algorithme de sélection distante des variantes

Un algorithme de sélection distante de variantes est un algorithme qui permet à un serveur, au lieu de l'application cliente concernée, de choisir la meilleure variante du contenu source demandée par le client. L'utilisation d'un algorithme de sélection côté serveur peut accélérer le processus de négociation de contenu en évitant les échanges inutiles de messages entre le client et le serveur.

L'algorithme de sélection distante vérifie si les informations transmises par l'application cliente sont suffisantes pour prendre une décision de choix ; dans le cas positif, l'algorithme sélectionne la meilleure variante.

Dans [52], la version 1.0 de l'algorithme de sélection distante des variantes pour le protocole HTTP (RVSA/1.0) est définie. L'algorithme est souvent utilisé pour des requêtes particulières qui concernent des ressources négociables d'une manière transparente. L'algorithme utilise comme entrée, la liste des variantes de la ressource originale, telle qu'elle est présente dans l'entête *Alternates* de la ressource. Il utilise aussi le sous-ensemble de capacités et de préférences du client, en se basant sur les informations transmises à l'aide des entêtes *Accept*. Sur la base de ces informations utilisées en entrée, l'algorithme exécute l'action la plus appropriée. Deux cas sont distingués :

- Les informations des entêtes *Accept* sont suffisantes pour faire un choix ; dans ce cas, l'algorithme retourne la meilleure variante dans une *réponse de choix* [35].

- Les informations des entêtes *Accept* ne sont pas suffisantes pour faire un choix ; dans ce cas, une liste de réponses est envoyée afin que l'application cliente puisse faire son choix.

2.3.3.3.1 Les valeurs de qualité L'algorithme se base sur le calcul des valeurs, appelées *valeurs de qualité*, associées aux différentes variantes d'une liste. Une valeur de qualité représente en quelque sorte le degré de fidélité de la variante par rapport à la ressource source et le degré de satisfaction des contraintes du client. Le RVSA/1.0 définit la valeur de qualité Q d'une variante, comme suit :

$$Q = round5\ (qs * qt * qc * ql * qf)$$

Où, $round5$ est une fonction qui arrondit une valeur réelle en une valeur réelle à cinq décimal après la virgule. Les facteurs qs, qt, qc, ql et qf sont relatifs à la qualité source, la qualité du type de média, la qualité du jeux de caractères, la qualité du langage et la qualité des fonctionnalités (section 2.3.3.2) respectivement. Le tableau 2.2 montre un exemple de calcul de valeur de qualité pour une variante.

La meilleur variante, telle qu'elle est définie par l'algorithme RVSA/1.1, est la variante ayant la plus grande valeur de qualité. Dans le cas où plusieurs variantes partagent la même valeur, la première variante qui apparaît dans la liste des variantes est choisie.

2.3.3.4 Négociation et utilisation du cache

Afin de supporter une gestion de cache des réponses négociées et assurer plus d'extensibilité que le protocole HTTP/1.0, HTTP/1.1 inclut l'entête de réponse *Vary*

Tab. 2.2 – Exemple de calcul de valeur de qualité

Information utile	Valeur
Description de la variante	'document.html.en' 0.7 type text/html language fr
Accept	text/html :q=1.0, */* :q=0.8
Accept-Language	en ;q=1.0, fr ;q=0.5
Valeur de qualité	round5(0.7*1.0*0.5)= 0.35000

[35]. Cet entête comporte des champs de sélection qui participent à la sélection d'une variante du contenu source. Pour utiliser une variante particulière d'une réponse cachée, les champs de sélection doivent correspondre exactement au champ de la requête d'origine. Ce mécanisme est efficace pour plusieurs stratégies de négociation. Cependant, il ne permet pas d'appliquer des stratégies intelligentes de cache, par exemple dans le cas où la valeur d'un champ est compatible avec la valeur originale sans que les deux valeurs soient obligatoirement égales.

Afin de rendre la gestion du cache plus explicite, le protocole HTTP/1.1 définit la nouvelle entête *Cache-Control* qui permet d'étendre les directives de contrôle du cache qui peuvent être transmises dans les réponses des serveurs et dans les requêtes d'applications clientes. L'ensemble de ces directives est très large (voir [35]). Par exemple, concernant la transformation du contenu, le protocole définit la directive *no-transform* qui permet de contrôler les transformations du contenu. Pour éviter la dégradation des images reçues à travers un proxy de transcodage d'images, un client peut utiliser cette directive pour éviter toute transformation de contenu.

Le protocole HTTP/1.1 essaie aussi de définir une stratégie appelée négociation transparente de contenu [53] [52] ; cependant ces efforts ne font pas partie de la spécification du protocole [35].

2.3.3.5 Optimisation des ressources réseau

Le protocole HTTP/1.1 assure une meilleure gestion d'utilisation des ressources réseau, comparée à celle du protocole HTTP/1.0. En appliquant le protocole HTTP/1.1, un client peut envoyer une requête visant une partie du contenu existant côté serveur. Cela est utile dans certaines situations, telles que la visualisation de parties des grands documents, ou le téléchargement contrôlé des ressources. L'approche des requêtes sur des parties du contenu a été déjà abordée par Ari Luotonen et John Franks [36] en utilisant des extensions de la syntaxe des URLs au lieu d'utiliser des champs d'entête. HTTP/1.1 définit les requêtes de type *Range* qui permettent d'envoyer une requête vers une portion de contenu. Les parties sélectionnées sont toujours spécifiées en octets ce qui représente une limitation si on vise d'appliquer des sélections plus proches de la sémantique du contenu, par exemple des sélections de chapitres dans un document, des images dans une vidéo, etc. Afin de recevoir une partie de contenu, l'application cliente insère l'entête *Range* dans sa requête en spécifiant un ou plusieurs intervalles d'octets. Le serveur pourra donc répondre à cette requête en envoyant une ou plusieurs parties du contenu source. Le serveur indique la partie retournée ainsi que sa taille en utilisant l'entête : *Content-Range*. Le nouveau type MIME *multipart/byteranges* permet la transmission de plusieurs parties du contenu en une seule réponse.

Les requêtes de type *Range*, peuvent être exploitées de plusieurs façons, par exemple pour lire la partie initiale d'une ressource afin de déterminer ses caractéristiques et pouvoir préparer l'application cliente à la traiter ; pour finir un transfert de données qui a été interrompu à cause de l'instabilité du réseau ; pour construire des profils de ressources afin de négocier le contenu à recevoir sans être obligé de recevoir le contenu entier.

2.3.4 Bilan sur la négociation dans HTTP

Dans les deux sections précédentes 2.3.2 et 2.3.3, nous avons présenté les stratégies de négociation de contenu du protocole HTTP. Ces stratégies sont utilisées dans les architectures basées sur l'utilisation de HTTP pour la communication entre l'ensemble des clients, des serveurs et éventuellement des proxy. La spécification du protocole HTTP/1.0 discute très brièvement le problème de la négociation de contenu. La stratégie de négociation du protocole est basée sur l'envoi des capacités de l'application cliente et des préférences de l'utilisateur au serveur. Le serveur choisit ensuite la meilleure forme du contenu source qui correspond aux exigences du client. La variante choisie est envoyée au client. Cette simple stratégie a montré ses limitations. Non seulement elle ne considère pas tous les aspects de la négociation, tels que les dimensions de la négociation, la gestion efficace du cache et de l'utilisation des ressources, ou les directives explicites de contrôle, mais elle présente aussi des gro problèmes de performances dûs à l'envoi d'énormes entêtes qui peuvent s'avérer inutiles.

Le protocole HTTP/1.1 définit une stratégie de négociation plus complète et plus efficace. Le protocole considère la négociation comme un outil permettant de servir l'application cliente avec la meilleure version (ou variante) disponible, c'est-à-dire celle qui est la plus fidèle au contenu source et qui satisfait au mieux les contraintes du client cible. Les stratégies étudiées peuvent être classées selon l'entité responsable du processus de négociation. Comme nous avons déjà vu, trois schémas sont définis : la négociation orientée serveur, la négociation orientée client et la négociation basée sur la combinaison de ces deux dernières, en utilisant les caches.

L'intérêt de la première classe est que l'entité responsable des traitements de négociation est le serveur. Ceci évite que les clients de l'environnement hétérogène, qui sont souvent limités, se chargent du calcul engendré par le processus de la négociation. Comme nous l'avons vu, ce type de stratégie n'exclut pas que l'application cliente facilite l'exécution du processus de négociation en envoyant des entêtes de description. Cependant, cette stratégie comporte quelques inconvénients. Le serveur est incapable d'avoir une idée globale du contexte et du profil du client, c'est-à-dire de ce que le client est réellement capable de faire et de comprendre. Ce point peut conduire à des mauvaises décisions, incompatibles avec les principes de la négociation. Un autre problème qui se pose est celui de l'expressivité des clients. En effet, le client ne peut pas décrire d'une manière précise son contexte, sa manière de présenter le contenu et tout ce qu'il est capable de faire. En outre, la spécification du protocole, ne définit pas l'intégration d'un algorithme de négociation précis pour le serveur.

La deuxième classe de négociation définit le client comme l'entité responsable du processus de négociation de contenu. L'avantage de cette catégorie de négociation est que le client est l'élément de l'environnement le mieux placé pour connaître ses propres

capacités et préférences. Ce type de négociation implique une gestion importante des caches afin de garantir des performances raisonnables du système. L'inconvénient majeur de cette stratégie est que les clients sont souvent face à des limitations matérielles et logicielles non négligeables, et ne peuvent pas assurer une stratégie de négociation avancée. En outre, la communication client/serveur implique deux échanges de messages : un pour la liste des variantes et un pour la réception de la variante choisie.

La troisième classe est une combinaison des deux approches précédentes. Bien que les performances du système peuvent être améliorées par rapport aux précédentes propositions, la stratégie garde quelques inconvénients comme le manque d'expressivité, l'absence de définition précise d'une stratégie de cache avancée, etc. Le choix de la combinaison des approches est tellement ouvert que les stratégies proposées peuvent être très variées et très éloignées du modèle d'origine.

Du point de vue global, la négociation du protocole HTTP ne peut pas être adoptée pour assurer un système de négociation avancée. La négociation est complètement basée sur une approche de gestion de variantes, ce qui implique que l'auteur du contenu doit fournir de multiples présentations du même contenu (la *multi-édition*), afin que les ressources soient négociables. La négociation HTTP ne considère pas l'aspect dynamique de transmission et de transformation du contenu. Les seules directives qu'on peut trouver sont celles du genre *no-transform*, qui permet d'indiquer si le client accepte de recevoir un contenu transformé ou non. Le principe des types MIME utilisé est limité et ne permet pas d'exprimer tous les types de média qui peuvent être utilisés dans une présentation multimédia. Pour la négociation d'une ressource média, le type est nécessaire mais pas suffisant. En effet le type représente une dimension de négociation parmi une grande diversité d'autres caractéristiques. D'autre part, la spécification du protocole HTTP ne donne pas une définition complète et normalisée de la déclaration des capacités et des préférences du client.

2.3.5 Langage SMIL 2.0

La version 2.0 du langage SMIL (Synchronized Multimedia Integration Language) [109] est un modèle de document basé sur le langage XML qui permet la représentation des présentations multimédia interactives. Le langage permet principalement de décrire l'organisation temporelle des objets média, les relations entre les objets utilisés, la disposition spatiale des objets ainsi que les liens qui peuvent être utilisés dans une présentation. D'autre part, la spécification du langage vise à offrir la possibilité de réutiliser la syntaxe et la sémantique de SMIL dans d'autres langages basés sur XML. Les composants temporels et de synchronisation de SMIL sont ainsi utilisés pour intégrer un aspect temporel tel qu'il est le cas dans XHTML [131] et dans SVG [107].

2.3.5.1 Documents SMIL 2.0

Un document SMIL comporte une structure d'éléments composites appelés opérateurs. Un opérateur porte une certaine sémantique temporelle qui permet de définir le placement temporel des ressources qu'il contient. L'élément *seq* permet de présenter les ressources médias en séquence. L'élément *par* permet de présenter les ressources en parallèle. L'élément *excl* permet de présenter les ressources d'une manière exclusive, c'est-à-dire qu'un seul objet est joué parmi l'ensemble des objets contenus dans l'opérateur.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<smil xmlns="http://www.w3.org/2000/SMIL20/CR/Language">
<head>
    <layout>
        <region id="main" left="10" .../>
    </layout>
</head>
<body>
    <par id="T" dur ="120s">
        <img id="photo" src="Ray_of_Light_CD.jpg" region="main"/>
        <audio id="song" src="Frozen.mp3" begin="id(song)+2s"/>
    </par>
</body>
</smil>
```

Fig. 2.12 – Exemple de document SMIL 2.0

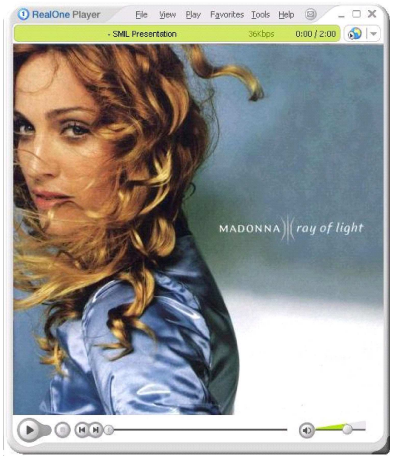

Fig. 2.13 – Exemple de présentation d'un document SMIL 2.0

Les opérateurs constituent une structure arborescente qui permet de définir une synchronisation des objets média d'une manière globale. SMIL offre la possibilité de détailler cette synchronisation en spécifiant des arcs de synchronisation ou des événements. Ces arcs sont définis à l'aide des attributs *begin* et *end* qui sont associés aux opérateurs et aux objets média. Figure 2.12 montre un exemple de document SMIL. La Figure 2.13 montre la présentation du document par une application cliente.

La sémantique du document SMIL représenté par la Figure 2.12 est décrite comme suit : les objets média identifiés par *photo* et *song* se jouent en parallèle. La ressource *song* se joue deux secondes après le début de l'objet *photo* grâce à la synchronisation définie par *begin="id(song)+2s"*. Le document spécifie aussi des informations de type spatial : la ressource *photo* est associée à une partie visuelle de l'espace d'affichage. Cette partie est définie par l'élément *region*. Dans un document SMIL, une nette séparation est faite entre le scénario temporel, spécifié par les opérateurs, et la dimension spatiale définie en termes de régions.

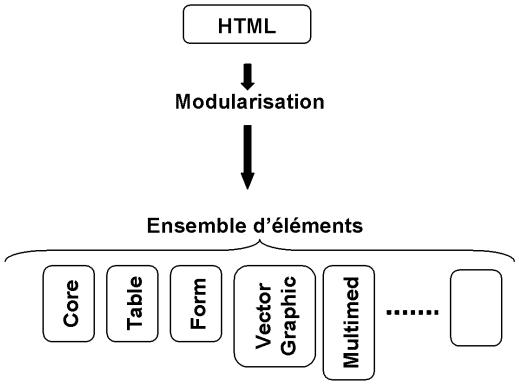

FIG. 2.14 – Exemple de modularisation de XHTML

2.3.5.2 La modularisation et les profils de langage

La spécification du langage SMIL 2.0 adopte une approche de *modularisation*. Cette approche consiste à décrire les fonctionnalités du langage sous forme d'un ensemble de *modules*. Un module est simplement un ensemble d'éléments, d'attributs et de valeurs d'attributs qui décrivent un certain aspect sémantique des présentations multimédia. Le langage utilise aussi le principe de la définition des profils qui revient à définir de nouveaux langages XML, basés sur la combinaison des modules déjà définis, afin de satisfaire un besoin applicatif particulier.

Le concept de la modularisation représente une approche très utile pour répondre à la diversité des applications et des clients dans un environnement hétérogène. Cette approche a été utilisée par d'autres langages tels que le langage XHTML (Extensible Hypertext Markup Language) [131]. XHTML est une reformulation du langage HTML 4.0 [97] sous forme d'une application XML. XHTML est organisé sous forme d'un ensemble de modules indépendants pour la description des éléments HTML (voir Figure 2.14), tels que les titres, les paragraphes, les listes, etc. Un module est défini par une description de fonctionnalités sous forme d'un ensemble d'éléments et attributs. La modularisation utilisée dans XHTML a permis d'utiliser ce langage dans beaucoup d'applications, notamment dans WAP (Wireless Application Protocol) 2.0 [89] [88], un langage destiné aux téléphones mobiles.

Grâce à la combinaison de modules, SMIL 2.0 offre une approche de négociation qui permet de supporter les contraintes de plusieurs contextes et applications clientes. Par exemple, un terminal mobile peut supporter uniquement un ensemble prédéfini de modules. Le contenu transmis à ce terminal ne doit pas contenir les modules non supportés. Il est donc possible de définir des *profils* de langage [109] spécifiques pour différentes classes de clients. L'organisation modulaire donne la possibilité aux serveurs d'appliquer des processus de négociation basés sur les modules du contenu, mais aussi aux clients de distinguer les parties à traiter dans le cas où l'application cliente reçoit un module non reconnu.

SMIL est un langage extensible permettant la description de plusieurs aspects d'une représentation multimédia. Cette extensibilité permet de couvrir une grande

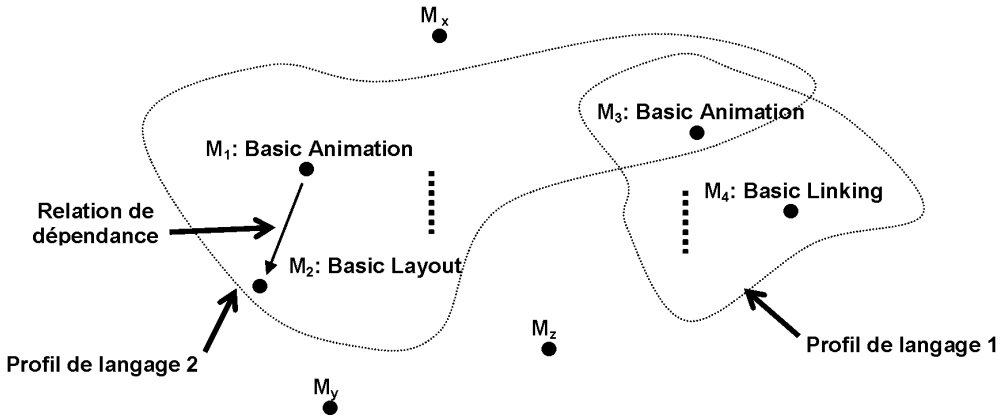

FIG. 2.15 – Principe des profils de langages SMIL

diversité de fonctionnalités et facilite la conception de stratégies de négociation. SMIL est utilisé dans un grand nombre d'applications pour les plates-formes conventionnelles mais aussi pour les plates-formes mobiles, comme celles de la téléphonie de troisième génération 3GPP (the third Generation Partnership Project) [**?**] et son format MMS, pour les présentations multimédia [117], qui se base essentiellement sur le langage SMIL Basic [127].

Les modules SMIL 2.0 peuvent être complètement indépendants ou reliés par des relations de dépendance. Par exemple, les modules *BasicLayout* et *BasicLinking* sont des modules indépendants, alors que le module *AudioLayout* dépend du module *Basic-Layout*.

Un langage de profil SMIL 2.0 est un sous-ensemble de modules SMIL 2.0 qui inclut tous les modules reliés par une relation de dépendance avec un module déjà inclus dans la langage (voir Figure 2.15). Chaque module possède un unique identificateur. Par exemple, l'identificateur du module *Structure* est *http ://www.w3.org/-2000/SMIL20/CR/Structure*.

Dans le contexte des environnements hétérogènes qui présentent beaucoup de limitations et de contraintes en termes de ressources et de capacités de traitement, il est important de définir un noyau commun et extensible de fonctionnalités supportées. Une telle définition permet la construction d'un contenu plus générique et plus flexible selon le contexte d'utilisation cible. Le langage de profil SMIL Basic [127] représente un outil efficace pour atteindre cet objectif. En effet, le langage permet d'éditer des présentations multimédia pour un grand nombre de clients dont la plate forme peut être avancée ou très limitée. Le profil SMIL Basic consiste en une collection réduite de modules SMIL 2.0. Un contenu SMIL Basic peut être présenté par une grande majorité d'applications clientes, même par celles qui tournent sous des plate formes mobiles et qui présentent des limitations concernant le dispositif d'affichage, les caractéristiques du réseau, etc. Un exemple d'application SMIL Basic est son utilisation par le projet 3GPP dans les descriptions de scènes pour les serveurs et les clients PSS (*Packet switched Streaming Service*) [**?**]. La collection de modules PSS SMIL inclut les modules définis dans le langage SMIL 2.0 Basic avec trois modules additionnels.

2.3.5.3 Négociation de l'affichage dans SMIL

SMIL 2.0 inclut les informations relatives à l'affichage spatial des objets en utilisant les éléments <*layout*>, <*root-layout*> et <*region*>. Le langage inclut un ensemble de modules pour des types spécifiques d'affichage comme, par exemple, l'affichage des fenêtres multiples. L'approche de SMIL 2.0 sépare clairement les informations relatives à l'espace d'affichage et le contenu à afficher. En variant ces informations d'affichage, la présentation SMIL finale pourra être changée sans modifier le contenu à afficher. De la même manière, le contenu peut être modifié sans affecter les informations d'affichage. En SMIL, seuls les identificateurs des régions relient le contenu avec l'espace d'affichage.

Cette approche offre la possibilité de développer des stratégies de négociation relatives à la dimension spatiale des présentations. Ces stratégies peuvent définir des mécanismes qui permettent d'associer des affichages différents à des variables de contexte. Par conséquent, l'affichage des objets d'un contenu pourra être négocié, au moment de la transmission, selon l'état courrant du contexte. Ce genre de stratégie peut être appliqué du côté serveur (ou proxy), avant que le contenu soit transmis, ou du côté de l'application cliente qui sera responsable de négocier l'affichage le plus approprié. Ces possibilités de négociation de SMIL ont été exploitées par quelques travaux récents sur la variation des stratégies d'affichages [125], l'adaptation de l'affichage selon le contexte [84] et la définition d'un nouveau langage de marquage pour l'indépendance aux terminaux [25]

2.3.5.4 Négociation basée sur l'interaction

La spécification du langage SMIL 2.0 [109] définit de nouveaux mécanismes d'interaction qui étendent l'interaction déjà offerte par HTML [97]. En particulier, la notion des liens hypermédia a été étendue afin de supporter :

- Le déplacement vers un instant précis de la présentation[6] (voir Figure 2.16).
- Les liens de navigation temporelle qui sont accessibles seulement durant des périodes de temps définies par la présentation.

Ces mécanismes offrent de nouvelles techniques pour la négociation et l'adaptation du contenu. Les liens hypermédia peuvent être utilisés pour assurer une décomposition du contenu afin que les longues présentations puissent être divisées en petites parties selon les limites indiquées par l'auteur ou selon l'état courant du contexte de transmission. D'autre part, les liens temporels permettent de présenter le contenu uniquement avec les options de navigation qui sont appropriées au contexte de la présentation. Puisque seul un sous-ensemble de liens est nécessaire à un moment donné de la présentation, la stratégie de négociation peut réduire l'espace d'affichage réservé aux liens ainsi que la diversité du choix de liens.

2.3.5.5 Modules de contrôle de contenu

Les modules de contrôle de contenu de SMIL 2.0 [17] incluent les éléments et les attributs qui offrent la possibilité de transmission d'un contenu optimisé et de la sélection du contenu au moment de la présentation. Les fonctionnalités du contrôle de

[6] L'élément *anchor*, utilisé pour cela dans SMIL 1.0 [108], a été remplacé par l'élémént *area* dans SMIL 2.0

```
<video
src="inria.mpeg"
region="video"
title="Presentation INRIA"
alt="Presentation INRIA Rh{\^o}ne Alpes, France"
abstract="Les projets de recherche de l'INRIA">

<area id="WAM" begin="0s" end="50s" title="Projet WAM"/>
<area id="Sardes" begin="15s" end="65s" title="Projet Sardes"/>

</video>
```

FIG. 2.16 – L'intereaction de SMIL 2.0

contenu de SMIL sont organisées sous forme de quatre modules :

1. Le module *BasicContentControl* qui contient des éléments de sélection de contenu et définit un ensemble d'attributs de base pour tester le contexte.

2. Le module *CustomTestAttribute* qui contient des éléments et attributs personnalisés pour tester le contexte.

3. Le module *PrefetchControl* qui contient des éléments et attributs d'optimisation de la présentation.

4. Le module *SkipContentControl* qui contient un attribut qui gère une évaluation selective de contenu.

La spécification des modules de contrôle du contenu de SMIL 2.0 [17] définit une liste de douze attributs pour l'évaluation du contexte tels que la langue naturelle de l'utilisateur, la taille de l'écran du terminal, le système d'exploitation utilisé, etc. Les attributs du contexte forment des expressions booléennes à évaluer. Une expression peut être associée à un ou plusieurs éléments de la présentation. Si un ou plusieurs attributs du contexte sont évalués à *faux*, l'élément comportant cet attribut est ignoré dans la présentation.

SMIL 2.0 assure une gestion complète des alternatives en utilisant l'élément *switch*. Ce dernier permet de spécifier un ensemble d'alternatives pour un élément de la présentation. Chaque alternative comporte des attributs de contexte qui sont évalués. La première alternative acceptable est présenté. Quelques travaux essaient de définir d'autres moyens pour assurer la gestions des alternatives, comme par exemple le model *CMIF* [19] [18] qui propose une autre manière d'expression des alternatives mais qui est équivalente à un ensemble d'éléments *switch*.

La Figure 2.17 montre un exemple d'utilisation du *switch*. Dans cet exemple, l'objet audio *welcome_to_inria.wav* est jouée en même temps que l'image sélectionnée. L'image *inria_1024_1280.gif* est réservée pour les terminaux ayant un espace d'affichage suffisant, l'image *inria_480_640.gif* pour les terminaux ayant un espace d'affichage moyen et l'image *inria_240_320.gif* est réservée pour les terminaux ayant un espace

```
...
<par>
<audio src="welcome_to_inria.wav" .../>
<switch>
    <img src="inria_1024_1280.gif" systemScreenSize="1024X1280"/>
    <img src="inria_480_640.gif" systemScreenSize="480X640"/>
    <img src="inria_240_320.gif" systemScreenSize="240X320"/>
    <img src="inria_default.gif"/>
</switch>
</par>
...
```

FIG. 2.17 – L'élément *switch* de SMIL 2.0

d'affichage réduit. L'application cliente doit évaluer si le terminal est capable ou non d'afficher les différentes ressources. Si aucune ressource n'est sélectionnée, l'alternative *inria_default.gif* est sélectionnée car cette alternative n'est associée à aucun attribut de contexte.

Afin d'offrir plus de flexibilité dans la sélection des objets, les attributs du contexte peuvent aussi être utilisés en dehors des éléments *switch*, et cela en les associant directement avec les ressources de la présentation. comme dans l'expression suivante :

$< textstream\ src = "presentation\_speech\_translation.rt"\ systemLanguage = "fr" / >$

Ce genre de sélection est appelé *test en ligne* [109]. L'utilisation des tests en ligne simplifie énormément la spécification des directives de négociation et d'adaptation du contenu, en particulier dans le cas où plusieurs alternatives indépendantes existent.

Bien que la spécification des alternatives, que ce soit à l'intérieur ou à l'extérieur de l'élément *switch*, s'effectue manuellement par l'auteur de la présentation, l'usage des stratégies de négociation basées sur l'évaluation des attributs de contexte permet de calculer automatiquement les alternatives appropriées. Malheureusement, l'ensemble des attributs de contexte est limité. Il est donc difficile avec un ensemble restreint de dimensions, d'effectuer des adaptations raffinées pour satisfaire les contraintes du client cible.

2.3.6 Modèle AHM

Le modèle *AHM* (Amsterdam Hypermedia Model) [49] a été développé afin de fournir un cadre de travail pour la combinaison de deux domaines : le multimédia et l'hypermédia. Le modèle a été défini en combinant le modèle hypertexte *Dexter* [47] [44] avec le modèle multimédia *CMIF* [18] et en définissant quelques extensions, relatives à l'hypermédia.

Comme le modèle Dexter, AHM permet l'usage d'un ensemble d'attributs de présentation spécifiques à des composants individuels du contenu. AHM étend cet ensemble d'attributs en définissant le concept des *canaux* (ou *channels*). Les canaux sont des entités abstraites associées à la présentation d'un élément du contenu multimédia. Un canal regroupe un ensemble de caractéristiques qui concernent la présentation de l'élément multimédia, tels que le style ou la taille d'un texte, le volume d'un objet audio, etc. Lorsqu'un document multimédia est joué, les canaux sont instanciés selon

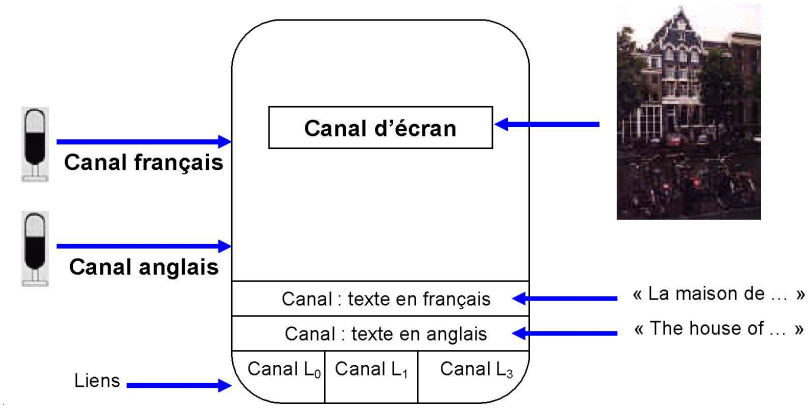

FIG. 2.18 – Le modèle *AHM*

les caractéristiques physiques du terminal.

La Figure 2.18 [49] montre un exemple d'utilisation des canaux AHM pour une présentation multimédia. Deux canaux audio sont utilisés, un canal pour un commentaire en français et un autre pour le même commentaire en anglais. La présentation contient trois liens représentés par trois canaux différents qui pointent vers des portions différentes de contenu. Deux ressources textes sont utilisées et qui fournissent des informations sur la présentation en deux langues. Un canal AHM possède un attribut d'activation qui permet à l'utilisateur de spécifier si le canal doit être utilisé ou non au moment de la présentation.

Les canaux du modèle AHM permettent de personnaliser une présentation multimédia pour différents contextes. Cependant, le modèle ne définit pas une stratégie explicite pour la gestion et la sélection des canaux. Les attributs des canaux, qui correspondent aux dimensions du contexte dans le cadre de la négociation, restent limités et ne peuvent pas assurer une négociation avancée qui soit dynamique. Le modèle néglige aussi le degré de préférence entre différentes variantes tel qu'il existe dans HTTP/1.1 avec les variables de qualité. Ce simple paramétrage des présentations a été repris et amélioré par quelques autres modèles, comme le modèle Z_YX que nous discutons dans la section suivante.

2.3.7 Modèle Z_YX

Z_YX [11] [12] représente un modèle qui a été développé sur la base des avantages des modèles de documents qui existaient déjà et de quelques recherches précédentes faites dans le contexte de la présentation multimédia dans les systèmes de gestion de base de données. Avec un nouveau modèle de document, Z_YX essaie d'atteindre quelques objectifs qui incluent la réutilisation des ressources, l'adaptation et l'indépendance vis-à-vis de la présentation. Bien qu'aucune stratégie stratégie de négociation de contenu ne soit définie, l'approche de négociation tirée du modèle Z_YX est basée sur les fonctionnalités du contenu lui-même, et non pas sur des entités externes comme les serveurs ou les proxy.

Pour assurer la réutilisation du contenu, Z_YX définit plusieurs granularités de composants réutilisables : les objets médias, les fragments de documents et les documents

entiers. Les fragments, qui peuvent avoir une taille arbitraire, peuvent être réutilisés dans n'importe quel autre document. Ce principe est similaire à l'approche adoptée par Madeus [68]. Le modèle essaie d'offrir la possibilité de réutiliser la structure du document en y rajoutant des informations de disposition afin d'utiliser les différentes granularités des composants avec les nouvelles informations rajoutées. Concernant la sélection et l'identification des différentes granularités des composants, le modèle utilise des annotations et ajoute aux composants des méta données de description du contenu.

L'objectif de l'adaptation dans Z_YX est d'éviter que l'adaptation soit dépendante d'un ensemble prédéfini d'attributs exploités par les mécanismes d'adaptation (comme c'est le cas dans SMIL 1.0 [108]). L'approche adoptée dans Z_YX permet que cet ensemble d'attributs soit plus ouvert afin de refléter le contexte du système et des clients compliqués [10]. Basé sur les avantages des modèles SMIL 2.0 [109] et AHM [49], Z_YX propose une modélisation statique des *alternatives de présentation* définies pour assurer la présentation du contenu dans différents contextes. Le modèle définit aussi des primitives pour déterminer le besoin des alternatives au moment où le contenu est demandé.

Afin que le modèle puisse assurer une indépendance vis-à-vis de la présentation du contenu, Z_YX propose un modèle de données qui décrit le document multimédia avec un niveau sémantique élevé. Ce niveau sémantique se base sur un modèle fortement structuré et une modélisation riche des primitives. Comme dans la plupart des modèles de documents qui ont précédé Z_YX, le modèle adopte une structure basée sur une organisation hiérarchique. Il sépare les informations de disposition spatiales des informations de la structure, comme SMIL.

2.3.7.1 Vue globale sur les documents Z_YX

La structure des documents Z_YX est une structure arborescente dont les nœuds sont les éléments de présentation. Chaque élément possède une ou plusieurs variables qui peuvent le lier avec un autre élément de présentation. Un élément peut avoir aussi des variables, dites de *projection*, qui spécifient sa disposition spatiale dans le document (voir Figure 2.19 [12]).

Les éléments de présentation peuvent représenter des objets média ou des éléments de structure temporelle, spatiale ou d'interaction. La Figure 2.20 [12] montre un exemple de représentation arborescente d'un document Z_YX. Les éléments *seq* (élément séquentiel) et *par* (élément parallèle) sont des éléments de structure temporelle comme dans SMIL. L'élément *par*, par exemple, synchronise les deux éléments de média *Vidéo* et *Audio* qui sont liés aux variables v_1 et v_2 de *par*. La sémantique de ce fragment est que les ressources *Vidéo* et *Audio* sont jouées en parallèle. Notons que les variables v_1 et v_5 de l'élément *seq* ne sont liées à aucun élément[7]. Une réédition du document peut associer ces deux variables à un titre et un résumé de la séquence respectivement.

[7]Les variables non liées sont appelées *libre*

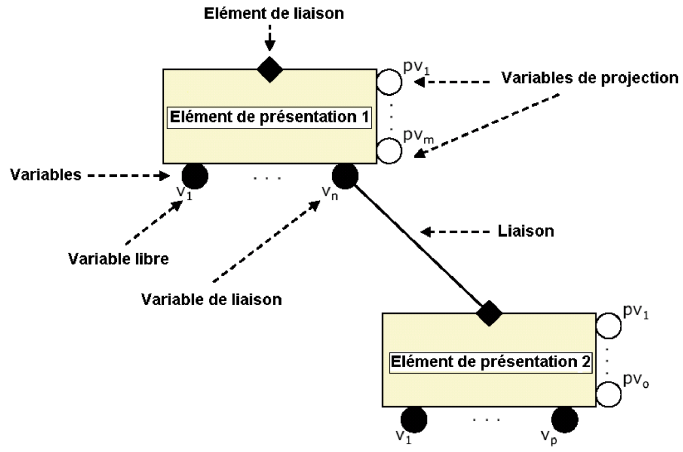

FIG. 2.19 – Eléments du modèle Z_YX

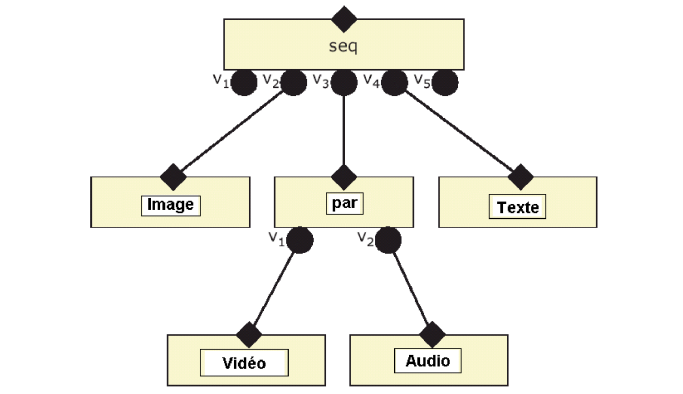

FIG. 2.20 – Exemple de fragment de document Z_YX

2.3.7.2 La réutilisation

Z_YX définit les éléments *selector* (sélecteurs) pour la réutilisation des éléments médias. Ces éléments sélectionnent les parties des éléments médias qui seront représentées et réutilisées. Le sélecteur temporel (*temporal-s*) spécifie le début et la durée de la séquence sélectionnée. Le sélecteur spatial (*spatial-s*) spécifie, en utilisant un polygone, une partie d'une ressource visuelle (une image par exemple). Le modèle définit la notion de *template* pour offrir la possibilité de réutiliser les entités de la présentation. Ce principe est assuré en utilisant des variables libres (voir Figure 2.20) qui peuvent être instanciées par une future édition mais aussi par un processus d'adaptation.

Les variables libres peuvent aussi être associées à des fragments complexes qui encapsulent plusieurs éléments de présentation. En outre, il est aussi possible d'utiliser des fragments externes qui proviennent d'autres documents. Un document ou un fragment de document peut être réutilisé en liant une variable libre à son élément racine.

Le modèle Z_YX utilise des éléments, dits *projecteurs*, afin de contrôler la présentation d'un fragment (position spatiale, cadence d'une vidéo, etc.). Un *projecteur* peut être associé à un élément de présentation ou à son sous-arbre. La même structure de présentation peut donc être réutilisée en changeant, ou en ajoutant, seulement les éléments *projecteurs*. Notons que le principe des *projecteurs* est similaire à la notion des *régions* définie dans SMIL [109]. Afin de faciliter l'identification et la sélection des fragments de présentation, Z_YX associe à chaque fragment un ensemble de méta-données qui décrit le contenu sous forme de couples d'attributs et de valeurs.

2.3.7.3 L'adaptation

Le modèle Z_YX définit trois éléments de présentation au service de l'adaptation des documents : l'élément *switch*, l'élément *decide* et l'élément *query*. Les deux premiers éléments (*switch* et *decide*) sont utilisés pour spécifier des alternatives pour une partie du document. A chaque alternative, définie sous un élément *switch* ou *decide*, des méta données, qui décrivent un contexte particulier, sont associées. Le contexte décrit représente l'état du système dans lequel l'alternative correspondante offre la meilleure présentation qui respecte les contraintes de l'environnement.

Le modèle suppose donc que lors de la présentation, le contexte est évalué et la meilleure variante est sélectionnée. La différence entre les éléments *switch* et *decide* est que dans le cas du *decide* le choix d'alternatives est définitif, c'est-à-dire une fois une alternative choisie, la présentation ignore les autres alternatives, alors que dans le cas du *switch* une nouvelle sélection d'alternative peut être appliquée si les conditions de la présentation changent.

L'élément *query* spécifie, en utilisant des méta-donnés, le fragment à utiliser lors de la présentation du document. Quand un document est présenté, l'élément *query* est évalué et l'élément est remplacé par le fragment qui correspond le mieux aux méta-données spécifiées par l'élément *query*.
Exemple :

query [(sujet, "météo du jour"), (type, "résumé"), (durée, 5min)]

L'élément *query* de l'exemple, permet d'inclure au moment de la présentation le résultat transmis comme réponse à la requête ayant comme paramètres les couples (*attribut*, *valeur*) donnés. Le système d'information utilisé doit trouver le fragment qui correspond mieux à la présentation.

2.3.7.4 Neutralité vis-à-vis de la présentation

La neutralité d'un modèle de document vis-à-vis de la présentation est fortement liée à sa spécification et à la possibilité de réutilisation des structures définies. Le modèle Z_YX s'appuie sur les *projecteurs* pour offrir un maximum de neutralité par rapport à la présentation finale. Les variables de projection peuvent être utilisées au dernier moment pour associer la structure à un contexte particulier de présentation.

2.3.8 Cadre de travail InfoPyramid

Smith et al. définissent dans [110] un système extensible de transmission du contenu multimédia. Le système utilise un nouveau modèle de données, appelé *InfoPyramid*. Le modèle représente un cadre de travail pour la gestion et la manipulation du contenu multimédia composé d'objets vidéos, images, audio ou texte. *InfoPyramid* manipule les différentes variantes des ressources médias, avec différentes modalités et niveaux de fidélité, et sélectionne les alternatives afin d'adapter le contenu sources à plusieurs terminaux. L'approche adoptée par le modèle *InfoPyramid* se base sur une description hiérarchique des médias afin de faciliter la recherche et l'extraction du contenu.

2.3.8.1 InfoPyramid

Comme le montre la Figure 2.21, *InfoPyramid* manipule différentes variations des ressources média avec plusieurs modalités (vidéo, image, texte et audio) et fidélité (abstraction, compression et variations extensibles) [80]. Le modèle essaie aussi de fournir quelques méthodes de traduction et d'abstraction qui génèrent de nouvelles variantes des ressources médias.

Dans *InfoPyramid*, chaque ressource média est représentée par une case (voir Figure 2.21). Le niveau inférieur représente les ressources avec la plus grande qualité (haute résolution, grande taille, etc.). La qualité se dégrade (faible résolution, compression, etc.) en allant vers les niveaux plus haut. Les cases du même niveau représentent les différents types d'alternatives. Par exemple, dans la Figure 2.21, la case de ressource *texte* représente une alternative à la ressource *vidéo*. Les ressources en haut de *texte* représentent cette ressource avec plus d'abstraction et de compression. La case *audio* représente une forme audio de la ressource *texte*, elle peut être obtenue par exemple par une transformation de synthèse vocale.

Le modèle *InfoPyramid* inclut un ensemble de classes pour le traitement des quatre types de modalités : vidéo, image, texte et audio, et définit des méthodes pour le transcodage des ressources. Ces méthodes sont : la conversion *texte-vers-audio*, le transcodage des images et de la vidéo, l'extraction d'image à partir d'une vidéo et l'abstraction et la traduction du texte. Le modèle se limite seulement au traitement des ressources média qui peuvent être utilisés dans une présentation, mais pas aux documents multimédia qui utilisent ces ressources. En effet, l'approche du modèle offre des possibilités de substitution et de transcodage de quelques types de média indépendamment des relations

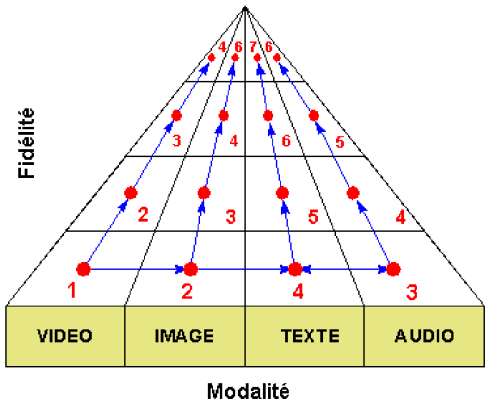

FIG. 2.21 – Le modèle *InfoPyramid*

qui peuvent exister entre les ressources elles-mêmes ou entre les ressource et le scénario de la présentation.

2.3.8.2 La transmission du contenu adapté

InfoPyramid distingue deux types de transmission du contenu :

1. *La transmission adaptative*
2. *Le transcodage en ligne*

2.3.8.2.1 La transmission adaptative Dans la *transmission adaptative*, le serveur utilise le modèle *InfoPyramid* pour gérer et sélectionner les différentes variantes des ressources média. A chaque ressource média on associe une entité *InfoPyramid*, qui n'est autre que la collection des différentes variantes de la ressource.

Afin d'effectuer la sélection des variantes sur la base des entités *InfoPyramid*, le système utilise des *valeurs de contenu* pour chaque ressource média. Ces valeurs peuvent être mesurées automatiquement selon le degré de fidélité de la variante par rapport à la ressource source. La valeur de contenu d'une variante augmente quand la fidélité de la variante vis-à-vis du contenu source diminue. La Figure 2.21 illustre des exemples de valeurs de contenu concernant les différentes variations d'une ressource *vidéo*. Dans cet exemple, la ressource *vidéo* possède la plus grande valeur de contenu. La manipulation de cette ressource *vidéo* selon les axes de fidélité ou de modalité diminue la valeur de contenu.

Le modèle définit quelques règles pour respecter trois types simples de contraintes. Ces contraintes sont relatives à la taille des ressources envoyée, au temps de chargement et à l'espace d'affichage du terminal. Ces règles sont définies comme suit : dans le premier type de contrainte les meilleures variantes des ressources utilisées, dans le document source, sont choisies de manière que la somme des valeurs de contenu soit maximale et la taille du document résultant soit inférieur à la taille autorisée. Dans le deuxième cas, les meilleures variantes sont choisies de manière que la somme des valeurs de contenu soit acceptable[8] et la taille du document résultant soit minimale.

[8] le système suppose qu'une valeur minimale de contenu est donnée

Dans le dernier cas, concernant l'espace d'affichage, les meilleures variantes sont choisies de manière que la somme des espaces d'affichage de toutes les variantes sélectionnées soit inférieure à la taille de l'espace d'affichage du terminal.

Le support de contraintes du modèle est très pauvre. La définition des règles de négociation, sous forme de simples propriétés mathématiques, limite le support des contraintes compliquées. Cette approche peut appliquer des abstractions sur quelques détails de contraintes du terminal et peut donner dans beaucoup de situations un échec de négociation alors qu'un contenu adapté peut être transmis. Par exemple, concernant l'espace d'affichage, il n'est pas nécessaire de considérer la totalité de l'espace d'affichage occupé par les variantes pour voir si le contenu est adaptable ou non. En effet, la gestion de l'espace d'affichage est plus compliquée et ne peut pas se faire indépendamment du scénario de la présentation.

2.3.8.2.2 Le transcodage en ligne Dans *le transcodage en ligne*, *InfoPyramid* est utilisé comme une structure temporaire dans le transcodage des ressources médias vers les modalités et les fidélités les plus appropriées. Le système essaie de définir un processus dynamique qui assure une sélection automatique des méthodes de transcodage pour répondre aux contraintes du client. Afin d'atteindre cet objectif, le modèle décrit les caractéristiques des méthodes de transcodage sur la base de quatre dimensions :

1. modalité en entrée/sortie
2. fidélité en entrée/sortie
3. taille de données en entrée/sortie
4. dimension spatiale-temporelle en entrée/sortie

Par exemple, une méthode de transcodage L_k qui résume des ressources texte à 50%, est décrite dans *InfoPyramid* comme suit :

$$L_k = (L_k^{in}, L_k^{out})$$

Où : $L_k^{in} = (M_k^{in}, F_k^{in}, D_k^{in}, S_k^{in})$, $L_k^{out} = (M_k^{in}, 0.5F_k^{in}, 0.5D_k^{in}, 0.5S_k^{in})$

Cela signifie que la méthode produit en sortie, une ressource de la même modalité que la ressource source, mais avec une réduction de 50% de la taille des données et de la taille spatiale.

Le système utilise ces descriptions afin de sélectionner l'ensemble optimal des méthodes de transcodage à appliquer. L'approche de sélection des méthodes est très similaire à la sélection des variantes d'une ressource média. En effet, le principe est de supposer l'existence de certains paramètres et de définir des fonctions objectifs à minimiser selon les valeurs de ces paramètres.

Par exemple, vis-à-vis du temps d'adaptation, le système associe un paramètre appelé *taux de temps de réponse*, noté ρ_k, pour chaque méthode de transcodage L_k. La valeur du paramètre doit donner une idée sur le temps moyen qu'une méthode de transcodage peut prendre pour adapter une taille donnée de ressource. La fonction objective dans ce cas là, revient à sélectionner les différentes méthodes de transcodage dont la somme totale des valeurs ρ est minimale.

2.3.9 OPES

L'architecture OPES (Open Pluggable Edge Services) [5] est une architecture proposée par l'IETF [58] afin de permettre le déploiement des services appliqués sur les données d'une application par des entités intermédiaires du réseau. L'objectif de cette architecture est de définir un protocole et un ensemble d'APIs pour l'application d'un ensemble de services assurant une transmission efficace d'un contenu complexe et l'utilisation de services relatifs au contenu. Les entités intermédiaires peuvent être des proxy ou des serveurs auxiliaires au système.

Une architecture OPES se base sur trois principaux composants :

1. Les entités : des applications (processus) OPES qui sont utilisées dans le réseau.
2. Les flux : des flux de données générés d'une manière coopérative par les entités OPES.
3. Les règles : qui spécifient quand et comment les services OPES sont exécutés et sur quel flux de données.

Une entité OPES effectue un traitement sur un flux de données entre un serveur de contenu et une application cliente. Une application OPES peut être une application de service qui analyse et éventuellement transforme les données échangées entre le serveur et le client ou un processus qui invoque des applications OPES sur la base de règles OPES. L'architecture OPES ne considère ni le serveur de contenu ni l'application cliente comme une entité OPES, ce qui limite la stratégie de négociation et d'adaptation du contenu à une stratégie basée sur l'utilisation des proxy.

Les règles utilisées dans OPES consistent en un ensemble de conditions et d'actions. Les règles déterminent les services qui vont être appliqués sur un flux de données. Malheureusement, l'architecture OPES ne définit pas de mécanisme ouvert de configuration des règles ce qui représente un aspect très important dans une architecture d'adaptation et de négociation de contenu. La spécification de l'architecture met la configuration des règles en étant la responsabilité des différentes implémentations qui doivent être conformes à quelques exigences relatives à la sécurité et l'authentification de l'usage des services. Les règles sont utilisées pour établir une correspondance entre les requêtes des clients et le serveur d'origine, ou entre les réponses du serveur d'origine et le client. Une fois la correspondance faite, une action spécifiée est exécutée. Cette action peut faire appel à une entité distante (appelée généralement serveur de *callout*) pour faire exécuter un code distant. Le protocole de l'adaptation *ICAP* représente un des protocoles de communication qui peuvent être utilisés pour assurer les interactions entre les entités intermédiaires OPES et les serveurs de *callout*.

2.3.10 Protocole ICAP

Le forum ICAP (Internet Content Adaptation Protocol) [57] propose le protocole *ICAP* pour les applications réseau basées sur des systèmes récents de transmission de contenu. Le protocole est conçu pour assurer une distribution et gestion de cache de contenu sur le Web. ICAP définit la distribution du contenu entre les serveurs d'origine, les proxy de cache et les serveurs ICAP. Ces serveurs sont généralement utilisés pour assurer certaines valeurs ajoutées au contenu source. Ces valeurs ajoutées peuvent inclure le filtrage du contenu, la traduction des langages, l'insertion de contenu, etc. En outre, le protocole considère l'adaptation de contenu afin d'assurer

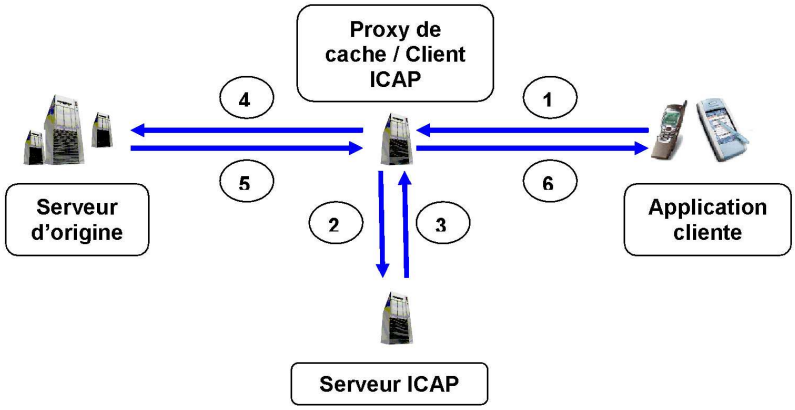

FIG. 2.22 – Architecture ICAP de modification de requête

une transmission adaptée pour certains environnements limités, tels que les systèmes incluant des terminaux mobiles.

Du point de vue des performances, le protocole permet d'alléger la charge des serveurs d'origine et du réseau en affectant la tâche des services ajoutés à des serveurs ICAP. On retrouve aussi ce gain de performance dans le cas des clients ICAP, qui sont des clients capables d'assurer l'adaptation de contenu, ce qui évite d'effectuer l'adaptation par d'autres entités du réseau.

L'approche ICAP peut être appliquée, par exemple, pour dédier une tâche de transformation et de cache du contenu à une entité local du réseau, au lieu de faire appel au serveur d'origine. Du point de vue interopérabilité avec les protocoles existants, ICAP assure une interface standard pour l'adaptation des message HTTP, ce qui peut être bénéfique pour les applications basées sur le modèle HTTP.

2.3.10.1 Architecture

Le protocole ICAP est principalement un protocole basé sur les appels distants de procédures HTTP qui permet à une entité du réseau, un proxy de cache par exemple, de transférer des messages HTTP à une application serveur sans être amené à recharger la mémoire du cache et augmenter le temps de réponse. Cela signifie simplement que l'approche ICAP permet l'envoi de messages HTTP à des serveurs ICAP au service de l'adaptation [31].

Ces services sont souvent fournis par des sites Web qui offrent des applications aux différent clients. Dans les architectures classiques, la manipulation du contenu source, en ajoutant du nouveau contenu par exemple, est généralement faite par le site d'origine ou par le fournisseur d'accès au site. Dans l'approche ICAP, les services ajoutés ciblent mieux le contexte (ou le profil) de l'utilisateur final : sa position géographique, ses centres d'intérêt, etc. ICAP définit trois types d'architectures : l'architecture basée sur la modification de requête, l'architecture basée sur la satisfaction de requête et l'architecture basée sur la modification de réponse.

Dans le premier type d'architecture (voir Figure 2.22), quand une application

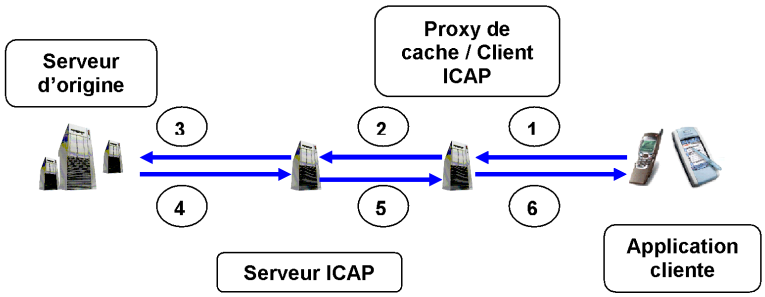

FIG. 2.23 – Architecture ICAP de satisfaction de requête

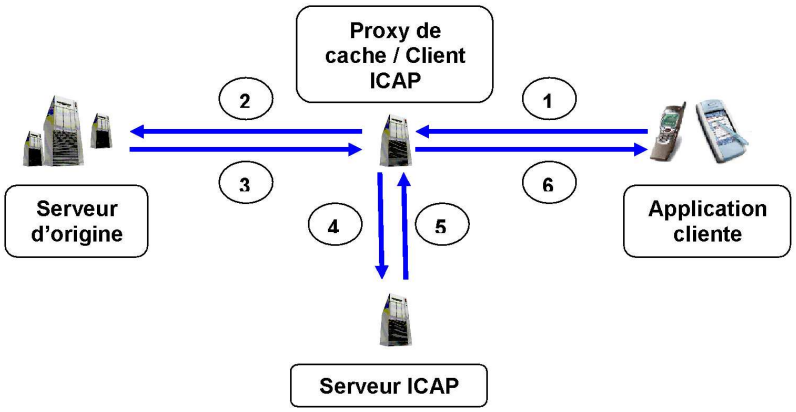

FIG. 2.24 – Architecture ICAP de modification de réponse

cliente envoie sa requête vers un serveur d'origine, la requête est redirigée vers un serveur ICAP à l'aide d'un proxy. Le serveur ICAP modifie le message de requête et renvoie le nouveau message au proxy. Le proxy analyse le message modifié et l'envoie au serveur d'origine. Le serveur reçoit donc la requête modifiée et transmet sa réponse au client à travers le proxy.

Dans le deuxième type d'architecture (voir Figure 2.23), l'application cliente envoie une requête au serveur d'origine. La requête est redirigée vers un serveur ICAP par un proxy. Le serveur ICAP modifie la requête et envoie la nouvelle requête directement au serveur d'origine. Après le traitement de la requête modifiée, la réponse du serveur d'origine est envoyée au client via le serveur ICAP et le proxy.

Dans le dernier type d'architecture, l'application cliente envoie une requête HTTP à un proxy conforme au protocole ICAP. Le proxy envoie la requête au serveur d'origine. Le proxy redirige la réponse du serveur vers un serveur ICAP. Ce dernier exécute la requête ICAP demandée et envoie sa réponse au proxy. Le proxy envoie au client la réponse reçue, qui est éventuellement différente de celle du serveur d'origine.

Dans les différents types d'architecture, les clients et les serveurs ICAP s'échangent les requêtes standard *GET* et *POST* du protocole HTTP. Par exemple, dans le dernier type d'architecture que nous avons vu, le client ICAP utilise une requête *POST* dans laquelle, la requête de l'application cliente et la réponse du serveur d'origine sont encapsulées au début du corps du message HTML. Une autre similitude entre ICAP

```
REQMOD icap://serveur-ICAP.org/serveur
ICAP/1.0
Host: serveur-ICAP.org
Encapsulated: req-hdr=0
GET / HTTP/1.1
Host: www.serveur-origine.com
Accept: text/html, text/plain
Accept-Encoding: compress
Cookie: gh34frtuio@pRRff
If-None-Match: "azzer", "qaserrtttt"
```

FIG. 2.25 – Exemple de requête ICAP

```
ICAP/1.0 200 OK
Date: Tue, 08 Jul 2003 09:07:07 GMT
Server: ICAP-Server-Software/1.0
Connection: close
Encapsulated: req-hdr=0
GET /modified-path HTTP/1.1
Host: www.serveur-origine.com
Accept: text/html, text/plain, image/gif
Accept-Encoding: gzip, compress"
If-None-Match: "azzer", "qaserrtttt"
```

FIG. 2.26 – Exemple de réponse ICAP

et HTTP est que ICAP se base sur le principe d'échange requête/réponse exactement comme HTTP. Cependant, ICAP n'est pas conforme à HTTP et ne peut pas être considéré comme une application basée sur HTTP. Les communications utilisées dans ICAP sont souvent effectuées sous TCP/IP.

Les Figures 2.25 et 2.26 montrent un exemple de requête et de réponse ICAP.

Comme nous l'avons vu, le protocole ICAP définit un cadre de travail pour l'adaptation du contenu sur le Web. Ce cadre de travail se limite à l'échange des messages HTTP et néglige d'autres protocoles qui peuvent transporter et contrôler la transmission des ressources utilisées par le contenu (par exemple les protocoles de transmission des flux de médias). En outre, le protocole définit seulement une manière d'échanger des requêtes d'adaptation ce qui représente la première étape d'un processus de négociation de contenu adapté, mais ne constitue pas une stratégie de négociation complète.

2.3.11 MPEG 21

MPEG-21 [13] est une norme multimédia élaborée par le groupe de travail MPEG (Moving Picture Experts Group : Groupe d'experts sur les images animées). L'objectif est de concevoir une norme qui garantit une utilisation transparente et enrichie de ressources multimédias dans une vaste gamme de réseaux et d'appareils. MPEG-21 permet l'intégration de processus pour créer, manipuler, utiliser, gérer et diffuser des fichiers multimédia. La norme comprend des éléments qui s'appuient sur l'identification et la description de ressources numériques, le traitement et l'utilisation du contenu, la gestion et la protection de la propriété intellectuelle, etc.

La norme vise un modèle multimédia commun qui facilite la coopération entre des infrastructures différentes. Notons que les différences entre systèmes limitent l'interopérabilité mais aussi la protection des ressources (problème de la propriété intellectuelle). Les aspects : IPMP (Intellectual Property Management and Protection), RDD (Rights Data Dictionary) et REL (Rights Expression Language) intégrés dans le travail de MPEG-21 sont dédiés à ce problème [66].

La norme MPEG-21 introduit la notion *d'élément digital* (*digital item*) qui représente une abstraction d'un contenu multimédia incluant différents types d'objets reliés entre eux. Par exemple, une présentation multimédia peut comporter plusieurs discours encodés en plusieurs formats qui correspondent à des terminaux avec des capacités de traitement différentes. La présentation peut aussi inclure d'autres types d'objets tels que des images, des informations sur le droit d'utilisation des images, etc. L'ensemble du contenu est considéré dans MPEG-21 comme un élément digital, où des descripteurs (exemple : les droits d'utilisation des images) sont associés aux ressources qui peuvent être dans notre exemple, les images. Un élément digital est défini par une structure standardisée avec une représentation définie, une identification et une description [13].

MPEG-21 définit aussi le concept d'interaction de l'utilisateur avec les éléments digitaux. L'objectif est de définir les mécanismes nécessaires qui garantissent l'accès, l'échange, l'utilisation et les autres manipulations de l'utilisateur vis-à-vis des éléments multimédia d'une manière efficace, transparente et interopérable. Afin d'assurer cet objectif, le cadre de travail définit les éléments nécessaires et les opérations applicables à ces éléments afin de garantir une chaîne efficace de transmission de contenu. Dans les différentes parties de la spécification de norme, ces éléments sont élaborés en définissant leur syntaxe et la sémantique de leurs différentes caractéristiques.

Dans MPEG-21, un utilisateur est défini comme une entité qui interagit avec l'environnement ou qui utilise un élément digital. Aucune distinction n'est faite entre un fournisseur et un consommateur de contenu. Par conséquent, toutes les entités qui interagissent dans un environnement MPEG-21 sont considérées comme des utilisateurs qui peuvent utiliser le contenu de différentes manières. Dans son niveau basique, MPEG-21 considère un cadre de travail avec un utilisateur qui interagit avec un autre utilisateur dont l'objet de l'interaction est l'élément digital (le contenu). Plusieurs types d'interactions ont été définis, parmi ces types on trouve la création du contenu, l'archivage, la transmission, et la consommation. Ces types d'interaction sont considérés comme différentes sortes d'utilisation et l'entité qui applique une interaction est considérée comme un utilisateur.

2.3.11.1 La déclaration des éléments digitaux (DID)

L'objectif des déclarations des éléments digitaux (DID : Digital Item Declaration) de MPEG-21 est de décrire un ensemble de termes et de concepts afin de former un modèle de définition des éléments digitaux. Le but du modèle est de proposer un cadre de travail qui soit le plus flexible et le plus général possible pour qu'il soit utilisé dans des modèles de niveau supérieur de MPEG-21.

Les déclarations DID de la norme MPEG-21 sont faites à l'aide de :

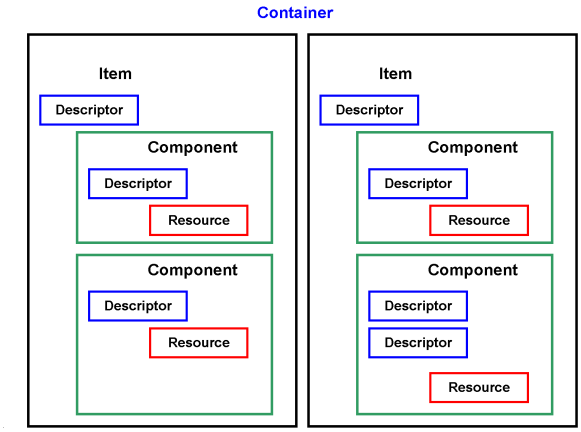

FIG. 2.27 – Déclaration des éléments dans MPEG-21

1. Modèle : Le modèle DID décrit un ensemble de termes et concepts pour la définition d'un élément digital.

2. Représentation : décrit la sémantique et la syntaxe de chaque déclaration d'élément digital.

3. Schéma : un schéma XML pour la représentation XML de la déclaration de l'élément digital.

Le modèle de déclaration des éléments digitaux définit quatorze éléments et inclut les éléments : *container*, *resource*, *component*, *item* et *descriptor*. Un élément *container* (conteneur) est une structure qui permet de grouper un ensemble d'éléments *item* et/ou *containers*. Ce regroupement peut être utilisé par exemple pour former un paquetage logique pour la transmission ou l'échange des éléments groupés. Les éléments *descriptors* (descripteurs) permettent d'étiqueter un *container* avec les informations propres au groupe telles que les instructions de transmission. Un élément *resource* (ressource) représente un objet de contenu tel qu'une vidéo, une image ou un texte. Un élément *component* (composant) lie une ressource avec tous ses descripteurs. Ces descripteurs peuvent contenir des informations de contrôle ou de structure de la ressource (telle que le débit d'une vidéo, des information de cryptage, etc.). L'élément *item* permet de grouper des sous-items et/ou des composants liés par des descripteurs. Les *items* peuvent contenir des éléments de choix qui permettent de sélectionner ou de configurer les *items*. La Figure 2.27 représente un exemple qui montre les éléments les plus importants du modèle de déclaration des éléments digitaux de MPEG-21.

2.3.11.2 L'adaptation des éléments digitaux (DIA)

MPEG-21 vise l'accès universel au contenu au moyen d'une adaptation des éléments digitaux (DIA : Digital Item Adaptation) [87]. Cette approche est illustrée par la Figure 2.28.

MPEG-21 est basé sur l'utilisation d'un module de description et d'adaptation de ressources qui permet de fournir un élément adapté avec une nouvelle description. La description de l'environnement dans MPEG-21 décrit les capacités du terminal, les ca-

ractéristiques du réseau, l'utilisateur et l'environnement naturel comme les conditions physiques de l'entourage de l'utilisateur tel que niveau de bruit, la luminosité, l'heure, la position géographique, etc.

Les capacités du terminal : Les descriptions des capacités du terminal sont définies afin de garantir une compatibilité de formats des média et différentes formes d'adaptation pour les terminaux. Les classes de description suivantes font partie de l'adaptation de MPEG-21 :

- capacités des codecs : spécifie le format que le terminal peut encoder ou décoder,
- capacités des entrées/sorties : inclut une description des capacités d'affichage, d'audio et les différentes propriétés relatives aux modalités d'interaction avec le terminal,
- propriétés de l'appareil : caractérise les attributs relatifs à l'énergie de l'appareil ainsi que les capacités de stockage et les périphériques d'entrées/sorties.

Les caractéristiques du réseau : MPEG-21 considère deux catégories principales : les capacités du réseau et les conditions d'utilisation du réseau. Les capacités du réseau sont décrites par des attributs statiques comme la capacité maximale et minimale de la bande passante.

Les caractéristiques de l'utilisateur : les caractéristiques de l'utilisateur peuvent être classées comme suit :

- les informations de l'utilisateur, les préférence sur l'usage du contenu et l'historique de cet usage,
- les préférences de présentations, telles que la configuration préférée pour l'utilisation d'une ressource audio, la luminosité d'affichage pour une image, etc.
- les caractéristiques d'accessibilité qui sont relatives aux capacités des personnes qui reçoivent le contenu,
- les caractéristiques de la position qui couvrent la description de la mobilité et de la destination dans le temps.

Les caractéristiques de l'environnement naturel : ces caractéristiques concernent deux types de descriptions : 1) la position et le temps, et 2) l'environnement audiovisuel. Le premier type concerne l'endroit où se trouve l'élément digital et le moment de son utilisation ; le second type décrit les attributs audiovisuels de l'entourage de l'utilisateur et qui peuvent influencer la réception du contenu. Par exemple, pour les ressources audio, MPEG-21 décrit le niveau de bruit de l'environnement.

La description de syntaxe Bitstream (BS) : L'objectif de la description de la syntaxe Bitstream (BS) est de permettre à un nœud du réseau ou à un proxy d'effectuer une adaptation de média sans être confronté à la complexité des différents formats des médias. Le processus d'adaptation, basé sur la description BS, est donné comme suit : au début, un utilisateur de l'environnement MPEG-21 génère la description BS du flux d'entrée. La description, écrite en XML, détaille essentiellement l'organisation des différentes couches de données. Avec cette description, un processus d'adaptation peut appliquer une transformation du contenu XML. Le processus d'adaptation peut ensuite générer un flux de données adapté en utilisant la description transformée. Malheureusement, ce mécanisme ne permet d'appliquer que des adaptations très simples tels que le filtrage de données.
Afin de supporter ce type d'adaptation basé sur les BS, MPEG-21 définit deux

mécanismes qui sont : le langage de description de syntaxe bitstream (BSDL), pour la description des formats de codage, et la syntaxe de bitstream générique (gBS) pour la description des ressources binaires de façon indépendante des codecs.

Le terminal et la qualité de service du réseau : L'adaptation des éléments digitaux de MPEG-21 définit un ensemble de mécanismes afin de garantir une stratégie d'adaptation optimale. En particulier, il est possible de spécifier des relations entre les contraintes du terminal et du réseau que les processus d'adaptation doivent satisfaire. Il est également possible de définir des valeurs de qualité basées sur différentes opérations d'adaptation et paramètres d'opérations. Dans ce contexte, MP2G-21 définit de nombreuses contraintes possibles, opérations de contraintes et métriques pour l'expression de relations plus riches.

l'adaptabilité des métadonnées : Cette description indique des informations qui peuvent réduire la complexité de l'adaptation des instances de métadonnées. Ces informations ont été définies pour deux classes d'applications. La première classe concerne le filtrage d'instances de description importante en taille, et la seconde concerne l'intégration d'une ou plusieurs instances de description.

la mobilité des sessions : La déclaration des éléments digitaux (DID) permet de spécifier la construction d'un élément digital d'une manière flexible. Un élément digital peut être déclaré en spécifiant ses ressources, ses méta-données, les différentes relations entre les composants de l'élément, etc. Une DID peut être configurée en utilisant un ensemble de mécanismes de choix et de sélection définis dans le cadre MPEG-21. L'instanciation des choix et des sélections dans une DID est appelée l'état de configuration de l'élément digital. La mobilité de session concerne le transfert des informations de l'état de configuration d'un appareil, utilisant un élément digital, vers un autre appareil. Cela permet l'utilisation de l'élément digital par un deuxième appareil d'une manière adaptée.

2.3.11.3 Bilan

L'adaptation des éléments digitaux de MPEG-21 offre de nombreux mécanismes de description pour garantir une adaptation des éléments de contenu. Ces mécanismes doivent être exploités pour la transmission du contenu MPEG-21 aux terminaux à capacités limités. L'approche MPEG-21 représente une approche d'adaptation efficace mais qui se base essentiellement sur la définition de mécanismes d'adaptation pour un contenu MPEG-21. Afin de considérer la diversité des autres formats existants dans les environnements hétérogènes, une approche d'adaptation doit adopter une stratégie plus globale qui prend en compte non seulement les contenus au format MPEG-21 mais aussi les autres composants tel que le serveur, le client et les contenus de formats différents. De plus, il n'existe pas actuellement d'outils ou de systèmes qui permettent d'exploiter de façon pratique les mécanismes définis par MPEG-21 dans le cadre de l'adaptation et de la négociation de contenu. Les seules applications qui exploitent MPEG-21 se limitent à l'application de quelques mécanismes simples tels que l'extraction des objets [6].

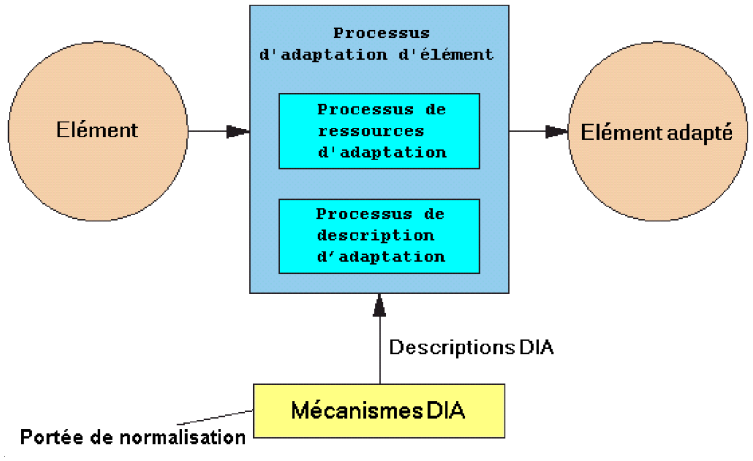

FIG. 2.28 – Architecture d'adaptation de MPEG-21

2.4 Classification des systèmes d'adaptation

Le processus d'adaptation représente un mécanisme qui s'applique sur une forme d'un contenu et produit en sortie une nouvelle forme alternative. L'adaptation peut être simple, comme par exemple une approche de sélection de versions, ou complexe, en changeant les modalités du contenu comme dans l'adaptation d'une représentation textuelle vers une présentation visuelle.

Dans cette section, nous discutons les différents niveaux dans lesquels le processus d'adaptation de contenu peut être appliqué. Nous distinguons trois niveaux possibles selon le chemin de transmission du contenu du serveur vers le client cible : niveau serveur, niveau client et niveau intermédiaire. Les processus d'adaptation peuvent coopérer ; par conséquent le contenu adapté peut être le résultat de plusieurs tâches d'adaptation appliquées à plusieurs niveaux.

2.4.1 Adaptation côté serveur

L'adaptation du contenu peut être effectuée du côté du serveur si le client cible présente certaines limitations (capacités de traitement, méthode d'accès, taille d'écran, mémoire, etc.) et n'est pas capable de recevoir ou d'adapter le contenu source. L'adaptation côté serveur peut aussi être appliquée afin de respecter le contexte de la transmission du contenu, par exemple pour optimiser l'utilisation des ressources du réseau si la bande passante est limitée ou pour réduire le temps de réponse et les délais de transmission du contenu. Le scénario d'adaptation, c'est-à-dire la manière dont les tâches d'adaptation sont organisées et exécutées, dépend du protocole de communication utilisé pour la transmission du contenu. Dans certaines situations, et afin d'assurer une adaptation efficace, le serveur exige un retour de rapport sur la transmission. On trouve ce genre de mécanisme dans certains protocoles tels que RTP, RTCP et RSVP. Plusieurs architectures adoptent l'adaptation côté serveur [55] [8] [130] puisqu'elle est facile à mettre en œuvre et n'implique que l'entité serveur du réseau.

2.4.1.1 Sélection de versions

Ce mécanisme d'adaptation consiste à choisir la meilleure version du contenu. La sélection est appliquée sur les versions disponibles du côté du serveur sur la base de la

requête du client. Le processus de sélection du serveur analyse les caractéristiques des variantes et détermine si une version respecte ou pas le contexte du client cible.

2.4.1.2 Adaptation

Le serveur peut appliquer une méthode d'adaptation sur un contenu demandé par un client en utilisant les différentes techniques disponibles. De telles techniques peuvent être des transformations structurelles, des adaptations de ressources médias, etc. (voir Section 2.5)

2.4.1.3 L'utilisation des méta données

Le serveur peut associer certaines méta données relatives à l'expiration d'une ressource demandée afin d'éviter le téléchargement fréquent d'une ressource surtout quand le réseau est limité ou ses performances sont dégradées. Ces informations peuvent guider la réception des ressources appliquées par le client ou par les systèmes intermédiaires.

2.4.1.4 La décomposition

Le serveur peut appliquer une décomposition du contenu source pour l'adapter aux limitations physiques du terminal[9] et afin de faciliter la navigation et l'accès au contenu [70] [42]. Le processus de décomposition du serveur peut utiliser les informations fournies par l'auteur du contenu (les éléments de HTML par exemple) et les informations du contexte du client.

2.4.2 Adaptation côté client

L'adaptation du contenu peut être assurée au niveau client si ce dernier est capable d'effectuer certaines tâches sans avoir besoin de l'aide des autres entités du réseau. Les techniques d'adaptation peuvent être appliquées complètement ou partiellement selon les caractéristiques du client telles que sa capacité de mémoire ou sa vitesse de traitement.

2.4.2.1 Retaillage d'images

Le retaillage des images est généralement appliqué afin que les ressources images utilisées dans le contenu et reçu par le client, soient affichées dans un espace limité. Le retaillage d'images peut se baser sur le ré-encodage des ressources, la sélection des versions ou l'utilisation de modèles de contenu scalables, comme SVG.

2.4.2.2 Substitution de police de caractères

A cause de leurs capacités de traitement, les terminaux mobiles peuvent supporter un ensemble réduit de polices de caractères. Le client peut donc appliquer une substitution de police de caractères vers une autre police supportée.

2.4.2.3 Transformation et transcodage

Certains terminaux peuvent être capables d'appliquer des transformations ou un transcodage du contenu par exemple en analysant des feuilles de style XSLT ou en

[9] par exemple, la taille d'une carte WML pour un téléphone mobile

exécutant certains scripts. Ces techniques peuvent inclure la conversion ou la traduction du contenu, le filtrage, la sélection, etc. Afin d'appliquer ce genre de techniques, le client doit être capable d'analyser les règles du langage de transformation ou d'appliquer les directives des programmes de transformation. Géneralement, les techniques de transformation au niveau client sont réduites et se limitent à des domaines d'applications bien particuliers.

2.4.2.4 Adaptation des interfaces

L'application cliente peut appliquer une adaptation des interfaces utilisées dans un contenu afin de les rendre plus faciles à utiliser dans un espace d'affichage limité. L'adaptation des interfaces peut être assurée en appliquant la réduction de la partie visuelle des interfaces, le changement des positions des contrôles, la transformation des types des contrôles ou la décomposition des interfaces en répartissant les différents contrôles sur plusieurs interfaces.

2.4.3 Adaptation côté proxy

Les adaptations intermédiaires du contenu sont généralement appliquées dans les architectures à base de proxy. Ces architectures se basent sur l'ajout d'une troisième entité entre l'ensemble des serveurs et l'ensemble des clients. Un proxy peut être vu comme un processus qui reçoit le contenu, à la place du client, et qui applique un certain traitement sur le contenu avant de le transmettre au client cible. Ce type d'architecture représente une approche efficace pour traiter le problème de l'hétérogénéité des clients et des serveurs. En effet, dans une architecture à base de proxy, la plate forme du réseau n'est pas modifiée et les caractéristiques de l'environnement déjà existantes sont préservées. Ce type d'architecture peut être très compliqué en incluant plusieurs proxy dédiés pour des tâches particulières telles que le transcodage de la vidéo et des images, la diffusion de la vidéo, etc.

2.4.3.1 L'adaptation intermédiaire

Le proxy est l'entité responsable de l'extraction du contexte et des requêtes des clients et d'exécuter les adaptations possibles sur le contenu transmis par le serveur. Le contenu adapté est transmis au client en respectant son propre contexte et ses limitations. Le tâche du proxy peut être complètement transparente par rapport au reste des entités du réseau. Notons que dans ce type d'architecture, la gestion des variantes du contenu est plus complexe que dans le cas de l'adaptation au niveau du serveur. En effet, les variantes ne sont pas disponibles sur le proxy, par conséquent le contrôle du proxy vis-à-vis des variantes disponibles et de leurs caractéristiques reste limité. Ce problème peut être résolu en faisant appel à un échange supplémentaire de messages entre le proxy et le serveur d'origine. Cependant, ce type d'échange peut dégrader les performances.

2.4.3.2 La sélection du serveur

Un des types d'adaptation qui puisse être assuré au niveau du proxy est la sélection et la redirection des serveurs, par exemple vers le meilleur serveur miroir qui peut répondre à la requête d'une application cliente. Certains serveurs disposent d'une variété de miroirs qui peuvent stocker le contenu et le transmettre sans formes. La sélection du proxy se base généralement sur la connectivité du terminal, les capacités

et les préférences du client, la position géographique du serveur choisi, etc. Une fois la sélection faite, le proxy applique une conversion entre les URIs du client et les ressources du réseau.

2.4.3.3 L'adaptation du protocole de transmission

L'adaptation du proxy peut concerner le protocole utilisé dans la transmission du contenu adapté. Après l'application d'une adaptation particulière, par exemple une transformation HTML vers WML, le proxy peut contrôler la manière dont le contenu adapté est transmis au terminal cible. En effet, le proxy peut utiliser un nouveau protocole, différent du protocole d'origine utilisé entre le proxy et le serveur. Ce type de proxy est appelé en général, *proxy de protocole*. Le proxy peut aussi contrôler la transmission du contenu en appliquant des algorithmes de compression afin de réduire la taille du contenu transmis ou en intégrant plusieurs objets dans la même ressource afin de minimiser l'échange de requêtes dans le réseau.

2.5 Quelques techniques d'adaptation

2.5.1 Transformation structurelle

Cette catégorie de techniques concerne les transformations appliquées sur l'organisation globale ou l'arbre logique du document. Quelques exemples de telles transformations [74] [73] sont la transformation des documents HTML vers des documents XHTML Basic pour les terminaux mobiles, le filtrage des documents HTML, la transformation d'un contenu textuel écrit en XML vers une représentation graphique en SVG, etc. La transformation structurelle peut préserver les ressources médias utilisées par le document source comme elle peut les filtrer. Dans un système avancé, la transformation structurelle peut aussi lancer une transformation externe de médias, afin d'adapter les ressources au contexte cible.

2.5.1.1 XSLT

Le langage XSLT [135] permet la définition d'un ensemble de règles (appelé feuille de transformations) permettant de transformer un document XML en un autre document. Ce nouveau document peut être un document XML, HTML ou textuel. Beaucoup de travaux dans le domaine de l'adaptation utilisent XSLT pour assurer la partie restructuration du contenu. Dans [92], XSLT est utilisé pour assurer la réorganisation de la structure spatiale d'un document. Le langage XSLT peut aussi être utilisé pour créer des documents pour différents clients. Cela peut être fait en développant plusieurs variantes de feuilles de transformation XSLT pour chaque terminal cible, tout en gardant le même document cible. Dans ce cas, la gestion des versions concerne les feuilles de styles et non pas le document lui-même. Cette approche a été utilisée dans certains travaux comme [126], où plusieurs feuilles de transformation sont exploitées pour engendrer des documents multimédia destinés à différents terminaux.

2.5.2 Transformation des médias

Dans cette catégorie, on trouve les méthodes de transformation qui concernent l'adaptation et l'encodage des médias. Par exemple l'adaptation des images et de la

vidéo en appliquant une réduction de couleurs ou de niveau de gris, un retaillage ou une conversion de format d'encodage. Cette catégorie de transformation s'appliquent au niveau bas de l'encodage des ressources médias et nécessite la connaissance approfondie de l'encodage source et cible.

Beaucoup de travaux ont développé des techniques et des applications pour l'adaptation des ressources médias, tels que l'adaptation des images pour les terminaux mobile [23], l'adaptation de la vidéo dans les environnements mobiles [26], etc.

2.5.3 Adaptation sémantique

Dans plusieurs situations, et afin d'assurer une adaptation qui produit un contenu cohérent, les techniques d'adaptation doivent prendre en compte l'aspect sémantique du contenu source. La sémantique d'un contenu dépasse l'encodage ou l'organisation structurelle du contenu en associant un sens aux différentes parties du contenu et aux relations qui peuvent exister entre ces parties et les objets utilisés dans le contenu. Une bonne connaissance de la sémantique du contenu permet la définition de techniques d'adaptation plus évoluées. Dans ce contexte, l'adaptation contextuelle du Web représente un des importants défis du Web sémantique [128]. Généralement, le contexte est exprimé dans un langage, tel que RDF [101] qui est recommandé par la communauté du Web sémantique. Les attributs et les valeurs du contexte sont ensuite extraits et utilisés pour faire une correspondance entre le contexte et le contenu à personnaliser.

Dans la littérature de l'adaptation sémantique, on trouve beaucoup d'efforts qui ont été faits dans ce domaine, tel que le projet Kontii [118] qui vise à définir un cadre pour la conception et l'utilisation de profils des applications ; ou le cadre de travail présenté dans [34], qui définit une approche pour l'adaptation sémantique en focalisant sur la dimension temporelle des documents multimédia.

2.6 Conclusion

Dans ce chapitre, nous avons passé en revue plusieurs approches relatives au problème de l'adaptation et de la négociation du contenu. De cette étude il ressort les lacunes suivantes :

- L'absence d'une architecture d'adaptation et de négociation complète qui prend en compte les caractéristiques des environnements hétérogènes et les contraintes de la diversité des clients qui existent ;

- La considération du contexte est partielle et ne prend pas en compte toutes les entités qui peuvent influencer le processus de transmission d'un contenu adapté ;

- Pour le moment, il n'existe pas de protocole de négociation et de traitement des caractéristiques de l'environnement hétérogène ce qui rend les possibilités d'adaptation limitées.

Une architecture d'adaptation et de négociation complète doit donc couvrir différents aspects tel que la considération de toutes les entités de l'environnement, la définition de la manière dont ces entités peuvent interagir au profit de l'adaptation

et les techniques d'adaptation qui permettent enfin de transformer le contenu de son état d'origine vers un nouvel état plus conformes aux caractéristiques du client cible. La considération de ces aspects, sous forme d'une nouvelle architecture, fait l'objet du chapitre suivant.

Chapitre 3

Architecture et entités d'adaptation et de négociation du contenu de NAC

Résumé

L'objectif de ce chapitre est de présenter l'architecture NAC et de détailler les principales fonctionnalités des différentes entités utilisées. Le chapitre explique les organisations possibles de l'architecture et détaille comment les entités de l'architecture coopèrent entre elles afin d'atteindre l'objectif d'une adaptation contextuelle basée sur les contraintes de l'environnement hétérogène.

Contenu

3.1 Introduction

Ce chapitre présente l'architecture d'adaptation et de négociation NAC (Negotiation and Adaptation Core) qui assure, dans un environnement hétérogène, une transmission de contenu adapté aux contraintes du contexte du client cible.

L'approche adoptée dans la conception et la définition de l'architecture NAC consiste à définir des entités qui coopèrent pour transmettre des contenus qui respectent les caractéristiques et les limitations de l'environnement. Chaque entité de l'architecture a un rôle bien défini et assure un ensemble de fonctionnalités qui peuvent être exploitées à tout moment par le système global.

Globalement, NAC inclut deux grandes parties qui sont : *la description du contexte* et l'adaptation pour la satisfaction du contexte. La description du contexte assure une formulation des caractéristiques de l'environnement et de ses différents composants. Tout composant concerné par la transmission d'un contenu, qui a été demandé par un client, peut être l'objet d'une description. Ces descriptions sont utilisées par le système d'adaptation afin d'appliquer une adaptation de contenu sur la base des contraintes posées.

La validation de cette approche repose principalement sur la réalisation d'une architecture complète dont nous décrivons les principaux objectifs et fonctionnalités dans la section suivante.

Ce chapitre est consacré à la présentation des entités de l'architecture et à leurs principales fonctionnalités. Comme nous allons voir, NAC représente une architecture flexible et supporte plusieurs organisations, selon l'affectation du module d'adaptation et de négociation. Le chapitre discute aussi l'aspect adaptation de NAC sur les deux plans de la description de l'environnement et de l'application des mécanismes d'adaptation contextuelle.

3.2 L'architecture NAC

Avant de donner une vue plus détaillé sur l'architecture NAC et ses différents composants, nous discutons d'abord les grands principes qui ont présidé à la conception de l'architecture. Nous présentons ensuite deux vues d'ensemble de NAC, une vue selon les principales fonctionnalités et une vue en termes de composants. Les fonctionnalités fondamentales de l'architecture font l'objet des chapitres 4, 5 et 6.

3.2.1 Principes

L'approche adoptée dans la conception de l'architecture NAC essaie de réaliser l'objectif de l'accès universel au contenu indépendamment de l'endroit, des méthodes d'accès et du terminal utilisé. L'approche tente d'assurer une utilisation *automatique* d'un contenu distant par une variété de clients qui peuvent être très différents en termes de capacités de traitement, format supportés, protocole de communication, etc.

NAC vise à assurer un système d'accès au contenu utilisé par l'application cliente. Par ailleurs, le système tente de fournir une analyse et une gestion efficace du contexte de l'environnement dont l'état peut changer d'un instant à un autre. L'aspect adaptation du système est donc confronté aux contraintes causées par l'état du système, la diversité du contenu existant dans le réseau et des capacités et préférences de l'utilisateur final.

L'accès universel du Web nécessite d'effectuer un traitement d'adaptation sur le contenu avant de procéder à la transmission du contenu adapté. Le contenu peut

provenir de n'importe quel serveur du réseau qui peut donc ne disposer d'aucun moyen d'adaptation. Le processus d'adaptation peut donc être intégré au niveau d'une troisième entité entre l'application cliente et le serveur du contenu. Cette entité est appelée généralement *proxy*. Ce qui est intéressant dans une approche qui utilise une entité intermédiaire est que le proxy peut inclure tous les traitements nécessaires pour assurer une transmission de contenu qui respecte les contraintes du client. En outre, le proxy est indépendant du serveur du contenu et il peut donc être utilisé pour adapter tout contenu du réseau. Il est clair que rien n'empêche dans de telles architectures, d'utiliser plusieurs proxy qui coopèrent entre eux.

Vis-à-vis de l'analyse du contexte, le proxy représente l'entité idéale pour analyser le contexte de l'environnement. En effet, les caractéristiques de toutes les entités de l'environnement peuvent influencer l'accès au contenu par l'application client. Dans ce cas, le proxy joue le rôle d'un processus central qui tente de rassembler une image globale de l'environnement incluant au minimum les descriptions des entités concernées par l'accès au contenu à un moment donné. En confiant au proxy les traitements relatifs à l'adaptation et à la négociation du contenu on allège la tâche du reste des composants du réseau, ce qui augmente les performances du système.

NAC essaie d'exploiter les avantages de ce type d'organisation par l'utilisation d'un module proxy tout en donnant la possibilité d'intégrer ce module au niveau du serveur de contenu. Cette possibilité est très intéressante dans certains types d'applications (par exemple, pour activer l'adaptation au niveau d'un serveur particulier) ou pour améliorer les performances du système dans certains cas d'adaptation, comme par exemple dans la gestion des versions.

Du côté de la description et de la gestion du contexte de l'environnement, l'approche déclarative des contraintes utilisateur et de son environnement permet le traitement automatique des descriptions. En effet, à partir des ensembles de descriptions des profils, cette approche permet une extraction facile des caractéristiques nécessaires pour l'adaptation du contenu dans un contexte particulier. L'organisation sémantique des contraintes de l'environnent en utilisant des modèles de description tel que RDF [101], permet d'automatiser le traitement des contraintes et, par conséquent, d'automatiser l'exécution des règles et des processus d'adaptation qui dépendent des caractéristiques du client et de son contexte global. Un exemple typique de ces traitements, est l'adaptation du contenu en fonction de la langue de l'utilisateur. Du point de vue de la description, le modèle adopté spécifie les valeurs des entités sémantiques prédéfinies, telle que la langue de l'utilisateur. Le processus d'adaptation exécute des traitements liés à la sémantique et aux valeurs de ses entités données par les profils, par exemple, le choix d'une version correspondante à une langue donnée ou la traduction en temps réel.

La sauvegarde de profils statiques, qui décrivent l'ensemble des clients et des caractéristiques du système global (serveurs, connexions, réseau, etc.), au niveau de l'entité qui assure l'adaptation du système hétérogène (le proxy ou le serveur) améliore les performances du système d'adaptation pour que les processus d'adaptation exploitent directement le contenu des profils. Dans un système hétérogène, il est fréquent que de nouveaux terminaux apparaissent et changent de configuration matérielle ou logicielle. Il est aussi fréquent que l'utilisateur change ses préférences (langue, style du contenu, filtrage, etc.) vis-à-vis du contenu demandé. Le système du cache des profils peut donc être augmenté par des mécanismes qui assurent plus d'interactions

orientées négociation entre le client et le serveur. Qui dit interaction, dit protocole de communication qui assure une sorte de dialogue sous forme de question/réponse entre les deux parties de la négociation. L'utilisation d'un protocole simple et efficace dédié à la négociation est donc essentielle. Comme il existe une grande diversité de protocoles de communication pour l'accès au contenu (HTTP, RTP, RTSP, RSVP, etc.), il est intéressant de séparer les deux aspects de négociation et de transfert. Cette idée de base permettra aussi d'utiliser le même protocole de négociation dans des modes hétérogènes d'accès au contenu. L'utilisation du protocole de négociation doit être optionnelle pour les terminaux qui ne peuvent pas supporter l'exécution de protocoles supplémentaires autres que leurs propres protocoles de transfert.

Maintenir les descriptions des caractéristiques du système et de ses différentes entités constitue une tâche importante dans un système d'adaptation et de négociation de contenu. Afin d'assurer des meilleures performances du système et d'alléger les traitements des différents composants du système, le principe de répartition de tâches dans un système distribué représente une approche efficace. En particulier, la gestion du contexte peut être dédié à une entité du système distribué, qui assure tous les traitements relatifs aux profils : la sauvegarde, l'interrogation et la mise à jour des profils. Ici encore, un mécanisme d'interaction, entre les entités consommatrices des descriptions et les gestionnaires de ces descriptions, est nécessaire. L'approche des services Web, s'avère intéressante pour assurer un tel objectif. En effet, les traitements des profils peuvent être vus comme un service (fourni par une entité du réseau) qui est utilisé par les processus d'adaptation. Un service peut donner le résultat d'une interrogation de profil, une extraction d'une caractéristique en relation avec le contenu demandé par le client, l'extraction de plusieurs parties des profils, etc. L'entité qui propose ces services peut aussi effectuer des modifications sur les descriptions, telle qu'une mise à jour de la base des profils après une simple requête de l'application cliente. L'approche des services Web assure plus d'automatisation dans le processus d'adaptation. En effet, des interactions de demande et d'utilisation de services peuvent être effectuées d'une manière complètement automatique entre les processus d'adaptation et les fournisseurs de services.

Du côté de l'adaptation du contenu pour différents contextes, les processus d'adaptation essaient généralement d'appliquer une réorganisation et une restructuration du contenu original. Le format et la structure d'origine du contenu joue donc un rôle principal dans le processus d'adaptation. Afin de faciliter ce processus, le contenu ne doit pas être lié, au moment de sa création, à une plate-forme particulière. Le modèle du contenu doit éviter d'inclure des informations spécifiques à des caractéristiques particulières telle qu'une résolution donnée, une grande taille d'écran, etc. Ces principes relatifs aux modèles de contenu sont appelés : *principes d'indépendance vis-à-vis des terminaux*. Suivre ces principes dans la conception et l'adoption de modèles de contenu facilite l'application des processus d'adaptation et rend l'adaptation plus universelle.

L'architecture NAC part de ces différents constats et principes pour proposer une solution au problème de négociation et d'adaptation du contenu dans les systèmes hétérogènes. NAC tire avantage des technologies et des standards existants pour l'adaptation et la négociation du contenu en proposant un noyau d'architecture flexible et extensible. L'architecture définit un ensemble d'entités et associe un ensemble de fonctionnalités à chacun des composants du système. Ces entités sont organisées dans une architecture dont les différents composants coopèrent en se basant sur la

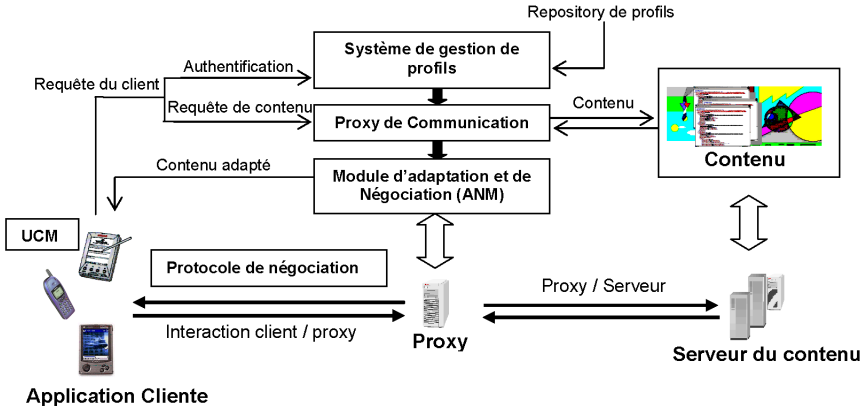

FIG. 3.1 – Organisation générale de l'architecture NAC

description et la gestion du contexte, les techniques d'adaptation et les protocoles de communication et de négociation.

3.2.2 Architecture générale

Dans cette section nous allons présenter l'architecture générale de NAC (Négociation and Adaptation Core). Cette architecture est organisée sous forme de cinq entités qui coopèrent. (voir Figure 3.1) :

- **Le proxy de communication** : Il assure l'accès au contenu et la communication qui se déroulent classiquement entre le client et le serveur. Le proxy reçoit les requêtes des applications clientes et les réponses des serveurs de contenu. Il assure la transmission des requêtes des clients aux serveurs et de la même manière les réponses des serveurs aux applications clients. Souvent, ces réponses sont adaptées ou modifiées avant qu'elles soient délivrées à leurs destinations finales. En réservant un port additionnel, le proxy assure aussi la communication orientée négociation avec un module existant côté client (le module UCM).

- **Le module d'adaptation et de négociation (ANM)** : Il assure l'adaptation et la négociation du contenu de l'architecture. Cela est effectué grâce à l'application d'un ensemble de méthodes de transformation structurelle et d'adaptation de contenu. L'adaptation est dans la majorité des cas dynamique et dépend des valeurs que prennent les dimensions du contexte telles que : l'application cliente utilisée, les formats acceptés par le client, la taille d'écran du terminal, etc. Le module coopère avec les autres entités de l'architecture afin de prendre la meilleure décision de négociation : choix de version de contenu, choix de méthode d'adaptation, choix de méthode de transmission du contenu final, etc.

- **Le module du contexte de l'utilisateur (UCM)** : Il représente un module utilisé côté client afin d'assurer une négociation avancée du contenu. Grâce à l'UCM, le terminal peut effectuer une configuration du proxy et du port d'autorisation utilisés dans la négociation. L'UCM participe dans la négociation du contenu d'une manière complètement indépendante de l'application cliente

utilisée. Cela veut dire que le terminal peut changer d'application cliente ou utiliser plusieurs applications cliente en même temps tout en gardant la même configuration de l'UCM. Ce module permet aussi de sélectionner le profil client et de le changer à n'importe quel moment. Comme nous allons voir, la participation de l'UCM dans la négociation est faite d'une manière très optimisée afin d'éviter la dégradation des performances des terminaux qui sont souvent limitées.

– **Le protocole de négociation** : Grâce à ce protocole, une stratégie de négociation avancée est assurée. Le protocole de négociation définit un mode d'interaction entre le module UCM côté client et le processus de négociation du module ANM. Cette interaction est définie sous forme de requêtes et de réponses. Par exemple, le processus de négociation peut interroger le terminal sur ses capacités et ses préférences courantes ou pour tester si le profil client a subi un changement ou non.

– **Le système de gestion de profils** : Il assure l'analyse et la gestion des descriptions du contexte (la description du client, du contenu, des méthodes d'adaptation, etc.) au profit de l'adaptation du contenu. Les contraintes des profils sont exploitées afin que les processus d'adaptation puissent être sélectionnés et exécutés au mieux. Le système inclut aussi l'exploitation des services de l'architecture NAC pour assurer l'extraction des profils, des parties de profils ou des caractéristiques élémentaires d'un profil donnée. Ce système inclut aussi un mode d'interaction optimisée avec les repository de profils qui peuvent maintenir des bases de profils du contexte.

Un terminal qui fait partie de l'environnement hétérogène utilise une application cliente pour pouvoir accéder et utiliser un contenu qui existe dans le réseau. Le contenu peut être simple, par exemple sous forme d'un contenu textuel ou d'une page HTML, ou complexe, par exemple sous forme d'une présentation multimédia qui utilise des ressources média telles que des vidéos de haute résolution, des images, etc. Lors de l'accès au contenu, le proxy de communication reçoit la requête de l'application cliente à travers le port de communication. Le proxy essaie d'identifier le profil courrant du terminal et son contexte grâce au protocole de négociation qui peut interroger directement le module UCM en cas de changement de profil ou en cas de besoin d'une propriété particulière du profil de l'utilisateur. La connaissance du contexte client peut aussi faire appel au système de gestion de profils afin d'exploiter les services des repository de profils existants dans l'architecture. Le contexte de l'application cliente, qui est relatif à la requête courante, représente un ensemble de contraintes que le module de négociation et d'adaptation ANM doit prendre en compte avant la transmission du contenu final. Cet ensemble de contraintes est construit à l'aide du système de gestion de profils qui se base sur le contenu des profils disponibles. Le traitement du module ANM peut varier selon la nature des contraintes, les méthodes d'adaptation disponibles, les versions disponibles du contenu demandé et les caractéristiques du contenu source et du serveur qui détient ce contenu. A n'importe quel moment, et afin de faire la correspondance entre les caractéristiques du contenu source, les capacités du système d'adaptation et les exigences du terminal, le module ANM peut enrichir son ensemble de contraintes à l'aide du protocole de négociation et des respository des profils. ANM essaie d'établir un *graphe d'adaptation* qui liste toutes les étapes et techniques d'adaptation possibles qui permettent de démarrer du contenu source non adapté et d'arriver, après l'application des techniques

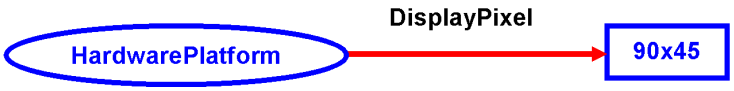

FIG. 3.2 – Représentation graphique d'une déclaration RDF

d'adaptation, à un contenu adapté respectant l'ensemble des contraintes relatives au contexte de la requête de l'application cliente. Le module essaie de trouver le chemin, composé d'un ensemble de tâches d'adaptation, qui respecte le mieux le contexte de l'environnement. Le module ANM applique les étapes d'adaptation, et grâce au proxy de communication le contenu adapté est transmis à l'application cliente.

3.2.3 Vue d'ensemble sur le système de gestion de contexte

La gestion du contexte de l'architecture NAC inclut la définition d'un **modèle de description** des caractéristiques de l'environnement du terminal et du réseau utilisé, **le stockage des descriptions** et le **traitement** local et distribué de l'ensemble des descriptions disponibles.

Le modèle de description manipule un ensemble de caractéristiques, ou des profils, décrits selon un format déclaratif sous forme d'un marquage descriptif. Le modèle de description assure la description des caractéristiques statiques et dynamiques. Les profils peuvent être créés par édition d'un auteur ou par génération automatique à partir du contexte de l'environnement.

La gestion du contexte utilise un modèle de description basé sur plusieurs extensions des cadres de travail CC/PP [43] et RDF [101] définis par le consortium W3C. CC/PP représente un cadre de travail, basé sur XML, qui permet de décrire les caractéristiques logicielles et matérielles d'un terminal mais aussi les informations relatives aux préférences de l'utilisateur. CC/PP se base sur RDF, un modèle pour l'utilisation des méta données sur le Web.

Le modèle de description de NAC, appelé UPS (schémas universels pour la description des profils) [77], utilise RDF pour représenter les différents profils de l'environnement hétérogène en se basant sur un modèle de données. Formellement, RDF peut être définit par :

- Un ensemble de *Ressources*
- Un ensemble de *Littéraux*
- Un sous-ensemble de *Ressources* appelé *Propriétés*
- Un ensemble *Déclarations* qui inclut des triplets de la forme : (p, s, o), où p représente une propriété, s est une ressource et o est une ressource ou un littéral.

Un ensemble des déclarations peut être représenté par un graphe étiqueté dont les ressources et les littéraux sont représentés par des nœuds. Un triplet (p, s, o) est représenté par les deux nœuds s et o liés par une arête, étiquetée par p, dont l'origine est s et l'extrémité est o. Un triplet (p, s, o) peut être interprété comme suit : o est la valeur de p pour s, ou de gauche à droite, s a une propriété p avec une valeur o, ou encore le p de s est o. La Figure 3.2 donne une représentation graphique de la

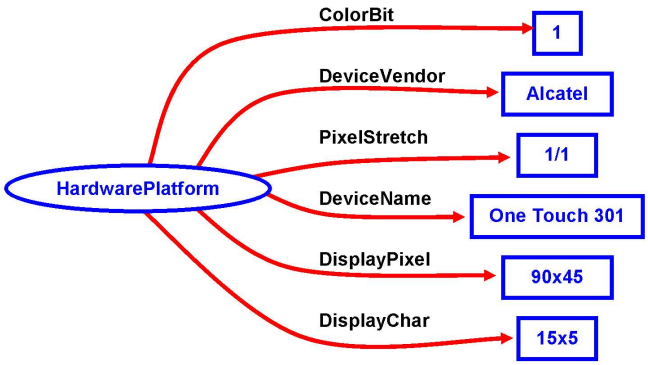

FIG. 3.3 – Représentation graphique d'un composant UPS

déclaration HardwarePlatform extraite d'un profil écrit en UPS [77] (voir le profil "ups-client-profile9.rdf" du repository de profils de NAC [72]). La déclaration donne, entre autre, l'information suivante : la taille du dispositif d'affichage de la plate-forme matérielle est de "90x45".

Le modèle de données des contraintes UPS exprimées par l'ensemble des triplets des profils peut être représenté dans d'autres formats. Par exemple, en utilisant le format N-Triples [59] qui est une représentation textuelle linéaire des graphes RDF. La représentation N-Triples de la déclaration (DisplayPixel, HardwarePlatform, "90x45") est donnée par :

```
<HardwarePlatform> <DisplayPixel> "90x45" .
```

Comme nous allons voir dans le chapitre 4, le modèle de description UPS inclut la définition des profils client, ressources client, instances de document, ressources de document, méthodes d'adaptation et réseau. La Figure 3.4, montre une partie d'un profil client écrit en UPS [72]. UPS adopte un principe qui consiste à définir des composants séparés pour décrire les différents aspects de l'environnement. Un composant inclut différents types de contraintes qui peuvent être représentées par des triplets RDF de la forme (p, s, o), où p est la propriété à décrire, o est la valeur de la propriété et s représente le composant du profil UPS. La Figure 3.4 représente le composant "HardwarePlatform" (plate forme matérielle) d'un profil client UPS.

Ce mémoire inclut une évaluation de performances vis-à-vis de l'utilisation du modèle de description UPS.

Pour le stockage des descriptions, l'architecture NAC permet de sauvegarder les profils à trois niveaux : niveau client, niveau proxy et niveau repository de profils. Quel que soit le niveau de sauvegarde, les caractéristiques des profils - nécessaires pour l'adaptation - doivent être communiquées aux processus d'adaptation. Cette communication nécessite un échange de messages à travers le réseau dans le cas où les profils sont maintenus par le client (grâce au module UCM) ou par le repository de profil. Le premier type d'échange de message fait appel au protocole de négociation appliqué entre le module ANM et le module UCM, alors que le deuxième type d'échange fait appel à l'utilisation des services Web proposés par le repository de profils. Les échanges réseau sont optimisés de manière à véhiculer uniquement l'information nécessaire

```
1:  <ccpp:component>
2: <rdf:Descriptionrdf:ID="HardwarePlatform">
3: <neg:DeviceName>One Touch 301</neg:DeviceName>
4: <neg:DisplayPixel>90x45</neg:DisplayPixel>
5: <neg:DisplayChar>15x5</neg:DisplayChar>
6: <neg:ColorBit>1</neg:ColorBit>
7: <neg:DeviceVendor>Alcatel</neg:DeviceVendor>
8: <neg:PixelStretch>1/1</neg:PixelStretch>
9: </rdf:Description>
10: </ccpp:component>
```

FIG. 3.4 – Exemple de représentation de profil en UPS

quand il le faut.

Le traitement des descriptions consiste à extraire les informations appropriées et à faire la correspondance entre ces informations. L'extraction des informations peut être statique ou dynamique. L'extraction statique fait appel à une interrogation paramétrée de la base des profils. Cette interrogation est appliquée d'une manière locale ou distante selon le niveau de sauvegarde adopté. Les paramètres de l'interrogation dépendent du contexte courant de la requête émise par l'application cliente. En effet, les caractéristiques du contexte de la requête (type de requête, format du contenu demandé, application cliente utilisée, etc.) déterminent le type des informations nécessaires pour la requête d'interrogation.

L'extraction dynamique fait appel à l'application de certaines méthodes déjà disponibles pour le système d'adaptation. Ces méthodes permettent de calculer les valeurs de certaines caractéristiques de l'environnement qui sont généralement de nature dynamiques, telles que la bande passante disponible au moment de la réception de la requête, la charge du serveur, la charge de la mémoire du terminal cible, etc.

3.3 Présentation des principaux acteurs de l'architecture NAC

3.3.1 Les applications clientes

L'application cliente utilisée par un terminal de l'environnement hétérogène représente un acteur principal de l'architecture NAC. C'est cet acteur qui permet au terminal de communiquer avec le reste du système, de recevoir le contenu et de le présenter à l'utilisateur. L'environnement hétérogène peut inclure une grande diversité d'applications clientes qui varient des applications clientes limitées comme par exemple les navigateurs MMS embarqués sur des téléphones mobiles, jusqu'aux applications clientes les plus sophistiquées qui peuvent traiter un contenu plus complexe et qui disposent de plus de ressources tel que l'espace de stockage, la vitesse de traitement, la bande passante, etc.

Afin de supporter la diversité des applications clientes qui peuvent exister dans l'environnement hétérogène, NAC sépare les deux aspects de négociation et d'accès au contenu. Les services d'adaptation et de négociation de l'architecture peuvent être uti-

lisés par n'importe quelle application cliente indépendamment de ses caractéristiques. Pour cela, il suffit que l'accès au réseau soit configuré pour passer par le module ANM. La mise en œuvre de cette configuration est très simple. En effet, il suffit que l'application cliente utilise le module ANM comme proxy en utilisant le port de communication approprié. L'utilisateur peut à n'importe quel moment effectuer cette configuration en utilisant l'interface de son application cliente. Cette configuration n'est pas forcément permanente et peut concerner uniquement un sous ensemble d'applications dont l'utilisateur souhaite qu'elles reçoivent un contenu adapté. Une fois la configuration effectuée, toutes les requêtes passeront par le module ANM qui assurera la transmission d'un contenu compatible avec les exigences de l'application cliente.

Concernant la négociation, les applications clientes transmettent certaines informations relatives à leur contexte. Cela est assuré à travers le protocole utilisé pour l'accès au contenu. Par exemple, en utilisant le protocole HTTP [7], l'application cliente transmet son identificateur à l'aide de l'entête *user agent* (voir Chapitre 2). Dans ce cas bien particulier, le module ANM fait une correspondance automatique entre l'identificateur de l'application cliente et un ensemble de caractéristiques prédéfinies qui correspondent à l'application. Par exemple, l'identificateur de l'application peut donner une idée sur les protocoles d'accès supportés. D'autres informations transmises par l'application client donnent une idée sur les formats de contenu supportés que ce soit par l'application d'origine ou par les *plugins* installés. Les informations transmises par les applications clientes sont reçu par le module ANM et font partie des dimensions du contexte utilisées par le processus de négociation de contenu.

3.3.2 L'application PocketSMIL

Comme nous avons vu dans le Chapitre 2, le langage SMIL représente un modèle de documents efficace pour la spécification et l'utilisation des présentations multimédia. SMIL permet de créer des présentations riches qui combine plusieurs ressources médias (audio, vidéo, etc.) synchronisées. Grâce à la modularisation de SMIL, les présentations multimédia peuvent être adaptées aux contextes limités [103] (voir Chapitre 2). Par conséquent, un terminal utilisant un navigateur compatible avec le langage SMIL peut profiter des possibilités d'adaptations assurées par le langage. Le terminal peut donc accéder à des présentations riches qui seront personnalisées selon ses capacités et préférences.

L'architecture NAC inclut l'application PocketSMIL (voir Figure 3.5) qui a été développée pour les systèmes embarqués et les terminaux mobiles tel que les ordinateurs de poche (PDA). PocketSMIL permet l'accès et la présentation des documents SMIL 2.0 Basic [127]. NAC inclut plusieurs mécanismes d'adaptation du langage SMIL pour les contextes limités. L'adaptation est effectuée selon les caractéristiques du terminal cible et de son environnement (bande passante disponible, charge mémoire du terminal, etc.). Dans le cadre de la négociation du contenu, le développement de l'application PocketSMIL a été amélioré de telle sorte que l'application envoie les caractéristiques et les informations compatibles avec le contexte du client, tel que les formats supportés, l'identificateur de l'application, le langage de l'utilisateur, etc.

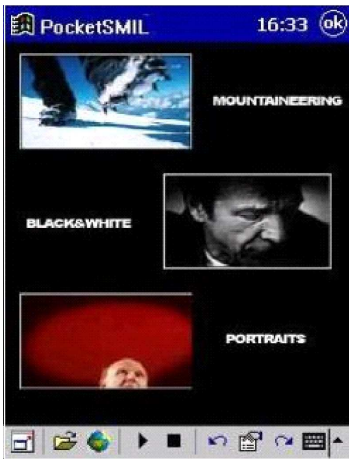

FIG. 3.5 – L'application PocketSMIL

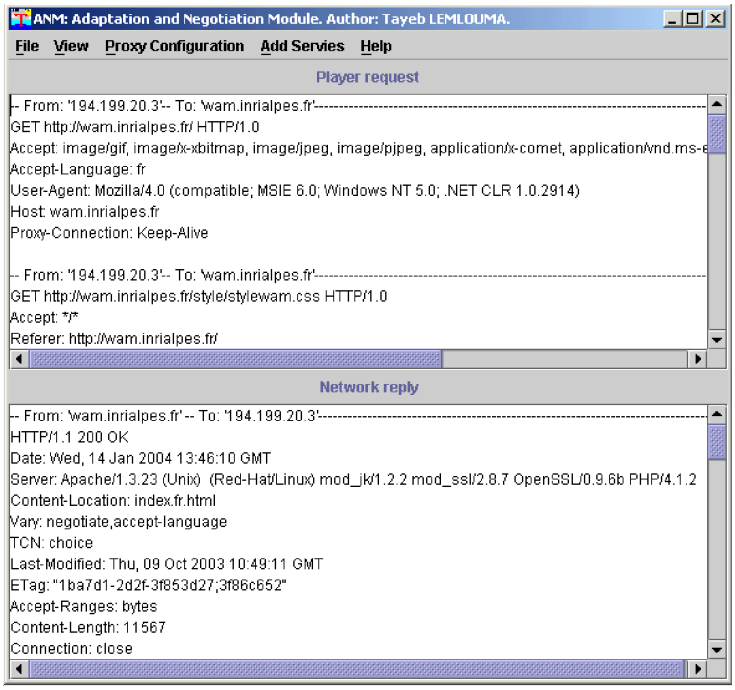

FIG. 3.6 – Interface d'administration du module ANM

3.3.3 Module d'adaptation et de négociation (ANM)

Comme nous l'avons introduit au début de ce chapitre, le module d'adaptation et de négociation ANM (Adaptation and Negociation Module) est l'entité responsable de la négociation et de l'adaptation du contenu de l'architecture NAC. L'interface du module ANM est donnée par la Figure 3.6. Elle comporte un menu et deux zone de texte : une utilisée pour l'affichage des requêtes des applications clientes et une pour l'affichage des réponses des serveurs d'origines.

Le menu (voir Figure 3.8) permet de configurer les ports utilisés pour l'accès au contenu du réseau (le port de communication) et le port réservé pour le protocole de négociation utilisé par l'ANM et l'UCM (le port de négociation). En outre, le menu permet d'associer les versions et les *profils documents* [77] (disponibles côté ANM) au contenu qui peut être demandé par les applications clientes. La déclaration des *profils documents* (qui décrivent le contenu) et de la disponibilité des versions associées aide le processus de négociation à prendre la meilleure décision après la reception de la requête du client. Cela est fait grâce à la connaissance des caractéristiques du contenu demandé et des alternatifs qui peuvent le remplacer. Le menu du module ANM permet aussi de configurer l'inclusion d'autres services. Ces services seront rajoutés au document source demandé par l'application cliente.

Afin d'assurer les services d'adaptation et de négociation pour les terminaux de l'environnement hétérogène, le module ANM doit fonctionner en permanence. Les applications clientes doivent utiliser les mêmes paramètres de configuration de ANM, en particulier les mêmes valeurs de ports. Une fois exécuté, le module ANM lance deux serveurs d'écoute (appelés *player listener* et *UCM listener*) sur les ports de communication et de négociation. Le serveur *player listener* s'exécute en permanence et détecte toute nouvelle connexion d'une application cliente. D'un autre part, le serveur *UCM listener* détecte toute connexion de la part des terminaux dotés du module de contexte utilisateur UCM. Les deux serveurs s'exécutent en même temps et supportent un nombre de connexions illimitées [1]. Si un des deux serveurs détecte une connexion, une instance du serveur est crée afin de traiter la session en cour du terminal. Si une connexion de la part d'un module UCM est détectée, le serveur *UCM listner* crée une instance de serveur appelée *One UCM server*. De la même manière, si une connexion de la part d'une application cliente est détectée, le serveur *player listener* crée une instance de serveur appelée *One server* (voir Figure 3.7).

L'algorithme de négociation et les processus d'adaptation utilisent les informations produites par les instances des serveurs *One server* et *One UCM server*. Ces informations concernent la session courante du terminal : le contenu demandé, le serveur d'origine, le profil de l'application cliente, etc. Dans le cas d'absence de respository de profils, le module ANM emploie une stratégie de cache des profils clients et cela afin de minimiser les échanges de profils avec les terminaux qui utilisent généralement une connexion limitée. La stratégie de cache se base, dans l'identification des applications clientes, sur l'adresse IP du terminal. Les interactions entre une instance de serveur *One UCM server* et un module UCM sont définies par le protocole de négociation de NAC.

[1] la seule limitation du nombre de connexions simultanées est imposé par la capacité de la machine où le module ANM est exécuté

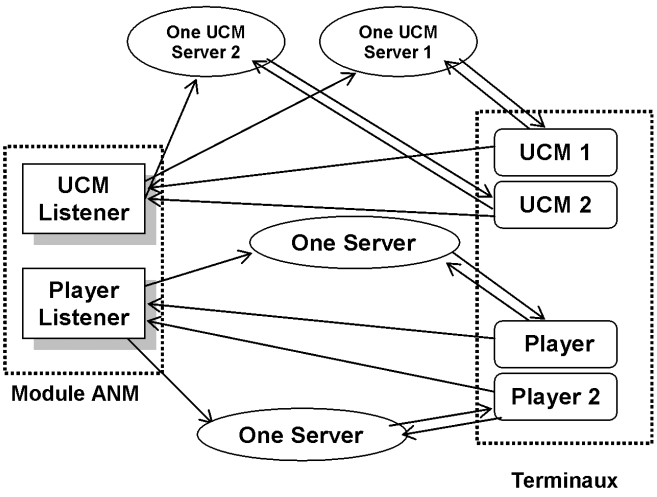

FIG. 3.7 – Les interactions entre les terminaux et le module ANM

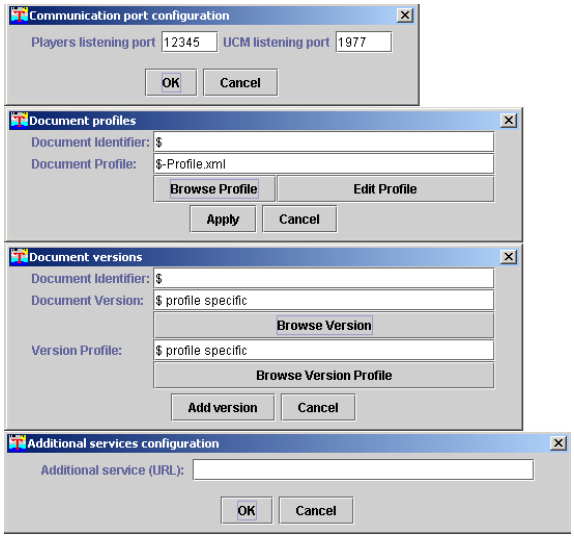

FIG. 3.8 – Menus du module ANM

3.3.4 Module du contexte de client (UCM)

Le module du contexte de l'utilisateur UCM (User Context Module) est un module qui fait partie de l'architecture NAC et qui fonctionne du côté du client. L'utilisation de l'UCM n'est pas obligatoire pour tous les terminaux de l'environnement hétérogène vu que certains terminaux ne supportent pas le rajout d'autres composants dans leurs plates-formes. Lorsqu'un module UCM est installé du côté d'un terminal, le système d'adaptation et de négociation de NAC peut assurer une stratégie de négociation avancée grâce au protocole de négociation et l'échange de caractéristiques en temps réel entre les modules UCM et ANM. Le protocole de négociation a été optimisé afin de minimiser les échanges d'informations relatives à la négociation (optimisation des ressources du réseau) et alléger le traitement des terminaux mobiles (optimisation des ressources du client). Par exemple, un profil utilisateur n'est transmis au module ANM qu'une seule fois tant qu'il n'a pas subit de changement. Le module ANM utilise une gestion de cache afin de garder la trace des caractéristiques des terminaux et des changements qu'ils subissent. L'identification de ces caractéristiques (i.e. les profils) est basée sur l'adresse IP des terminaux.

Le module UCM a été implémenté pour les systèmes embarqués des terminaux mobiles tels que les ordinateurs de poches, les téléphones mobiles, etc. Une version d'émulation a été implémenté aussi pour les plates-formes conventionnelles (ordinateur de bureau, etc.). Le module UCM a été optimisé afin de respecter les caractéristiques des terminaux mobiles et leurs limitations de ressources mémoire et de capacité de traitement. Le module permet de configurer les paramètres de négociation avec l'ANM, et de visualiser les événements liés au protocole de négociation. Ces événements concernent la réception des requêtes de l'instance de serveur *One UCM server*, les réponses de l'UCM vis-à-vis du protocole de négociation, la sélection et le changement de profil du terminal.

La Figure 3.9 montre le menu de configuration des paramètres de négociation de l'UCM sur un ordinateur de poche (PDA). Le menu permet :

1. L'édition du nom du client [2]
2. L'édition de l'URL, ou l'adresse IP, de la machine où le module de négociation et d'adaptation de NAC s'exécute. C'est avec cette machine que l'UCM échange les informations de négociation.
3. La configuration du port de négociation. La même valeur est utilisée par l'instance *One UCM server* du module ANM
4. La sélection et le changement du profil du terminal. A tout moment le client peut sélectionner un nouveau profil dans le cas où les préférences ou les caractéristiques matérielles de l'utilisateur subissent un changement quelconque. Les profils suivent le modèle de description UPS et sont stockés dans des fichiers d'extension *xml* ou *pro*. Ils peuvent être édités ou modifiés et par la suite sélectionnés par l'UCM (voir Figure 3.10).
   Exemple : la description UPS suivante correspond à une partie des types mime d'un terminal mobile (un téléphone portable, voir [72]). L'élément *neg :OnlySupportedMimeTypes* inclut tous les types mime que le terminal supporte.

```
<neg:OnlySupportedMimeTypes>
  <rdf:Bag>
```

[2] ce champ est facultatif

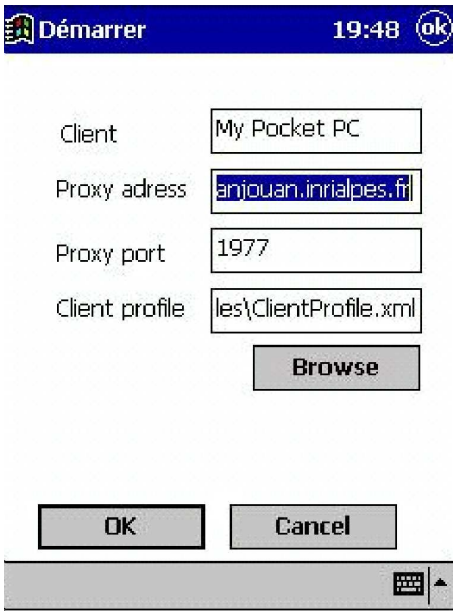

FIG. 3.9 – Interface utilisateur de la configuration de l'UCM

```
        <rdf:li>application/vnd.wap.wmlc</rdf:li>
        <rdf:li>application/vnd.wap.wmlscriptc</rdf:li>
        <rdf:li>image/vnd.wap.wbmp</rdf:li>
        <rdf:li>application/vnd.wap.multipart.mixed</rdf:li>
        <rdf:li>application/vnd.wap.multipart.header-set</rdf:li>
        <rdf:li>text/vnd.wap.wml</rdf:li>
        <rdf:li>text/vnd.wap.wmlscript</rdf:li>
    </rdf:Bag>
</neg:OnlySupportedMimeTypes>
```

Supposant que les caractéristiques du terminal ont changé, par exemple en utilisant un nouveau composant logiciel (un midlet ou autre) qui prend en compte le nouveau type mime : $image/gif$. Dans ce cas là, le profil courant du terminal doit refléter ce changement afin de déclarer que le client supporte le nouveau type mime et par conséquent le module ANM peut lui envoyer ce type de contenu. En terme de description UPS, cela peut être effectué en rajoutant la ligne : <rdf :li>*image/gif*</rdf :li> à l'élément *neg :OnlySupportedMimeTypes*

Afin qu'un terminal puisse participer dans la négociation avancée durant les différentes sessions d'accès au contenu de l'application cliente, l'UCM doit être lancé à l'initialisation du terminal. Le rôle de l'UCM peut être assimilé à une tâche du système d'exploitation du terminal qui doit être exécutée au démarrage du terminal et qui doit fonctionner en permanence. Le module UCM peut aussi être exploité pour assurer la simulation d'une diversité de terminaux. En effet, en préparant plusieurs profils qui correspondent à plusieurs terminaux différents, l'UCM peut simuler un nouveau terminal dont les caractéristiques sont celles stockées dans le profil courrant.

3.4 L'utilisation des services Web au profit de la négociation

Dans une architecture de négociation et d'adaptation, la gestion de profils est très importante. Cette gestion inclut toutes les tâches relatives au traitement, à la sauve-

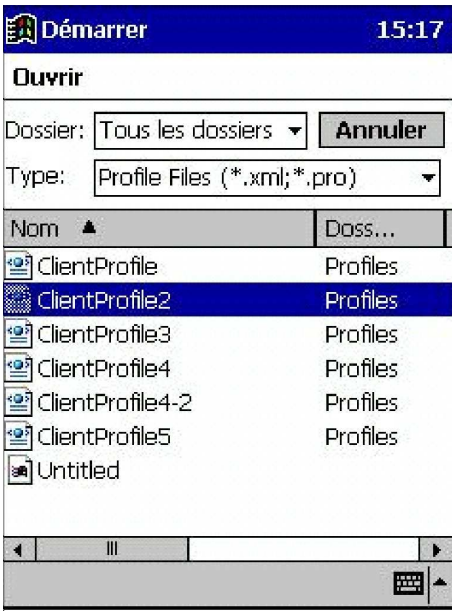

FIG. 3.10 – Sélection de profils avec l'UCM

garde et à l'interrogation des profils. Ces tâches peuvent être exécutées par n'importe quelle entité du système : serveur, proxy ou client. Un système d'adaptation efficace doit éviter de dédier ces taches à ces entités classiques de l'architecture. En effet, l'exécution de ces traitements peuvent ralentir considérablement le fonctionnement des différentes entités.

Un repository de profils est défini comme un serveur qui répond à des requêtes spécifiques en envoyant des profils. Dans l'architecture NAC une séparation est faite entre l'adaptation du contenu et la gestion (sauvegarde, traitement, extraction, etc.) des profils. NAC utilise un repository basé sur les services SOAP (Simple Object Access Protocol) [114] [70]. Le repository de profil interagit avec le reste du système comme le montre la Figure 3.11. Le module ANM qui est l'entité responsable d'appliquer les différentes méthodes d'adaptation sur le contenu d'origine est considéré comme un client SOAP. Les requêtes (ou les appels) RPC (remote procedure calls) sont utilisés pour interroger les profils du repository afin d'extraire le contexte du terminal concerné par l'adaptation du contenu demandé par le client.

Le repository utilisé par NAC prend en compte cent dix huit profils de terminals [72] écrit en UPS [77] et incluent une multitude de terminaux récents.

Les principaux services SOAP du repository sont présenté dans la Table 3.1. Le repository utilise le langage XQuery [134] afin d'assurer les interrogations contextuelles des profils et l'extraction du contexte du terminal selon les paramètres des requêtes RPC. Une fois reçus, les paramètres instancient des interrogations préparées ce qui permet de retourner le résultat voulu. L'exemple présenté par la Figure 3.12 montre comment peut on invoquer une requête RPC.

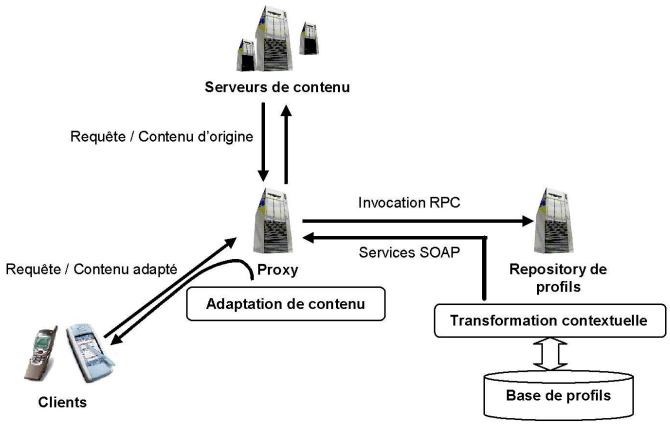

FIG. 3.11 – L'architecture d'adaptation

```
URL url = new URL (Routeur_RPC_NAC); ..
call.setMethodName("getContextAtomicValue"); ..
Response resp = call.invoke (url, "" ); ..
Parameter result = resp.getReturnValue();
```

FIG. 3.12 – Invocation d'appel RPC

TAB. 3.1 – Méthodes exposées du repository SOAP

Méthodes exposées	Paramètres	Signification
getProfile	profileID	interroge un profil dans le repository
getContextAtomicValue	profileID CCPPComponentID contextEntity	interroge une valeur dans un profile
getSubContext	profileID XPathExpression	interroge un sous contexte dans un profile
updateContextAtomicValue	profileID CCPPComponentID contextEntity	met à jour une valeur dans un profile

3.5 Organisations possibles de NAC

Deux organisations de l'architecture NAC sont possibles. Une organisation basée sur l'utilisation d'un proxy et une organisation basée sur l'utilisation du serveur d'origine. Dans le premier type d'organisation, le module d'adaptation et de négociation de NAC est intégré au niveau du serveur du contenu. Dans le deuxième type d'organisation, le module ANM s'exécute au niveau d'une entité intermédiaire entre le terminal et le serveur.

L'architecture basée sur l'utilisation du serveur d'origine est préférable dans le cas ou l'on souhaite activer un type bien particulier d'adaptation et de négociation dans une entité du système global. Un exemple de telles organisations est la conception d'un serveur de contenu HTML qui soit adaptable pour un ensemble de terminaux mobiles qui supportent un sous ensemble de fonctionnalités du langage HTML (HTML Basic par exemple).

Dans ce type d'architecture, le module ANM doit être doté principalement des mécanismes et des méthodes nécessaires pour appliquer le type d'adaptation ou de négociation voulu. Le module proxy de communication de l'ANM sera inutile puisque toutes les requêtes reçues par l'ANM concerneront toutes le même serveur de contenu. Dans ce type d'organisation, l'ANM possède plus de contrôle sur le contenu du serveur puisqu'il s'exécute à son niveau. En effet, aucun échange de messages n'est nécessaire entre l'ANM et le serveur pour que le module d'adaptation analyse les caractéristiques du contenu demandé par un terminal. Cet avantage est important dans le cas de gestion de variantes du contenu source. Dans le cas d'une architecture basée sur l'utilisation d'un proxy, l'ANM ne possède pas une image globale sur les versions disponibles du contenu source et leurs caractéristiques sans être obligé à faire appel aux échanges réseau.

L'organisation basée sur l'utilisation d'un proxy dédie toutes les fonctionnalités d'adaptation et de négociation du système à une entité du réseau (le proxy) autre que l'ensemble des serveurs et des clients. Dans ce genre d'organisation, tout le trafic passe par le proxy que ce soit le trafic du contenu ou le transport des profils et des requêtes relatives au protocole de négociation. Au contraire de la première organisation, l'adaptation du système est plus universelle et ne se limite pas à un serveur particulier. En effet, un terminal peut envoyer sa requête vers n'importe quel serveur de contenu et c'est au proxy d'acheminer les requêtes vers le bon serveur et d'adapter les réponses des serveurs selon les caractéristiques des clients.

Du côté description de l'environnement, ce type d'organisation permet d'avoir une image globale plus détaillée des caractéristiques du système en centralisant la collection des profils au niveau du proxy. Le proxy collecte les descriptions du contexte quand cela est nécessaire. Les informations du contexte peuvent donc être utilisées par n'importe quel entité du réseau en utilisant les services du proxy. En outre, le proxy permet d'alléger la tâche des serveurs du contenu et de jouer le rôle de l'acteur qui interagit avec les différentes entités particulières de l'architecture telle que le repository des profils et le module du contexte de l'utilisateur.

Les deux types d'organisations de l'architecture NAC permettent de s'adapter aux besoins des applications pour utiliser les services d'adaptation et de négociation de

contenu dans des cas bien spécifiques. Il est clair que l'utilisation du proxy représente la solution la plus adéquate afin d'assurer l'adaptation d'une manière plus universelle qui couvre l'ensemble des serveurs et des clients de l'environnement hétérogène.

3.6 Adaptation du contenu dans NAC

NAC manipule un format déclaratif des descriptions de l'environnement et des capacités des processus et des méthodes d'adaptation du contenu. Cette manipulation est faite dans un cadre extensible que ce soit les profils de description des clients qui peuvent évoluer ou les mécanismes d'adaptation qui peuvent être ajoutés à n'importe quel moment afin d'enrichir les capacités d'adaptation de l'architecture.

3.6.1 Modèle de description pour l'adaptation

Comme nous allons voir en détail dans le chapitre 4, NAC défini le modèle UPS (Universal Profiling Schema) pour la description des profils de l'environnement. UPS décrit les contraintes de l'environnement selon un format déclaratif sous forme d'un marquage descriptif. Des balises, sont définies afin de décrire les caractéristiques des différentes entités du système hétérogène qui peuvent influencer la transmission finale du contenu. Ces balises peuvent être organisées pour former des structures plus complexes. L'assemblage des balises est défini en suivant plusieurs modes de regroupement et d'ordre donnant ainsi une sémantique aux structures. Par exemple, une liste d'éléments peut être représentée par un ensemble ordonné, un ensemble non ordonné, un ensemble d'exclusion, etc.

UPS utilise, dans la description d'une entité (par exemple un client, un serveur, etc.), un regroupement sous forme de *composants*. Chaque composant comporte un ensemble d'informations homogènes relatives à un côté particulier des caractéristiques de l'entité décrite. Il existe plusieurs types de composants : le composant des caractéristiques matérielles, le compostant des caractéristiques logicielles, les caractéristiques des formats de contenu supportés, etc. Un composant peut inclure aussi des éléments composés d'un ensemble de marques pour décrire une caractéristique complexe de l'entité. Par exemple, la description d'un écran est décomposée en la description du nombre de couleurs supportées, la longueur du dispositif d'affichage, la largeur, etc.

Le modèle UPS couvre la description du client, du serveur et du réseau. Les profils de description suivent six schémas définis dans le langage RDF schema [99]. L'ensemble des descriptions est traduit en un ensemble de contraintes sur l'utilisation d'un contenu donné. Les caractéristiques du contenu et du contexte de transmission sont extraites afin de voir si la transmission du contenu source est conforme aux exigences de l'application cliente cible ou non. Ces caractéristiques sont très variées et concernent, entre autres, la structure du contenu, le format des ressources médias utilisées, les caractéristiques physiques des ressources, etc. Chaque caractéristique correspondante à une contrainte posée par le client est analysée. Le système d'adaptation applique les mécanismes nécessaires pour que toutes les caractéristiques non conformes soient cohérentes avec les contraintes posées. Comme nous allons voir dans le Chapitre 5, ces mécanismes sont variés et peuvent impliquer une transformation du contenu source, une sélection d'alternative, un filtrage, etc.

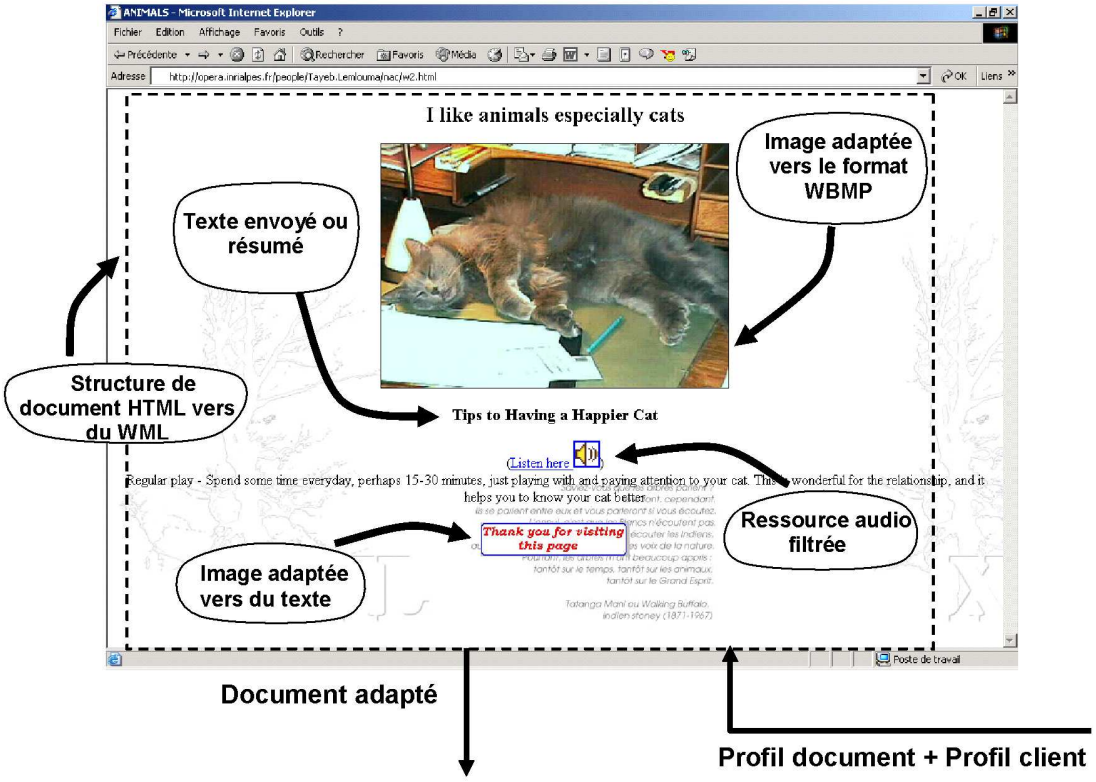

FIG. 3.13 – L'adaptation des ressources d'un contenu

La Figure 3.13 montre un exemple de contenu à adapter pour un téléphone mobile. Le contenu est sous forme d'un document HTML qui utilises différentes ressources médias : ressources textuelles, des images et une ressource audio. Les caractéristiques du terminal cible (le téléphone mobile) ne permettent pas une utilisation directe du contenu source. Le terminal dispose d'un espace d'affichage limité, et il ne supporte que le format WML sans les ressources audio. Comme le montre la Figure 3.13, le système d'adaptation applique plusieurs types d'adaptations : une transformation structurelle du format HTML vers le format WML, une sélection de variante pour la substitution de l'image JPEG avec la variante correspondante en WBMP, une adaptation d'image en utilisant l'attribut HTML ALT d'une image, un filtrage de la ressource audio et d'une ressource image (l'image du fond). Grâce au profil du document, le système d'adaptation détermine, selon les capacités du système, les méthodes d'adaptation à appliquer afin de respecter les contraintes imposées par le contexte du client. Par exemple, la sélection de variante est appliquée lorsque une ressource originale ne correspond pas au contexte cible et qu'une version de la ressource peut être utilisée. La Figure 3.14 montre le résultat de l'adaptation.

3.6.2 Adaptation contextuelle

A cause de la variété des caractéristiques du contenu et des autres dimensions du contexte, le nombre des mécanismes de transformation nécessaires peut être très important. NAC définit un mécanisme d'adaptation contextuel qui permet la réutilisation des transformations pour plusieurs contextes. NAC s'appuie sur deux principales approches : la définition des méthodes et des règles d'adaptation paramétrées [70], et

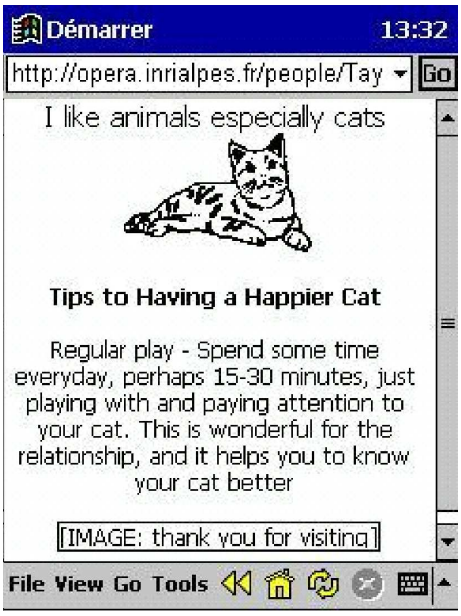

FIG. 3.14 – Le contenu adapté pour un PDA

l'adaptation dynamique [74]. Dans la première approche, les méthodes d'adaptation dépendent du contexte où le contenu sera adapté. Cette dépendance est assurée en définissant des variables utilisées par les méthodes de transformation. Les variables représentent indirectement des dimensions du contexte. Elles sont instanciées selon la description du contexte dans lequel le contenu est transmis. Par exemple, une méthode d'adaptation peut définir la variable *largeur d'écran* pour adapter le contenu pour différentes tailles d'écran. C'est au moment de la transmission du contenu que la variable est instanciée et par conséquent l'adaptation est appliquée avec la valeur appropriée de la variable.

Dans la deuxième approche, les règles d'adaptation sont générées à partir du profil du client. Selon les valeurs des informations élémentaires du contexte du client (que nous appelons dimensions du contexte), des règles d'adaptation sont sélectionnées et instanciées pour correspondre à l'adaptation nécessaire pour le type de profil client. Une règle d'adaptation est une procédure appliquée pour transformer une structure ou une ressource média d'un état vers un autre. Par exemple, une règle peut juste remplacer le nom d'un nœud dans un arbre ou changer les dimension d'une image. Dans cette approche, l'adaptation est appliquée en deux passes : une génération de règles d'adaptation à partir du profil client, et l'application des règles générées sur le contenu demandé par le client.

L'adaptation de NAC est faite de manière à minimiser le temps de réponse et les échanges de messages entre les différentes entités de l'architecture notamment entre les modules ANM et UCM et entre le module ANM et le repository du profils.

3.7 Conclusion

Dans ce chapitre, nous avons présenté les principales entités de l'architecture de négociation et d'adaptation NAC. Nous avons discuté les principes de bases adoptés lors de la conception de l'architecture ainsi que les principales fonctionnalités des

différents modules de l'architecture. Comme nous avons vu, NAC définit le modèle de description UPS afin d'assurer la gestion des caractéristiques de l'environnement hétérogène. La description concerne les différentes entités de l'environnement qui peuvent influencer la transmission finale du contenu.

Le modèle de description adopté assure une description extensible de l'environnement et une représentation déclarative des différentes caractéristiques. Cette description, sous forme de composants et d'éléments simple et complexes, est exploitée par le système d'adaptation afin d'effectuer la correspondance entre les contraintes du client et les capacités du système en terme de contenu et de mécanismes d'adaptation de contenu.

L'architecture NAC définit une variété d'entités et de modules qui coopèrent entre eux afin d'atteindre l'objectif de l'adaptation du contenu. La définition de ces entités permet une organisation flexible des différents modules. Elle permet aussi de dédier des parties spéciales du réseau pour assurer certaines tâches du service de négociation et d'adaptation du contenu.

De la discussion ce chapitre, il ressort que la conception d'une architecture d'adaptation et de négociation efficace doit fournir :

- Un modèle de description de l'environnement et une stratégie d'analyse et de gestion du contexte,

- La définition des règles d'interaction entre les différentes entités du système et l'exploitation des descriptions pour l'adaptation du contenu,

- Des mécanismes d'adaptation qui assurent la transformation du contenu de son état initial vers un état final qui répond au contexte des clients cibles.

Le chapitre qui suit est consacré au premier point, c'est-à-dire à la manipulation du contexte des différents acteurs du système. L'ensemble des descriptions est exploité par le système d'adaptation afin de satisfaire les contraintes imposées par la diversité des terminaux.

Chapitre 4

Description et gestion des contextes

Résumé

Ce chapitre présente le système de description et de gestion des contextes de l'architecture NAC. Ce système permet de prendre en charge différentes contraintes imposées par l'environnement et les différentes exigences des terminaux et de leurs utilisateurs. Le système définit un modèle de description accompagné d'un cadre d'adaptation automatique qui se base sur les différentes informations du contexte. Le cadre d'adaptation permet d'appliquer une stratégie d'adaptation et de négociation de contenu basée sur le contexte et de transmettre finalement un contenu qui satisfait le contexte.

Contenu

4.1 Introduction

Dans le chapitre précédent, nous avons décrit les principales entités de l'architecture d'adaptation et de négociation NAC. Nous avons montré que l'adaptation de contenu, qui vise à satisfaire les contraintes du client cible, fait intervenir plusieurs entités (telles que le client, le serveur, le repository de profils) qui coopèrent pour transmettre un contenu adapté suite à la requête client. Nous avons souligné que la qualité de l'adaptation appliquée dépend profondément de l'état de ces différentes entités. Les décisions prises par le module d'adaptation et de négociation sont prises sur la base de l'image globale de l'environnement. Cette image globale concerne toute entité susceptible d'influencer le processus de la transmission finale du contenu. Pour fournir une image de l'état de environnement, il est nécessaire de définir un moyen qui permet de décrire les caractéristiques de l'environnement et une manière d'analyser ces caractéristiques au profit du système d'adaptation et de négociation du contenu. C'est l'ensemble de ces caractéristiques qui détermine les exigences de l'application cliente vis-à-vis du contenu demandé mais aussi les capacités du système à fournir un contenu adapté en appliquant plusieurs techniques.

Un système d'adaptation et de négociation de contenu doit se doter d'un ensemble de mécanismes de description universelle qui couvre tous les détails des caractéristiques de l'environnement. Cependant, sachant que les environnements hétérogènes présentent beaucoup de limitations, que ce soit sur le plan du réseau de communication ou des terminaux utilisés, la gestion (le stockage, le traitement, la transmission..) des descriptions doit être optimisée afin d'éviter l'augmentation de délais de réponse. Ce besoin d'optimisation représente un vrai problème lorsqu'on veut lier un bon niveau de détail des descriptions avec un transfert rapide et un traitement minimal de ces descriptions. Ce problème pousse non seulement à définir un modèle universel et flexible de description de l'environnement, mais aussi à adopter une stratégie efficace de traitement des descriptions.

Ce chapitre présente le système de description et de gestion des contextes de l'architecture NAC. Ce système permet d'assurer une prise en charge des différentes contraintes imposées par l'environnement et les différentes exigences des terminaux et de leurs utilisateurs. La démarche adoptée consiste à définir un modèle de description des caractéristiques de l'environnement qui ne soit pas limitée au client qui représente un acteur parmi d'autres dans le processus d'adaptation du contenu. Le modèle de description est accompagné d'un cadre d'adaptation automatique qui se base sur les différentes informations du contexte.

Les traitements (ou les calculs) basés sur le contexte ont été discutés pour la première fois par Schilit et Theilmer en 1994 [111]. Un traitement basé sur le contexte a été défini comme un logiciel qui s'adapte à l'endroit de l'utilisation, à l'ensemble des gens et des objets qui partagent le même entourage, et aux changements que ces objets peuvent subir dans le temps. Cette définition générale a été raffinée par la suite par de nombreux travaux qui ont adopté le traitement basé sur le contexte pour des besoins plus spécifiques. [30]

Le système de traitement des contextes est conçu d'une manière à définir et extraire les différentes contraintes imposées par le contexte, à vérifier la correspondance entre un couple de contextes et à définir les règles d'adaptation à appliquer.

Les dimensions du contextes peuvent être très diverses et par conséquent le niveau de détail d'un descripteur de contexte peut être très profond. Ce fait pause un problème d'efficacité du système d'adaptation. L'idéal est d'inclure toutes les dimensions du contexte dans un descripteur et d'utiliser ces dimensions pour vérifier la correspondance entre le contenu et le contexte client lors de la réception d'une requête cliente. Malheureusement cette option est très coûteuse en termes de stockage, de transfert et de complexité de traitement. Ce coût se multiplie dans le cas des environnements hétérogènes qui soufrent de plusieurs limitations tels que la faible bande passante du réseau, les capacités de traitement réduites des terminaux, etc. Afin de répondre à ces particularités de l'environnement tout en considérant les contextes des différents éléments de la transmission d'un contenu adapté, le système de NAC adopte une approche qui vise à minimiser le coût des algorithmes de traitement des descripteurs du contexte ainsi que le transfert de l'information lors de la négociation de contenu et des méthodes d'adaptation à appliquer. NAC considère aussi l'aspect sémantique de la satisfaction des dimensions contextuelles. La considération de ce type de dimensions, permet de garder un lien entre les préférences et l'intérêt d'un utilisateur vis-à-vis du contenu et l'adaptation automatique basée sur les caractéristiques matérielles et logicielles du client.

4.2 Principe de modèle UPS pour la description de l'environnement

Un certain nombre de systèmes, tel que GStreamer [46] et les implémentations de la négociation du protocole HTTP [7] [35], utilisent une négociation de contenu basée sur l'utilisation des caractéristiques du client et de l'environnement sous forme d'un ensemble de variables utilisées par des scripts et des programmes prédéfinis. La négociation dépend des valeurs de variables appartenant à cet ensemble défini avant l'application de la négociation dans un contexte précis.

L'utilisation des langages de programmation pour l'analyse et la description du contexte comporte deux inconvénients majeurs. Le premier inconvénient, est le fait que la description n'est pas évolutive pour inclure de nouvelles contraintes du contexte. Les environnements hétérogènes subissent fréquemment des changements de contexte, ce que rend les approches qui figent les ensembles de dimensions inefficaces. Le deuxième inconvénient est relatif à l'ouverture de la stratégie de négociation appliquée. En effet, dans ce genre d'approche, la description du contexte n'est exploitable que par le script ou le programme qui la définit. Les descriptions ne peuvent pas être réutilisées, échangées ni modifiées par les autres entités du système ce qui rend la stratégie de négociation fermée.

Avec une approche déclarative pour la description du contexte de l'environnement, basée sur XML une grande partie des inconvénients cités ci-dessus disparaissent. De plus, l'approche déclarative, inhérente aux systèmes de contraintes, ouvre des perspectives plus larges au traitement automatique des descriptions. En effet, à partir

des différentes caractéristiques contenues dans une description, il est possible de définir des mécanismes qui se prêtent mieux aux traitements adaptatifs, voire génériques, contrairement aux descriptions impératives qui utilisent des programmes qui sont par nature moins adaptables. XML [33] est un standard d'échanges de données universel. Il s'agit d'un métalangage, qui a donné naissance à de nombreux autres standards. Il permet la représentation de données sous format déclaratif en utilisant des balises et des attributs. XML garantit une exploitation et une pérennité des données ce qui constitue un avantage non négligeable dans le contexte de la négociation de contenu. Les descriptions XML du contexte peuvent être fournies par n'importe quelle entité du système qui intervient pour l'adaptation, le traitement ou la réception du contenu (client, serveur, proxy, etc.). Le contenu de ces descriptions est facilement échangé, mis à jour et interrogé par les nombreux outils conçus pour XML.

Le langage XQuery [134], défini par le W3C, permet d'interroger des collections d'arbres, qui sont des représentations de documents XML. Le langage utilise des expressions de requêtes, appelées expressions flower, qui sont "for", "let", "where" et "return". Ce dernier élément correspond à la faculté de construire le document que l'on souhaite publier en résultat. Globalement, XQuery permet d'effectuer toutes les opérations que l'on peut faire avec les langages de bases de données classiques. Il est toutefois beaucoup plus puissant. En effet, il permet de restructurer totalement les arbres en entrée. Il permet également une recherche d'information assez poussée pour l'extraction des composants de description de l'environnement.

UPS exploite les avantages de XML grâce aux différents schémas utilisés qui étendent le cadre de travail CC/PP et qui définissent plusieurs types de descriptions pour les différentes entités du système d'adaptation et de négociation. CC/PP (Composite Capabilities/Preference Profiles) [43] est un standard défini par le W3C pour la description du contexte pour l'adaptation de contenu. Bien qu'il soit très récent, il semble être le dénominateur commun des applications mettant en jeu l'adaptation. C'est en fait un cadre de travail qui s'appuie à la fois sur XML et sur RDF (Resource Description Framework) [101]. CC/PP permet de décrire les caractéristiques logicielles et matérielles d'un terminal, mais aussi les préférences de l'utilisateur. Dans sa version courante, il se limite à ces aspects du contexte.

Le standard RDF, défini par le W3C, permet de rendre plus *intelligent* le traitement des informations. RDF associe à toute ressource décrite un ensemble de descripteurs qui caractérisent au mieux la ressource, on parle alors de *métadonnées*. Dans ce contexte, il est nécessaire de construire des langages de définition de corpus homogènes de métadonnées. C'est la fonction d'un métalangage de définition de modèles, souvent appelés *vocabulaires*. Ces vocabulaires RDF (qui relèvent plutôt de la syntaxe) sont spécifiques à un domaine bien précis, tel que dans le cas d'UPS où les descriptions concernent l'environnement hétérogène au profit de l'adaptation de contenu. Une application de traitement des informations, décrites en RDF, peut alors par simple analyse des types de données définies dans le modèle, retrouver la sémantique applicable aux métadonnées liées à ce modèle. L'utilisation de RDF garantit aussi l'échange de connaissances par des modules logiciels. Cet échange a un intérêt particulier dans le contexte de la négociation de contenu où plusieurs entités du système peuvent être impliquées afin d'adapter le contenu d'une manière efficace.

La structure générale d'un profil client CC/PP prend la forme d'un arbre à deux

niveaux : les composants et les attributs, chaque composant ayant la possibilité de faire appel à un ensemble de valeurs. Un profil CC/PP contient un ou plusieurs *composants*, qui contiennent à leur tour un ou plusieurs attributs (voir Figure 4.1). Chaque composant est représenté par une ressource de type *ccpp :Component* (ou par certaines sous-classes RDF de celle-ci) et relié à la ressource du profil client par une propriété *ccpp :component*. La Figure 4.1 montre un exemple de profil CC/PP. Dans cet exemple, le type du premier composant CC/PP est défini dans l'espace de noms *http ://www.example.com/schema#HardwarePlatform* donné par l'attribut *ressource* de l'élément *rdf :type*.

Bien que les descriptions UPS définissent différents schémas qui ne se limitent pas à la description du client, comme dans CC/PP, la structure des profils UPS est similaire à la structure des profils CC/PP.

L'espace de noms ccpp est *http ://www.w3.org/2002/11/08-ccpp-schema#*. Dans le modèle UPS, les éléments précédés par le préfixe *ccpp* proviennent de l'espace de noms du modèle CC/PP, ceux précédé par les préfixes *rdf* proviennent de l'espace de noms du modèle RDF. Le modèle UPS exploite les avantages du cadre de travail CC/PP et définit un cadre complet pour l'adaptation et la négociation du contenu.

4.3 Modèle UPS d'expression des contraintes en XML

Une contrainte d'utilisateur est une expression logique qui inclut une information particulière du contexte de l'utilisateur. Cette information concerne soit les capacités du client, soit ses préférences vis à vis du contenu demandé. Par exemple, une contrainte peut exprimer que le terminal est capable de jouer du son, qu'il supporte un contenu de type vidéo ou que l'utilisateur préfère recevoir les textes en anglais.

Le modèle d'expression de contraintes que nous avons défini dans UPS comprendre :

1. des outils pour l'expression des formules logiques,
2. une organisation structurée (propriétés, structures des éléments et des relations entre les structures et les propriétés, etc.),
3. et un processus de traitement des expressions afin de faciliter la tâche des fournisseurs de contenu après la réception du contexte utilisateur.

4.4 Variables de contraintes

Une variable de contrainte représente la principale entité d'une expression de contrainte. Considérons la contrainte suivante :

'*Le client ne peut pas jouer du son*'

Ici, la variable de contrainte peut être '*support du son*' ou '*son activé*'. Par conséquent, la contrainte précédente peut être réécrite comme suit :

'*support du son* = faux' ou '*son activé* = faux'

```
<?xml version="1.0"?>
<rdf:RDF
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ccpp="http://www.w3.org/2002/11/08-ccpp-schema#"
         xmlns:ex="http://www.example.com/schema#">

  <rdf:Description rdf:about="http://www.example.com/profile#MyProfile">

    <ccpp:component>
      <rdf:Description
          rdf:about="http://www.example.com/profile#TerminalHardware">
        <rdf:type
            rdf:resource="http://www.example.com/schema#HardwarePlatform" />
        <ex:displayWidth>320</ex:displayWidth>
        <ex:displayHeight>200</ex:displayHeight>
      </rdf:Description>
    </ccpp:component>

    <ccpp:component>
      <rdf:Description
          rdf:about="http://www.example.com/profile#TerminalSoftware">
        <rdf:type
            rdf:resource="http://www.example.com/schema#SoftwarePlatform" />
        <ex:name>EPOC</ex:name>
        <ex:version>2.0</ex:version>
        <ex:vendor>Symbian</ex:vendor>
      </rdf:Description>
    </ccpp:component>

    <ccpp:component>
      <rdf:Description
          rdf:about="http://www.example.com/profile#TerminalBrowser">
        <rdf:type
            rdf:resource="http://www.example.com/schema#BrowserUA" />
        <ex:name>Mozilla</ex:name>
        <ex:version>5.0</ex:version>
        <ex:vendor>Symbian</ex:vendor>
        <ex:htmlVersionsSupported>
          <rdf:Bag>
            <rdf:li>3.2</rdf:li>
            <rdf:li>4.0</rdf:li>
          </rdf:Bag>
        </ex:htmlVersionsSupported>
      </rdf:Description>
    </ccpp:component>
  </rdf:Description>
</rdf:RDF>
```

FIG. 4.1 – Exemple de profil CC/PP

```
<imageSupport>
    <longueur>200</longueur>
    ...
</imageSupport>
```

FIG. 4.2 – Exemple 1 d'utilisation de la variable *longueur*

```
<ecran>
    <longueur>360</longueur>
    ...
</ecran>
```

FIG. 4.3 – Exemple 2 d'utilisation de la variable *longueur*

Une variable de contrainte possède un type et appartient à une structure simple ou complexe. Dans le contexte XML, une variable simple peut être représentée par un nom d'élément. Exemple : '*support du son*' peut être représenté par l'élément XML <supportSon> ou <sonActive>. Du point de vue de la syntaxe, le nom de l'élément et sa valeur dépendent du schéma adopté. La sémantique d'un élément dépend du chemin allant de la racine de la structure XML globale jusqu'à cet élément. Ce chemin est appelé généralement *chemin de contexte*. La sémantique d'un élément varie donc selon le chemin du contexte. Par exemple, une variable 'longueur' peut concerner la longueur des images supportées (voir Figure 4.2) ou la longueur d'écran d'un terminal (voir Figure 4.3).

Nous distinguons deux types de variables :

1- Variables atomiques : Une variable est appelée atomique si elle supporte des valeurs de type simple (entier, booléen, chaîne de caractères, un ensemble fini de valeurs, etc.). Exemple : taille d'écran (entier), écran couleur (booléen), police de caractère (un ensemble fini de valeurs possibles), etc.

2- Variables complexes : Une variable complexe est un ensemble ordonné ou non ordonné de valeurs atomiques. La structure associée à cet ensemble est utilisée pour exprimer comment la variable est composée à partir des variables atomiques. Si l'ordre est important pour la sémantique de la variable, un ensemble ordonné est utilisé, sinon un ensemble non ordonné est utilisé par défaut.

Exemple : formats de son acceptés = {WAVE, MP3}[ensemble non ordonné], langues acceptés = {Français, Anglais, Allemand}[ensemble ordonné. L'ordre indique les préférences].

Les variables complexes peuvent être représentées en utilisant quelques structures RDF [101] déjà définies telles que le RDF *Bag* pour les ensembles non ordonnés et le RDF *Seq* pour les ensembles ordonnés.

4.5 Expressions de contraintes

Les expressions de contraintes sont des expressions logiques qui se basent sur l'utilisation des variables et de leurs valeurs pour décrire un sous ensemble du contexte

client. Une expression de contraintes peut être évaluée à VRAI ou FAUX.
L'analyse et l'évaluation des contraintes du client est faite côté serveur (ou proxy) afin de transmettre au client un contenu adapté à ses contraintes. Pour faciliter cette tâche, le modèle de description des contraintes doit être simple.

4.5.1 Expressions atomiques

Une expression atomique est définie comme suit : V = y, où V est le nom de la variable et y est la valeur de V. La forme XML la plus simple pour présenter cette contrainte est : <A>y</A>, où A est le nom d'élément associé à V.

Exemple : 'Screen width = 240 pixels' représente une contrainte atomique. La représentation XML de cette contrainte peut être donnée comme suit : <width>240</width>

Une contrainte atomique peut utiliser une variable complexe. Exemple : 'formats d'images acceptés = {BMP, GIF, SVG}'. La représentation XML d'une contrainte atomique qui utilise une variable complexe nécessite l'utilisation de nouvelles structures. RDF *Bag* et *Seq* représentent des structures adéquates pour la description de variables sous forme d'ensembles ordonnés et non ordonnés respectivement.

Une forme simplifiée de la contrainte précédente peut être donnée comme suit :

```
<acceptedImageFormats>
<Bag>
<li>BMP</li>
<li>GIF</li>
<li>SVG</li>
</Bag>
</acceptedImageFormats>
```

4.5.2 Construction d'expressions complexes de contraintes

La définition introduite précédemment permet seulement d'exprimer des contraintes de type 'variable = valeur'. Afin de décrire les capacités du client d'une manière plus efficace, nous devons enrichir le modèle de description des contraintes de telle manière qu'il soit capable d'exprimer les autres relations logiques qui sont '< et NON' (*inférieur à* et *la négation logique*), ou bien '> et NON' (*supérieur à* et *la négation logique*).

La représentation XML des relations logiques utilisées dans des contraintes complexes peut être assurée de différentes manières. Nous identifions deux manières de descriptions qui peuvent être combinées :

1-*L'utilisation des noms d'éléments XML* : Ici, l'approche est d'utiliser un nom d'élément qui porte en lui-même une sémantique logique inspirée de la contrainte. Pour simplifier, la contrainte suivante : 'taille d'image supportée $\leq$ 30000 Octets' peut être représentée par : <maxTailleImage>30000</maxTailleImage>

2- *La définition des formats des valeurs* : Ici, le format de la valeur associée à un élément est défini pour inclure la contrainte logique. Par exemple : 'LE' peut être

Tab. 4.1 – Combinaison d'expressions atomiques

Type d'expression	Exemple
Conjonction	E1 AND E2
Disjonction	E1 OR E2
Négation	NOT E
Exclusion	Exclude (E1) from E (équivalent à 'E AND NOT E1')

réservé pour exprimer la relation 'inférieur ou égale'. Une contrainte : 'taille d'image supportée $\leq$ 30000 Octets' peut être écrite comme suit :

```
<dimension>
    <contrainte>taille LE 30000</contrainte>
</dimension>
```

Un parseur de contraintes doit donc être capable d'analyser les différents formats des valeurs d'éléments pour extraire les contraintes du contexte.

4.5.3 Combinaison des expressions de contraintes

Une fois assurée la définition complète des variables atomiques, des expressions et la manière de les représenter en XML, des mécanismes permettant la construction d'expressions complexes doivent être définis. Ces mécanismes doivent définir la manière d'exprimer les différentes combinaison de contraintes. Nous identifions les combinaisons présentées par le tableau 4.1 où E, E1 et E2 sont des expressions de contraintes.

Les conjonctions d'expressions sont utilisées pour représenter les capacités et les préférences de l'utilisateur. Par exemple pour les capacités d'affichage des images : on peut représenter les formats acceptés *et* la taille maximale des image *et* ...etc. Généralement, la conjonction est la relation logique prise par défaut lors de l'analyse d'un profil de contraintes dont les relations logiques ne sont pas explicites.
Pour rendre le modèle de description des contrainte plus complet et éviter toute ambiguïté lors de l'analyse des profils de contraintes ; nous définissions trois types de structures simples pour l'expression des 'AND', 'OR' et 'NOT'.
Le 'ET' logique est représenté par une structure similaire aux conteneurs RDF. Nous appelons cette structure *le conteneur du ET logique*.

Exemple :

```
<andBagConstraints>
    forme XML de la contrainte 1
    forme XML de la contrainte 2
    ...
    forme XML de la contrainte n
</andBagConstraints>
```

Les disjonctions sont utilisées pour exprimer qu'au moins une des contraintes données doit être satisfaite. Cela est utile dans le cas où on décrit plusieurs propriétés d'un seul élément, par exemple pour exprimer une expression du genre : '(taille maximale d'image = 10000 *and* format = "JPEG") *or* (taille maximale d'image =

30000 *and* format = "BMP")'

De la même manière que pour les conjonctions, une représentation XML des disjonctions est assurée par *un conteneur de OU logique* :

```
<orBagConstraints>
    forme XML de la contrainte 1
    forme XML de la contrainte 2
    ...
    forme XML de la contrainte n
</orBagConstraints>
```

La négation logique est importante pour exprimer que certaines expressions ne doivent pas être évaluées à VRAI. La négation sert aussi à réduire la liste des contraintes posées (en utilisant l'exclusion par exemple) ce qui évite de lister toutes les contraintes évaluées à VRAI. Exemple : 'le format d'image "JPEG" n'est pas accepté' ce qui est équivalent à 'NOT (formats acceptés = "JPEG")'. La représentation XML de la négation utilise *un conteneur de NON logique* qui peut inclure une expression ou plusieurs. La sémantique d'un tel conteneur est que toutes les expressions qui sont représentées sont liées par un ET logique et chaque expression est précédée par un NON. La représentation XML suivante exprime l'expression 'NOT C1', tel que C1 est une expression de contrainte.

```
<notBagConstraints>
    forme XML de C1
</notBagConstraints>
```

Pour exprimer 'NOT C1 AND NOT C2', on peut écrire :

```
<notBagConstraints>
    forme XML de C1
    forme XML de C2
</notBagConstraints>
```

Sans ajouter un conteneur andBagConstraints qui inclut C1 et C2.

L'utilisation des exclusions est utile pour exprimer des contraintes qui excluent quelques objets d'un ensemble prédéfini d'éléments. Cela permet d'écrire des profils courts et évite de réécrire des ensembles énormes d'éléments. La représentation XML de l'exclusion utilise *un conteneur d'exclusion*. Un exemple d'exclusion peut exprimer que : 'les modules acceptés sont des modules SMIL 2.0 sans les deux modules *animation* et *content control*'. La représentation XML de cette expression de contrainte est donnée comme suit :

```
<modulesSupport>
    <modulesLanguage>SMIL 2.0</modulesLanguage>
    <excludeBag>
        <li>animation</li>
        <li>content control</li>
    </excludeBag>
</modulesSupport>
```

Les éléments d'un conteneur d'exclusion doivent avoir le même type d'éléments appartenant à l'ensemble à partir duquel l'exclusion est faite. Dans l'exemple précédent, l'exclusion est faite sur un ensemble de modules du langage SMIL 2.0. Les éléments à exclure (i.e. "animation" et "content control") sont en effet deux modules du même langage.

Les différentes représentations des expressions logiques ont été identifiées pour simplifier l'écriture et le traitement des profils de contraintes. Notons que les structures définies pour le ET et le NON logique forment un système complet pour la description des autres combinaisons d'expressions logiques. La même situation est pour les structures définies pour le OU et le NON logique.

4.6 Contexte de transmission (delivery context)

Le contexte est défini généralement par toutes les informations qui peuvent être utilisées pour caractériser la situation d'une entité. Une entité peut être une personne, un endroit ou un objet qui peut intervenir dans les interactions entre l'utilisateur et l'application y compris l'utilisateur et l'application eux mêmes. [1]

Schilit et Theimer [111] définissent le contexte par rapport à la localisation et les identités des personnes et des objets voisins et des changements de ces objets. Ce genre d'approche qui adopte une définition par exemples est difficile à appliquer. Le problème se pose quand on veut vérifier si un type d'information, non listé par la définition, fait partie du contexte ou non.

D'autres travaux fournissent des définitions en employant quelques synonymes du mot contexte, comme par exemple, environnement ou situation [16] [129]. Ces définitions restent générales et sont difficiles à appliquer dans la pratique. Schilit et al. [112] et Pascoe [91] proposent d'autres définitions plus pratiques. Schilit et al identifient les aspects importants du contexte par l'identité de l'entité, la localisation de l'entité et les ressources du même environnement. Pascoe définie le contexte comme un sous ensemble d'états physiques et conceptuels importants d'une entité particulière. Ces définitions sont trop spécifiques. Il est difficile d'énumérer tous les aspects importants d'une entité puisque ces derniers peuvent changer d'une situation à une autre. Il existe aussi d'autres définitions qui s'intéressent uniquement à une entité particulière du système, comme par exemple l'ensemble des paramètres de configuration d'une application [102] et négligent le reste des entités.

Nous définissons le contexte, du point de vue de l'adaptation de contenu, comme *l'ensemble de toutes les informations de l'environnement qui peuvent influencer le processus de l'adaptation et de la transmission du contenu vers l'utilisateur final.*

La description du contexte peut inclure les caractéristiques des méthodes d'accès au contenu, les préférences de l'utilisateur, les capacités du terminal utilisateur, l'état du réseau, etc. Comme nous avons vu dans le chapitre précédent, de nombreux travaux ont été menés pour assurer une correspondance entre le résultat des applications et les contraintes du client. Ces efforts incluent le protocole HTTP/1.0 [53] qui se base sur l'utilisation des entêtes pour inclure quelques capacités et préférences utilisateur, le cadre de travail CC/PP [43] et beaucoup d'autres vocabulaires, spécifiques à des domaines particuliers, qui définissent des attributs concernant un ou plusieurs

contextes [116] [63].

L'objectif d'un vocabulaire de description de contexte est de fournir une image détaillée du client, de son identité, de ses besoins, de ce qu'il peut traiter correctement et éventuellement d'autres informations complémentaires qui peuvent être utiles lors du processus d'adaptation de contenu. Un bon vocabulaire doit être simple et doit éviter l'utilisation des structures complexes afin de faciliter la tâche d'analyse des descriptions. Le vocabulaire doit aussi éviter toute ambiguïté qui peut résulter, par exemple, en utilisant les mêmes noms d'éléments pour différentes sémantiques, ou l'attribution de différentes valeurs au même attribut. Dans [21] on trouve de telles directives concernant le vocabulaire CC/PP.

Le schéma universel de profils (UPS) [77] est défini dans notre approche pour servir comme base de modélisation des différents profils de l'environnement. Comme la transmission du contenu final est influencée par plusieurs paramètres de l'environnement global, UPS joue un role central dans le processus de l'adaptation et de la génération du contenu.

UPS peut être vu comme une extension de CC/PP. CC/PP se limite à la description des capacités et des préférences du client. UPS ajoute une description détaillée des capacités des serveurs et des proxy, les caractéristique du contenu, l'état du réseau et les méthodes d'adaptation qui peuvent exister au niveau du serveur ou des proxy.

4.6.1 Schémas de modèle UPS

Le modèle UPS à été défini pour fournir une description universelle de l'environnement. Cela veut dire que n'importe quel élément susceptible d'influencer la négociation du contenu et la satisfaction d'une requête cliente est décrit sous forme d'un descripteur de contexte appelé *profil*. En plus du client et du contenu demandé, les descriptions utiles pour la négociation peuvent être élargies pour couvrir par exemple les caractéristiques du réseau, les capacités du système d'adaptation, etc.

Le modèle UPS définit plusieurs types de schémas RDF (voir Annexe A). Les schémas sont relatifs au client, au contenu, aux méthodes d'adaptation et au réseau utilisé pour envoyer la requête cliente et recevoir la réponse du serveur.

4.6.1.1 Les schémas relatifs au client

Les schémas qui donnent les descriptions du contexte relatifs aux clients sont :

1. ***Le schéma du profil client*** : Ce schéma décrit les caractéristiques générales du client. Il inclut la description des plate-formes matérielles et logicielles utilisées par le terminal de l'utilisateur.

   Le composant de la plate-forme matérielle décrit les caractéristiques physiques du terminal telles que le nom du terminal (le modèle), les capacités d'affichage, la résolution, etc. Les informations décrites ont une relation directe avec la transmission finale du contenu adapté au terminal. Par exemple, si le terminal est un téléphone mobile, le système d'adaptation réduit les dimensions d'une

image (en se basant sur les dimensions de l'écran du terminal).

Le composant de la plate-forme logicielle décrit le logiciel du terminal : le système d'exploitation utilisé, la version du système, etc.

Le composant de l'application cliente décrit le navigateur utilisé par l'utilisateur du terminal dans l'accès au contenu du réseau. Ce composant peut inclure dans un profil l'identificateur du navigateur, la version du navigateur, le protocole d'accès, etc. Ce composant a une importance particulière dans la négociation du contenu. Il inclut une collection d'informations concernant l'accès et la présentation du contenu du serveur. Cette collection est organisée sous forme de trois catégories : 1) les ressources supportées, 2) le ressources préférées et 3) les ressources non supportées. La première et la troisième catégorie sont décrites en utilisant l'élément RDF *rdf :bag* [101]. La deuxième catégorie est décrite en utilisant l'élément *rdf :seq*. La différence entre ces deux types d'élément est que l'ordre est important pour les préférences mais ne l'est pas pour les ressources. Pour la deuxième catégorie, l'ordre est donné par la valeur de l'élément *neg :prioriyLevel*.

Afin de minimiser la taille du profil client et d'utiliser uniquement l'information utile quand cela est nécessaire, le schéma du profil client définit l'élément *neg :profile* qui peut être utilisé dans les catégorie une, deux ou trois. Cet élément permet de donner un lien vers le profil détaillé d'une ressource incluse dans le profil du client. Le système d'adaptation et de négociation utilise le lien donné, quand c'est nécessaire, afin d'extraire les caractéristiques détaillées de la ressource.

2. ***Le schéma du profil ressource client*** : Ce schéma donne plus de détails sur la description d'une ressource déclarée dans le profil client. Le schéma définit un composant (appelé *content requirement*) qui décrit les exigences du client vis-à-vis du contenu. Le composant contient une description de contraintes du client vis-à-vis de la ressource concernée. Le composant peut aussi utiliser les éléments *OnlySupportedResources*, *NonSupportedResources* et *PreferredSupportedResources* afin de donner une spécification plus détaillée sur l'utilisation d'une ressource particulière.

4.6.1.2 Les schémas relatifs au contenu

Les schémas qui donnent les descriptions du contexte relatives au contenu sont :

1. ***Le schéma du profil d'instance de document*** : Ce schéma considère un document qui existe sur le serveur comme une ressource. Les différentes caractéristiques de cette ressource sont décrites dans un profil instance de document. Une ressource document est décrite indépendamment des différentes ressources utilisées par le document tels que les images, les vidéos, etc. Le schéma définit trois types de composants : 1) la description de l'instance du document, 2) la description du contenu multimédia et 3) la description des ressources adaptables.

Le dernier composant déclare la liste des variantes et des méthodes d'adaptation qui peuvent être utilisées pour adapter le document. Chaque ressource d'adaptation (variante ou méthode d'adaptation) peut être associée à un élément, appelé *neg :ResourceAdaptabilityValue*, qui indique le niveau d'adaptation de la ressource. Cette valeur est utilisée dans la stratégie de négociation appliquée par le système d'adaptation et de négociation. La description détaillée d'une ressource d'adaptation (c'est-à-dire son profil) est indiqué dans le profil d'instance de document en utilisant le même élément *neg :profile* utilisé dans le schéma du profil client. Dans le cas où plusieurs instances de documents partagent le même profil, il est plus efficace de sauvegarder les profils de ces instances sous le même nom ce qui donne un gain dans l'espace de stockage des profils. Cette possibilité est intéressante pour les documents statiques qui gardent toujours le même contenu. Cependant, dans le cas ou le contenu des documents risque de changer, il est préférable de stocker les profils des instances séparément.

2. ***Le schéma du profil ressource*** : Ce schéma est similaire au schéma du profil ressource client. Il complète la description du profil instance de document pour un ou plusieurs instances de documents[1]. Le schéma définit deux composants : 1) la description des ressources médias et 2) la description des ressources d'adaptation. Le dernier composant joue exactement le même rôle que le composant de description des ressources d'adaptation du profil d'instance de document. Ici, l'entité considérée est la ressource utilisée par l'instance de document et non pas le document lui-même.

4.6.1.3 Le schéma relatif aux méthodes d'adaptation

Le schéma qui décrit les descriptions du contexte relatif aux méthodes d'adaptation est appelé *schéma de profil de méthode d'adaptation*. Le schéma définit deux composants : 1) la description de ressource d'adaptation et 2) la description de méthode d'adaptation.

Le premier composant décrit la méthode d'adaptation en tant que ressource comme n'importe quelle autre ressource telle qu'une image, un document, etc. La description peut contenir le nom et le format de la méthode d'adaptation, le chemin de la méthode d'adaptation, etc. Le composant de description de méthode d'adaptation décrit les capacités de la méthode d'adaptation en termes de conditions en entrée nécessaires pour l'application de la méthode et de description de la ressource générée après l'application de la méthode. La Figure 4.4 donne un exemple de profil de méthode d'adaptation. L'adaptation décrite par le profil est sous la forme d'une feuille XSLT qui transforme des documents XML vers des documents LaTeX.

4.6.1.4 Le schéma relatif au réseau

Le schéma qui décrit les descriptions du contexte relatif au réseau est appelé *schéma de profil de réseau*. Le schéma définit un composant (*CurrentNetworkDescription*) de description courante du réseau. Le profil définit des dimensions pour la description du réseau utilisé. Les valeurs de ces dimensions sont utilisées afin d'effectuer certaines adaptions relatives à l'état courant du réseau, telles que les adaptations basées sur la

[1] dans le cas où les instances de document utilisent les mêmes ressources

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:ccpp="http://www.w3.org/2000/07/04-ccpp#"
xmlns:neg="http://www.inrialpes.fr/opera/people/Tayeb.Lemlouma/NegotiationSchema/
AdaptationMethodProfileSchema-03012002#">
    <rdf:Description rdf:ID="AdaptationMethodProfile">
       <ccpp:component>
           <rdf:Description rdf:ID="AdaptationMethodResourceDescription">
              <rdf:type rdf:resource="http://www.inrialpes.fr/opera/people/
              Tayeb.Lemlouma/NegotiationSchema/AdaptationMethodProfileSchema
              -03012002#AdaptationMethodResourceDescription"/>
                   <neg:ResourceType>XSLT style sheet</neg:ResourceType>
                   <neg:ResourceFormat>xsl</neg:ResourceFormat>
                   <neg:ResourceName>XML2LaTeX.xsl</neg:ResourceName>
                   <neg:ResourceSize>19568Bytes</neg:ResourceSize>
               <neg:ResourceLocation>
               opera/people/Tayeb.Lemlouma/MULTIMEDIA/XSLT/
               </neg:ResourceLocation>
               <neg:ResourceServer>http://www.inrialpes.fr/</neg:ResourceServer>
           </rdf:Description>
       </ccpp:component>
       <ccpp:component>
        <rdf:Description rdf:ID="AdaptationMethodDescription">
           <rdf:type rdf:resource="http://www.inrialpes.fr/opera/people/
           Tayeb.Lemlouma/NegotiationSchema/AdaptationMethodProfileSchema
           -03012002#AdaptationMethodDescription"/>
            <neg:SubAdaptationMethods>
               <rdf:Bag>
                      <rdf:li rdf:parseType="Resource">
                         <neg:InputRequirements>
                           <rdf:Bag>
                             <rdf:li rdf:parseType="Resource">
                              <neg:ResourceType>XML</neg:ResourceType>
                              <neg:ResourceFormat>xml</neg:ResourceFormat>
                              <neg:ResourceDTD>http://www.inrialpes.fr/opera/people/
                              Tayeb.Lemlouma/MULTIMEDIA/XSLT/LaTeXDTD.dtd
                              </neg:ResourceDTD>
                             </rdf:li>
                           </rdf:Bag>
                         </neg:InputRequirements>
                         <neg:OutputDescription>
                           <rdf:Bag>
                             <rdf:li rdf:parseType="Resource">
                              <neg:ResourceType>LATEX</neg:ResourceType>
                              <neg:ResourceFormat>tex</neg:ResourceFormat>
                             </rdf:li>
                           </rdf:Bag>
                         </neg:OutputDescription>
                      </rdf:li>
               </rdf:Bag>
            </neg:SubAdaptationMethods>
        </rdf:Description>
       </ccpp:component>
    </rdf:Description>
</rdf:RDF>
```

FIG. 4.4 – Exemple de profil méthode d'adaptation

bande passante disponible pour la transmission d'un flux vidéo.

Grâce à l'adoption du cadre de travail CC/PP et de son extension, les profils UPS restent extensibles et supportent l'ajout de nouvelles dimensions du contexte. Afin de garantir une prise en charge de l'espace des nouvelles dimensions, le système d'adaptation doit définir les règles de correspondance nécessaires et les méthodes d'adaptation qui peuvent être appelées pour répondre aux contraintes des dimensions de profils.

4.6.2 Transformation de vocabulaires de description

Les vocabulaires utilisés pour la description du contexte sont très variés. Ces vocabulaires concernent généralement un aspect bien particulier du contexte. Par exemple, le vocabulaire UAProf [116], une spécification du WAP Forum, est conçu pour la description des téléphones mobiles. Malheureusement, le modèle reste limité à une seule catégorie de terminaux et à l'architecture WAP, qui est une architecture fermée souffrant de nombreuses limitations pour l'accès au contenu.

Afin de tirer profit des descriptions UAProf déjà existantes pour les terminaux mobiles et des descriptions fournîes généralement par l'industrie de la téléphonie mobile, nous avons défini une transformation structurelle des descriptions UAProf vers UPS. Cette transformation permet de transformer les différents types de profils UAProf et prend en charge les sept différents espaces de noms des schémas UAProf. UPS résout le problème du schéma UAProf concernant la variation de structures utilisées pour la description de certains aspects des terminaux mobiles. Par exemple, les différentes versions du schéma UAProf n'utilisent pas toutes la même structure XML pour la description de l'élément *prf :CcppAccept-Charset*. En effet, la même entité est décrite en tant qu'élément XML simple dans certaines versions de UAProf et en tant que structure complexe dans d'autres versions. Pour décrire cette entité, le schéma UPS utilise l'élément *neg :OnlySupportedCharset* sous forme d'un élément RDF de type *rdf :bag*. L'Annexe D donne les détails de la transformation structurelle UAProf vers UPS.

4.6.3 Modélisation

Nous modélisons la description d'un contexte C par l'ensemble de variables (dimensions) d_i qui appartiennent chacune à un type prédéfini $T(d_i)$. Une dimension de contexte est une variable simple ou complexe qui décrit une information du contexte, par exemple, la taille d'écran, la bande passante du réseau, le format d'une image, etc.

Nous notons $Contexte(e)$ le contexte qui décrit l'entité e. e peut représenter un terminal mobile, un document, une méthode d'adaptation de contenu, etc. Une instance de contexte, notée c, est un ensemble de couples de dimensions de contexte et de valeurs. Chaque couple regroupe une dimension et sa valeur. Formellement, on peut écrire :

$$c = \{(d_i, v_i) \quad i = 1..n\}$$

où v_i représente la valeur de la fonction d'évaluation $value(d_i)$ de la variable d_i, avec $value(d_i) \in T(d_i)$. Le résultat de l'évaluation d'une dimension de contexte peut varier

selon le moment d'évaluation.

Du point de vue sémantique, une dimension de contexte d_i peut dépendre d'une autre dimension d_j dans le même contexte. Cela veut dire que si $v_i = value(d_i)$ change alors $v_j = value(d_j)$ change aussi et vice-versa. Formellement, on peut représenter cette dependence comme suit :

$$d_i \rightarrow d_j \Leftrightarrow v_i \rightarrow v_j$$

Exemples : *temps_transmission_image* $\rightarrow$ *taille_image* (*temps_transmission_image* dépend sémantiquement de *taille_image*), *dimension_ecran_terminal* $\rightarrow$ *longueur_ecran_terminal* (*dimension_ecran_terminal* dépend sémantiquement de *longueur_ecran_terminal*)

Une dimension d_i peut être déduite à partir d'une autre dimension d_j dans un contexte C (i.e. $d_i \triangleleft d_j$), s'il existe un processus P, pour chaque instance c de C, qui permet d'évaluer v_i à partir de v_j sans faire intervenir d'autres dimensions de contexte[2]. Formellement, on écrit :

$$d_i \triangleleft d_j \Leftrightarrow \exists\ P\ \forall c \in C\ v_i = P(v_j) \wedge \forall P'\ \forall d_k\ d_k \neq dj\ \ v_i \neq P'(v_k)$$

Exemples : *longueur_ecran_terminal* $\triangleleft$ *dimension_ecran_terminal*, *dimension_ecran_terminal* $\triangleleft$ $\{longueur\_ecran\_terminal, largeur\_ecran\_terminal\}$, $\neg(temps\_transmission\_image \triangleleft taille\_image)$. La valeur de la dimension *temps_transmission_image* dépend sémantiquement de la dimension *taille_image* mais elle ne peut pas être déduite d'elle seule : la bande passante du réseau intervient aussi.

Pour éviter l'explosion de taille des descriptions et faciliter leur transport, un profil qui décrit un contexte ne doit pas inclure des dimensions de contexte qui peuvent être reliées par des relations de déduction. Le processus d'analyse du contexte doit prévoir des méthodes qui peuvent appliquer la relation de déduction afin d'extraire d'autres dimensions de contexte à partir d'une description donnée. Par exemple, le couple (d, v) = $(format\_contenu$, *'smil basic'*) implique beaucoup d'autres dimensions qui peuvent être utilisées par un processus d'analyse de contexte. En effet, (d, v) implique que le contenu supporte un ensemble prédéfini de modules SMIL [109] tels qu'ils sont définis par la spécification du langage SMIL Basic [127].

Afin de comparer les dimensions de contexte, nous définissons la notion *d'équivalence de dimensions*. Deux dimensions $w1$ et $w2$ sont sémantiquement équivalentes (noté $w1 \equiv w2$) si les deux dimensions ont le même type et décrivent une information similaire du contexte (éventuellement en utilisant des noms différents). L'équivalence sémantique n'implique pas que les dimensions équivalentes doivent avoir la même valeur dans une instance de contexte.

Si une dimension de contexte $w1$ peut être déduite d'une autre dimension $w2$ et vice-versa, alors les deux dimensions sont également sémantiquement équivalentes :

$$(v \triangleleft w \wedge w \triangleleft v) \Leftrightarrow v \equiv w$$

[2]Ceci est équivalent à dire que le processus P ne dépend d'aucune variable de contexte autre que v_i

Exemple : la dimension "$prf : Model$" du schema UAProf [116] est équivalente à la dimension "$neg : DeviceName$" du schéma UPS [77]. On peut donc écrire : "$prf : Model \equiv neg : DeviceName$". Les deux dimensions ont le même type et la même sémantique i.e. le nom du terminal donné par le fabricant. L'équivalence sémantique est une notion très importante pour la comparaison et la transformation de vocabulaires ainsi que pour la correspondence entre contextes.

4.6.4 Extraction du contexte

Comme nous avons vu dans le chapitre précédent (Section 3.2.3), les profils peuvent être créés par un auteur ou engendrés automatiquement à partir du contexte de l'environnement. Le processus d'extraction de contexte consiste à instancier les dimensions du contexte pour une session client/serveur. L'instanciation est effectuée afin de fournir un contenu adapté à l'instance courante du contexte. L'évaluation des dimensions du contexte est faite avant ou durant la requête du client. Les dimensions dynamiques du contexte (i.e. les dimensions qui risquent de changer de valeur d'une session à une autre) sont souvent ré-évaluées après la réception de chaque nouvelle requête émise par client. Les dimensions du contexte sont instanciées à partir de :

1. Repository de profils ;
2. Variables utilisateur (exemple : préférence utilisateur) mises à jour ;
3. Dimensions évaluées en temps réel (exemple : vitesse du réseau, protocole de connexion.)

L'évaluation en temps réel est effectuée en utilisant des méthodes déjà préparées qui permettent, à tout moment, de calculer la valeur courante d'une dimension et de la mettre à jour. Ces méthodes concernent l'évaluation des descriptions des ressources médias, l'extraction des dimensions du contexte client et réseau à partir des requêtes des applications clientes, ainsi que l'évaluation du contexte réseau par des méthodes utilisées dans le module ANM.

Les méthodes appliquées sur les ressources médias permettent d'instancier les valeurs des dimensions des médias, par exemple, la longueur ou la largeur d'une ressource vidéo, la résolution ou le nombre de couleurs d'une image, etc. Ce type de méthode est appliqué lorsqu'aucun profil n'est associé à une ressource[3] ou lorsque le profil de la ressource n'inclut pas la dimension du contexte recherchée.

L'extraction des dimensions à partir des requêtes des applications clientes consiste à analyser les requêtes émises par le client et à instancier le maximum de dimensions de contexte transportées par la requête. La requête représentée par la figure 4.5 permet, par exemple, l'instanciation des dimensions du contexte suivantes :
(Protocol, 'HTTP/1.0'), (PlatformName, 'Windows CE (POCKET PC)'), (PlatformVersion, '3.0'), (ColorBit, '4') , (DisplayPixel, '240x320').

[3]Ce type de méthode peut être utilisé pour générer automatiquement des profils de ressources déjà existantes et parconséquent enrichir le repository des profils par le profil généré

```
GET people/Tayeb.Lemlouma/PICS/NAC-on-Mobile.JPG HTTP/1.0
Accept: */*
UA-OS: Windows CE (POCKET PC) - Version 3.0
UA-color: color16
UA-pixels: 240x320
UA-CPU: ARM SA1110
User-Agent: Mozilla/2.0 (compatible; MSIE 3.02; Windows CE;
240x320)
Host: 194.199.20.23
```

FIG. 4.5 – Extraction de dimensions de contextes à partir des requêtes clientes

L'extraction des dimensions du contexte à partir des profils est basée sur l'utilisation d'un langage de requêtes qui analyse le profil et extrait la valeur de la dimension voulue. UPS [77] facilite l'extraction des dimensions de contexte grâce à l'utilisation du principe des composants de CC/PP [43] et à son extension sur tous les types de contextes de l'environnement. Le processus d'extraction peut donc facilement sélectionner les composants appropriés selon le contexte qu'on veut évaluer. Exemple : Les préférences utilisateur en UPS sont incluses dans le composant CC/PP qui satisfait *rdf :about = 'BrowserUA'*. La requête suivante est un exemple d'expression XQuery qui liste toutes les ressources du serveur ayant comme format *jpeg* et comme taille une valeur inférieure à 3000 :

```
let  $resource =
Document("resources.rdf")/rdf:RDF/rdf:Description/
neg:Resource/rdf:Description[neg:ResourceFormat="jpeg" and
neg:ResourceSize<3000]/neg:ResourceName
```

Comme nous allons le voir, dans le cas où les profils sont stockés dans une entité du réseau distante, l'analyse des profils nécessitera l'utilisation d'un protocole de communication efficace entre les deux entités du réseau.

4.6.5 Satisfaction du contexte

Lorsqu'un client envoie une requête pour recevoir un contenu C, le processus d'adaptation du proxy (ou du serveur) vérifie si C peut être supporté par le client et son utilisateur. Si c'est le cas, le processus d'adaptation envoie simplement le contenu d'origine. Sinon, la tâche d'adaptation de contenu est appliquée sur C afin de produire un contenu adapté au client. Vérifier si un contenu est adapté aux capacités et préférences d'un client/utilisateur revient à vérifier si le contexte du contenu correspond au contexte de l'utilisateur. Un contexte $C1$ correspond à un contexte $C2$ ($C1 \bowtie C2$) : si toutes les dimensions de $C1$ correspondent aux dimensions de $C2$. Formellement, on écrit :

$$C1 \bowtie C2 \Leftrightarrow (\forall d_i \in C1 \ \forall d_j \in C2 \ d_i \equiv d_j \Rightarrow d_i \bowtie d_j)$$

Exemple : Si durant une session donnée, l'instance du contexte client inclut le couple unique $(d, v) = (format\_contenu,\ 'gif')$ alors toutes les ressources médias qui ont le *format* gif correspondront au contexte du client. Notons que seule la dimension $format$ des ressources a été considérée car $format$ est la seule dimension qui est

TAB. 4.2 – Types de contraintes de profils

Type de contrainte	Exemple
Ensemble de valeurs autorisées	(format_accept, {'jpeg', 'gif', 'xhtml1.0'}). d_i correspond à d_j si chaque valeur atomique de d_i appartient à d_j
Ensemble de valeurs nécessaires	(modules_SMIL_necessaires, {'Basic-ContentControl', 'CustomTestAttributes'}) d_i correspond à d_j si toutes les valeurs atomiques de d_j appartiennent à d_i
Ensemble de valeurs non-autorisées	(modules_SMIL_non_supportés, { 'PrefetchControl Module'}) d_i correspond à d_j si d_i n'inclut aucune valeur qui appartient à d_j
Valeur unique autorisée	(format_contenu, 'xhtml1.0') d_i correspond à d_j si $value(d_i) = value(d_j)$.
Intervalle de valeurs	(max_taille_image, 3000) d_i correspond à d_j si toute valeur atomique de d_i appartient à l'intervalle dénoté par le couple (d_j, v)
Ensemble ordonné de valeurs autorisées	(langages_préférés, {'fr', 'en', 'de'}) d_i correspond à d_j si chaque valeur atomique de d_i appartient à d_j. L'ordre donnée par le couple (d_j, v) doit être considéré comme critère de sélection si plusieurs valeurs sont acceptées.

sémantiquement équivalente à la dimension $format\_contenu$ du contexte client.

Vérifier si une dimension d_i d'un contexte $C1$ correspond à une dimension d_j d'un contexte $C2$ dépend de la contrainte logique exprimée par le couple formé par le nom de la dimension et sa valeur, i.e. (d_j, v). Ce couple appartient à l'instance du contexte $C2$. La contrainte logique du couple (d_j, v) dépend de la valeur et de la sémantique de la dimension d_j.

Nous distinguons six types de contraintes logiques qui peuvent être interprétées à partir du couple (d_j, v). Le tableau 4.2 énumère ces types de contraintes.

Dans certains cas, les dimensions de contexte qu'on veut comparer ne sont pas sémantiquement équivalentes. Exemple : d_1 = taille_image et d_2 = neg :MmsMaxResolutionImage. Un processus de correspondence efficace doit chercher les couples de dimensions qui sont équivalentes sémantiquement en se basant sur les relations de déductions (voir section 4.6.3). Formellement, ceci est équivalent à la recherche des couples (x, y) satisfaisants :

$$d_i \in C1, d_j \in C2, x \triangleleft d_i \wedge y \triangleleft d_j \wedge x \equiv y$$

Exemple : supposons que d_1 = taille_image et d_2 = neg :MmsMaxResolutionImage. On peut obtenir : x = MmsMaxTailleImage qui est sémantiquement équivalent à d_1,

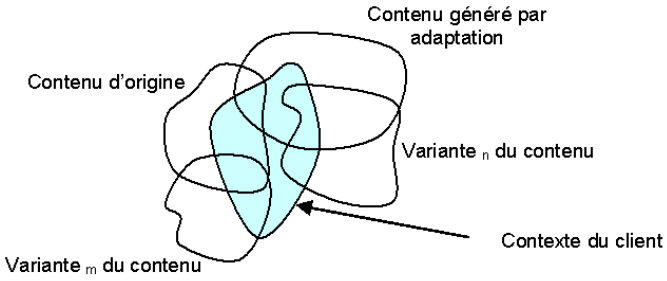

FIG. 4.6 – Les contextes de l'environnement

car MmsMaxTailleImage ◁ neg :MmsMaxResolutionImage. On a donc $x \equiv d_1$, par conséquent les deux dimensions d_1 et x peuvent être comparées.

4.7 Cadre de l'adaptation basée sur les contextes

Après la réception d'une requête client, le processus d'adaptation peut transmettre le contenu d'origine, une version de ce contenu ou un contenu généré en appliquant une méthode d'adaptation sur le contenu demandé par le client (voir figure 4.6).

Un contenu e n'est pas transmis a un client u si le contexte du contenu ne correspond pas au contexte du client ; i.e. $\neg(C1 \bowtie C2)$ où $C1 = Context(e)$ et $C2 = Context(u)$. Le processus de négociation peut appliquer des déductions afin de rendre les dimensions de C1 et C2 équivalentes sémantiquement[4]. Selon notre précédente définition, cela peut être écrit formellement comme suit :

$$\exists d_i \in C1,\ \exists d_j \in C2,\ d_i \equiv d_j \wedge \neg(d_i \bowtie d_j)$$

Les dimensions incluses dans le contexte du contenu et qui ne correspondent pas à certaines dimensions du contexte client représentent les dimensions à adapter afin de rendre le contenu utilisable par le client cible. Nous appelons cet ensemble de dimensions *la différence* entre les deux contextes du contenu C1 et du client C2 notée $\Delta(C1, C2)$:

$$\Delta(C1, C2) = \{d_i,\ d_i \in C1 \wedge \exists d_j \in C2,\ d_i \equiv d_j \wedge \neg(d_i \bowtie d_j)\}$$

L'objectif de toute méthode d'adaptation appliquée à un contenu e est de créer un contenu e' tel que $\Delta(Context(e'), C2) = \emptyset$. Cela peut se faire en choisissant une version de e qui satisfait la condition précédente ou par l'application de certaines tâches qui permettent de changer les valeurs des dimensions de l'ensemble $\Delta(Context(e), C2)$.

4.7.1 Stratégie d'adaptation

Après la réception d'une requête cliente et dans le cas où il n'y a pas de correspondance entre le contenu source et le contexte du client, le système d'adaptation cherche une variante (une version) du contenu source qui pourra être supportée par le client cible. Si une telle variante n'existe pas, le système cherche une méthode d'adaptation

[4]Si l'application des déductions est impossible, cela signifie que le système de description de l'architecture d'adaptation et de négociation n'est pas efficace. Dans ce cas là, le contenu ne peut pas être adapté

TAB. 4.3 – Exemple de réponses d'adaptation

Réponse d'adaptation	Contexte
Transformation de document	format (contenu) = 'SMIL' format_supporté = 'SMIL Basic'
Filtrage de modules du document	format (contenu) = 'SMIL' module_non_supporté = 'animation'
Filtrage de ressource média	format_non_supportés = 'MPEG'
Substitution de variante	format_supporté = 'SMIL Basic' une version du contenu existe en SMIL Basic
Conversion de ressource	format (contenu) = 'JPEG' format_supporté = 'WBMP'
Compression de ressource	format (contenu) = 'JPEG' taille (contenu) = '3000Ko' taille_maximale_supportée = '1200Ko'
Réduction de dimension de ressource	dimension (contenu) = '200x300' dimension_ecran = '101x80'
Réduction du nombre de couleurs	colorBit(Contenu) = 16 colorBit(Client) = 4
Réduction du nombre de trames vidéo	Bande_passante = '56Kbps' format (contenu) = 'MPEG' à 30fps
Adaptation de protocole de communication	protocole du serveur = 'HTTP' protocole du client = 'réseau cellphone'

afin de transformer le contenu source vers une autre forme qui satisfait les contraintes du client. Si plusieurs variantes du contenu source existent, le système d'adaptation sélectionne la variante qui a la plus petite taille en espace afin de minimiser le temps de réponse.

Le système d'adaptation extrait, à partir de l'environnement, les paramètres et les variables liés au contexte du client. Le système d'adaptation essaie de satisfaire les dimensions du contexte client qui ne sont pas satisfaites par le contenu source. Grâce au profil d'adaptation UPS, le système d'adaptation peut appliquer la méthode d'adaptation qui satisfaite les dimensions du contexte client. Si aucune méthode d'adaptation ne peut satisfaire les dimensions du contexte liées aux capacités du client, le système d'adaptation envoie une réponse négative au client. Si les dimensions du contexte qui ne sont pas satisfaites sont liées aux préférences de l'utilisateur, le contenu est envoyé avec ces dimensions. Cette stratégie évite l'échange inefficace de contenu dans le cas où le client cible est incapable d'utiliser le contenu source.

Dans le cas ou certaines dimensions du contexte ne sont pas disponibles dans les profils, le système d'adaptation essaie d'extraire ces dimensions en utilisant les méthodes d'extraction disponibles, telles que les méthodes d'évaluation de l'état courant du réseau.
Le Tableau 4.3 rassemble les réponses appliquées par des règles d'adaptation dans diverses situations de contexte.

(1) *content* = C ;
(2) **from** i = 1 **to** $|p$ | **do**
(3) *content*=M_i(*content*) ;
(4) deliver *content* ;

FIG. 4.7 – Algorithme d'exécution de chemin d'adaptation

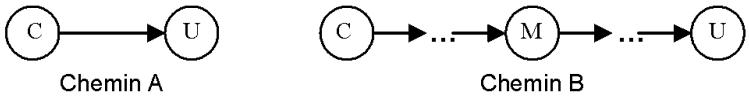

FIG. 4.8 – Chemins d'adaptation

4.7.2 Graphe d'adaptation

Une méthode d'adaptation transforme le contenu d'un état vers un autre afin de satisfaire un contexte cible. Dans certaines situations, plusieurs méthodes d'adaptation sont appliquées sur le contenu d'origine pour arriver à satisfaire les contraintes du contexte cible. Nous représentons les tâches d'adaptation par un graphe acyclique appelé *graphe d'adaptation*. Le graphe d'adaptation inclut des chemins à partir d'une unité du contenu C vers le client cible U. Un chemin peut être de type A (voir Figure 4.8) si l'unité originale du contenu C peut être directement transmise au client. Un chemin de type B est un chemin qui contient des nœuds M_i (Figure 4.8) dont la sémantique est d'appliquer toutes les méthodes d'adaptation M_i dans l'ordre de leur apparition de C vers U. L'application d'un chemin p est équivalente à l'exécution de l'algorithme de la figure 4.7 où (3) signifie que le contenu généré par une méthode d'adaptation M_i est utilisé par la méthode M_{i+1}. M_1 est toujours appliquée sur le contenu d'origine qu'on veut adapter. La méthode $M_{|p|}$ transmet le résultat final de l'adaptation directement au client.

Dans un graphe d'adaptation, un chemin quelconque p est équivalent à un chemin de type A dont le nœud C représente le même contenu généré par la méthode $M_{|p|}$ dans le chemin p. Le contexte d'entrée d'une méthode M (voir Figure 4.9) est le contexte du contenu sur lequel M est appliquée. Le contexte de sortie de M est le contexte du contenu généré par M. Un arc peut relier un nœud M_i à un nœud M_j si le contexte de sortie de M_i correspond aux contexte d'entrée de M_j.

4.7.3 Evaluation de l'adaptation

Une méthode d'adaptation peut consister à supprimer quelques éléments du contenu d'origine (exemple : le filtrage des images d'une page HTML), transformer le format et la structure du contenu (exemple : la conversion des images JPEG vers des images GIF, transformer XHTML vers HTML Basic) ou sélectionner une variante du contenu (voir Figure 4.9).

Si $t(M_i^c)$ est le temps nécessaire pour appliquer la méthode M_i^c dans un chemin p_c

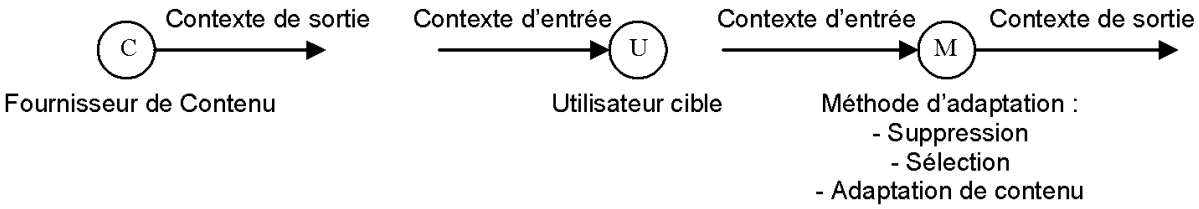

FIG. 4.9 – Nœuds du graphe d'adaptation

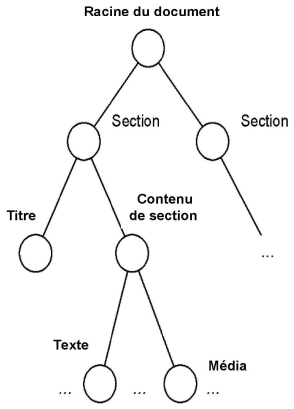

FIG. 4.10 – Organisation hiérarchique

allant du contenu c vers le client U, alors le temps nécessaire pour adapter c égale à $\sum_{i=1}^{|p_c|} t(M_i^c)$. L'adaptation d'un contenu composé C (un contenu qui référence d'autre éléments de contenu, tel qu'une page HTML qui fait référence à des images) peut être représenté par l'union de tous les chemins allant d'un élément de contenu vers le client. Si $C = \bigcup_{i=1}^{n} c_i$, le temps nécessaire pour adapter C égale à $\sum_{c \in C} \sum_{i=1}^{|p_c|} t(M_i^c)$.

Dans le cas où il existe plusieurs chemins qui lient le même élément de contenu avec le client, le processus d'adaptation choisit le chemin qui nécessite un temps d'exécution minimal.

4.8 Adaptation sémantique basée sur l'organisation hiérarchique

Ce type d'adaptation consiste à adapter le contenu selon sa sémantique et en se basant sur la structure hiérarchique du contenu original. L'organisation hiérarchique du contenu concerne deux aspects :

- la sémantique du contenu,
- la présentation du contenu.

Pour satisfaire un tel besoin, différents niveaux de hiérarchie sont définis afin d'appliquer une adaptation de contenu selon le centre d'intérêt de l'utilisateur et les capacités du client. Un document peut être vu comme un ensemble de sections. Le plus bas niveau sémantique d'une section commence par son titre suivi du contenu de la section (voir Figure 4.10)

Lors de la transmission de réponse, la présentation du contenu d'une section dépendra des capacités d'affichage et de traitement du client. Le processus de décomposition du contenu en plusieurs unités de présentation, appelé processus de pagination de contenu [119], considère la dimension *taille_max_page* du contexte client. Cette dimension est instantiée à partir du contexte de transmission et dépend de la taille du dispositif d'affichage de l'appareil client.

4.8.1 Modèle de contenu indépendant des terminaux

Beaucoup de travaux qui ont été effectués sur l'adaptation du contenu pour des appareils limités sont basés sur l'association de différentes présentations du même contenu aux différents types d'appareils [93, 83, 54].

La difficulté de cette approche est qu'elle nécessite beaucoup d'efforts humains pour l'édition et le maintien des différentes variantes. Certaines études ont essayé de résoudre ce problème en fournissant des mécanismes d'adaptation qui soient plus dynamiques en se basant sur des modèles de contenu déjà existant comme HTML [50]. Malheureusement, la majorité des modèles existant dépendent des plate-formes cibles et incluent déjà des informations de présentation qui ne sont pas idéales pour tous les clients (voir Figure 4.13.a).

Puisque la présentation du contenu final dépend des capacités d'affichage et de traitement du client, qui ne sont connus que lors de l'étape de transmission de contenu par le serveur, une approche plus abstraite est adoptée en se basant sur la définition d'un modèle qui soit indépendant des caractéristiques des terminaux et qui exclue, le plus possible, les informations relatives à une présentation spécifique du contenu.

Notre modèle profite des avantages de quelques modèles existants qui utilisent des mécanismes d'indépendance aux terminaux (comme SMIL [109]). Un document du modèle peut être vu comme un ensemble d'unités (ou section au sens contenu du terme) où chaque section possède un titre et peut être organisé selon plusieurs niveaux de détail (voir Figure 4.10). Le titre d'une section représente le niveau le plus bas du contenu d'une section (voir Figure 4.13.b). Chaque unité ou section du document peut utiliser une ressource média (vidéo, audio, image, etc.) en utilisant un élément XML appelé *REF* qui indique uniquement une référence vers une ressource mais qui ne spécifie pas de chemin vers une ressource unique. Une référence de ressource dénote un ensemble de médias qui représentent la même information avec des niveaux de détail et de richesse de présentation différents (voir Figure 4.11). Durant la transmission du contenu final, le processus d'adaptation envoie uniquement la ressource qui correspond le mieux au contexte de transmission.

Les relations entre ressources sont stockées sous forme de profils gérés par le serveur de contenu ou par une entité intermédiaire. La figure 4.11 représente un exemple de profil qui définit des relations entre les ressources R_i (i = 1..6). Comme le montre la figure 4.12, le profil inclut trois références : $R_1 \rightarrow R_3 \rightarrow R_5$, $R_2 \rightarrow R_6$ et $R_4 \rightarrow R_3 \rightarrow R_6$. La définition des relations d'équivalence et de hiérarchie peut dépendre du niveau de détail de l'information qu'une ressource peut donner, de la resolution graphique, etc.

Le modèle considère le contexte de transmission en offrant aux auteurs la possibilité de spécifier les paramètres d'un contexte sous forme d'un ensemble d'attributs et de

```
<?xml version="1.0" ?> <rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:ccpp="http://www.w3.org/2000/07/04-ccpp#"
xmlns:neg="http://www.inrialpes.fr/opera/people/Tayeb.Lemlouma/NegotiationSchema/
RelatedResourcesSchema-18042003#"> <rdf:Description
rdf:ID="RelatedResourcesProfile">

<ccpp:component> <rdf:Description
rdf:ID="RelatedResourcesDescription"> <rdf:type
rdf:resource="http://www.inrialpes.fr/opera/people/
Tayeb.Lemlouma/NegotiationSchema/ResourceProfileSchema-03012002#
RelatedResourcesDescription"/>

<neg:RelatedResources> <neg:Resource idf="ref1">
    <item idf="R2" path="http://http://opera.inrialpes.fr/Smil/T/
    NEGOTIATION/WAP/images/background.jpg" profile="http://opera.inrialpes.fr/
    people/Tayeb.Lemlouma/NegotiationSchema/Profile3.xml"/>
    <item idf="R1" path="location1" profile="profileLocation1".../>
    <item idf="R3" path="location3" profile="profileLocation3".../>
    <item idf="R5" path="location5" profile="profileLocation5".../>
</neg:Resource> <neg:Resource idf="ref2">
        <item idf="R2" path="location2" profile="profileLocation2" .../>
        <item idf="R6" path="location6" profile="profileLocation6" .../>
</neg:Resource> <neg:Resource idf="ref3">
    <item idf="R4" path="location4" profile="profileLocation4" .../>
    <item idf="R3" path="location3" profile="profileLocation3" .../>
    <item idf="R6" path="location6" profile="profileLocation6" .../>
</neg:Resource> </neg:RelatedResources>

</rdf:Description> </ccpp:component>

</rdf:Description> </rdf:RDF>
```

FIG. 4.11 – Les relations entre les ressources médias représentées par un profil

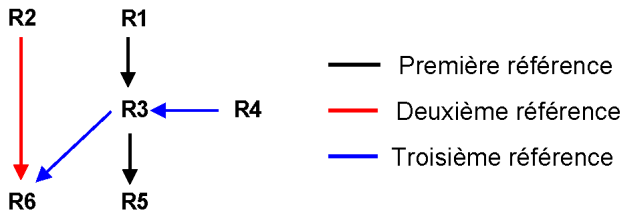

FIG. 4.12 – Référence des ressources

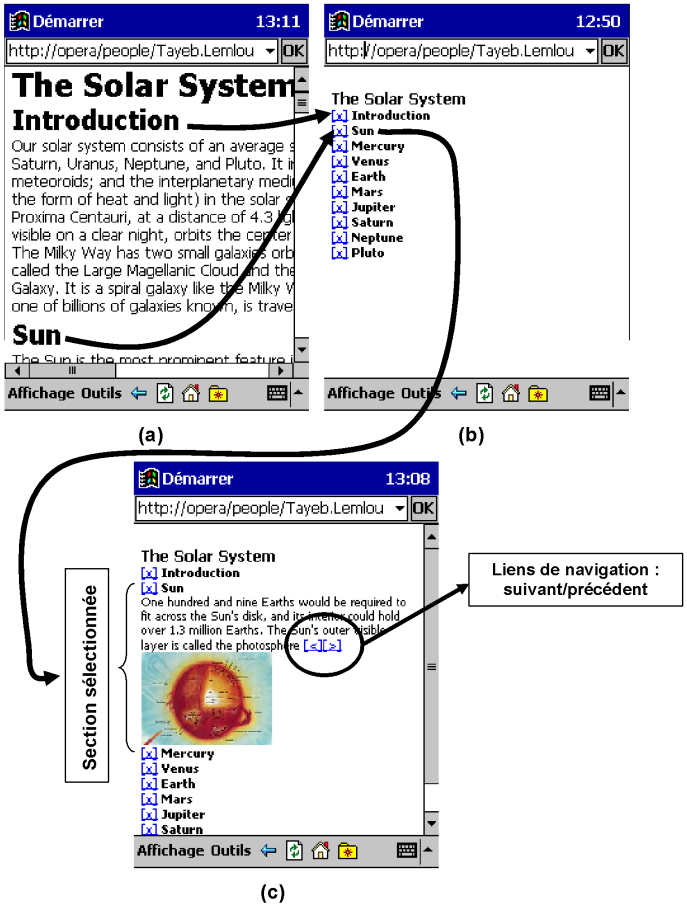

FIG. 4.13 – Adaptation basée sur la hiérarchie et la pagination du contenu

valeurs (exemple : $bande_passante = 512$) associés à une ressource média en utilisant l'élément appelé $SWITCH$ similaire à l'élément SWITCH de SMIL [109].

4.8.2 Pagination et liaison des unités de présentation

Dans quelques situations, l'adaptation du contenu nécessite l'envoi du contenu d'origine en plusieurs parties selon certaines caractéristiques du contexte de transmission (bande passante, capacité d'affichage du terminal, etc.). Ce processus représente une adaptation particulière appelée 'pagination du contenu' [119]. Cette adaptation nécessite une bonne prise en compte des paramètres du contexte et une stratégie qui facilite la navigation entre les différentes parties du contenu.

NAC applique une stratégie de pagination qui n'utilise pas de mécanisme de cache pour éviter le problème d'explosion de taille du contenu sauvegardé. Les parties générées incluent des liens hyper-texte vers des adaptations temps réel qui existent sur le serveur ou sur le proxy. Le lien vers une adaptation est accompagné de paramètres qui permettent de réutiliser la même méthode d'adaptation pour naviguer dans d'autres parties du contenu (voir Figure 4.13.c). En effet, le processus de pagination utilise ces paramètres pour appliquer la tâche d'adaptation la plus appropriée : selection et transmission de la partie du contenu qui correspond à la navigation de l'utilisateur (Exemple : une nouvelle section, la section suivante de la section cou-

rante, etc.). Ce processus est exécuté quand l'utilisateur clique sur le lien de navigation.

La Figure 4.13.a montre un contenu HTML tel qu'il est visualisé par un PC de poche dans une architecture classique (i.e. une architecture sans adaptation ni négociation de contenu). Le même contenu - édité selon notre modèle d'indépendance aux terminaux (voir Figure 4.10) - a été accédé par le terminal sous l'architecture NAC. La figure 4.13.b montre comment le contenu est dynamiquement adapté et visualisé par le terminal. Dans cette situation, l'adaptation résulte en l'extraction dynamique et la transmission du niveau zéro de la hiérarchie du contenu d'origine. Le contenu transmis, inclue des liens générés qui stockent des paramètres permettant d'instancier des des méthodes d'adaptation au niveau proxy. Ces méthodes d'adaptation permettent de suivre l'organisation hiérarchique du contenu que ce soit structurelle ou concernant les ressources médias. Ainsi un lien peut générer une compression ou un retaillage de ressources médias si par exemple l'utilisateur se contente de recevoir un niveau d'hiérarchie pas trop détaillé du contenu ou d'une partie du contenu. La Figure 4.13.c montre le contenu reçu après une interaction de l'utilisateur. L'adaptation transmet uniquement la partie demandée par l'utilisateur et la pagine (divise en plusieurs pages) selon les capacités d'affichage du terminal cible.

4.9 Traitement de la structure et des ressources médias

Un document existant côté serveur est considéré selon deux aspects :

1. la structure globale du document, comme par exemple la structure d'un document SMIL qui décrit le scénario temporel d'une présentation multimédia.

2. les ressources utilisées par le document, comme dans le cas d'un document SMIL qui utilise plusieurs ressources médias telles que des vidéos, des images, etc.

Les dimensions d'un contexte client, déclarées dans les différents profils, concernent un des deux aspects précédents du document demandé. Par exemple, le profil d'un téléphone mobile qui ne supporte que le format MMS et les images de type GIF comporte deux types de contraintes : le premier type concerne la structure des documents supportés et le deuxième type concerne le format des ressources médias utilisées.

Afin d'adapter le contenu source aux contraintes du client, il est donc important que le système d'adaptation se dote de mécanismes permettant la création de profils de contenu (dans le cas où de tels profils n'existent pas sur le serveur) et de mécanismes d'adaptation des différentes ressources qui peuvent être utilisées dans le document demandé par l'application cliente.
Dans certaines situations, où le contenu source ne possède pas de descripteur, il est important de définir un mécanisme qui permet la création du profil du contenu.

L'algorithme présenté par la Figure 4.14 montre comment créer un profil UPS à partir de l'arbre du document source. L'algorithme fait appel à la procédure récursive *Treat_node* appliquée sur la racine du document (ligne 1). La procédure *Treat_node* s'applique sur un nœud d'arbre (ligne 2) et explore toutes les ressources médias du nœud afin d'extraire les caractéristiques de la ressource utilisées par le nœud père.

```
Algorithm1:

1: Treat_node(Document_root_node);
2: Procedure Treates_node(n)
   {
3:  If (n represents a media resource)
    Then
4:        create an entry in the output UPS profile;
5:        explore the attributes of node n;
6:        create media output attributes;
    Else
7:        if (n contains other child  nodes)
          Then
8:            for each child s do Treat_node(s);
          fi;
    fi;
   }
```

FIG. 4.14 – Création de profil à partir d'un document structuré

L'algorithme génère en sortie un profil UPS qui contient toutes les caractéristiques des ressources médias utilisées dans le document. La Figure 4.15 montre un exemple de profil UPS qui décrit une page HTML.

L'algorithme présenté par la Figure 4.16 permet la transmission de ressources média adaptées d'un document décrit par le profil UPS généré par l'algorithme de la Figure 4.14 pour un profil client donné. Le principe de cet algorithme est de transmettre les ressources médias supportées par le client cible (ligne 4) et de substituer les ressources non supportées par les variantes correspondantes (lignes 5-8) ou par les ressources générées par les méthodes d'adaptation nécessaires (lignes 9-12). Les procédures de recherche utilisées dans cet algorithme font appel à d'autres types de profils UPS tels que le profil de méthode d'adaptation ou le profil de ressource client (voir Section 4.6.1).

4.10 Conclusion

Dans ce chapitre, nous avons présenté le système de description et de gestion des contextes de l'architecture NAC. Ce système fondé sur les différents schémas du modèle de description universelle UPS permet de prendre les meilleures décisions de négociation et d'appliquer ces décisions sur la base du contexte de l'environnement. Ces décisions peuvent être la transmission du contenu source, la sélection de contenu ou des variantes ou l'adaptation du contenu (transformation structurelle et adaptation des ressources médias).

Le chapitre qui suit est consacré au système d'adaptation et au protocole de négociation de NAC qui se base évidemment sur les mécanismes de description et de gestion des contextes étudiés dans ce chapitre.

```
<?xml version="1.0"?> <rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:ccpp="http://www.w3.org/2000/07/04-ccpp#"
xmlns:neg="http://www.inrialpes.fr/opera/people/Tayeb.Lemlouma/
NegotiationSchema/DocumentInstanceSchema-03012002#">
<rdf:Description rdf:ID="DocumentInstanceProfile">
    <ccpp:component>
     <rdf:Description rdf:ID="DocumentInstanceDescription">
        <rdf:type rdf:resource="http://www.inrialpes.fr/opera/people/
        Tayeb.Lemlouma/NegotiationSchema/DocumentInstanceSchema
        -03012002#DocumentInstanceDescription"/>
        <neg:InstanceType>EXAMPLE.html</neg:InstanceType>
        <neg:InstanceFormat>html</neg:InstanceFormat>
        <neg:InstanceVersion>4.01</neg:InstanceVersion>
        <neg:InstanceDoctype>
          html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
          "http://www.w3.org/TR/html4/loose.dtd"
        </neg:InstanceDoctype>
        <neg:InstanceSize>910Bytes</neg:InstanceSize>
        <neg:InstanceLocation>Smil/T/NEGOTIATION/WAP/</neg:InstanceLocation>
        <neg:InstanceServer>LocalServer</neg:InstanceServer>
      </rdf:Description>
    </ccpp:component>
    <ccpp:component>
     <rdf:Description rdf:ID="MultimediaContent">
        <rdf:type rdf:resource="http://www.inrialpes.fr/opera/people/
        Tayeb.Lemlouma/NegotiationSchema/DocumentInstanceSchema
        -03012002#MultimediaContent" />
        <neg:content>
        <rdf:Bag>
        <rdf:li rdf:parseType="Resource">
            <neg:ResourceType>image</neg:ResourceType>
            <neg:RessourceName>background.jpg</neg:RessourceName>
            <neg:ResourceFormat>jpg</neg:ResourceFormat>
            <neg:ResourceSize>25824Bytes</neg:ResourceSize>
            <neg:ResourceDimension>800x600</neg:ResourceDimension>
            <neg:colorBit>24</neg:colorBit>
            <neg:RessourceLocation>smil/images/</RessourceLocation>
            <neg:RessourceServer>http://yap.inrialpes.fr/</neg:RessourceServer>
            ...
        </rdf:li>
        <rdf:li rdf:parseType="Resource">
            <neg:ResourceType>audio</neg:ResourceType>
            ...
        </rdf:li>
        <rdf:li rdf:parseType="Resource">
            <neg:ResourceType>video</neg:ResourceType>
            ...
        </rdf:li>
          ...
        </rdf:Bag>
        </neg:content>
      </rdf:Description>
    </ccpp:component>
  </rdf:Description>
</rdf:RDF>
```

FIG. 4.15 – Exemple de profil UPS généré

```
Algorithm2:

1: For each
    (media resource category X contained in the profile of the document to be
    delivered)
2: Do
3: If
        (All attributes and elements match to the corresponding resource
        Category X' (in the client profile))
    Then
4:      X is to be delivered
    Else
5:      Look for X-related resources (equivalent-to or
        adapted-to);
6:      Evaluate the resources;
7:  If (an appropriate resource exist)
    Then
8:      deliver it
    Else
9:      Look for available methods to make X adaptable regardless
        X' Constraints;
10:     If (such methods)
        Then
11:         Apply the method to X;
12:         Deliver the result
        Else
13:         Remove X;
        Fi;
    Fi;
    Fi;
 End for each
```

FIG. 4.16 – Traitement d'un profil document

Chapitre 5

Système d'adaptation et protocole de négociation de NAC

Résumé

Ce chapitre présente les processus d'adaptation de l'architecture NAC ainsi que les stratégies et protocoles de négociation adoptés. Comme nous allons voir, le système d'adaptation défini permet de considérer les différents aspects et fonctionnalités d'un contenu. Quant au protocole de négociation, il permet de prendre la meilleure décision d'adaptation qui prend en compte les caractéristiques du contexte de l'environnement.

Contenu

5.1 Introduction

Nous avons vu dans le chapitre précédent que la description et la gestion du contexte de l'environnement représentent des éléments de base pour assurer une adaptation efficace du contenu. Comme ces descriptions concernent plusieurs entités distantes du réseau, il est nécessaire que ces entités communiquent entre elles et coopèrent afin d'aider à atteindre l'objectif de l'adaptation de contenu. Cette coopération est

importante afin de considérer toutes les contraintes de l'environnement mais aussi afin d'exploiter les possibilités d'adaptation du système.

Ce chapitre présente les processus d'adaptation de l'architecture NAC, qui s'applique à toutes les dimensions d'un contenu accessible à l'ensemble des terminaux, ainsi que le protocole de négociation adopté. Le chapitre inclut un formalisme du système d'adaptation afin de comprendre les différentes techniques utilisées pour adapter un contenu donné pour un contexte cible donné. Comme l'application des différents mécanismes d'adaptation dépend profondément de la stratégie de la négociation adoptée, nous discutons en détail l'approche de négociation de contenu de NAC. Nous présentons la négociation de ressources de contenu ainsi que le protocole de négociation pour l'acquisition du contexte. Ce dernier représente une manière optimale pour assurer la prise en charge de l'acquisition du contexte directement du client mais surtout la considération des changements de contexte qui peuvent être fréquents dans un environnement hétérogène.

Comme nous allons voir, le système de négociation représente le processus qui lie l'aspect déclaration avec l'aspect action d'une architecture d'adaptation. L'aspect déclaration est assuré par le système de description du contexte, tandis que l'aspect action est assuré par l'ensemble des méthodes et des techniques d'adaptation de contenu disponibles dans le système.

La première partie de ce chapitre est consacrée à la présentation du processus d'adaptation, ses principes et les différents mécanismes d'adaptation appliqués. Dans le reste du chapitre nous présentons l'aspect stratégie et protocoles de négociation. Nous montrons comment les entités de l'environnement hétérogène peuvent suivre des règles d'échange de messages de négociation afin de permettre de prendre une décision qui reflète les capacités du système d'adaptation et qui respecte les contraintes de l'environnement.

5.2 Structure des documents multimédia

Une bonne modélisation des documents multimédia est nécessaire pour mettre en place des mécanismes d'adaptation efficaces. Les fonctionnalités d'un contenu multimédia peuvent être décrites à l'aide des quatre dimensions suivantes :

1. La dimension logique
2. La dimension spatiale
3. La dimension temporelle
4. La dimension hypermédia

La spécification de l'organisation logique correspond à la décomposition d'un système d'information en composants. L'auteur détermine le sujet à présenter et commence à définir les différents composants et la façon selon laquelle ils seront synchronisés. Par exemple, pour créer un contenu multimédia présentant une équipe de recherche, on peut décomposer le sujet en deux composants : une partie générique qui introduit l'équipe et une partie thèmes de recherches qui présente les activités de recherche au sein de l'équipe (voir l'exemple de la Figure 5.1).

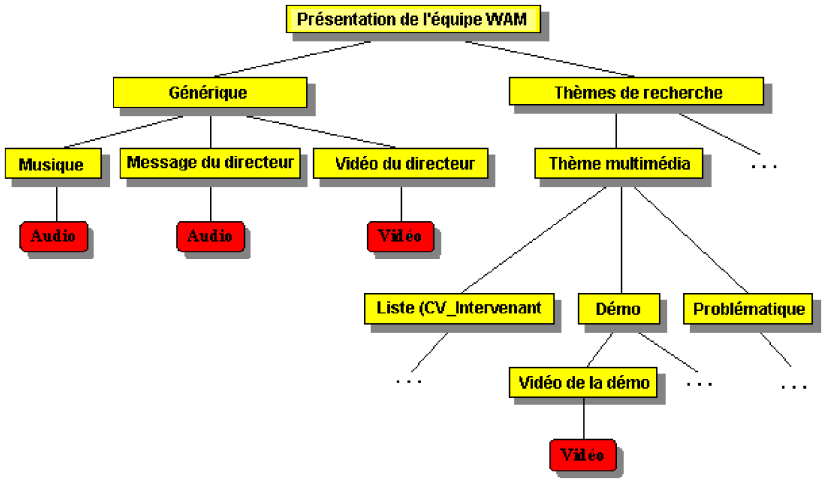

FIG. 5.1 – Structure logique d'une présentation de contenu multimédia

L'opération de décomposition est récursive, c'est-à-dire que chaque composant peut être également décomposé en d'autres composants. Par exemple, le composant "Générique" peut être décomposé en une musique de fond avec un message audio et une vidéo du directeur de l'équipe (voir Figure 5.1).

L'opération de décomposition termine lorsque le composant représente un média de base (exemple : texte, image, vidéo, audio, etc.). Le résultat de la décomposition donne une structure logique hiérarchique pour la présentation du contenu multimédia.

La dimension spatiale concerne l'affectation dans le temps des composants multimédia d'un document aux ressources physiques. Par exemple, un élément vidéo doit disposer d'un espace géométrique sur l'écran délimité par une certaine zone. D'une manière similaire, un élément audio doit disposer d'un canal audio pour toute la durée de sa présentation. L'opération de mise en correspondance est appelée formatage spatio-temporel [68] [20].

La spécification de l'organisation temporelle définit l'ordonnancement temporel de la présentation des différents composants afin de construire le scénario souhaité. Quatre formes principales de synchronisation temporelle sont distinguées : la synchronisation intra-objet, la synchronisation inter-objet, la synchronisation avec l'environnement et la synchronisation de groupe [9] [68] [61] [115].

De par la complexité de la tâche de spécification temporelle, il est nécessaire d'avoir un mécanisme efficace pour structurer les scénarios temporels. De plus, ce mécanisme doit pouvoir être supporté par un ensemble d'algorithmes pour vérifier la cohérence du scénario spécifié. Lorsque la cohérence du document est vérifiée l'exécution est possible grâce à un système de présentation [68].

Comme nous avons vu, la structure logique d'un contenu multimédia est fondamentalement hiérarchique. La représentation hiérarchique n'est pas adaptée pour décrire l'ensemble des relations qui peuvent exister entre les éléments du même document ou entre des documents différents. Dans [120], l'américain V. Bush a introduit pour

la première fois les notions qui se trouvent à la base des hypermédias et des hypertextes. L'objectif était de concevoir un système, Memex [121], permettant de faire face à l'explosion d'informations qui devait s'inspirer du principe associatif régissant le fonctionnement du cerveau humain. Cette idée a été reprise au début des années 1960 par D. Englebert, dans un projet "d'augmentation de l'intellect humain" (to augment the human intellect) [32] qui permet de se déplacer dans un espace multidimensionnel d'informations et de communiquer avec les autres utilisateurs ayant accès à cet espace. Les liens permettant de définir des relations de type sémantique entre les documents ou les parties de documents sont appelés liens hypermédia. La dimension hypermédia met donc en place un réseau de parties de documents dépassant les structures qui les englobent et constituent ainsi un support pour la navigation dans l'espace d'information.

5.3 Processus d'adaptation de contenu

Assurer la tâche de l'adaptation du contenu dans une architecture globale nécessite la considération de plusieurs paramètres de l'environnement. Dans cet environnement une requête de demande de contenu peut être reçue et doit être satisfaite par le fournisseur du contenu. Satisfaire une requête revient à satisfaire toutes les contraintes qu'elle peut comporter : le type du contenu, la nature de la connexion, les contraintes du réseaux utilisé, etc. D'un autre côté, des mécanismes de base doivent être disponibles pour pouvoir couvrir les différents types d'adaptation. Ces types sont nécessaires pour l'accomplissement de la tâche d'adaptation et de transmission du contenu final. Dans cette section nous détaillons les mécanismes de base nécessaires à la création d'un processus d'adaptation de contenu qui prend en compte les différentes sortes de fonctionnalités qu'un contenu peut inclure.

5.3.1 Principe de l'adaptation de contenu

Le problème du contenu existant actuellement sur le Web est qu'une grande partie de ce contenu a été créé initialement pour des utilisateurs utilisant des plates-formes "riches". Ces modèles, souvent appelés "modèles d'ordinateur de bureau", sont riches en ressources mémoire, capacité de traitement (logiciels de présentation sophistiqués, codecs de médias..), bonne connectivité, etc. Beaucoup d'auteurs adoptent donc cette vision du client cible et développent leur contenu en suivant un modèle de presentation unique. La conséquence de ce fait est que les terminaux qui présentent des ressources limitées que ce soit logiciel, matériel ou réseau de communication, sont incapables d'accéder et utiliser ce contenu. Il est donc nécessaire de transformer cette base de contenu de son état initial vers un état plus accessible et plus utilisable sous ces nouvelles plates-formes. Dans la même direction, le mode d'édition du contenu devra suivre des nouvelles directives universelles afin que le contenu gagne plus de visibilité. Créer un contenu dans un modèle qui soit adaptable et plus indépendant des plates-formes économise l'effort de création de multiples versions et facilite la génération du contenu pour différentes plates-formes. Un contenu peut donc être créé en XML et transformé en HTML, VoiceXML ou HTML Basic. Les efforts de quelques groupes de travail, notamment le groupe d'indépendance au terminaux [119](DI : Device Independence Working Group) et le groupe d'interaction multimodale [45] (MMI : Multimodal Interaction Working Group) du consortium W3 vont dans le sens de développer cette vision universelle d'accès et d'édition.

5.3.2 Mécanismes d'adaptation

L'approche la plus évidente pour assurer un système d'adaption de contenu consiste à créer plusieurs formes du même contenu et cela pour chaque client cible du système. Chaque forme de contenu, qu'on appelle aussi version ou variante, doit satisfaire les contraintes du client cible.

Cette approche, appelée *multi-édition* [48] [106] [124] [113] [123] [81], présente plusieurs inconvénients :

1. L'utilisation d'un espace de stockage important du côté fournisseur de contenu spécialement si la masse du contenu d'origine est importante (une grande base de documents par exemple).
2. Nécessite un effort non négligeable durant l'étape de création et de maintenance du contenu. Cela peut aussi résulter en l'utilisation inutile d'un espace de stockage dans le cas où le contenu est de nature dynamique. Pour éviter une telle explosion de l'espace, la maintenance du contenu source nécessite une suppression continue de documents et de fragments de documents due au changement fréquent du contenu.
3. L'approche de multi-édition de contenu ne peut ni tenir compte de toute la diversité des terminaux qui peuvent exister dans un système d'adaptation, ni considérer les nouvelles contraintes imposées par des nouveaux clients qui peuvent apparaître dans le futur.

Ce dernier point montre que l'ensemble des clients et des fournisseurs de contenu doivent coopérer dans une architecture globale afin d'offrir une meilleure exploitation du contenu existant dans un système multimédia.

Certains travaux ont discuté l'adaptation du contenu en se basant uniquement sur un seul type de ressource multimédia. Balabanovic introduit dans [4] l'adaptation des documents multimédia, (basée sur un modèle de vecteur d'espace [104]) en traitant les ressources purement textuelles qui peuvent exister dans un contenu. Ce genre d'approche reste non applicable sur la large base de documents qui existe aujourd'hui. En effet cette base connaît déjà une explosion d'utilisation de différentes resources multimédia (vidéo, son, animations, etc.).

L'objectif du système d'adaptation de l'architecture NAC est de considérer toutes les ressources qui peuvent exister dans un contenu fourni par les serveurs. Une ressource de contenu peut représenter un texte, une image, une video, une ressource audio, etc. Considérer les différents types de ressources permet d'assurer l'envoi d'une réponse adaptée qui peut être exploitée par les différents clients.

Un système d'adaptation efficace doit couvrir le plus large ensemble des formats existants, tel que HTML Basic, VoiceXML, XHTML, etc. Cela revient à assurer, au niveau de l'entité de transformation, les méthodes d'adaptation qui s'appliquent sur le contenu d'origine et donnent en sortie le format cible.

Le but principal du processus d'adaptation est de rendre le contenu utilisable par le client cible et cela en appliquant des transformations sur le contenu d'origine. Afin d'atteindre cet objectif, il est indispensable de tenir compte de la description du

contexte de l'utilisateur, ou en d'autres termes de ses capacités et préférences vis-à-vis de l'utilisation du contenu. Comme nous avons vu, quelques mécanismes de description du contexte existent déjà tels que les en-têtes *accept* du protocole HTTP et les éléments *ALT* [53] [35] [41]. CC/PP [43] représente un autre cadre de travail proposé par le W3C pour la description des capacités et des préférences des terminaux. Malheureusement ces mécanismes présentent des limitations et ne peuvent pas être utilisés dans une architecture d'adaptation efficace. En effet, le système d'adaptation doit être doté d'un mécanisme de description plus riche qui considère les différentes fonctionnalités du contenu et les caractéristiques des clients. En outre, la description doit couvrir d'autres entités qui peuvent jouer un rôle dans l'adaptation finale du contenu demandé.

Dans le processus d'adaptation, la description des caractéristiques des clients est importante mais elle n'est pas suffisante. Il est aussi important de fournir une description du serveur du contenu et de son environnement. Une telle description doit au moins inclure les informations relatives à :

1. Ce qui existe côté serveur
2. Les capacités du serveur

Le premier point implique que le contenu du serveur doit être décrit. Aucune restriction n'est faite sur le type du contenu. Le deuxième point est nécessaire pour la définition d'une stratégie de négociation, comme nous allons voir par la suite. Cela nécessite, du point de vue adaptation, la définition de méthodes d'adaptation (feuilles de styles XSLT par exemple) et la description de ces méthodes en termes de format (ou modèle) d'entrée et de sortie. Le format de sortie est celui du contenu généré par une méthode d'adaptation.

Dans ce qui suit, nous essayons de formaliser quelques aspects du système d'adaptation et de négociation. Notre objectif est de déterminer le rôle et les tâches des différents composants du système et de bien comprendre le fonctionnement de l'architecture globale.

Soit S un système d'adaptation et de négociation de contenu. Pour simplifier, nous considérons que S comporte un seul client et un seul serveur. Un système d'adaptation et de négociation dans un environnement hétérogène est défini comme suit :

$$S = (UA\_P, T, D\_P)$$

Où,

- UA_P représente l'ensemble des profils de l'application cliente[1]. Comme nous avons vu, les profils de l'application cliente sont écrits en utilisant le mécanisme de description du système.

- T est l'ensemble des méthodes d'adaptation et de transformation utilisées par le serveur du contenu. T est appliqué sur l'ensemble du contenu original et donne en sortie un contenu adapté (voir Figure 5.2).

- D_P est l'ensemble des profils documents qui existent au niveau du serveur : c'est à dire la description du contenu en termes de fonctionnalités et de modules

[1] ici, on suppose qu'il y a qu'une seule application cliente par terminal

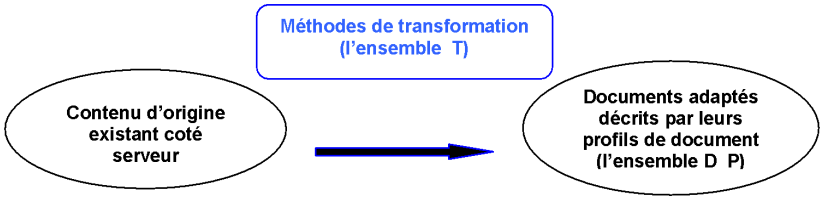

FIG. 5.2 – Adaptation du contenu

supportés. Selon la stratégie de la négociation, cet ensemble peut être statique ou dynamique.

5.3.3 Adaptation structurelle

Le processus d'adaptation consiste à appliquer des transformations (l'ensemble T, voir Figure 5.2) sur le contenu multimédia qui existe au niveau du serveur du contenu.

Adapter un contenu multimédia revient à considerer toutes ses dimensions et déterminer à chaque fois le traitement nécessaire pour adapter la dimension considérée. Cela ne nécessite pas que les différentes dimensions doivent être indépendantes. En effet les sous tâches d'adaptation coopèrent d'une manière permettant la transmission d'un contenu qui assure une utilisation optimale des ressources disponibles dans l'environnement cible. Par exemple, dans un environnement où l'espace d'affichage est limité, la technique d'adaptation pour la gestion économique de la dimension spatiale pourrait être l'utilisation d'une même région par différents objets en profitant de la connaissance préalable du scénario temporel.

5.3.3.1 Adaptation logique

Le contenu original peut être vu comme un ensemble d'objets multimédia (texte, images, vidéo, audio, etc.) qui composent le scénario multimédia final. Afin d'adapter la présentation de ces objets sous un contexte particulier, l'organisation logique du contenu peut être réorganisée vers une nouvelle forme qui considère mieux les contraintes du nouveau contexte. Le paragraphe 4.8 discute un cas particulier de telles réorganisations de structure logique basée sur une sémantique hiérarchique du contenu.

Exemple 1 : le contexte utilisateur inclut les deux contraintes suivantes : {'liste de listes supportée', 'tables non supportés'}. Action d'adaptation : le processus d'adaptation transforme toutes les 'tables' du document en liste de listes. Cela peut être appliqué en utilisant une simple transformation XSLT ou en appliquant des algorithmes de transformation avancés [82].

Exemple 2 : le contexte utilisateur inclut les contraintes suivantes : {'texte supporté', 'images non supportées', 'audio supporté'}. Il est clair dans ce cas là que le processus d'adaptation ne doit pas transmettre au client un contenu qui contient des ressources images. Le processus d'adaptation doit transformer les images en un format supporté. Les images peuvent être représentées par un texte équivalent [73], ou par un format

audio prononcé lorsque l'utilisateur clique sur le lien de l'image (voir Figure 5.3). La décision finale du choix entre les deux formes de présentation dépendra des préférences de l'utilisateur et cela indépendamment des capacités du terminal.

5.3.3.2 Adaptation spatiale

Ce type d'adaptation vise à optimiser les ressources spatiales du système de présentation cible [14] [15]. Ici, l'adaptation considère un vocabulaire décrivant les relations qui peuvent exister entre les différents objets et éléments composites de la présentation multimédia. Ces relations sont appelées *relations spatiales* [122]. Les relations spatiales décrivent principalement l'alignement d'objets média sur l'espace d'affichage dans les deux dimensions : horizontale et verticale [20]. Ces relations peuvent représenter les types de placement possibles comme : le centrage (verticale ou horizontale), l'alignement (de droite, de gauche, de haut ou d'en bas), l'espacement (droit, gauche, de haut ou d'en bas) et le décalage (droit, gauche, de haut ou d'en bas).

L'adaptation peut être assurée en réduisant la qualité des unités de présentation. Par exemple, dans le cas d'adaptation d'images, la résolution des images peut être réduite (voir la technique $T1$, Figure 5.3). Une autre possibilité dans le cas d'optimisation spatiales, est de partager les régions d'affichage entre plusieurs objets média. Considérons par exemple un contenu original qui comporte cinq images sans ou avec des intervalles de visualisation non chevauchés. Afin d'adapter de tel contenu, une technique d'adaptation pour un espace d'affichage réduit consiste à afficher régulièrement (pendant un intervalle de temps raisonnable) les cinq images dans la même région spatiale. Cette technique permet d'exploiter l'introduction d'une dimension temporelle au contenu d'origine qui peut comporter qu'une dimension spatiale pure. Dans ce cas précis, le processus d'adaptation doit être sûr que le système de présentation cible supporte les fonctionnalités temporelles rajoutées.

5.3.3.3 Adaptation temporelle

Si la dimension spatiale du contenu multimédia considère les deux dimensions de l'espace, le modèle temporel se base sur une unique dimension qui est le temps [62]. Le temps représente un ordonnanceur pour le traitement des objets multimédia dans un système de présentation. Plusieurs langages assurent, de différentes manières, une spécification des règles de présentations temporelles. SMIL [109] exprime le positionnement des objets multimédia en utilisant les opérateurs parallèles et séquentiels associés à des intervalles de temps. Magic [28] et Madeus [68] utilisent des restrictions de l'algèbre d'Allen sur les intervalles temporelles [2]. La spécification des règles temporelles peut être assurée en utilisant des relations qualitatives définies entre objets. Exemple, des relations du genre : "commencer avant", "commencer après", "finir avec", etc. [68]. Dans ce genre de spécifications, le système de présentation est l'entité responsable du calcul et de la planification des scénarios à jouer. Ce mécanisme est appelé *le formatage temporel*.

Certaines contraintes du contexte du terminal cible (plus généralement du contexte de transmission du contenu) peuvent rendre impossible de respecter le scénario temporel tel qu'il a été spécifié par l'auteur du contenu source. Par exemple, il est impossible de respecter la visualisation simultanée de plusieurs objets multimédia si l'espace d'affichage est très réduit. Afin de satisfaire les contraintes du contexte de transmission, le contenu multimédia doit être adapté avant d'être passé au système

de présentation cible. Plus le système d'expression de contraintes est expressif, plus le résultat de l'adaptation est satisfaisant [77] [76].

Kim et al proposent dans [65] une approche d'adaptation temporelle basée sur la notion de temps élastique afin d'adapter la durée d'une présentation aux préférences de l'utilisateur. Dans [34], une spécification de modèle temporel (basée sur l'algèbre d'intervalles temporelles d'Allen [2]) ainsi qu'une sémantique d'adaptation sont introduites. L'adaptation sémantique est discutée en termes de deux catégories : la catégorie des *adaptations de raffinage* et celle des *adaptations transgressives*. La première catégorie comporte les adaptations dont le modèle du document adapté est celui du document source, tandis que la deuxième catégorie comporte les adaptations dont le modèle du document adapté est le plus proche que possible du modèle source. Le travail inclut un formalisme de certains types de contraintes (tel que la contrainte de visualisation simultanée d'objets) ansi que quelques métriques qui peuvent être considérées dans la proximité des modèles tel que la distance conceptuelle entre intervalles et entre modèles, etc. [34].

L'architecture NAC inclut des techniques d'adaptation temporelles plus riches [71] [75] basées sur un système d'expression de contraintes plus flexible [77]. UPS assure une expressivité forte en se basant sur des contraintes posées sur la spécification des modèles du contenu source (exclusions, conteneurs, conjonctions, disjonctions, etc., voir Chapitre 4) en plus des contraintes prédéfinies (voir Annexe A). Ces contraintes permettent d'effectuer des traitements raffinés sur des fonctionnalités temporelles surtout si le modèle source est très expressif tel que le langage SMIL [75]. NAC adopte une approche additionnelle qui permet de satisfaire les contraintes temporelles des clients limités en éliminant les dimensions temporelles du contenu source. Dans ce cas, le processus d'adaptation se charge de la partie analyse de scénario temporel et génère un flux de type unique supporté par l'environnement du client cible [71].

Une technique d'adaptation temporelle concernant les contenus à unités de présentation uniformes est de réduire la cadence de présentation de ces unités (ou le nombre d'unités de présentation par intervalle de temps noté NPU) [79] [10]. Cette approche est appliquée dans la suppression de frames des flux vidéo [71] [51], l'élimination de groupes d'images [24], l'association de priorités à des frames [22] [86]. Afin de mesurer, et par conséquent contrôler, le taux d'adaptation dans ce genre de processus nous définissons le taux ($Taux$) comme suit :

$$Taux = 100\ (1 - (\frac{Nouveau\ NPU}{Ancien\ NPU})$$

Où $Nouveau\ NPU$ est le le nombre d'unités de présentation par unité de temps relatif au contenu adapté et $Ancien\ NPU$ est celui relatif au contenu source. Cette formule est valable si $Nouveau\ NPU$ est fixe pour tout le contenu généré. Dans le cas contraire, la formule suivante est adoptée :

$$Taux = 100\ (\sum_{i=1}^{nf} \frac{1 - \frac{Nouveau\ NPU_i}{Ancien\ NPU_i}}{nf})$$

Cette formule donne le taux d'adaptation moyen si $Nouveau\ NPU$ est variable avec nf est le nombre total de frames du flux initial. La variable $Taux$ peut donc être contrôlée selon les contraintes du contexte de transmission. Un exemple pratique

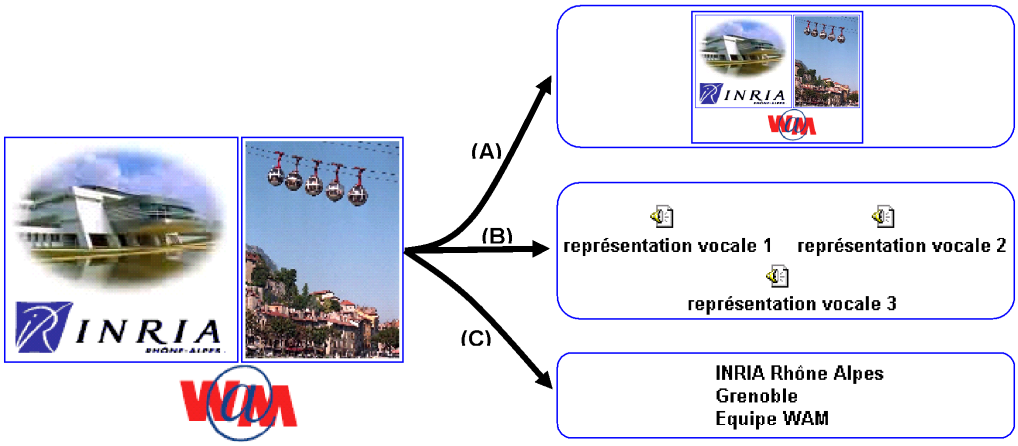

FIG. 5.3 – Techniques d'adaptation

serait de modifier la valeur de *Taux* selon la vitesse du réseau ou les ressources du client [71]. Dans [56], le flux vidéo transmis (par conséquent, la valeur de la variable *Taux* entre autres) dépend principalement de la vitesse du réseau allant du serveur du contenu vers l'utilisateur final.

Notons que le système d'adaptation peut disposer de plusieurs techniques qui peuvent être utilisées pour adapter les différentes dimensions d'un contenu multimédia. Le choix d'une méthode d'adaptation dépend de l'environnement de transmission et de ses contraintes. Ce choix est déterminé par le processus de négociation.

5.4 Principe de la négociation de contenu

L'objectif du processus de négociation de notre architecture est d'assurer une forme de consensus entre les fournisseurs du contenu et les clients. Les fournisseurs du contenu sont représentés par l'ensemble des serveurs et des proxy intermédiaires - jouant le rôle de serveurs auprès de leurs clients - qui existent dans un système hétérogène. Une architecture d'adaptation et de négociation peut être représentée par une boîte noire qui accepte en entrée les requêtes des clients et retourne en sortie des services adaptés destinés à être utilisés par ces clients (voir Figure 5.4). Cette boîte noire, comporte principalement deux processus :

1. Le processus de négociation
2. Le processus d'adaptation

Plusieurs requêtes peuvent être échangées entre le client et le serveur afin d'accomplir l'étape de négociation. Le choix final du contenu à transmettre représente le résultat de cette étape.

Assurer une stratégie de négociation complète revient à :

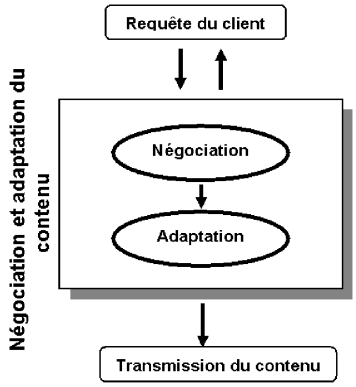

FIG. 5.4 – Vue globale sur le système d'adaptation et de négociation du contenu

1. Considérer la diversité des capacités des clients et de leurs préférences. Cela inclut la prise en compte des contraintes des applications clientes et l'application des règles permettant de faire une correspondance entre les contraintes des clients et les fonctionnalités du contenu demandé.

2. Déterminer les formats cibles en termes de fonctionnalités sélectionnées.

3. Déterminer les transformations à appliquer (lesquelles et comment ?)

4. Supporter les contraintes du contexte global de transmission : client, réseau, contenu, protocole de communication, etc.

Une solution efficace de négociation de contenu exige la connaissance approfondie du modèle cible du contenu (c'est à dire le modèle du contenu adapté). Cependant, les solutions de négociation partagent un ensemble commun de concepts et de techniques qui sont indépendants de l'environnement cible sous lequel la solution d'adaptation doit être opérationnelle.

5.4.1 Système de négociation

La stratégie de négociation la plus connue est celle basée sur le principe de sélection des variantes (voir Chapitre 2). Cette approche suppose que le contenu d'origine est édité en plusieurs versions, par exemple en plusieurs langues ou avec différents niveaux d'abstraction. Nous évaluons le nombre de versions à éditer dans ce genre d'approche à $\prod_{i=1}^{nd} |d_i|$, tel que nd est le nombre de dimensions de contexte considérées et $|d|$ est le cardinal de la dimension d. Par exemple, le nombre de versions à éditer pour produire un contenu adapté en langue Française et Anglaise et pour des terminaux de type *ordinateur de bureau* et *assistant personnel* est égal à 4 ($|langue|.|typedeterminal| = 2.2$).

La multi-édition dépend de la diversité des contraintes des clients cibles. La sélection du contenu à transmettre revient à trouver la meilleure version qui satisfait le plus les contraintes de l'application cliente. Cette approche ne devra pas interdire l'application d'autres techniques de transformation sur les versions existantes quand

cela est nécessaire.

Cette approche suppose que les fonctionnalités du contenu sont explicites. En outre, elle exige la connaissance de la diversité des contraintes existantes des clients et cela au moment de la création du contenu. Cette approche peut être appliquée dans les systèmes de négociation où l'hétérogénéité des applications clientes et le volume du contenu à servir sont réduits. Une approche plus efficace est celle basée sur l'édition unique où il y a moins de suppositions sur des contraintes explicites du contexte cible. La multi-édition est à minimiser quand cela est possible.

Le processus de négociation doit inclure trois principaux éléments :

1. *Une description du contenu du serveur*, telle que nous l'avons déjà définie comme profils de document.

2. *Une description des contraintes du client.*

3. *Un protocole de négociation*, c'est-à-dire l'ensemble des règles qui doivent être suivies pour adapter le contenu du serveur en respectant les contraintes de capacités et de préférences du client. Afin de répondre aux requêtes clientes, les règles de négociation appliquent des mécanismes prédéfinis pour adapter et/ou sélectionner le contenu.

5.4.2 Qualité du service de la négociation

Dans tout système fournisseur de services à un ensemble de clients, le paramètre qualité de service ne doit pas être ignoré. La qualité de service permet d'avoir une idée sur la manière dont les services sont utilisés et de contrôler l'efficacité du système appliqué. Afin d'évaluer et de contrôler la qualité du service de négociation de l'architecture NAC, nous identifions les facteurs suivants :

1. Le temps de réponse, c'est le temps pris entre la requête cliente et la réponse du module d'adaptation et de négociation ANM.

2. Le temps de l'adaptation.

3. L'utilisation du réseau de communication en termes de quantité d'informations de négociation échangées dans le réseau.

4. Le temps d'extraction des dimensions du contexte, que ce soit en utilisant des profils locaux ou un respository distant.

5. Le nombre de dimensions adaptées dans un contenu multimédia source.

6. Les changements entre le contenu source et le contenu adapté.

Ces paramètres permettent d'évaluer la qualité de service de la négociation de NAC comme nous allons voir plus tard. Ils permettent aussi d'orienter le choix d'application des différents mécanismes de négociation comme dans le cas de traitement de profils et d'extraction partielle de profil en utilisant l'entité repository.

5.5 La stratégie de négociation

L'objectif des mécanismes de négociation est de trouver un consensus entre l'utilisateur et le serveur concernant le contenu demandé par la requête cliente. L'approche de négociation de l'architecture NAC évite que l'application de tels mécanismes soit de la responsabilité de l'application cliente. Ceci est justifié par le fait que l'application cliente n'a qu'une vision partielle du contexte de l'environnement, et que ces applications clientes sont souvent limitées dans les environnements hétérogènes et, par conséquent, ne peuvent pas appliquer des stratégies de négociation évoluées.

La stratégie de négociation de NAC s'appuie sur l'ensemble des déclarations de profils des différents éléments de l'environnement ainsi que sur l'organisation de répartition de tâches du module d'adaptation et de négociation ANM (voir Chapitre 3). Afin de simplifier la compréhension de la section suivante, nous réutilisons le formalisme du système d'adaptation introduit au début de ce chapitre. Rappelons que nous avons défini le système d'adaptation par le triple $S = (UA\_P, T, D\_P)$. L'image de base du système S, nécessaire pour la négociation, inclut le contenu existant au niveau serveur, le profil du client et le contenu de la requête cliente.

Du point de vue de l'adaptation, il est important de connaître l'ensemble des méthodes d'adaptation disponibles, c'est-à-dire l'ensemble T. Pour la stratégie de négociation, il est important de connaître ce qu'une méthode d'adaptation t peut générer en sortie et quelles sont les conditions d'entrée nécessaires pour que la méthode soit appliquée correctement. Le schéma UPS relatif aux méthodes d'adaptation répond à ce besoin. Dans NAC, l'ensemble T inclut de nombreuses méthodes d'adaptation structurelles et de médias et de combinaison de ces deux types d'adaptation en plus des mécanismes d'adaptation contextuelle (voir Section 3.6.2). La formule $t(x) = y$ signifie que l'application de la méthode d'adaptation t sur le contenu x génère en sortie un contenu ayant le profil y. Cette formule sera utilisée dans la suite de ce chapitre dans la définition de la stratégie de négociation.

La stratégie de négociation de NAC peut être appliquée dans un cadre incluant :

1. Un ensemble de méthodes d'adaptation qui peuvent transformer un contenu source vers une nouvelle forme.

2. Un processeur d'adaptation qui permet à tout moment d'interpréter les règles d'adaptation.

3. Une description en termes de profil du contenu du serveur afin d'assurer une bonne qualité d'adaptation.

4. La description des caractéristiques et des préférences relatives au client.

Un contenu peut être décrit par le couple $<$ $Identificateur\ de\ contenu$, $Identificateur\ de\ profil$ $>$, où $Identificateurdecontenu$ identifie une unique ressource qui existe au niveau serveur[2], et $Identificateurdeprofil$ identifie le profil associé à cette ressource.

[2]un tel identificateur peut être simplement l'URL de la ressource. Une attention particulière doit

Notons que la valeur de l'identificateur de profil peut être la même pour plusieurs ressources de contenu. Ceci est possible quand plusieurs ressources partagent les mêmes caractéristiques.

La stratégie de négociation consiste à effectuer le meilleur effort pour : 1) trouver le contenu demandé par la requête cliente, 2) transmettre ce contenu au client si les contraintes du client sont satisfaites sinon chercher une version du contenu qui peut satisfaire ces contraintes, sinon 3) trouver la méthode d'adaptation qui peut générer un contenu adapté au client.

Le contenu est aussi contraint par les caractéristiques de l'environnement telles que les caractéristiques du réseau utilisé pour la transmission du contenu au client. Effectuer le meilleur effort signifie appliquer le processus optimal qui permet la sélection ou la génération du contenu à transmettre. La stratégie de négociation de NAC est basée sur le principe de suivant :

La négociation de contenu doit fournir en sortie un contenu qui n'inclut pas de fonctionnalités non supportées par le client tel qu'il est décrit dans son profil.

Une fonctionnalité de contenu peut être décrite par n'importe quelle dimension du contexte telle que le support des ressources vidéo, la taille maximale des images acceptées, etc. Nous illustrerons le précédent principe de la négociation par l'exemple suivant : supposons que le client supporte les fonctionnalités {X, Y, Z} qui décrivent les capacités de l'application cliente, et supposons que les ressources de contenu du serveur sont décrites par les fonctionnalités suivantes : $\{Y\}$, $\{X,\ Y\}$ et $\{X,\ Y,\ Z,\ T\}$. Il est clair que l'envoi du contenu décrit par l'ensemble $\{X,\ Y,\ Z,\ T\}$ n'est pas permis car le client ne supporte pas la fonctionnalité T. En outre, l'envoi du contenu décrit par $\{Y\}$ n'est pas préféré car il existe une autre ressource de contenu qui couvre plus de fonctionnalités supportées par le client cible. La stratégie de négociation doit donc prendre la décision de transmettre la ressource décrite par $\{X,\ Y\}$.

L'objectif est de rendre la sélection des ressources automatique et de prendre en compte les descriptions qui correspondent à des ressources pouvant être générées par l'ensemble des méthodes d'adaptation (l'ensemble T).

Nous détaillons dans ce qui suit les principaux champs de requêtes utilisés dans notre approche de négociation de ressource. Le format des requêtes est simplifié afin de faciliter la compréhension de l'approche :

Requête cliente :
<Identificateur du client, Identificateur du serveur, Identificateur de contenu, Fonctionnalités supportées, Fonctionnalités préférées>

Réponse du client :
<Identificateur du client, Identificateur du serveur, Identificateur de contenu, Profil sélectionné>

Requête du serveur :

être apportée au maintien du profil de la ressource dans le cas où le contenu de la ressource change, par exemple, si le contenu de la ressource représente l'état de la météo de la journée courante

<Identificateur du client, Identificateur du serveur, Identificateur de contenu, Ensemble de Profils>

Réponse du serveur :
<Identificateur du client, Identificateur du serveur, Contenu>

Le champ *identificateurduserveur* peut identifier le serveur du contenu comme il peut identifier un proxy. Le champ *Fonctionnalités supportées* indique la partie capacités du profil du terminal. Il peut être transmis sous forme de partie de profil UPS ou sous forme d'un identificateur permettant l'extraction des capacités à partir du profil. Ce champ peut varier pour le même terminal selon l'application cliente utilisée. Le champ *Fonctionnalités préférées* décrit les préférences de l'utilisateur vis-à-vis du contenu demandé. Ce champ augmente l'ensemble des contraintes données par le champ *Fonctionnalités supportées* et aide à définir des priorités afin de faire des restrictions dans le cas où il y a un choix multiple de ressources. Par exemple, l'application cliente peut supporter les animations de formats AVI et GIF mais préfère la réception des animations GIF. La différence majeure entre les deux précédents champs est que la stratégie de négociation peut transmettre un contenu supporté mais non préféré. L'inverse est incorrect.

5.5.1 Abréviations

Dans ce qui suit nous utiliserons les abréviations suivantes :

CI : Identificateur de client
SI : Identificateur de serveur
DI : Identificateur de contenu
P_S : Ensemble de profils, utilisé pour la déclaration et la recherche d'une ressource de contenu
CC_P : Contraintes de profil client, c'est un ensemble de contraintes qui permettent au client de choisir une ressource, lorsque le serveur propose plusieurs choix
SC_P : Contraintes additionnelles du contexte, telles que les caractéristiques du réseau
SP : Profil sélectionné, c'est le profil sélectionné par le client
D : Contenu, c'est le contenu adapté transmis en final
US : Fonctionnalités supportées
UP : Fonctionnalités préférées
USP : Fonctionnalités supportés et préférées
T_S : Ensemble de profils qui peuvent exister après l'application des méthodes d'adaptation de l'ensemble T.

5.5.2 L'évaluation *TL*

Nous définissons, pour la négociation de ressources de contenu, une méthode d'évaluation, liée à une échelle de niveaux de priorités, noté *TL* (Tailored Levels priorities evaluation). Cette méthode est définie pour faciliter la sélection des différents profils de ressources. L'approche consiste à définir un ordre de priorité qui guide la sélection d'un profil dans un ensemble donné. Le module d'évaluation de TL est noté *TL_evaluator*.

L'évaluateur TL accepte en entrée : un ensemble d'éléments I et une collection ordonnée d'ensembles CS. Il donne en sortie un ensemble d'éléments O, où les éléments

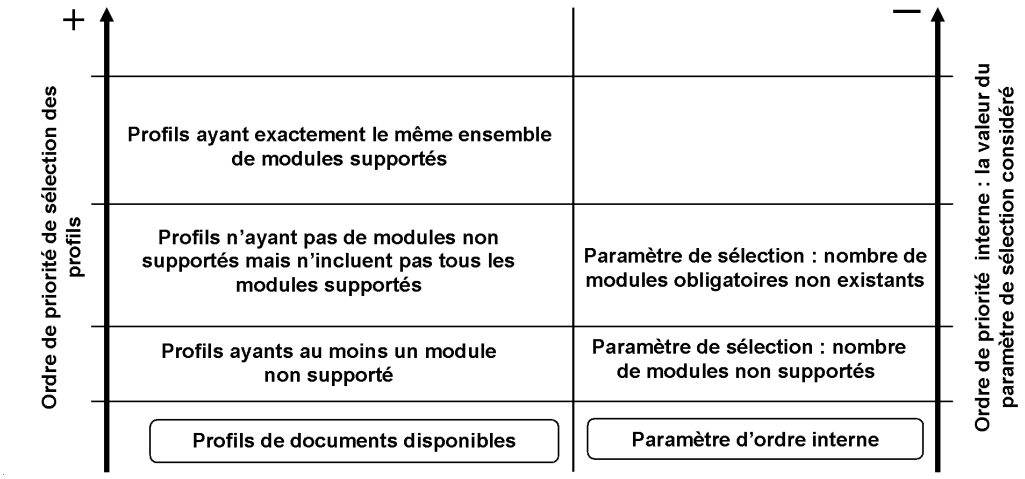

FIG. 5.5 – Les priorités de sélection

sont ordonnés selon un ordre prédéfini.

Nous distinguons deux cas d'utilisation de l'évaluation TL. Afin de comprendre le principe de l'évaluation, nous détaillons ces cas d'utilisation par des exemples.

Premier cas :

Format général : O = TL_evaluator(I,CS).

Exemple d'application : P_S= TL_evaluator(US,D_P).

Dans cet exemple, l'ensemble d'éléments en entrée (I), est donné par l'ensemble US, c'est-à-dire l'ensemble de fonctionnalités supportées. La collection d'ensembles (CS) dans lequel nous devons choisir le meilleur ensemble (ou la meilleure collection d'ensemble) selon l'ordre prédéfini de priorité est donné par D_P qui représente l'ensemble de profils ressource de contenu existants du côté du serveur. L'ensemble de sortie P_S contiendra les profils ressource de contenu recherchés. Les profils sont ordonnés selon l'ordre de priorité donné dans la Figure 5.5.

Les profils sont ordonnés selon un ordre croissant de priorité en trois collections. Dans chaque collection, les profils sont ordonnés selon un autre ordre décroissant de priorités. L'ordre décroissant dépend d'un paramètre associé à chaque collection (voir Figure 5.5). Le rôle de l'évaluateur TL est d'inclure tous les profils ordonnés dans l'ensemble P_S à l'exception des profils du dernier niveau, c'est-à-dire les profils qui incluent au moins un module non supporté selon l'ensemble US. Notons que l'évaluateur peut retourner un ensemble vide.

Considérons l'exemple suivant :

Soit US = $\{Y, T, U\}$ et D_P = {{X,W},{T},{X,Y,Z,T,U},{Y,U},{X,U,T},{T,U,Y},{Y,T, X},{T,Y}}.

Selon l'ordre de priorité, les profils sont ordonnés comme suit :

Première collection (bas priorité) :
$\{X, W\}$: ordre interne = 2, $\{X, Y, Z, T, U\}$: ordre interne = 2, $\{X, U, T\}$: ordre interne = 1, $\{Y, T, X\}$: ordre interne = 1.

Deuxième collection (priorité moyenne) :
$\{T\}$: ordre interne = 2, $\{Y, U\}$: ordre interne = 1, $\{T, Y\}$: ordre interne =1.

Troisième collection (priorité élevée) :
$\{T, U, Y\}$.

Après l'application de l'évaluateur TL sur l'ensemble de profils de ressources contenu, l'ensemble P_S contiendra, les éléments suivants dans cet ordre croissant :
P_S = {{T,U,Y},{Y,U},{T,Y},{T}}, ce qui correspond correctement au principe de la négociation.

Le premier cas d'utilisation de l'évaluateur TL est le même que le cas suivant : P_S= TL_evaluator(US,T_set). Ici, l'ensemble T_set représente tous les profils qui peuvent être obtenus par l'application des méthodes d'adaptation de T.

Deuxième cas :

Format général : O = TL_evaluator(CS,I).

Exemple d'application : P_S = TL_evaluator(P_S,USP).

Dans ce genre de situation, l'évaluateur TL accepte en entrée une collection d'ensemble CS (dans l'exemple c'est l'ensemble P_S) et un ensemble d'éléments I (dans l'exemple c'est l'ensemble USP, c'est-à-dire l'ensemble des fonctionnalités supportées et préférées).

Dans ce genre de cas d'utilisation, le principe est très simple. Il consiste à ordonner les ensembles de CS selon les éléments qui existent dans I. Dans l'exemple donné, les profils de l'ensemble P_S sont ordonnés selon les fonctionnalités supportées et préférées. Dans ce cas, l'ordre de priorité est défini comme suit : *Situation* (1) - Si les préférences UP décrivent uniquement des contraintes qui se rajoutent aux contraintes de l'ensemble US, le paramètre de priorité est égal au nombre de fonctionnalités préférées et l'ordre de priorité est croissant. *Situation* (2) - Si les préférences UP décrivent toutes et seulement toutes les fonctionnalités préférées, l'ordre de priorité est défini en utilisant deux paramètres : a) le nombre de fonctionnalités préférées avec un ordre croissant, et b) le nombre de fonctionnalités non préférées avec un ordre décroissant. Pour combiner ces deux paramètres, le plus simple est de définir l'ordre de priorité comme la valeur du premier paramètre moins la valeur du deuxième paramètre.

Considérons l'exemple précédant discuté dans le premier cas d'utilisation de l'évaluation TL. Considérons un exemple d'ensemble USP tel que USP = $\{Y, T\}$. Dans ce cas, les profils de P_S vont être ordonnés (après l'application de P_S = TL_evaluator(P_S,USP)) comme suit :

Situation (1) :

$\{T, U, Y\}$: paramètre de priorité = 2, $\{T, Y\}$: paramètre de priorité = 2, $\{Y, U\}$: paramètre de priorité = 1, $\{T\}$: paramètre de priorité =1.

Par conséquent P_S = {{T,U,Y},{T,Y},{Y,U},{T}}.

Situation(2) : (paramètre de priorité = nombre de fonctionnalités préférées - nombre de fonctionnalités non préférées)

$\{T, Y\}$: paramètre de priorité = 2-0 =2, $\{T, U, Y\}$: paramètre de priorité = 2-1 =1, $\{Y, U\}$: paramètre de priorité = 1-1 =0, $\{T\}$: paramètre de priorité = 1-0=1,

Par conséquent P_S = {{T,Y},{T,U,Y},{Y,U},{T}}.

Ce cas d'utilisation est le même que l'application de l'évaluateur TL dans les cas suivants P_S = TL_evaluator(P_S,SC_P) et TL_evaluator(P_S,CC_P).

SC_P et CC_P représentent un ensemble de contraintes additionnelles qui aident à guider plus la sélection des ressources. Ces contraintes additionnelles sont optionnelles.

5.5.3 L'échange de requêtes de négociation

L'échange d'informations de négociation, entre un couple de client et un module de négociation de ressources associé à un serveur, est défini par les deux algorithmes présentés dans les Figures 5.6 et 5.7 utilisés côté client et module de négociation respectivement.

Les algorithmes donnés se basent sur le contexte de l'environnement où la requête cliente est transmise pour accéder un contenu du réseau. Cependant, les algorithmes ne font aucune restriction sur le format de déclaration du contexte. L'ensemble des contraintes peut être extrait de n'importe quelle source de description. Dans NAC, le format de base des descriptions est le modèle UPS et l'ensemble des techniques d'évaluation dynamiques des dimensions contextuelles.

D'un point de vue sélection et négociation de ressources, les algorithmes sont flexibles et l'ordre de priorité adopté dans l'évaluation TL peut être modifié afin de favoriser un critère particulier de sélection. L'algorithme de négociation de ressources du côté module de négociation, considère l'augmentation de l'ensemble des profils de contenu, par les nouveaux profils créés ainsi que la sauvegarde des nouvelles variantes créées. Cette considération permet de réutiliser le contenu adapté, ce qui minimise le temps de réponse des prochaines requêtes qui peuvent provenir du même terminal ou d'autres nouveaux terminaux de l'environnement hétérogène.

Comme nous avons vu, les algorithmes adoptés ne posent aucune restriction sur l'organisation de l'architecture. En effet, le client peut envoyer sa requête au serveur d'origine comme il peut l'envoyer à un proxy intermédiaire. Dans le cas de l'utilisation du module de négociation au niveau d'un proxy, il suffit juste que les différents termi-

naux pointent leurs connexions vers le proxy afin d'utiliser le service d'adaptation de contenu.

5.5.4 Exemple de scénario d'exécution

Dans cette section nous présentons un exemple de scénario d'exécution des algorithmes de négociation de ressources présentés dans la section précédente. Nous considérons uniquement le côté interaction entre le client et le module de négociation de ressource sans détailler l'évaluation TL utilisée. Les différents cas d'utilisation de l'évaluation TL sont couverts par la section précédente. Pour simplifier, nous considérons que l'environnement hétérogène se limite à un serveur et une application cliente. Le principe reste le même dans les cas où plusieurs serveurs et applications clientes existent.

Soit {X, Y, Z, T, U, V, W} un ensemble de fonctionnalités atomiques et considérons :
- Un client avec :

Profil client = {*User Agent Supported Functionalities* + *User Agent Preferred Functionalities*}. Tel que :

User Agent Supported Functionalities = {Y,T,U}, *User Agent Preferred Functionalities* = {Y,T}.

- Le contenu du serveur est décrit par :

Ensemble de profil contenu (D_P) = {{X,W},{X,Y,Z,T,U},{U},{X,V,T},{W,T,X},{Y, U},{Y,T,U,W}}.

Les méthodes d'adaptation peuvent donner les profils suivants :
{{X,W},{T,Y},{X,Y,Z,T,U}}. Avec $t(ID = Identificateur\ de\ contenu)$={T,Y}, ce qui signifie que lorsque le système d'adaptation applique t (t appartient à l'ensemble T) sur le contenu identifié par $Identificateurdecontenu$, le contenu transformé aura comme profil {T,U,Y}. Nous supposons que SC_P et CC_P sont vides, c'est-à-dire qu'on considère pas de contraintes additionnelles de l'environnement autre que les contraintes du contenu et du client.

La négociation des ressources de contenu est initiée lorsque le client (qui peut être un terminal de capacités limitées) envoie une requête demandant un contenu du réseau. Le diagramme présenté par la figure 5.8 représente le scénario d'exécution qui correspond à notre exemple.

Les points représentés sur le diagramme de la figure 5.8 correspondent à l'exécution suivante :

C1 :
Préparation des différents champs de la requête du client
C2 :
Envoi de la requête du client au module de négociation qui existe du côté du serveur
S1 :

```
.
.
User Agent Supported Functionalities
= Determine_Actual_User_Agent_Supported_Functionalities() ;
//Permet de déterminer les fonctionnalités supportées de l'application cliente,
//retourne un identificateur, une structure UPS,
//un pointeur, ou autres. Le contexte peut être déjà sauvegardé,
//calculé, etc.
User Agent Preferred Functionalities
= Determine_Actual_User_Agent_Preferred_Functionalities() ;
//Déterminer les préférences de l'application cliente. Cela peut changer
//d'une requête à une autre et peut définir un ordre de priorité
//des préférences.
Server Identifier
= Determine_Document_Server(Document Identifier) ;
//Détermine l'identificateur du serveur du contenu recherché : un serveur d'origine
//un proxy de cache, ou autre.
Client_Request
=
(<Client Identifier, Server Identifier, Document Identifier, User Agent
Supported Functionalities, User Agent Preferred Functionalities>) ;

send (Client_Request) to Server Identifier ;
.
.
repeat forever
{

when receiveServer_Request from Server Identifier do
{
(CI,SI,DI,P_S) = Server_Request ;
SP = TL_evaluator(P_S,CC_P) ;
// Sélectionner un profil parmi les profils proposés
//par le serveur. La sélection peut faire intervenir d'autres
//contraintes décrites dans (CC_P) en plus de
//l'ordre définit par SP.
// CC_P peut être vide et donc SP ne sera pas modifié
send (CI,SI,DI,SP) to SI ;
}//end when

when receiveServer_Reply from Server Identifier do
{
(CI,SI,D) = Server_Request ;
if D <> ∅ then use D ;
}//end when

}//end repeat
.
.
```

FIG. 5.6 – Algorithme de la négociation des ressources : côté client

```
.
.
repeat forever
{
when receiveClient_Request from Client Identifier do
{(CI,SI,DI,US,UP)= Client_Request ;
USP = UP-(UP-US) ; //determine the supported AND preferred modules
P_S = TL_evaluator(US,D_P) ; {use case 1}
//chercher les profils du serveur en utilisant
//l'évaluation TL basée sur les fonctionnalités supportées ;
//le résultat est sauvegardé dans l'ensemble de profils : P_S.
P_S= P_S ⋃ TL_evaluator(US,T_set) ; //{use case 1}
//chercher les profils qui peuvent exister après
//l'application des méthodes d'adaptation qui peuvent satisfaire le profil
//de l'application cliente(US).
if (P_S<>∅) then
P_S = TL_evaluator(P_S,USP) ; //{use case 2}
// appliquer une sélection TL sur la base des préférences du client
P_S = TL_evaluator(P_S,SC_P) ; //{use case 2}
//SC_P contient des contraintes additionnelles en plus des contraintes du client,
//par exemple, le serveur peut détecter que la bande passante du réseau devient
//très faible et par conséquent le serveur doit éviter de transmettre des flux
//vidéos avec un débit élevé d'images. SC_P peut être vide et par conséquent P_S
//ne sera pas modifié.
Server_Request = <CI,SI,DI,P_S> ;
send (Server_Request) to CI
else
Server_Reply = <CI,SI,∅> ;
//Les profils disponibles ne peuvent pas satisfaire le profil cible.
//Par exemple, tous les profils disponibles peuvent contenir certaines
//fonctionnalités non supportées ; et le serveur ne peut pas adapter le contenu.
//Cette situation peut être évitée en enrichissant le système d'adaptation par
//d'autres méthodes d'adaptation.
send (Server_Reply) to CI ;
fi ; }//end when
when receive Client_Reply from Client Identifier do
{(CI,SI,DI,SP)=Client_Request ;
if SP exists_in D_P then send(CI,SI,Document(SP)) to CI
//send the document having as profile SP
else t = find_transformation(SP) ;
//l'existence de t est assurée car une réponse cliente est reçue
D = t(DI) ; //appliquer la méthode d'adaptation correspondante ;
D_P = D_P ⋃ SP ; //sauvegarder le profil du contenu ;
save(D) ; //sauvegarder la nouvelle version du contenu ;
send(CI,SI,D) to CI ; //envoyer le contenu adapté
fi ;}//end when
}//end repeat
.
.
```

FIG. 5.7 – Algorithme de la négociation des ressources : côté module de négociation

Le module de négociation reçoit la requête du client qui demande le contenu identifié par *Identificateur de contenu*
S2 :
Le module de négociation détermine l'ensemble P_S qui contient les profils qui peuvent être utilisé par le client. Le module cherche d'abord les profils correspondants au contenu et ses versions en appliquant : P_S = TL_evaluator(US,D_P), ce qui signifie trouver dans D_P les profils qui correspond à US. L'ensemble P_S trouvé contient {{Y,U},{U}}. Comme nous avons déjà vu, l'évaluation TL donne un ensemble de profils ordonnés selon leur degré d'adaptation pour le contexte de l'utilisateur. Afin d'assurer une meilleure adaptation, le module cherche les profils qui peuvent être générés par l'application des méthodes d'adaptation. Les profils trouvés sont ajoutés à l'ensemble de profil P_S. Cela est fait par l'exécution de : P_S= P_S $\bigcup$ TL_evaluator(US,T_set). L'ensemble P_S devient donc {{Y,U},{U},{T,Y}}. Le module de négociation applique par la suite un ordonnancement de profils selon les préférences de l'utilisateur : P_S = TL_evaluator(P_S,USP). L'ensemble P_S est l'ensemble final des profils préparés puisque il n'y a pas d'autres contraintes additionnelles (P_S n'est pas modifié par P_S = TL_evaluator(P_S,SC_P)).
S3 :
La requête du serveur est préparée avec les profils calculés. La requête est envoyée au client identifié par *Identificateur de client*
C3 :
Le client reçoit la requête du module de négociation
C4 :
Le client choisit le profil avec la priorité la plus élevée (puisque il n'y a pas de contraintes clientes additionnelles car CC_P est vide). Le profil sélectionnée est donc SP = {T,Y}
C5 :
Le message *réponse de client* est préparé et ensuite envoyé au serveur identifié par *Identificateur de serveur* avec {T,Y} comme profil sélectionné.
S4 :
Le module de négociation reçoit le message *réponse de client*
S5 :
Le module de négociation cherche le profil reçu dans son ensemble de profils. Puisque {T,Y} n'existe pas dans l'ensemble D_P, le module cherche la méthode d'adaptation correspondante qui peut générer un tel profil. La méthode d'adaptation trouvée est *t*. La méthode *t* est appliquée sur le contenu identifié par *Identificateur de contenu*. Le profil {T,Y} et le contenu adapté sont sauvegardés
S6 :
Le message *réponse du serveur* est préparé et ensuite envoyé au client
C6 :
Le client reçoit le message *réponse du serveur*
C7 :
Le client vérifie que champ du contenu n'est pas vide et utilise le contenu adapté décrit par le profil {T,Y}

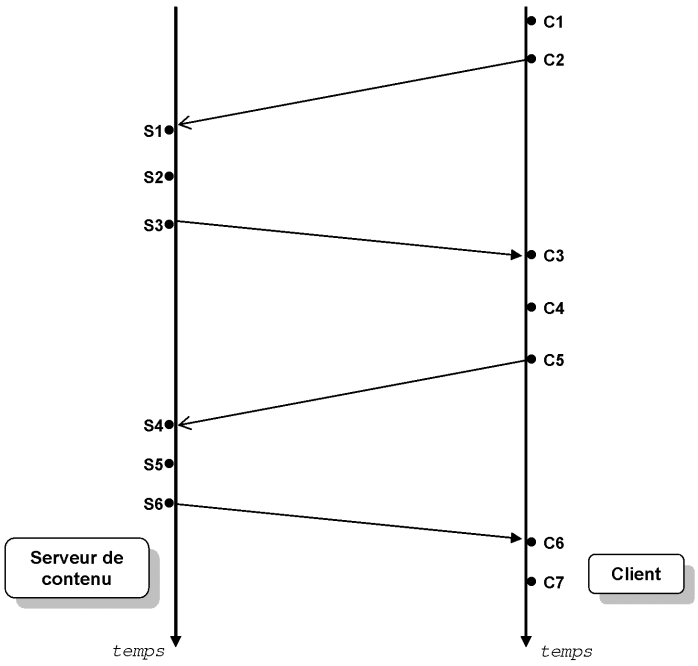

FIG. 5.8 – Un exemple de scénario

5.6 Le protocole de négociation pour l'acquisition du contexte client

L'ensemble des terminaux dans un environnement hétérogène subit souvent des changements de contexte dus à la localisation de terminal mobile, les préférences de l'utilisateur, l'utilisation de différentes applications clientes, etc. Afin de prendre en charge ces changements, l'architecture d'adaptation et de négociation NAC assure la définition et l'implémentation d'un protocole de négociation pour l'acquisition du contexte client. Le protocole représente un choix alternatif aux mises à jour du contexte effectué en passant le repository de profils en utilisant les méthodes SOAP (voir Section 3.4).

Comme nous avons vu déjà, les protocoles de communication, utilisés pour la transmission du contenu aux terminaux, sont généralement des protocoles sans états (*stateless protocols*). Ces protocoles sont appliqués entre le navigateur du client et serveur du contenu. Toutes les requêtes envoyées par l'application cliente suivent donc le protocole de communication. Le principe d'un protocole de communication est simple. Il suit le scénario général décrit par une requête et sa réponse. Dans le cas général, il n'y a pas de besoin pour garder des informations sur le client ou sur sa session courante. En effet, une fois la réponse du serveur envoyée (le contenu, un message d'erreur, un message de non modification, etc.) la communication entre le serveur et le client est fermée.

L'objectif du protocole de négociation pour l'acquisition du contexte est différent de celui du protocole de communication. Cependant, les tâches des deux protocoles sont complémentaires. Le protocole d'acquisition assure l'échange d'information de

négociation entre le client et le module d'adaptation et de négociation afin de permettre une meilleure adaptation du contenu du serveur. NAC adopte une approche qui sépare les deux protocoles de négociation et de communication afin de rendre le système indépendant d'un navigateur particulier et de couvrir une large diversité de clients.

Le protocole de négociation pour l'acquisition du contexte de NAC est conçu afin d'optimiser les ressources du client, du serveur et du réseau de communication utilisé. C'est pour cela que l'échange de messages de négociation est minimisé et ne véhicule que l'information utile. Le protocole prend en compte les changements de contexte et assure l'extraction des éventuels changements. Afin de garder le contexte d'un terminal et maintenir les différents contextes des différents terminaux d'une manière simultanée, le protocole applique une stratégie de cache temporaire basée sur l'adresse IP des terminaux.

Le protocole définit cinq types de messages :

1- GET_GLOBAL_PROFILE : Ce type de message est utilisé par le module ANM afin de demander le contexte courant du client.

2- OK_SENDING_PROFILE : Ce type de message est utilisé par le module UCM lors de l'envoi de son contexte.

3- OK_SENDING_CHANGE : Ce type de message est utilisé par le module UCM lors de l'envoi de son contexte dans le cas où ce dernier subit un changement depuis le dernier envoi de contexte.

4- NO_PROFILES_CHANGE : Ce type de message est utilisé par le module UCM si le contexte du client n'a subit aucun changement depuis le dernier envoi de contexte.

5- NO_PROFILE_ACQUISITION : Ce type de message est utilisé par l'UCM s'il est impossible d'extraire le contexte du client (exemple dans le cas d'erreur au niveau du client ou dans le cas d'absence de méthodes d'extraction de contexte).

Le protocole d'acquisition de NAC est appliqué entre le module de contexte client UCM et le module d'adaptation et de négociation ANM (voir Chapitre 3). Comme nous avons déjà vu, le module UCM a été développé pour les systèmes embarqués. Le module peut être utilisé sous une plate-forme classique grâce à une version d'émulation.

Les messages du protocole d'acquisition peuvent être échangés à n'importe quel moment après la configuration du module UCM (voir Figure 5.9). Le port de négociation utilisé par défaut est égal à 1977. Comme dans le protocole de communication, le module d'adaptation et de négociation doit utiliser le même port de négociation que le module UCM (voir Figure 5.10).

Après la configuration des ports et de l'adresse du module de négociation, le module ANM peut à tout moment vérifier le contexte des différents clients et cela grâce au principe de multi-threading adopté par NAC. Le contexte d'un client est décrit selon le modèle UPS et sauvegardé sous un format XML sur le terminal. Le changement de contexte est matérialisé par le changement que le profil UPS subit. A l'aide du mo-

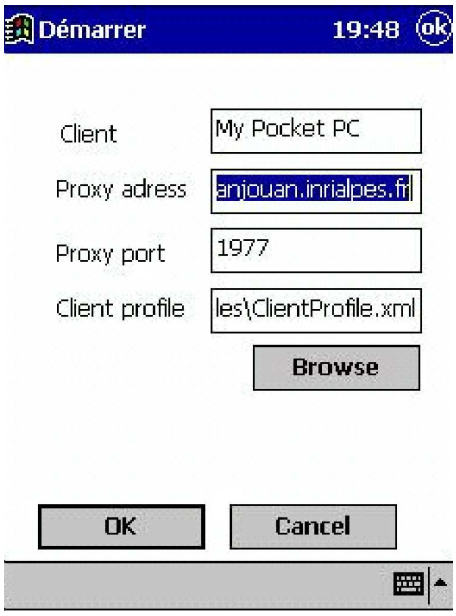

FIG. 5.9 – La configuration de négociation du module UCM

FIG. 5.10 – La configuration de négociation du module ANM

dule UCM, le client peut à n'importe quel moment changer son profil (voir Figure 3.10).

Pour une question d'optimisation, les changements locaux de profils ne sont pas transmis au module ANM quand cela n'est pas explicite. C'est uniquement lorsque le client envoie une requête vers un contenu du réseau, que le module ANM vérifie le changement de contexte à l'aide de la requête 'GET_GLOBAL_PROFILE'. Dans le cas où le contexte subit un changement, le module UCM répond par un message 'OK_SENDING_CHANGE' avec le nouveau contexte, sinon le client répond par un message 'NO_PROFILES_CHANGE'. Une fois qu'un changement de contexte est détecté par le module ANM, le module applique un nouveau processus de négociation sur le contenu demandé.

5.7 Conclusion

Dans ce chapitre nous avons présenté l'aspect processus d'adaptation et protocole de négociation de l'architecture NAC. Nous avons présenté une modélisation du système d'adaptation ainsi que différents éléments qui interviennent dans l'application de la stratégie et le protocole de négociation. Nous avons montré que la négociation du contenu dépend des capacités d'adaptation du système mais aussi de la bonne analyse et exploitation des descriptions du contexte de l'environnement. L'aspect acquisition du contexte et du changement de contexte a été présenté aussi. Cet aspect fondé sur l'utilisation d'un protocole entre le client est le module d'adaptation, permet d'assurer une adaptation qui reflète l'état courant du système tout en minimisant l'échange de

messages de négociation.

Comme nous avons vu, la stratégie de négociation repose sur l'ensemble des méthodes d'adaptation disponibles (l'ensemble T). Il est donc nécessaire de développer une architecture riche en adaptation afin que la négociation appliquée donne de meilleurs résultats. Dans le chapitre suivant nous discutons quelques méthodes et techniques d'adaptation développées dans l'architecture NAC. Le chapitre inclut aussi une évaluation détaillée de certains aspects de la négociation et de l'adaptation du contenu adoptés par NAC.

Chapitre 6

Techniques et méthodes d'adaptation du contenu dans NAC

Résumé

Ce chapitre présente en détail différentes techniques d'adaptation implémentées dans l'architecture NAC. Le système d'adaptation considère la transformation structurelle des documents ainsi que l'adaptation des différentes ressources médias. Afin de prendre en compte l'aspect du changement du contexte deux approches d'adaptation dynamiques sont détaillées. Le chapitre inclut une formulation de l'adaptation des ressources ainsi que des évaluations détaillées des techniques d'adaptation et de la gestion des profils.

Contenu

6.1 Introduction

Le système de négociation de contenu peut être amené à lancer une adaptation de contenu afin de satisfaire les contraintes du client cible. Le degré de satisfaction des contraintes du contexte dépend profondément des capacités du système en termes de techniques et de méthodes d'adaptation. Le résultat d'une stratégie de négociation ne peut être donc efficace que si le système considère plusieurs techniques d'adaptation.

Les besoins relatifs à la transformation des structures est apparu avec l'arrivé des documents structurés. Le principe de la transformation est de prendre un document

écrit dans un modèle donné, et de le réécrire dans un autre modèle. Le document en entrée de la transformation est appelé *document source*. Le document en sortie est appelé *document cible*. Ce processus nécessite de spécifier la manière dont le document doit être transformé. Cette spécification peut être exécutée ou interprétée. Selon le besoin de la transformation, la spécification peut s'appuyer sur le modèle du document source (s'il existe) ou sur l'instance source.

Les nombreuses recherches effectuées dans le domaine de la transformation structurelle (voir Section 2.5.1) ont conduit à la définition de plusieurs langages de transformation de documents [105]. Les applications clientes des différents terminaux de l'environnement hétérogène considèrent le contenu comme un tout : structure et média. La présentation du contenu s'appuie donc sur la structure et sur les différentes ressources médias utilisés. Les dernières années ont connu une explosion d'utilisation des documents multimédias. Les documents sont devenus de plus en plus riches en ressources multimédias tels que les vidéos, les images de haute résolution, etc. L'adaptation du contenu doit donc faire face à cette complexité de contenu en combinant les différentes techniques de transformation.

Nous avons vu dans le chapitre précédent l'importance de l'ensemble T (l'ensemble des méthodes d'adaptation) dans la stratégie de négociation. Ce chapitre présente en détail différentes techniques d'adaptation implémentées dans l'architecture NAC. Comme les environnements hétérogènes sont caractérisés par le changement dynamique et la diversité des capacités des terminaux, il est important de faire évoluer l'adaptation classique, qui consiste à appliquer des transformations figées sur des instances de contenu, vers une adaptation plus flexible et plus dynamique. Comme nous allons voir, l'adaptation dynamique, permet de considérer plusieurs dimensions du contexte et peut être ré-appliquée dans différentes situations. Le chapitre inclut une formulation de l'adaptation des différentes ressources afin de contrôler l'efficacité des processus d'adaptation appliqués dans l'architecture. Il inclut aussi des évaluations détaillées concernant les techniques d'adaptation utilisées ainsi que la gestion des profils.

Ce chapitre est organisé en cinq parties. La première est dédiée au processus d'adaptation de contenu statique et aux deux approches d'adaptation dynamique. La deuxième partie présente l'adaptation des ressources média et l'évaluation de l'adaptation par rapport au temps de transmission du contenu final. La troisième et la quatrième partie discutent l'adaptation des scénarios multimédia pour les terminaux de capacités limitées ainsi que la considération des dimensions dynamiques des capacités. La dernière partie présente une évaluation sur la gestion des profils UPS et l'utilisation du repository de NAC.

6.2 Le processus d'adaptation de contenu

Beaucoup de travaux récents ont proposé des méthodes pour associer un contenu multimédia à plusieurs terminaux. Une des approches les plus utilisées consiste à extraire le contenu d'un système de base de documents stockés dans un format XML [33], et le convertir vers le langage cible en utilisant le langage de transformation XSLT [135].

Avant d'appliquer un processus d'adaptation sur le contenu, le système de négociation doit déterminer le couple (*contexte initial*, *contexte final*) et la méthode d'adaptation qui correspond à ce couple. Le contexte initial englobe les conditions dans

lesquelles une méthode d'adaptation peut être appliquée. Le contexte final représente les différentes caractéristiques du contenu que l'on veut générer par adaptation. L'architecture d'adaptation et de négociation NAC distingue deux types d'adaptation selon le support des variations des différentes dimensions de contexte. Ces deux types sont l'adaptation statique et l'adaptation dynamique.

L'adaptation dynamique concerne les mécanismes d'adaptation paramétrés et les mécanismes d'adaptation pendant l'exécution. Les mécanismes d'adaptation paramétrés sont instanciées selon les valeurs des dimensions du contexte courant. Ce genre d'adaptation permet de couvrir plusieurs instances de contexte où les dimensions peuvent changer d'une instance à l'autre. Les mécanismes d'adaptation dynamiques pendant l'exécution, permettent d'assurer une adaptation continue pendant la présentation du contenu par l'application cliente. Ce genre d'adaptation, permet de considérer, en temps réel, les changements du contexte.

6.2.1 L'adaptation statique

L'adaptation statique de l'architecture NAC consiste à préparer une collection riche de méthodes pour les différents contextes. Chaque méthode considère un unique couple (*contexte initial*, *contexte final*) et ne peut pas être re-appliquée si le contexte change de dimensions. Lorsque il y a une correspondance entre les caractéristiques du contexte courant de l'environnement et les caractéristiques du couple de contexte de l'adaptation, la méthode est appliquée et le résultat est utilisé dans transmission finale du contenu adapté.

Les méthodes d'adaptation peuvent être spécifiées en utilisant un langage de programmation classique, comme elles peuvent être spécifiées en utilisant des langages spécifiques tels que les langages dédiés à la transformation de documents. Dans la thèse de Stéphane Bonhomme [105], on trouve un état de l'art complet de ces langages. De façon générale, les méthodes basées sur l'utilisation des langages de programmation concernent les adaptations appliquées sur les ressources média. NAC inclut de nombreux types de transformation de médias telle que la compression des images et des vidéos, la conversion de types des médias, le retaillage des ressources, etc. Les langages de transformation de documents sont utilisés pour concevoir de méthodes d'adaptation qui s'appliquent sur la structure des documents. Beaucoup de méthodes de transformations structurelles ont été développée dans NAC, telles que la transformation XHTML vers WML, XML vers SVG, MathML vers SVG, filtrage des documents, etc.

XSLT [135] est parmi les langages de transformation les plus utilisés pour assurer la transformation de documents. Comme nous avons vu dans le Chapitre 2, XSLT est le standard W3C défini pour transformer la structure et, dans une moindre mesure, le contenu des documents XML. Une transformation est spécifiée par un ensemble de règles associées à un sélecteur. Une règle de transformation permet de définir un fragment de document résultat de la transformation pour un ensemble de nœuds source identifiés à l'aide d'un sélecteur. Afin de contrôler la transformation, chaque règle inclut une liste d'instructions qui jouent un certain rôle dans la génération du document cible. Afin de développer des méthodes d'adaptation structurelles écrites en XSLT, il est donc nécessaire de définir les règles de correspondance entre le document source, demandé par la requête cliente, et la nouvelle forme du document qui peut

être utilisée directement par le client cible. Un exemple va nous permettre d'illustrer l'utilisation de XSLT dans l'adaptation des documents.

La partie de document qui suit, représente une partie d'un article écrit en XML suivant une DTD (voir Annexe B) que nous avons définie pour la transformation statique *XML vers LaTeX*.

```
<?xml version="1.0"?>
<LaTeXDocument Type="article">
   <Title>The Negotiation of Multimedia Content Services in
   Heterogeneous Environments</Title>
   <Author>
      <Name>Tayeb Lemlouma</Name>
      <Name>Nabil Layaida</Name>
   </Author>
   <Date>April 2001</Date>
   <Heading>
      <Organisation>
         <Name>OPERA Project, INRIA Rhone-Alpes</Name>
         <Adress>ZIRST-655 Avenue de l'Europe -38330
         Montbonnot Saint Martin France</Adress>
      </Organisation>
      <Phone>
         <Number>+33 4 76 61 52 81</Number>
         <Number>+33 4 76 61 53 84</Number>
      </Phone>
      <Fax>
         <Number>+33 4 76 61 52 07</Number>
      </Fax>
      <EMail>
         <Mail>Tayeb.Lemlouma@inrialpes.fr</Mail>
         <Mail>Nabil.Layaida@inrialpes.fr</Mail>
      </EMail>
   </Heading>
   ...
</LaTeXDocument>
```

Le document contient le titre de l'article, la liste des auteurs, la date et l'en-tête. Le contenu de l'élément *titre* du document source est utilisé pour la génération du document en sortie de la manière suivante :

```
<xsl:text>\title{\bf </xsl:text>
<xsl:apply-templates select="Title/text()"/>
<xsl:text>}</xsl:text>
```

La transformation XSLT définit donc une correspondance entre l'élément XML $<Title>$ du document source et la partie à générer dans le document cible qui est le texte :

```
\titre {titre de l'article source}
```

Les règles XSLT qui permettent d'appliquer le traitement de transformation sur la liste des auteurs, la date et l'en-tête sont définies de la manière suivante :

```
<xsl:apply-templates select="Author"/>
<xsl:apply-templates select="Date"/>
<xsl:apply-templates select="Heading"/>
```

L'instanciation se fait lorsque le nœud source en train d'être traité correspond au nœud : *Author*, *Date* ou *Heading* respectivement. L'identification des nœuds source est basée sur l'utilisation du langage XPath [133], par exemple l'identification du nœud racine en XPath est représenté par le caractère '/'. Les corps des règles sont donnés par :

```
<xsl:template match="Author">
   <xsl:for-each select="Name">
     <xsl:if test="position()=1">
     <xsl:text>
     \author{</xsl:text></xsl:if>
     <xsl:if test="not(position()=1)">
       <xsl:text> and </xsl:text>
     </xsl:if>
     <xsl:apply-templates select="text()"/>
   </xsl:for-each>
<xsl:text>}</xsl:text>
</xsl:template>
```

pour le traitement de la liste des auteurs, et par :

```
<xsl:template match="Date">
<xsl:text>
\date{</xsl:text>
<xsl:apply-templates select="text()"/>
<xsl:text>}</xsl:text>
</xsl:template>
```

pour le traitement de la date, et enfin par :

```
<xsl:template match="Heading">
<xsl:text>\begin{center}</xsl:text>
<xsl:apply-templates select="Organisation/Name/text()"/>
<xsl:text> \\ </xsl:text>
<xsl:apply-templates select="Organisation/Adress/text()"/>
<xsl:text> \\ </xsl:text>
<xsl:for-each select="Phone/Number">
<xsl:if test="position()=1"><xsl:text>
Phone: </xsl:text></xsl:if>
<xsl:if test="not(position()=1)"><xsl:text> / </xsl:text></xsl:if>
<xsl:value-of select="."/>
</xsl:for-each><xsl:text> \\ </xsl:text>
<xsl:for-each select="Fax/Number">
<xsl:if test="position()=1"><xsl:text>
Fax: </xsl:text></xsl:if>
<xsl:if test="not(position()=1)"><xsl:text> / </xsl:text></xsl:if>
<xsl:value-of select="."/>
</xsl:for-each><xsl:text> \\ </xsl:text>
<xsl:for-each select="EMail/Mail">
<xsl:if test="position()=1"><xsl:text>
E-mail: </xsl:text></xsl:if>
<xsl:if test="not(position()=1)"><xsl:text>, </xsl:text></xsl:if>
<xsl:apply-templates select="text()"/>
</xsl:for-each><xsl:text> \\ </xsl:text>
<xsl:text>
\end{center}</xsl:text>
</xsl:template>
```

pour le traitement de l'entête. Comme nous pouvons remarquer, le corps des règles est constitué de deux catégories d'éléments : les éléments XSLT, tel que l'élément *xsl :if*, et les éléments ou fragments appartenant au modèle du document cible, tel que l'élément *'Phone'*. L'application de ces règles sur la partie précédente du document XML génère la partie LaTeXsuivante :

```
\title{\bf The Negotiation of Multimedia Content Services in
Heterogeneous Environments}
\author{Tayeb Lemlouma and Nabil Laya\"{\i}da}
\date{April 2001}
\begin{center}OPERA Project, INRIA Rh\^{o}ne-Alpes \\
ZIRST-655 Avenue de l'Europe -38330 Montbonnot Saint Martin France \\
Phone: +33 4 76 61 52 81 / +33 4 76 61 53 84 \\
Fax: +33 4 76 61 52 07 \\
E-mail: Tayeb.Lemlouma@inrialpes.fr, Nabil.Layaida@inrialpes.fr \\
\end{center}
```

En suivant une approche statique d'adaptation, la variation des formats et des caractéristiques du contenu ainsi que la variation des dimensions du contexte de l'environnement hétérogène nécessitent la définition d'une grande variété de méthodes d'adaptation pour chaque couple (*contexte initial*, *contexte final*). Cette variété peut être explosive dans le cas où une dimension de contexte appartient à un intervalle de valeurs (par exemple, les dimensions d'une image, la taille de l'écran d'un terminal, etc.). Ce fait rend nécessaire le développement d'une approche d'adaptation plus dynamique qui s'adapte à la variation des dimensions du contexte et qui offre la possibilité d'appliquer les méthodes d'adaptation dans différents contextes.

6.2.2 L'adaptation dynamique paramétrée

Dans l'architecture NAC, le langage de transformation XSLT est souvent utilisé pour assurer la transformation structurelle des documents. Il est donc nécessaire de lier les feuilles de transformations développées aux différents profils des clients cibles pour assurer une adaptation dynamique de contenu. Afin d'éviter le développement d'une multitude de feuilles de transformation pour chaque profil client et chaque type de document, une des approches possibles et d'effectuer une concaténation du document source avec les contraintes données par le profil du client cible (voir Figure 6.1), et d'appliquer ensuite une feuille de transformation sur la structure résultante. La concaténation permet aux règles de la feuille de transformation d'effectuer des traitements et des sélections qui dépendent du contenu du profil cible. Dans la pratique, le développement de telles feuilles de transformation est très compliqué. Cela est dû au fait que les deux parties (contraintes de profil et document source) du document résultant sont indépendantes ce qui nécessite un traitement coûteux et lourd dans l'utilisation de la partie contraintes pour la transformation du document cible.

Une meilleure approche adoptée dans NAC consiste à définir une feuille de transformation générique qui peut s'appliquer sur un document de type profil et qui génère en sortie une nouvelle feuille de transformation (voir Figure 6.2). La génération est faite d'une manière automatique sur le profil courant, et la transformation générée est appliquée sur l'ensemble des documents demandés par la requête cliente. Ce processus permet la considération des contraintes du client cible qui sont déclarées dans son profil.

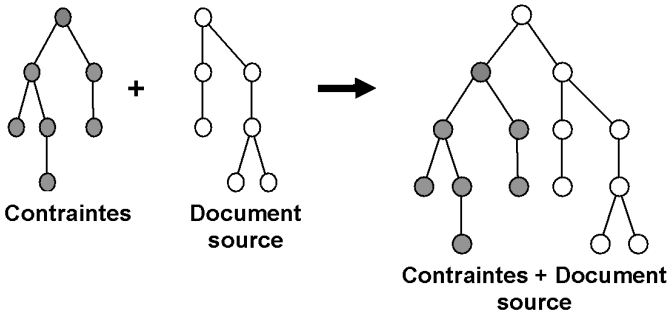

FIG. 6.1 – Concaténation des contraintes avec le document source

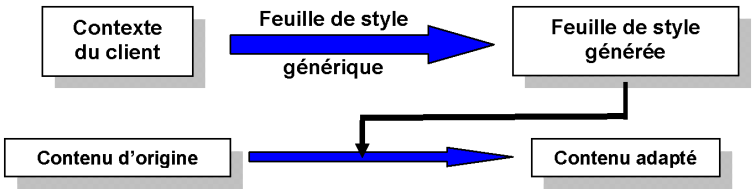

FIG. 6.2 – Génération automatique d'adaptation

Dans cette approche, le temps d'adaptation de n instances de documents pour un profil donné, est égal au temps de génération de la feuille de transformation plus le temps de l'adaptation des n instances. Ce temps d'adaptation est très réduit comparé avec le temps d'adaptation nécessaire dans la première approche de concaténation de contraintes. Dans la première approche, le temps d'adaptation est égal au temps de concaténation du profil avec le document source pour les n instances plus le temps nécessaire pour l'adaptation des n documents résultants. Le temps d'adaptation T est donc évalué à :

$$T = n.time(Concat(profil, document)) + n.time(Adaptation(document'))$$

Où $Concat$ représente le processus de concaténation et de création du profil avec le document source et $document'$ est le document généré après concaténation. Dans la deuxième approche, le temps d'adaptation est évalué à :

$$T = 1.time(Gen\_Transf(profil, feuille\_gen)) + n.time(Adaptation(document))$$

Où Gen_Transf représente le processus de génération de transformation en utilisant la transformation générique *feuille_gen*. Cette génération est faite une seule fois pour le même profil.

Nous illustrons l'application de cette approche, par l'exemple suivant. L'exemple représente un filtrage dynamique de documents de type SMIL. La Figure 6.3 donne un exemple simple de profil client qui indique que le client ne supporte pas les ressources médias de type *audio* ni les exécutions parallèles dans les scénarios SMIL. La feuille de transformation générique utilisée pour le filtrage dynamique est donnée dans l'Annexe

```
<ClientProfile>

<Service category="smil" name="Synchronized Multimedia Integration
Language">
  <SoftwarUsed>RealPlayer 8 Basic 6.0</SoftwarUsed>
  <ServiceComponent category="video" tagName="video" support="yes">
    <SourceType>
      <NotSupportedBag>
       <li>mpeg</li>
      </NotSupportedBag>
    <OnlySupportedBag>
    </OnlySupportedBag>
    </SourceType>
</ServiceComponent>

<ServiceComponent category="image" tagName="img" support="yes">
   <MaxDisplay>100x200</MaxDisplay>
   <MaxSize>2000K</MaxSize>
</ServiceComponent>

< !- - support=no, l'élément audio et son contenu ne sont pas supportés - ->

<ServiceComponent category="audio" tagName="audio" support="no"/>

< !- - support=noTag, seulement l'élément n'est pas supporté - ->

<ServiceComponent category="parallel execution" tagName="par"
support="noTag"/>
</Service>
</ClientProfile>
```

FIG. 6.3 – Exemple de profil déclaré

```
<?xml version="1.0" encoding="UTF-8"?>

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/
XSL/Transform" version="1.0">
<xsl:output omit-xml-declaration="yes"/>
<xsl:template match="*">
<xsl:copy>
<xsl:apply-templates select="@*"/>
<xsl:apply-templates/>
</xsl:copy>
</xsl:template>
<xsl:template match="@*">
<xsl:copy/>
</xsl:template>

<xsl:template match="audio">
</xsl:template>
<xsl:template match="par">
<xsl:apply-templates/>
</xsl:template>

</xsl:stylesheet>
```

FIG. 6.4 – Feuille de transformation générée

B (voir Section B.3).

L'application de la feuille de la transformation générique permet d'effectuer une sélection et instanciation de règles de transformation nécessaires selon le contenu du profil. L'application de la feuille générique sur le profil client de notre exemple, génère la feuille de transformation présentée par la Figure 6.5. La transformation générée permet de transformer les documents SMIL selon les contraintes du client, déclarées dans son profil. Appliquée sur le document SMIL suivant :

La transformation générée donne en résultat le document présenté par la Figure 6.6. Comme nous pouvons remarquer, le document en sortie respecte bien les contraintes du profil en entrée. Le scénario résultant n'inclut ni des ressources audio, ni des conteneurs d'exécution parallèle.

Les transformations de type XSLT sont appliquées au niveau structure des documents et ne peuvent pas assurer d'autres types d'adaptation plus avancées, tels que l'adaptation des ressources externes utilisées dans le documents (exemple, des images, des vidéos, etc.). Lorsqu'une application cliente envoie une requête demandant un certain document, le serveur envoie en plus du document, les différentes ressources utilisées à l'intérieur du document. Il est donc nécessaire d'enrichir les transformations structurelles par des transformations qui s'appliquent au niveau média afin de satisfaire le besoin d'adaptation de ressources. La section suivante est consacrée à l'adaptation des ressources médias.

Une méthode d'adaptation M est applicable sur une ressource (document ou média) R, si la description UPS de R correspond aux conditions d'entrée de M, et si la description UPS de sortie de M correspond aux contraintes du client cible (voir Chapitre 4). Pour éviter la définition de plusieurs méthodes d'adaptation statiques

```
<smil>
<head>
<layout>
<region id="r1" top="0" left="0" right="240" bottom="240"/>
<region id="r2" top="180" left="150"/>
<region id="r3" top="300" />
<region id="r4" top="40" left="0" right="240" bottom="220"/>
<region id="r5" top="130" left="0" right="240" bottom="277"/>
</layout>
</head>
<body>
<par>
<audio id="audio" src="techno.mp3" begin="0s" end="10s"/>
<seq>
<par>
<img id="img1" region="r1" src="mathilda5.jpg" begin="0" end="5s"/>
<img id="img2" region="r2" src="mathilda2.jpg" begin="0s" end="5s"/>
</par>
<par>
<audio id="audio1" src="techno.mp3" clip-begin="23s" clip-end="28s"/>
<img id="img3" region="r1" src="mathilda4.jpg" begin="0s" dur="5s"/>
<img id="img4" region="r2" src="mathilda3.jpg" begin="0s" dur="5s"/>
<img id="img5" region="r3" src="mathilda.jpg" begin="0s" dur="5s"/>
</par>
</seq>
</par>
</body>
</smil>
```

FIG. 6.5 – Source SMIL à adapter

```
<smil>
<head>
<layout>
<region id="r1" top="0" left="0" right="240" bottom="240"/>
<region id="r2" top="180" left="150"/>
<region id="r3" top="300"/>
<region id="r4" top="40" left="0" right="240" bottom="220"/>
<region id="r5" top="130" left="0" right="240" bottom="277"/>
</layout>
</head>
<body>
<seq>
<img id="img1" region="r1" src="mathilda5.jpg" begin="0" end="5s"/>
<img id="img2" region="r2" src="mathilda2.jpg" begin="0s" end="5s"/>
<img id="img3" region="r1" src="mathilda4.jpg" begin="0s" dur="5s"/>
<img id="img4" region="r2" src="mathilda3.jpg" begin="0s" dur="5s"/>
<img id="img5" region="r3" src="mathilda.jpg" begin="0s" dur="5s"/>
</seq>
</body>
</smil>
```

FIG. 6.6 – Document SMIL adapté

pour chaque type de ressource, il est plus efficace de définir des méthodes d'adaptation qui peuvent générer en sortie une variété de ressources de différentes caractéristiques et qui peuvent être re-appliquées dans différents contextes : c'est le principe de l'adaptation dynamique.

Dans les architectures classiques, l'adaptation dynamique est assurée en général par l'envoi de certains scripts et styles de formatage inclus dans le contenu final et évalués par le client cible lors de la réception du contenu. Ce genre d'approche connaît beaucoup de limitations : il dépend des capacités de traitement des terminaux et de leurs applications clientes (souvent limitées) et ne peut pas assurer une bonne prise en compte des paramètres du contexte.

Comme nous l'avons introduit dans le chapitre 3 (Section 3.6.2), NAC prend en compte l'adaptation dynamique en suivant deux approches [74] [70] :

1. La définition de méthodes d'adaptation paramétrées, et
2. La génération automatique d'adaptation.

La première approche est assurée par la définition de méthodes d'adaptation qui supportent le changement des dimensions du contexte, par exemple une méthode de conversion de ressources de différents formats d'entrée, une compression variable d'images, une extraction de contextes de profils avec des expressions XPath qui guident l'extraction, etc.

Afin de prendre en compte la variation des dimensions contextuelles, des variables sont associées à chaque méthode d'adaptation dynamique (exemple, le format d'entrée, le niveau de compression, etc.). Lors de la réception d'une requête cliente et après l'évaluation du contexte de l'environnement et de ses différentes dimensions, le processus de négociation applique la méthode d'adaptation correspondante en instanciant ses variables avec les valeurs des dimensions qui correspondent à la nature de l'adaptation. Par exemple, une adaptation dynamique de conversion peut être instanciée avec la valeur de format de sortie 'GIF' si le client cible ne supporte pas le format d'origine de la ressource source. Une adaptation de compression peut être instanciée avec une grande valeur de compression si le contexte courant spécifie que la bande passante du réseau est très faible.

Dans NAC, l'adaptation dynamique est aussi utilisée dans l'extraction de contexte en utilisant le repository de profils UPS. En effet, le processus d'extraction de contexte représente la même méthode qui accepte différentes variables en entrée. Ces variables permettent de guider la sélection des parties de profils UPS et de les envoyer par la suite au proxy. Les variables sont instanciées après la réception des différents appels RPC du proxy. (voir Section 3.4).

L'exemple suivant montre une application d'adaptation dynamique dans NAC. Dans cet exemple, la méthode d'adaptation utilisée assure l'application d'un algorithme de compression d'encodage des ressources images. La méthode permet de contrôler le degré de compression ainsi que la prise en compte de différents formats d'image (JPEG, GIF et TIF).

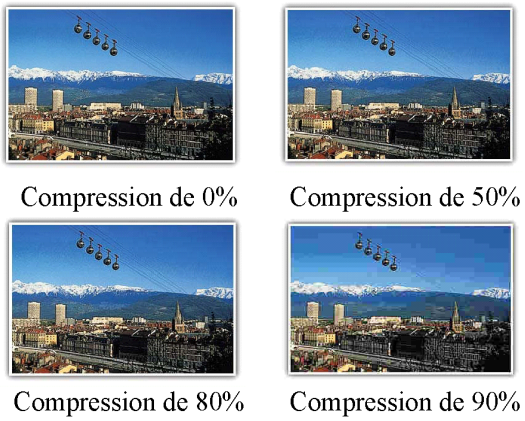

FIG. 6.7 – Transformation de ressources média (compression d'images)

La Figure 6.7 montre un exemple de ressources médias images, générées en appliquant différents degrés de compression. La première ressource représente la ressource source, les autres ressources sont créées en temps réel par adaptation de la ressource source.

La méthode d'adaptation dynamique permet au système de réagir dynamiquement selon le contexte de l'environnement et ses variations. Un des scénarios d'utilisation de cette adaptation, est de changer automatiquement la compression des ressources, selon la bande passante disponible du réseau. Le Tableau 6.1, donne quelques résultats d'évaluation après l'application d'une méthode d'adaptation dans différents contextes. Les expérimentations ont été effectuées en utilisant un PC avec un micro-processeur Intel Pentium 4 de 2.00 GHz et 266 Mo de mémoire principale. A travers les résultats donnés, nous pouvons remarquer que la transmission de la ressource source peut nécessiter des délais de téléchargement considérables lorsque la bande passante du réseau devient faible. Il est donc préférable d'appliquer une adaptation contrôlée afin de trouver un compromis entre la qualité de la ressource transmise et le temps de réponse du proxy.

La décision d'application des différentes méthodes d'adaptation dynamiques avec les meilleurs paramètres doit prendre en compte :

1. La qualité du contenu : ce qui revient à effectuer le meilleur choix pour que le contenu adapté ne viole pas les caractéristiques et la sémantique du contenu source.
2. L'application des méthodes efficaces : ce qui revient à éviter l'application de méthodes d'adaptation trop consommatrices en temps d'exécution.
3. La satisfaction du contexte courant de l'environnement : l'instanciation des différents paramètres doit respecter le contexte cible, par exemple, une méthode de conversion dynamique de formats doit être appliquée avec une valeur de format de sortie qui correspond au format accepté par le client.

Tab. 6.1 – Evaluation de l'adaptation dynamique

	Image source	**Image 2**	**Image 3**	**Image 4**
Taille (octet)	13131	10094	6576	5062
Temps de téléchargement de la ressource source par le proxy (ms)	25			
Adaptation : compression JPEG de :	0%	50%	80%	90%
Temps d'adaptation moyen (ms)	0	95	95	95
Gain de taille (octets)	0	3037	6555	8069
Temps total de récéption d'un client avec la bande passante de 128 Kbit/s (ms)	820,69	725,87	506	411,37
Temps total de récéption d'un client avec la bande passante de 256 Kbit/s (ms)	410,34	410,43	300,5	253,19
Temps total de récéption d'un client avec la bande passante de 512 Kbit/s (ms)	205,17	252,72	197,75	174,10

6.2.3 L'adaptation dynamique pendant l'exécution

Dans cette section, les terminaux cibles que nous considérons sont des terminaux qui ne possèdent pas de système de présentation évolué. Nous considérons plus précisément, les clients qui sont incapables d'assurer une analyse de scénarios multimédia complexes, incluant la gestion spatiale, temporelle et hypermédia d'un contenu multimédia. Le modèle de scénario multimédia que nous considérons est le langage SMIL 2.0 [109].

Comme nous avons vu dans le chapitre 2, le module de contrôle de contenu (content control module) du langage SMIL 2.0 permet l'utilisation de l'élément *switch* pour spécifier, dans une présentation SMIL, une collection d'objets alternatifs. La sélection de contenu est exprimée à l'aide des attributs de test système (les attributs de test définis dans SMIL 2.0 en plus des sept attributs définis dans SMIL 1.0 [17] [108]).

Dans les environnements hétérogènes, il est difficile d'attribuer la tâche de sélection de contenu aux terminaux à cause de leurs limitations et leurs capacités de traitement réduites. En outre, il est parfois impossible que le terminal parvienne à évaluer certains attributs de tests, en particulier les attributs liés aux caractéristiques du réseau. Il est donc plus efficace d'attribuer la sélection à une entité de l'architecture qui peut effectuer de tels traitements et qui peut avoir une image globale des caractéristiques du contexte de l'environnement. Dans NAC, c'est le module de négociation et d'adaptation qui est responsable de l'évaluation de la sélection de contenu.

Comme nous avons vu dans le chapitre 4, les caractéristiques du contexte de l'environnement peuvent être à tout moment extraites par le module ANM soit à l'aide des profils UPS soit à l'aide des évaluations dynamiques. Ce sont ces caractéristiques qui permettent l'évaluation des éléments de sélection du langage SMIL au niveau proxy ou au niveau serveur. Le module de négociation évalue donc les différents attributs de test système de SMIL selon le contexte actuel. Le premier élément SMIL qui correspond au contexte actuel est l'élément sélectionné qui remplace donc l'élément *switch* dans le document SMIL. Un élément sans attributs de test système est toujours acceptable. Si aucun élément ne correspond au contexte de l'environnement, l'élément

Tab. 6.2 – Vidéos utilisées dans les présentations SMIL

	Vidéo 1	**Vidéo 2**
Format en entrée	MPEG 1	Indeo Video 3
Dimensions	352x288	
Cadence (image/s)	25.0	
Taille (Ko)	8501	8770
Durée (s)	30.99	30

spécifié par défaut est choisi.

Considérons la partie suivante d'un document SMIL :

```
<switch>
<img region="subtitling_zone" src="subtitle1.gif"
begin="14.5" end="15.9" systemLanguage="fr"/>
<img region="subtitling_zone" src="subtitle2.gif" begin="14.5"
end="15.9" systemLanguage="en"/>
</switch>
```

Si la préférence de l'utilisateur est la langue française, le module de négociation de NAC, adapte la partie précédente comme suit :

```
<img region="subtitling_zone" src="subtitle1.gif" begin="14.5"
end="15.9" />
```

NAC inclut l'adaptation des présentations SMIL incluant de la vidéo vers un flux vidéo unique incluant toute la présentation. Les présentations vidéo générées présentent le même scénario temporel tel qu'il a été spécifié dans le scénario SMIL source. Ce type d'adaptation inclut l'analyse du scénario temporel SMIL en entrée et l'encodage de la vidéo en sortie. L'encodage de la vidéo consiste à décoder la vidéo source référencée par le scénario SMIL source en un format décompressé, et la génération de la vidéo finale. Les décodeurs vidéos utilisés dépendent de l'encodage de la ressource vidéo source. Les décodeurs sont configurés pour générer un format décompressé de chaque image de la vidéo qui est simplement la représentation en points de l'image. Chaque point appartient au format RBV représenté par trois octets qui correspondent à la valeur de couleur rouge, bleue et verte du point [71].

Dans ce qui suit, nous présentons quelques résultats expérimentaux concernant l'adaptation d'un contenu SMIL vers un contenu vidéo. Les expérimentations concernent la génération d'un contenu vidéo à partir de deux présentations SMIL (le même scénario avec différents éléments de sous-titrage) et en utilisant deux ressources vidéos en entrée. Les mesures données concernent uniquement le format non compressé de la vidéo générée. La Table 6.2 donne les caractéristiques des deux vidéos utilisées dans les présentations SMIL.

Le graphe temporel des présentations SMIL utilisées est représenté par la Figure 6.8. La Table 6.3 montre les caractéristiques des vidéos générées à partir des deux présentations SMIL en entrée ainsi que le temps de l'encodage. La Figure 6.9 montre la vidéo générée à partir du deuxième scénario temporel SMIL. Le module de négociation, lance le processus d'adaptation *SMIL vers Vidéo* avec les bons paramètres qui correspondent aux préférences de l'utilisateur. Il est donc possible de générer plusieurs

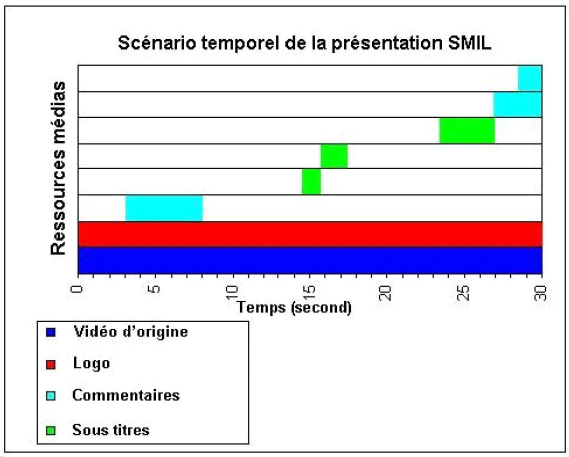

FIG. 6.8 – Scenario temporel des présentations SMIL

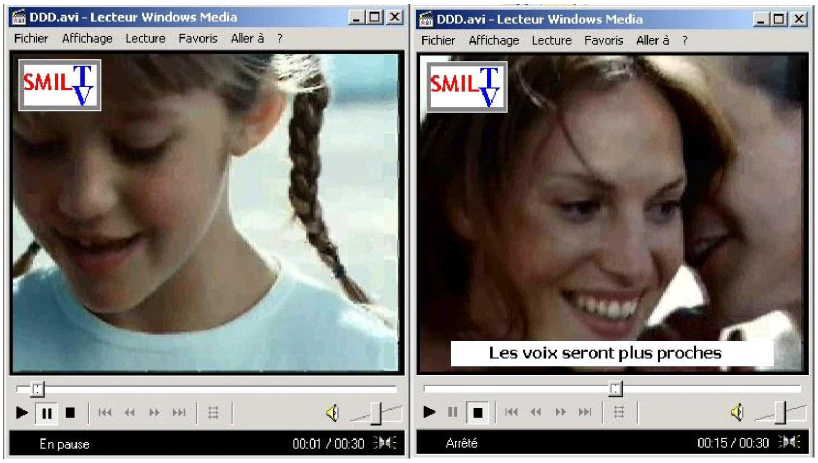

FIG. 6.9 – Vidéo générée à partir de scénario SMIL

vidéos avec des sous-titrages en différentes langues grâce à ce type d'adaptation.

Le processus d'adaptation de NAC peut aussi effectuer des adaptations de ressources médias utilisées à l'intérieur des documents SMIL. Les ressources sont donc adaptées et utilisées dans le nouveau document SMIL adapté et envoyées au client. La Figure 6.10 montre l'adaptation des ressources vidéo utilisées dans les documents SMIL. La partie *a* de la figure montre la vidéo source utilisée dans une présentation SMIL. La partie *b* montre le document SMIL adapté (tels qu'il est présenté par PocketSMIL sur un assistant personnel) avec un retaillage de dimensions de la ressource vidéo source. Les dimensions de la vidéo ont été adaptées afin de correspondre aux limitations d'affichage du client cible.

6.3 L'adaptation des ressources média

Le principe de l'adaptation des ressources média est que la transmission des différentes ressources utilisées dans un document ne doit se faire que s'il y a une correspondance entre les caractéristiques des ressources et les contraintes du client

TAB. 6.3 – Vidéos générées

	Vidéo 1	**Vidéo 2**
Format en sortie	Format décompressé (RGB 24 bits)	
Dimensions	352x288	
Cadence (image/s)	25.0	25.0
Taille (Ko)	197918	227940
Nombre d'images	801	750
Durée d'images (s)	0.0387	0.04
Temps d'encodage et de sauvegarde (ms)	35972	39577

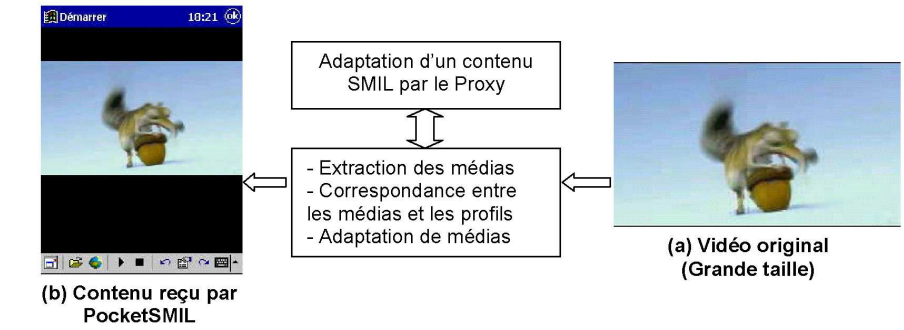

FIG. 6.10 – Adaptation des ressources média utilisées dans SMIL

cible. Pour satisfaire ces contraintes, les ressources peuvent être substituées, filtrées ou transformées du format original vers un nouveau format. La transmission d'une ressource peut aussi dépendre d'autres paramètres tels que les contraintes du réseau de communication. Par exemple, si la bande passante devient faible, une ressource peut être substituée par une variante qui consomme moins de bande passante. L'identification et l'extraction des ressources à partir d'un document permettent l'évaluation de l'effort d'adaptation à appliquer avant la transmission finale du contenu. Dans ce qui suit nous évaluons la relation entre les différentes ressources médias et le temps de transmission du contenu.

Nous supposons que le contenu source C utilise n ressources média MR et que le temps de recherche local d'une ressource est négligeable. Une ressource MR peut exister en une seule ou plusieurs versions. Un document qui référence des ressources média (par exemple un document SMIL ou XHTML) est aussi considéré comme une ressource.
Soit MR_i^v la v-ième version de la i-ième ressource, Size(MR_i^v) est la taille de la version MR_i^v, $Transformed_k(MR_i^v)$ est la ressource média obtenue après avoir appliquer la méthode d'adaptation k sur MR_i^v, $T\_Transformation_k(MR_i^v)$ est le temps nécessaire pour transformer MR_i^v en appliquant la méthode k.

La taille du contenu adapté est égal à :

$$Size' = \sum_{i=1}^{l} Size(MR_i^v) + \sum_{i=l+1}^{n} Size(Transformed_k(MR_i^v))$$

Où les ressources média de $l+1$ à n sont transformées en appliquant des méthodes d'adaptation, d'où la différence de taille entre le contenu source et le contenu adapté est égal à :

$$\Delta Size = \sum_{i=l+1}^{n} (Size(MR_i^v) - Size(Transformed_k(MR_i^v))...(I)$$

Si B_A est la valeur moyenne de la bande passante et D_{RTT} est le délai de retour du réseau, le temps de transmission du contenu adapté de C est :

$$D\_Time = D_{RTT} + \frac{Size'}{B_A} + \sum_{i=l+1}^{n} T\_Transformation_k(MR_i^v)...(II)$$

Le gain en temps de transmission est donc :

$$G\_Time = \frac{\Delta Size}{B_A} - \sum_{i=l+1}^{n} T\_Transformation_k(MR_i^v)...(III)$$

La formule (III) du gain de temps de transmission montre la relation entre l'adaptation des ressources média et le temps de transmission du contenu final. Le résultat de cette évaluation est utile pour ajuster la stratégie de négociation du contenu appliquée par le module ANM. Afin de respecter le contexte de transmission (les caractéristiques du client, les limitations du réseau, etc.), la stratégie de négociation peut juger si une

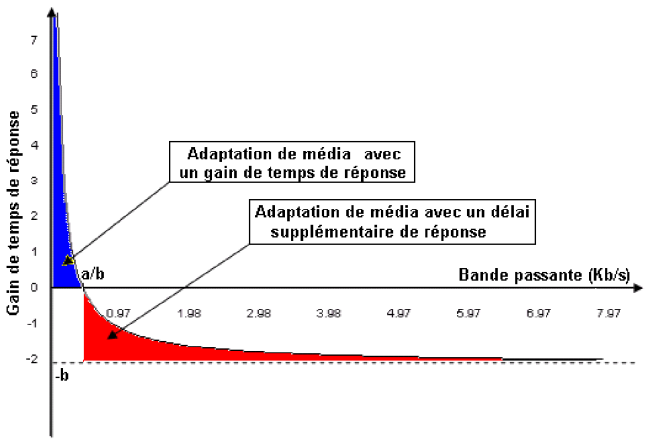

FIG. 6.11 – La relation entre le temps de transmission du contenu et la bande passante

adaptation ou une substitution de ressource média est nécessaire ou pas. Dans certains cas, l'application de l'adaptation des ressources, utilisées dans un document, peut diminuer considérablement le temps de transmission du document ce qui améliore les performances du système d'adaptation et de négociation.

La Figure 6.11 montre la relation entre le gain en temps de transmission de contenu et la bande passante du réseau, avec $a = \Delta Size$ et $b = \sum_{i=l+1}^{n} T\_Transformation_k(MR_i^v)$. Comme nous pouvons voir, le gain en temps de transmission reste positif quand la valeur de la bande passante est inférieure à $\frac{a}{b}$. Ce résultat est interprété dans le contexte de l'adaptation comme suit : l'adaptation des ressources média offre un gain de temps de transmission du contenu final, quand la bande passante courante du réseau est inférieure à $\frac{a}{b}$, sinon, la transmission du contenu adapté nécessite un délai supplémentaire par rapport à la transmission du contenu source. Indépendamment de l'importance de la transmission d'un contenu adapté qui soit compréhensible par le client cible, le résultat de l'évaluation montre que il est efficace d'appliquer une adaptation de ressources média quand le réseau de communication devient limité.

6.4 L'adaptation pour les dimensions dynamiques du contexte client

Dans certaines situations, l'adaptation du contenu dépend de quelques caractéristiques dynamiques du contexte client. Ces caractéristiques peuvent donc changer à tout moment ce qui nécessite que le processus d'adaptation mette à jour sa vision du contexte client à chaque transmission de contenu. NAC prend en charge ce type particulier d'adaptation en s'appuyant sur une réévaluation continue de la dimension du contexte concernée. Les dimensions dynamiques d'un contexte client sont généralement sous forme de préférences d'utilisateur (langue préférée, format de ressources, etc.). Cependant, dans certains cas, ces dimensions peuvent concerner les capacités du terminal lui-même.

Lors des différentes expérimentations du système d'adaptation et de négociation

de NAC, nous avons remarqué que les systèmes embarqués des terminaux de capacités limitées se bloquent souvent lorsqu'ils reçoivent du contenu très consommateur en ressource mémoire (par exemple, de complexes présentations multimédia). Généralement, un terminal ne peut pas avoir une idée sur la complexité de la présentation distante tant qu'il ne l'a pas encore reçue. Pour résoudre ce problème de blocage, une technique d'adaptation basée sur l'état courant de la mémoire du client cible a été intégrée au système d'adaptation de NAC. Ici, le principe de négociation est le suivant : le système doit fixer un seuil d'utilisation de la mémoire cible qu'il ne faut pas dépasser. Le système doit donc effectuer des adaptations de contenu dès que ce seuil est dépassé par le terminal afin d'éviter la saturation de mémoire.

Le module responsable du contexte client (UCM, voir Chapitre 3), a été enrichi par une fonctionnalité qui permet de reporter (à la demande ou régulièrement) la capacité et l'état courant de la mémoire du terminal mobile. Cette fonctionnalité permet au module de négociation et d'adaptation de classifier le terminal grâce à la connaissance de la capacité totale de la mémoire, et de suivre le comportement et l'état courant de l'utilisation de la mémoire selon la réception de la présentation multimédia.

Dans ce qui suit nous présentons en détail un scénario d'application de ce genre d'adaptation. La présentation multimédia considérée est sous forme d'un document SMIL, et le terminal est un assistant personnel (un iPAQ 3600) qui accède à la présentation en utilisant le protocole HTTP en passant par un proxy ANM. Le réseau de communication est un réseau sans fil de type 802.11b. Le navigateur client utilisé est PocketSMIL. Voici une partie de la présentation SMIL qu'on considère :

```
<par>
   <video id="vid" region="region_video" src="Videos/orange4_f.mpg"/>
   <img region="img_zone1" src="img1.gif" begin="1s" end="vid.end"/>
   <img region="img_zone2" src="img2.gif" begin="2s" end="vid.end"/>
</par>
```

Le scénario inclut une ressource vidéo qui se joue en même temps que deux différentes images avec un décalage de commencement dans le temps. La Figure 6.12 montre la variation de l'état de la mémoire du terminal telle qu'elle a été reportée par le module UCM. Comme nous pouvons voir, l'état d'utilisation de la mémoire change lorsque le navigateur PocketSMIL commence à présenter le document SMIL. Naturellement, l'utilisation de la mémoire augmente lors de la réception et du décodage des ressources médias. Dans ce scénario, l'utilisation maximale de la mémoire représente 57 % de la capacité totale de la mémoire du terminal.

L'adaptation consiste à réduire la cadence de la ressource vidéo source lorsque le pourcentage de l'utilisation de la mémoire commence à dépasser une valeur donnée α. Cette valeur dépend des caractéristiques du terminal cible et peut être estimée après plusieurs expérimentations. Le module d'adaptation du proxy réduit progressivement la cadence de la vidéo source jusqu'à ce que l'UCM reporte un nouvel état de mémoire qui soit inférieur à la valeur α. Lorsque l'état d'utilisation devient inférieur à α, la cadence adoptée reste la même jusqu'à ce que l'UCM reporte un nouvel état de mémoire (déterminé par une valeur prédéfinie) qui permet l'augmentation de la cadence de la vidéo. Cette approche évite le blocage des terminaux de capacités limitées et évite la transmission inutile de contenu à travers le réseau. Cependant,

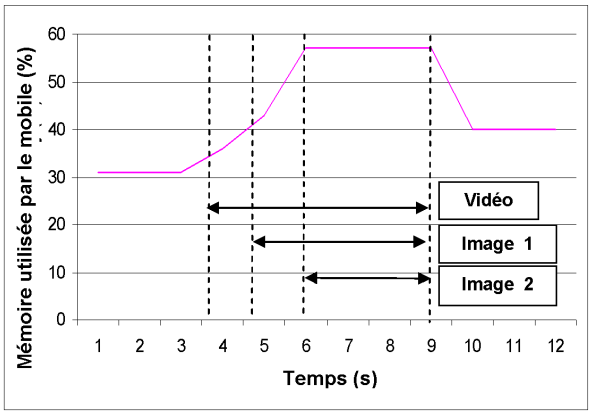

FIG. 6.12 – La variation d'état de mémoire de l'assistant personel

l'approche ne peut pas être appliquée pour tous les types de terminaux. En effet, pour les terminaux de capacités très limitées, la réduction de la cadence vidéo peut être très importante ce qui favorise la substitution à l'adaptation des ressources média. L'approche ne résout pas le problème de blocage quand la saturation de la mémoire du terminal est indépendante de la présentation du contenu multimédia reçu.

6.5 Evaluation de la gestion des profils

Nous présentons maintenant quelques mesures de performance des services du repository de NAC. Nous présentons aussi quelques mesures relatives à l'interrogation et la transmission de contextes et des composants de contextes des terminaux. Ces mesures sont effectuées à partir d'une base de profils réels : cent dix huit profils de terminaux mobiles utilisés. Les profils sont écrits en UPS [77] soit sur la base des descriptions techniques fournies par les fabricants des terminaux ou par une transformation XSLT (voir Annexe D) sur des descriptions déjà écrites en UAProf [116]. Ces mesures montrent que dédier la gestion des profils, en termes d'interrogation, de sauvegarde et d'interaction en utilisant un protocole de services, permet d'obtenir de bonnes performances vis-à-vis du processus global d'adaptation et de négociation du contenu. En outre, les mesures montrent que l'identification de parties de contexte à extraire et à traiter donne de meilleures performances que l'utilisation des contextes globaux.

Les expérimentations ont été effectuées dans une plate-forme qui comporte deux infrastructures de réseaux : un réseau LAN sans fil 802.11 et un réseau filaire à 100 Mbits/s. Trois types de terminaux sont utilisés pour pouvoir accéder au contenu du système : un PC de poche (Pocket PC) iPAQ 3600 sous une plate-forme Windows CE connecté à travers le réseau sans fil, un PC portable utilisant les deux connexions sans fil et filaire et un PC de bureau connecté à travers le réseau filaire. Le PC de poche simule différents types de terminaux mobiles en utilisant le module implémenté du contexte de client (UCM). La majorité des méthodes d'adaptation ont été implémentées en Java et Java Media Framework [60] pour l'encodage et l'adaptation des ressources multimédia et en utilisant le langage XSLT [135] et XQuery [134] pour les transformations structurelles. Le repository SOAP utilise l'implémentation Apache

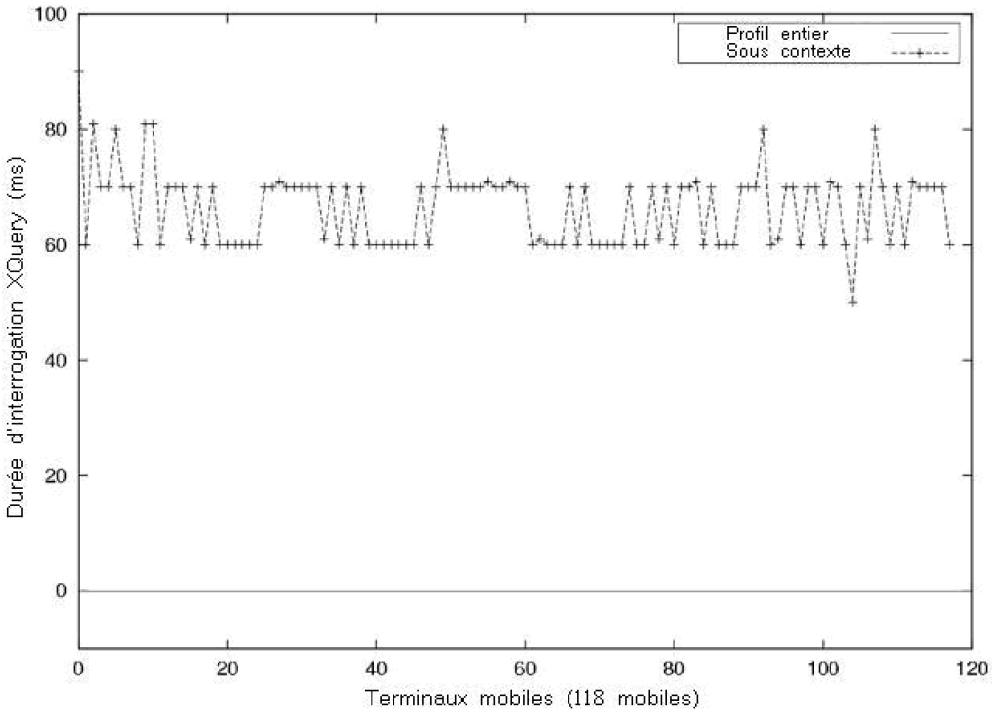

FIG. 6.13 – Performance des interrogations XQuery

de SOAP [3] et l'implémentation Tamino du langage XQuery [96]. Le repository fonctionne sous une machine Compaq PC avec un processeur Intel Pentium 4 de 2.00 GHz et 266 Mo de mémoire principale.

Afin de mieux observer le comportement de notre système, nous avons effectué quelques mesures de performance sur le repository de NAC sur la base des critères suivants :

- Le temps de traitement des requêtes (XQuery) appliquées sur les profils
- Le temps de transmission après un appel RPC
- Le temps de transmission des différents composants du même profil UPS

Aucun ordre n'est considéré dans l'ordonnancement des terminaux selon les axes des x des Figures 6.13 et 6.14.

La Figure 6.13 représente les mesures relatives au temps d'interrogation XQuery après la reception de deux types d'appels RPC SOAP : $GetProfile$ et $GetSubContext$. Ici, le sous contexte considéré est la partie $OnlySupportedResources$ (voir les schémas UPS [77]) de chaque profil. La Figure montre aussi que la valeur du temps de requêtes sur les profils entiers est nulle. Ceci s'explique simplement par le fait qu'aucune requête d'interrogation est nécessaire lorsque le repository reçoit une demande de profil entier. Dans ce cas là, le repository transmet la totalité du contenu du profil. Par contre, pour recevoir un sous contexte de profil, le repository instancie et évalue une requête d'interrogation afin de sélectionner les parties des profils. La figure 6.14 représente les mesures qui ont été effectuées sur le temps de transmission des profils entiers et des sous contextes de profils (le sous contexte $OnlySupportedResources$ [77]). Les mesures concernent dix appels RPC de type : $GetProfile$ pour les profils entiers et

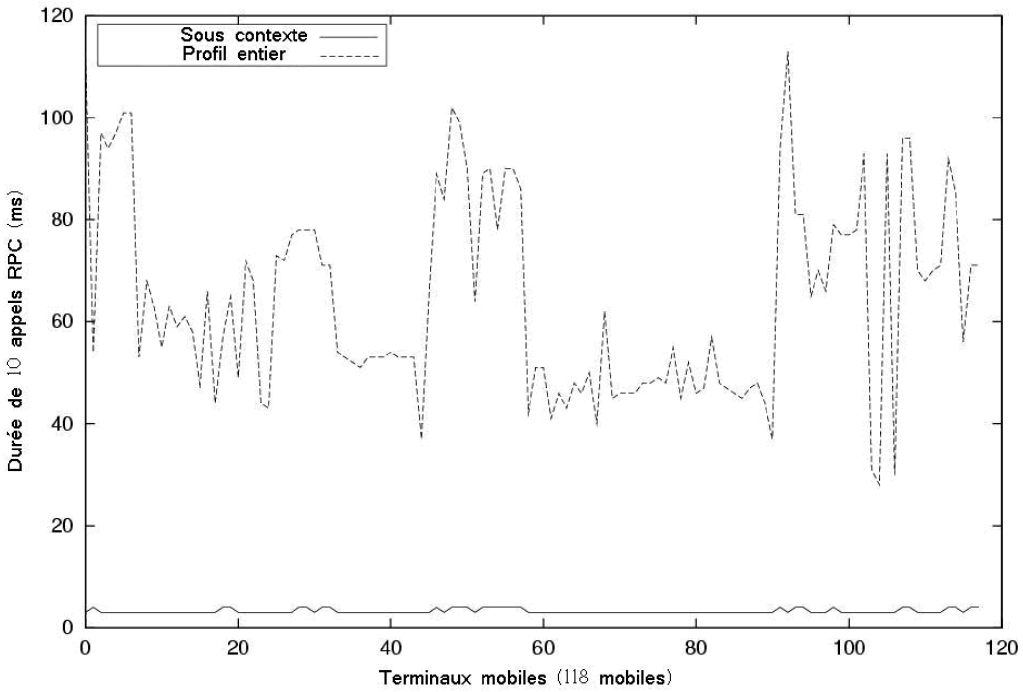

FIG. 6.14 – Performance des appels RPC du proxy vers le repository

GetSubContext pour les sous contextes.

Les mesures montrent que le repository met plus de temps pour répondre aux demandes des profils entiers comparant avec l'interrogation et la transmission des parties du contexte de l'environment qui ont une relation directe avec le processus d'adaptation de contenu.

Le but des mesures présentées par la Figure 6.15, est de comparer le temps de transmission des profils entiers et des différents composants. La Figure donne des mesures relatives à un terminal choisi : un téléphone mobile de type Sony Ericsson T68R502 [72]. Les mesures montrent qu'il est possible d'assurer une adaptation de contenu plus efficace en profitant de la structure des profils UPS qui est basée sur le principe des composants du CC/PP [43]. En effet, un composant UPS associé à une dimension de contexte peut être facilement identifié, par conséquent l'envoi du contexte peut se limiter à l'envoi de composants UPS.

6.6 Conclusion

Dans ce chapitre, nous avons présenté les techniques et les méthodes d'adaptation du contenu de l'architecture NAC. L'adaptation du contenu de NAC tire profit des langages de transformation de documents pour assurer l'aspect transformation structurelle. Comme la structure des documents peut référencer des ressources médias externes qui sont transmises à l'application cliente, l'adaptation de l'architecture combine la transformation structurelle avec des différentes techniques d'adaptation de

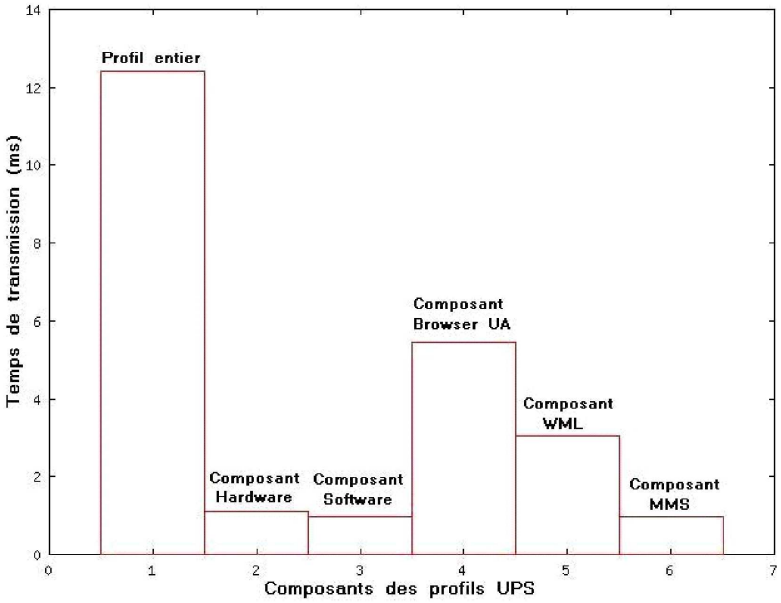

FIG. 6.15 – Temps de transmission des composants de profils

médias.

Comme nous avons vu, l'adaptation statique ne peut pas répondre à toutes les situations de transmission de contenu. Les approches d'adaptation dynamiques permettent de minimiser l'effort de conception des méthodes d'adaptation et de couvrir plusieurs situations où les dimensions du contexte peuvent varier. Dans un environnement à ressources limitées, il est important de contrôler la qualité du service d'adaptation ainsi que l'utilisation des ressources du système. La formulation de l'adaptation ainsi que les résultats d'expérimentations des techniques d'adaptation et de la gestion des descriptions, permettent dévaluer l'effort appliqué par le système d'adaptation au sein de l'architecture globale.

Chapitre 7

Conclusion

7.1 Rappel des objectifs

L'apparition récente d'une grande variété de moyens de communication, accompagnée d'un accroissement important de l'information multimédia rend la négociation et l'adaptation de contenu nécessaires. Cette nécessité se justifie par une demande croissante d'accès à l'information en tout lieu et sur des plates-formes très hétérogènes. Dans ce cadre, le projet Opéra, dans lequel s'est inscrit cette thèse, étudiait les problèmes d'édition, de maintenance et de traitement de documents structurés et multimédia. Le traitement des documents couvre de plus en plus l'adaptation et la réutilisation du contenu sous différents contextes. L'adaptation et la négociation représentent désormais un thème important dans le domaine multimédia. Le travail de cette thèse s'inscrit dans ce thème et concerne la conception et la réalisation d'une architecture d'adaptation et de négociation pour les environnements hétérogènes.

Les environnements hétérogènes sont caractérisés par des contraintes qui se situent à plusieurs niveaux : le terminal, l'utilisateur, le contenu, le réseau, le serveur et ses capacité d'adaptation. La négociation a pour objectif d'analyser ces contraintes et d'établir une correspondance entre les capacités d'une application cliente et les caractéristiques du contenu. L'adaptation a pour objectif de transformer le contenu de sa forme initiale vers une nouvelle forme qui puisse être traitée et utilisée par le client. Dans le cas général, la négociation a recours à l'adaptation quand le contenu d'origine n'est pas conforme au profil du client.

Les solutions proposées actuellement ne s'attaquent pas au problème de l'adaptation avec des architectures complètes, mais en essayant de fournir des solutions à des besoins très spécifiques tels que l'adaptation des images pour les mobiles ou le transcodage de la vidéo. En outre, ces systèmes ne considèrent pas les aspects fondamentaux de la négociation comme l'aspect protocolaire et la gestion de profils. Le cadre de travail CC/PP défini par le W3C prend en compte la description des capacités et des préférences de l'utilisateur. La publication de la recommandation CC/PP [43] est récente, mais il y manque plusieurs aspects, tels que le protocole, le traitement et l'échange des profils et la description des profils du contenu.

Les lacunes des solutions existantes et le manque d'une architecture plus complète, qui permette un accès universel à l'information dans un environnement hétérogène, montre que plusieurs aspects du problème de l'adaptation et de la négociation de contenu doivent encore être étudiés. Il donc est notamment nécessaire de définir des

mécanismes qui prennent en compte les contraintes de toutes les entités de l'environnement, l'échange des informations de négociation entre ces entités et l'application de différentes formes d'adaptation de contenu.

7.2 Démarche suivie dans le travail et bilan scientifique

Afin d'atteindre ces objectifs, nous avons analysé les concepts fondamentaux des systèmes de négociation et nous avons identifié les différentes techniques d'adaptation de contenu. Cette étude montre que la prise en compte de la description du contexte est fondamentale. C'est grâce à la description du contexte que le contenu peut être négocié et que les techniques d'adaptation peuvent être appliquées correctement. Nous avons montré les limitations des solutions existantes qui ne considèrent pas la description de la totalité de l'environnement hétérogène. Les descriptions qui se limitent parfois au client et même uniquement au format du contenu ne permettent pas d'assurer une stratégie de négociation efficace ni de tirer profit des capacités d'adaptation d'une architecture client/serveur. Les limitations s'étendent aussi à un manque d'interaction entre les entités de l'environnement et à l'absence de stratégies d'adaptation automatiques qui dépendent directement du contexte de l'environnement.

Le premier résultat de cette thèse est une architecture flexible qui permet de définir clairement les composants intervenant dans la négociation et l'adaptation du contenu, et qui décrit comment ces composants sont organisés. Nous avons proposé une organisation souple à base de proxy. L'utilisation d'un proxy intermédiaire permet de rendre l'adaptation transparente. En effet, le contenu de tout le réseau peut être traité en passant par le proxy utilisé par les différentes applications clientes. Le module d'adaptation peut également être intégré au serveur, afin de développer des techniques d'adaptation plus avancées dans certains cas. L'architecture proposée est extensible, et permet l'enrichissement du système par de nouvelles méthodes d'adaptation.

Une fois l'architecture et ses différents composants définis, il était nécessaire de définir un modèle pour la prise en compte du contexte. Nous avons cherché à capturer au mieux les différentes contraintes du contexte qui peuvent influencer la transmission d'un contenu adapté et qui peuvent être extraites des caractéristiques matérielles des terminaux et des préférences des utilisateurs. Nous nous sommes basés sur une approche déclarative pour la spécification du modèle UPS. Ce modèle permet de décrire le client (capacités matérielles et logicielles, préférences utilisateur), le contenu (fonctionnalités et ressources utilisées) les capacités d'adaptation (conditions d'application et description de contenu en sortie) et le réseau et ses caractéristiques. Le modèle UPS est inspiré du cadre de travail CC/PP [43]. Il inclut une extension de description pour des nouvelles entités de l'environnement et la prise en compte de nouvelles caractéristiques telles que les capacités du système d'adaptation. Le modèle UPS représente le second résultat de cette thèse. Un composant de l'architecture, le repository des profils, a été dédié au stockage et à la gestion des profils et fonctionne sous la forme d'un service Web exploité par le proxy de l'architecture NAC.

Le troisième résultat est une stratégie de traitement de profils UPS et un protocole de négociation et d'acquisition du contexte client. La stratégie de traitement des profils comprend la mise en correspondance des différentes dimensions du contexte. Le protocole de négociation sert à garantir une interaction efficace entre les différents composants de l'architecture : le client, le proxy et le serveur. Cette interaction

implique le module du contexte client, le module d'adaptation et de négociation ANM, le serveur et le repository des profils. L'interaction comporte la définition de plusieurs types de requêtes, échangées entre le module ANM et le terminal, et des services offerts par le repository et utilisés par le proxy.

Enfin, le dernier résultat de cette thèse est un système d'adaptation. Ce système, au cœur de notre architecture, met en œuvre des mécanismes permettant d'adapter le contenu pour différents contextes. L'adaptation porte sur la structure des documents ainsi que sur les ressources média utilisées. Les mécanismes d'adaptation sont instanciés en appliquant une stratégie de négociation pour un contexte donné. Le système d'adaptation est flexible et peut inclure, à tout moment, des nouvelles méthodes d'adaptation. En effet, il suffit de déclarer le profil d'adaptation UPS d'une nouvelle méthode pour qu'elle soit prise en considération par le système.

7.3 Bilan et évaluation de la réalisation

La mise en œuvre pratique des solutions qui résultent de notre étude du problème de l'adaptation et de négociation de contenu a représenté une part importante du travail accompli. La réalisation de l'architecture NAC avec tous ses composants a permis de mettre en œuvre une stratégie de négociation efficace et un système assez riche en techniques d'adaptation de contenu.

Le prototype NAC comporte plusieurs types de modules qui coopèrent entre eux. Ces modules sont exploités à plusieurs niveaux, au niveau client, serveur, proxy et respository. L'architecture est flexible et extensible. Le système d'adaptation peut être personnalisé pour répondre à des besoins d'adaptation spécifiques et il peut être enrichi afin d'améliorer la qualité de service. Le traitement et la gestion des profils permettent de lier la transmission de contenu aux caractéristiques courantes des terminaux. La mise en œuvre des services Web pour le traitement de profils et le protocole de négociation permettent d'enrichir l'échange des informations de négociation contrairement aux protocoles de communication classiques.

Comme nous l'avons vu dans la partie évaluation de cette thèse (Chapitre 6), le prototype NAC couvre la gestion d'un ensemble important de profils, la stratégie de négociation et les différentes techniques d'adaptation implémentées fonctionnent de manière satisfaisante avec des performances acceptables. NAC permet d'interagir avec le module de contexte du client et d'adapter le contenu pour différents contextes.

7.4 Perspectives

Cette thèse a permis d'apporter des réponses aux différents problèmes autour de la négociation et l'adaptation du contenu dans les environnements hétérogènes. Elle représente une première expérimentation pour la construction d'une architecture permettant d'adapter et de négocier le contenu sous différentes contraintes et pour plusieurs types de terminaux. Ce travail constitue une base qu'il faudra encore améliorer et compléter. Plusieurs perspectives sont envisageables. En plus de la consolidation des propositions de cette thèse, les perspectives peuvent être organisées autour de quatre grandes directions : le protocole de négociation, les techniques d'adaptation, la spécification de langages de transformation dynamiques et l'adaptation coopérative.

7.4.1 Négociation et traitement de profils

7.4.1.1 Le protocole de négociation

Dans le chapitre 5, nous avons proposé un protocole de négociation et d'acquisition du contexte du client. Ce protocole fait intervenir le module d'adaptation et de négociation (ANM), le module du contexte de client (UCM) et l'application cliente elle-même par les informations de négociation qu'elle peut transmettre. Les requêtes échangées entre les modules ANM et UCM sont limitées aux traitements qui concernent : la réception des profils, la réception des changements de profils et les en-têtes de réponses relatives aux différentes requêtes. Le protocole n'est donc pas complet et peut être étendu de plusieurs façons :

- **Amélioration des performances de la négociation.** Dans un système d'adaptation et de négociation, il est très important de minimiser le temps relatif à l'échange et au traitement des profils. Une piste possible est d'étudier comment améliorer encore les performances de la négociation en adoptant une stratégie avancée de gestion de sessions. Cette stratégie doit s'adapter aux caractéristiques des environnements hétérogènes et en particulier aux réseaux mobiles où une session doit être établie sachant que le client peut se déconnecter et se reconnecter de manière intempestive.

- **Authentification.** Dans le protocole proposé, nous n'avons pas considéré l'aspect authentification lors des différentes connexions des clients au module ANM. En effet, dans NAC, il suffit de connaître le port de la négociation (1977 par défaut) et l'adresse de la machine sur laquelle le module ANM s'exécute, pour qu'un client puisse utiliser le service d'adaptation. L'authentification permet de mieux gérer les différentes connexions compte tenu de l'importance des information des profils et facilite l'identification des terminaux qui se connectent au module ANM.

- **Sécurité.** L'accès au contenu par un terminal, en utilisant le module ANM, provoque l'exécution automatique au niveau proxy (ou serveur) d'un code relatif à la stratégie de négociation et aux différentes techniques d'adaptation. Dans tout protocole où les requêtes clientes provoquent l'exécution d'un code au niveau serveur, l'aspect sécurité est important. Il permet de contrôler et de maîtriser ces exécutions automatiques. Il n'est pas considéré dans NAC, et représente un travail à effectuer au niveau du protocole proposé. L'aspect sécurité inclut aussi l'encryptage des informations de négociation échangées. Rappelons que les informations de négociation incluent des informations sensibles concernant l'utilisateur et ses préférences. Dans certaines applications (par exemple, dans les échanges de transactions), il est important de garder ce genre d'information confidentiel entre l'application cliente et le serveur. Une attention particulière doit être accordée à l'encryptage surtout dans des environnements mobiles réputés très vulnérables.

- **Stratégies de cache.** Dans le but d'améliorer les performances de gestion et de traitement des profils clients, la conception d'une stratégie de gestion de cache est importante. Dans l'utilisation du repository, le protocole de négociation de NAC emploie une gestion simpliste de cache basée sur la sauvegarde des profils et une identification basée sur l'adresse IP du terminal. Il nous paraît nécessaire

de développer une stratégie de cache avancée afin de garantir de meilleures performances.

7.4.1.2 Séparation des préférences utilisateur et des capacités du terminal

Dans le modèle UPS, les préférences de l'utilisateur et les capacités d'un terminal sont décrites dans un même profil. Si un utilisateur donné utilise deux terminaux différents, le système associe au même utilisateur deux profils différents. Cela est dû à la non séparation, du point de vue profil, entre les préférences et les caractéristiques matérielles et logicielles. Une perspective intéressante serait de considérer l'utilisateur indépendamment du terminal utilisé de permettre ainsi de supporter plusieurs terminaux pour un utilisateur donné.

7.4.2 Les techniques d'adaptation

Les techniques d'adaptation développées dans NAC s'appliquent sur une grande diversité de modèles de document et de formats de ressources médias. Il est toujours possible de développer et de prendre en compte d'autres formats et modèles. En plus de la couverture des formats et des modèles, d'autres perspectives d'adaptation sont envisageables.

- **Adaptation sémantique.** Dans cette thèse, nous avons présenté quelques techniques d'adaptation sémantiques telle que la pagination du contenu selon les préférences de l'utilisateur. Le cadre de travail présenté dans [34] définit une approche pour l'adaptation sémantique en se focalisant sur la dimension temporelle des documents multimédia. Le travail effectué jusqu'à présent dans le domaine de l'adaptation sémantique des documents multimédia reste encore limité. Une perspective intéressante est d'étudier ce problème en profondeur et de développer des mécanismes avancés d'adaptation sémantique qui couvrent l'ensemble des dimensions.

- **Vérification statique de type.** Comme nous l'avons déjà vu, le langage XSLT a été utilisé pour effectuer une grande partie des transformations structurelles, que ce soit d'une manière statique ou par génération de feuilles de transformation. Un des problèmes majeurs de XSLT est l'impossibilité de garantir de façon statique (c'est-à-dire avant l'exécution d'une transformation) que quelle que soit l'instance du document source en entrée d'un processeur de transformation, le document produit respecte un modèle de document (sa DTD). Il n'y a donc aucune garantie que les feuilles de transformation XSLT utilisées produisent en sortie un document de présentation qui soit structurellement correct.

7.4.3 La spécification de langage de transformation dynamique

Les langages de transformation de structure, tels que XSLT, sont souvent utilisés pour assurer une adaptation statique. Cela veut dire que le contexte d'application d'une transformation est fixé durant sa conception. Cette approche est inadaptée pour les environnements hétérogènes caractérisés par un changement fréquent du contexte. En effet, avec les langages de transformation existants, il est nécessaire de définir une transformation pour chaque contexte. Dans cette thèse, nous avons proposé une approche d'adaptation dynamique qui dépend des valeurs des dimensions du contexte. Une perspective intéressante consiste à concevoir un langage de transformation

dynamique dont les règles de transformation sont liées directement aux variables du contexte. Ces variables peuvent être définies lors de la conception de la transformation et instanciées une fois le contexte évalué. La conception de tels langages serait très bénéfique pour les adaptations dynamiques qui peuvent être facilement réutilisées dans différents contextes. L'effort de la conception des transformations pourrait ainsi être considérablement minimisé.

7.4.4 L'adaptation coopérative

L'étude menée sur les systèmes d'adaptation existants a permis de constater que les solutions proposées ne fonctionnent que dans des cas précis, tels que les proxy de transcodage de la vidéo, les passerelles sans fils qui lient les téléphones cellulaires avec le réseau Internet, les proxy d'adaptation d'images, etc. Une voie intéressante consiste à unifier les adaptations proposées au sein d'un même système. Néanmoins, la négociation des méthodes d'adaptation devient plus complexe. En effet, dans ce genre de situations, la négociation implique plusieurs entités distantes ce qui complique la communication entre ces entités. Une perspective est donc de définir une stratégie coopérative afin de rendre le système d'adaptation plus puissant tout en minimisant le coût d'échange de messages de négociation. L'architecture NAC proposée dans cette thèse constitue une bonne base de départ pour spécifier de telles stratégies de coopération entre entités d'adaptation.

Annexe A

Schéma UPS

Contenu

A.1 Les schémas UPS

Les schémas UPS décrivent les capacités et les préférences des entités qui peuvent intervenir dans la chaîne d'adaptation allant du serveur de contenu vers le terminal cible. Les profils sont analysés syntaxiquement et tout information pertinente par rapport au processus d'adaptation est prise en compte. UPS inclut six schémas :

1. Le schéma de profil de client
2. Le schéma de profil des ressources du client
3. Le schéma de profil d'instance de document
4. Le schéma de profil des ressources du contenu
5. Le schéma de profil des méthodes d'adaptation
6. Le schéma de profil du réseau

A.2 Exemple de Schéma : Le schéma profil de client

```
< ?xml version="1.0" ?>
<rdf :RDF xmlns :rdf="http ://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns :rdfs="http ://www.w3.org/2000/01/rdf-schema#"
xmlns :neg="http ://www.inrialpes.fr/opera/people/Tayeb.Lemlouma/NegotiationSchema/
ClientProfileSchema-03012002#">

<!– This is the RDF Schema for "The Client Profile". Defined in the context of
content negotiation in heterogeneous environments. Author : Tayeb LEMLOUMA,
January 2002. –>

<rdf :Description rdf :ID="ProfileComponent">
<rdf :type rdf :resource="http ://www.w3.org/2000/01/rdf-schema#Class"/>
<rdfs :subClassOf rdf :resource="http ://www.w3.org/2000/01/rdf-
schema#Resource"/>
```

<rdfs :comment>
A class that includes different components for the Client Profile description
</rdfs :comment>
</rdf :Description>

<!– Common properties –>

<!– Main Client Profile Components –>

<rdf :Description rdf :ID=”HardwarePlatform”>
<rdf :type rdf :resource=”http ://www.w3.org/2000/01/rdf-schema#Class”/>
<rdfs :subClassOf rdf :resource=”#ProfileComponent”/>
<rdfs :comment>
Describe the hardware platform of the client device
</rdfs :comment>
</rdf :Description>

<rdf :Description rdf :ID=”SoftwarePlatform”>
<rdf :type rdf :resource=”http ://www.w3.org/2000/01/rdf-schema#Class”/>
<rdfs :subClassOf rdf :resource=”#ProfileComponent”/>
<rdfs :comment>
Describe the software platform used by the client device : OS, etc.
</rdfs :comment>
</rdf :Description>

<rdf :Description rdf :ID=”BrowserUA”>
<rdf :type rdf :resource=”http ://www.w3.org/2000/01/rdf-schema#Class”/>
<rdfs :subClassOf rdf :resource=”#ProfileComponent”/>
<rdfs :comment>
Describe players used by the client device. This component plays a major role in the content negotiation and may refer to external profiles. Note that the protocol used to access to the network, and player-related network description may be added here.
</rdfs :comment>
</rdf :Description>

<!– HardwarePlatform Profile Component –>
<rdf :Description rdf :ID=”DeviceType”>
<rdf :type rdf :resource=”http ://www.w3.org/2000/01/rdf-schema#Property”/>
<rdfs :domain rdf :resource=”#HardwarePlatform”/>
<rdfs :comment>
The type of the used device Example : ”Pocket PC, PC, Laptop, phone, WAP phone”
</rdfs :comment>
</rdf :Description>
<rdf :Description rdf :ID=”DeviceName”>
<rdf :type rdf :resource=”http ://www.w3.org/2000/01/rdf-schema#Property”/>
<rdfs :domain rdf :resource=”#HardwarePlatform”/>
<rdfs :comment>
The name of the used device Example : ”iPAQ 3600”
</rdfs :comment>
</rdf :Description>

<rdf :Description rdf :ID=”DeviceConstructor”>
<rdf :type rdf :resource=”http ://www.w3.org/2000/01/rdf-schema#Property”/>
<rdfs :domain rdf :resource=”#HardwarePlatform”/>
<rdfs :comment>
The device constructor Example : ”MyCompagny computer corp.”
</rdfs :comment>
</rdf :Description>
<rdf :Description rdf :ID=”screen”>
<rdf :type rdf :resource=”http ://www.w3.org/2000/01/rdf-schema#Property”/>
<rdfs :domain rdf :resource=”#HardwarePlatform”/>
<rdfs :comment>
The screen dimension of the used device Example : ”30X23mm”
</rdfs :comment>
</rdf :Description>
<rdf :Description rdf :ID=”screenColor”>
<rdf :type rdf :resource=”http ://www.w3.org/2000/01/rdf-schema#Property”/>
<rdfs :domain rdf :resource=”#HardwarePlatform”/>
<rdfs :comment>
The color displying capability of the screen. Example : ”yes”, ”no”
</rdfs :comment>
</rdf :Description>
<rdf :Description rdf :ID=”display”>
<rdf :type rdf :resource=”http ://www.w3.org/2000/01/rdf-schema#Property”/>
<rdfs :domain rdf :resource=”#HardwarePlatform”/>
<rdfs :comment>
The display of the used device Example : ”101x52Pixels”
</rdfs :comment>
</rdf :Description>
<rdf :Description rdf :ID=”PixelStretch”>
<rdf :type rdf :resource=”http ://www.w3.org/2000/01/rdf-schema#Property”/>
<rdfs :domain rdf :resource=”#HardwarePlatform”/>
<rdfs :comment>
Gives the (height/width) pixels ratio, used in images design Example : ”1.24”, means that the pixels are 24</rdfs :comment>
</rdf :Description>
<rdf :Description rdf :ID=”row”>
<rdf :type rdf :resource=”http ://www.w3.org/2000/01/rdf-schema#Property”/>
<rdf :type rdf :resource=”http ://www.w3.org/2000/01/rdf-schema#Bag”/>
<rdfs :domain rdf :resource=”#HardwarePlatform”/>
<rdfs :comment>
Gives the number of the screen rows. Can be given as a set of couple (type,value) Example : ”Latin, 5” and ”Chinese, 3”
</rdfs :comment>
</rdf :Description>
<rdf :Description rdf :ID=”col”>
<rdf :type rdf :resource=”http ://www.w3.org/2000/01/rdf-schema#Property”/>
<rdf :type rdf :resource=”http ://www.w3.org/2000/01/rdf-schema#Bag”/>
<rdfs :domain rdf :resource=”#HardwarePlatform”/>
<rdfs :comment>
The number of the screen columns. Can be given as a set of couple (type,value)

Example : ”14”
</rdfs :comment>
</rdf :Description>
<rdf :Description rdf :ID=”RAMSize”>
<rdf :type rdf :resource=”http ://www.w3.org/2000/01/rdf-schema#Property”/>
<rdfs :domain rdf :resource=”#HardwarePlatform”/>
<rdfs :comment>
The size of the device’s RAM memory Example : ”32 Mo”
</rdfs :comment>
</rdf :Description>
<rdf :Description rdf :ID=”ROMSize”>
<rdf :type rdf :resource=”http ://www.w3.org/2000/01/rdf-schema#Property”/>
<rdfs :domain rdf :resource=”#HardwarePlatform”/>
<rdfs :comment>
The size of the device’s ROM memory Example : ”16 Mo”
</rdfs :comment>
</rdf :Description>

<!– SoftwarePlatform Profile Component –>

<rdf :Description rdf :ID=”PlatformName”>
<rdf :type rdf :resource=”http ://www.w3.org/2000/01/rdf-schema#Property”/>
<rdfs :domain rdf :resource=”#SoftwarePlatform”/>
<rdfs :comment>
The name of the Operating System used by the user agent Example : ”Windows Professionnel”
</rdfs :comment>
</rdf :Description>

<rdf :Description rdf :ID=”PlatformVersion”>
<rdf :type rdf :resource=”http ://www.w3.org/2000/01/rdf-schema#Property”/>
<rdfs :domain rdf :resource=”#SoftwarePlatform”/>
<rdfs :comment>
The version of the Operating System used by the user agent Example : ”8.0”
</rdfs :comment>
</rdf :Description>

<!– BrowserUA Profile Component –>

<rdf :Description rdf :ID=”UsedPlayerName”>
<rdf :type rdf :resource=”http ://www.w3.org/2000/01/rdf-schema#Property”/>
<rdfs :domain rdf :resource=”#BrowserUA”/>
<rdfs :comment>
The name of the player used by the client, at a well determined session. The client can use many players. Example : ”Nethix”
</rdfs :comment>
</rdf :Description>

<rdf :Description rdf :ID=”UsedPlayerVersion”>
<rdf :type rdf :resource=”http ://www.w3.org/2000/01/rdf-schema#Property”/>

<rdfs :domain rdf :resource=”#BrowserUA”/>
<rdfs :comment>
The version of the player. Example : ”1.1”
</rdfs :comment>
</rdf :Description>

<rdf :Description rdf :ID=”OnlySupportedResources”>
<rdf :type rdf :resource=”http ://www.w3.org/2000/01/rdf-schema#Property”/>
<rdf :type rdf :resource=”http ://www.w3.org/2000/01/rdf-schema#Bag”/>
<rdfs :domain rdf :resource=”#BrowserUA”/>
<rdfs :comment>
The only supported set of resources or services. Profiles of these last are denoted using links (the profile element) This set is given generally when the it is small and not already described by an existed schema. Example : ”WML documents, wbmp images”
</rdfs :comment>
</rdf :Description>

<rdf :Description rdf :ID=”PreferredSupportedResources”>
<rdf :type rdf :resource=”http ://www.w3.org/2000/01/rdf-schema#Property”/>
<rdf :type rdf :resource=”http ://www.w3.org/2000/01/rdf-schema#Seq”/>
<rdfs :domain rdf :resource=”#BrowserUA”/>
<rdfs :comment>
A list that gives preferred resources ordered with a priority level value. The range of the value is left to the negotiation strategy. Example : ”wbmp,2,gif,1”
</rdfs :comment>
</rdf :Description>

<rdf :Description rdf :ID=”NonSupportedResources”>
<rdf :type rdf :resource=”http ://www.w3.org/2000/01/rdf-schema#Property”/>
<rdf :type rdf :resource=”http ://www.w3.org/2000/01/rdf-schema#Bag”/>
<rdfs :domain rdf :resource=”#BrowserUA”/>
<rdfs :comment>
The list of the excluded resources. Note that if the non supported resource depends to a particular set (WML, HTML, etc.), it’s preferable to include it on the corresponding profile. Example : ”the non support of a resource from a predefined set of supported resources”
</rdfs :comment>
</rdf :Description>

</rdf :RDF>

Annexe B

Transformations de structures

Contenu

B.1 La DTD des document XML pour la Transformations XML vers LaTeX

```
<!--
LaTeX Document Type Definition.
Author: Tayeb Lemlouma. June 2001.
-->
<!DOCTYPE LaTeXDocument [

<!ELEMENT LaTeXDocument (Title, Author?, Date?, Heading?, Abstract?, Key-
words?, LaTeXBody)>
<!ATTLIST LaTeXDocument Type CDATA #REQUIRED>
<!ELEMENT Title (#PCDATA) >
<!ELEMENT Author (Name)+ >
<!ELEMENT Name (#PCDATA) >
<!ELEMENT Date (#PCDATA) >
<!ELEMENT Heading (Organisation*, Phone?, Fax?, EMail?) >

<!ELEMENT Organisation (Name+, Adress) >
<!ELEMENT Adress (#PCDATA) >
<!ELEMENT Phone (Number)+ >
<!ELEMENT Number (#PCDATA) >
<!ELEMENT Fax (Number)+ >
<!ELEMENT EMail (Mail)+ >

<!ELEMENT Abstract (#PCDATA) >
<!ELEMENT Key-words (#PCDATA) >

<!ELEMENT LaTeXBody (Section+, Bibliography?)>
```

```
<!ELEMENT Section (Title, Body)>
<!ELEMENT Body (#PCDATA | Section | Cite | Figure | Items | Enumerates)* >
<!ELEMENT Cite (#PCDATA)>
<!ELEMENT Figure (path, label, Title)>
<!ELEMENT path (#PCDATA)>
<!ELEMENT label (#PCDATA)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Items (Item)+>
<!ELEMENT Item (#PCDATA)>
<!ELEMENT Enumerates (Item)+>
<!ELEMENT Bibliography (Citation)+>
<!ELEMENT Citation (Label, Authors, Title, Description)>
<!ELEMENT Label (#PCDATA)>
<!ELEMENT Authors (#PCDATA)>
<!ELEMENT Description (#PCDATA, Date)>


]>
```

B.2 Transformations de structures basées sur XSLT : XML vers LaTeX

```
<?xml version='1.0'?> <xsl:stylesheet
xmlns:xsl='http://www.w3.org/1999/XSL/Transform' version='1.0'>
<xsl:output omit-xml-declaration="yes" />

<xsl:template match="LaTeXDocument">

<xsl:text>
%************************************************************
%*                                                          *
%*   http://www.inrialpes.fr/opera/people/Tayeb.Lemlouma/   *
%*                 LTIMEDIA/XSLT/XML2LaTeX.xsl              *
%*             Author:  Tayeb Lemlouma, june 2001           *
%************************************************************
\documentstyle[11pt, epsfig]{
</xsl:text><xsl:value-of select="@Type"/><xsl:text>
}

%==============================Title of the document
\title{\bf </xsl:text>

<xsl:apply-templates select="Title/text()"/>

<xsl:text>}

</xsl:text> <xsl:apply-templates select="Author"/> <xsl:text>
</xsl:text>

<!-- ======= Date ====== -->

<xsl:apply-templates select="Date"/>

<!-- ======= Default properties ====== -->
```

```
<xsl:text>

\textwidth 16.8cm
\textheight 21.5cm
\oddsidemargin -.25in
\evensidemargin -.25in
\topskip 0cm
\footskip 1cm
\footheight 0cm
\headheight -1cm</xsl:text>
<!-- ======= Document Begining ======
--> <xsl:text>

\begin{document}
\maketitle </xsl:text> <!-- ======= Heading ====== -->
<xsl:apply-templates select="Heading"/>

<!-- ======= Abstract & Key Words ====== -->

<xsl:text>

%==============================Abstract and key words
\begin{abstract}
</xsl:text>
<xsl:apply-templates select="Abstract/text()"/>
<xsl:text>
\begin{flushleft}{\bf Keywords:}
</xsl:text>

<xsl:apply-templates select="Key-words/text()"/>

<xsl:text>
\end{flushleft}
\end{abstract}

%==============================Document Body
</xsl:text>
<xsl:apply-templates select="LaTeXBody/Section"/>
<xsl:apply-templates select="LaTeXBody/Bibliograhy"/>
<xsl:text>
\end{document}
%==============================End of Document
</xsl:text> </xsl:template>


<!-- ======= Authors ====== -->
<xsl:template match="Author">
<xsl:for-each select="Name">
<xsl:if test="position()=1">
<xsl:text>
\author{</xsl:text></xsl:if>
<xsl:if test="not(position()=1)">
<xsl:text> and
</xsl:text></xsl:if>
```

```
<xsl:apply-templates select="text()"/>

</xsl:for-each> <xsl:text>}</xsl:text>
</xsl:template>

<!-- ======= Date ====== -->
<xsl:template match="Date">
<xsl:text>
\date{</xsl:text>

<xsl:apply-templates select="text()"/>

<xsl:text>}</xsl:text> </xsl:template>

<!-- ======= Heading ====== -->
<xsl:template match="Heading">
<!-- ==Organisation== -->
<xsl:text>\begin{center}</xsl:text>

<xsl:apply-templates select="Organisation/Name/text()"/>

<xsl:text> \\ </xsl:text>
<!-- ==Adress== -->
<xsl:apply-templates select="Organisation/Adress/text()"/>
<xsl:text> \\ </xsl:text>
<!-- ==Phone== -->
<xsl:for-each
select="Phone/Number">
<xsl:if test="position()=1">
<xsl:text>
Phone:
</xsl:text></xsl:if>
<xsl:if
test="not(position()=1)">
<xsl:text> / </xsl:text>
</xsl:if>
<xsl:value-of select="."/>
</xsl:for-each><xsl:text> \\
</xsl:text>
<!-- ==Fax== -->
<xsl:for-each select="Fax/Number">
<xsl:if test="position()=1">
<xsl:text> Fax: </xsl:text></xsl:if>
<xsl:if test="not(position()=1)">
<xsl:text> / </xsl:text></xsl:if>
<xsl:value-of select="."/>
</xsl:for-each><xsl:text> \\
</xsl:text>
<!-- ==Mail== -->
<xsl:for-each select="EMail/Mail">
<xsl:if test="position()=1"><xsl:text> E-mail:
</xsl:text></xsl:if> <xsl:if test="not(position()=1)"><xsl:text>,
</xsl:text></xsl:if>

<xsl:apply-templates select="text()"/>
```

```
</xsl:for-each><xsl:text> \\ </xsl:text> <xsl:text>
\end{center}</xsl:text>
</xsl:template>

<!-- ======= Body ====== -->

<!-- Sections Processing -->
<xsl:template match="Section">

<xsl:text>
%============================
</xsl:text>
<xsl:value-of select="Title"/>
<xsl:text> \</xsl:text> <xsl:if
test="count(ancestor::node())>3">
<xsl:call-template
name="SubGenerator">
<xsl:with-param name="count"
select="count(ancestor::node())"/>
</xsl:call-template> </xsl:if>
<xsl:text>section{</xsl:text> <xsl:value-of
select="Title"/><xsl:text>} </xsl:text>

<!--Text Process -->
<xsl:apply-templates select="Body/node()"/>

</xsl:template>
<!-- End of Sections Processing -->


<!-- SubGenerator: A recursive function that generates "sub"
strings--> <xsl:template name="SubGenerator"> <xsl:param
name="count"/> <xsl:if test="$count != 3">
<xsl:text>sub</xsl:text>
<xsl:call-template name="SubGenerator">
<xsl:with-param name="count" select="$count - 2"/>
<!-- 2 because it follows the following organization:
Section/Body/Section ... etc. -->
</xsl:call-template> </xsl:if>
</xsl:template>
<!-- End of SubGenerator-->

<!-- Text Process Template -->
<xsl:template match="text()">

<xsl:variable name="S">
<xsl:value-of select="normalize-space()"/>
</xsl:variable>

<xsl:if test="string-length($S)>0">

<xsl:call-template name="LaTeXChar">

<xsl:with-param name="i" select="string-length($S)"/>
<xsl:with-param name="l" select="string-length($S)"/>
```

```
</xsl:call-template> </xsl:if>

</xsl:template>
<!-- End of the Text Process Template -->


<!-- LaTeXChar: A recursive function that generates LaTeX special
characters -->

<xsl:template name="LaTeXChar"> <xsl:param name="i"/>
<xsl:param name="l"/>

<xsl:variable name="SS">

<xsl:value-of select="substring(normalize-space(),$l - $i + 1,1)"
/>

</xsl:variable>

<xsl:if test="$i > 0">

<xsl:choose>
 <xsl:when test="$SS = '{\'e}'">
 <xsl:text>\'{e}</xsl:text>
 </xsl:when>
 <xsl:when test="$SS = '{\^e}'">
 <xsl:text>\^{e}</xsl:text>
 </xsl:when>
 <xsl:when test="$SS = '{\`e}'">
 <xsl:text>\`{e}</xsl:text>
 </xsl:when>
 <xsl:when test="$SS = '{\"\i}'">
 <xsl:text>\"{\i}</xsl:text>
 </xsl:when>
 <xsl:when test="$SS = '{\^\i}'">
 <xsl:text>\^{i}</xsl:text>
 </xsl:when>
 <xsl:when test="$SS = '{\`a}'">
 <xsl:text>\`{a}</xsl:text>
 </xsl:when>
 <xsl:when test="$SS = '{\'a}'">
 <xsl:text>\'{a}</xsl:text>
 </xsl:when>
 <xsl:when test="$SS = '{\^a}'">
 <xsl:text>\^{a}</xsl:text>
 </xsl:when>
 <xsl:when test="$SS = '\c{c}'">
 <xsl:text>\c{c}</xsl:text>
 </xsl:when>
 <xsl:when test="$SS = '{\^o}'">
 <xsl:text>\^{o}</xsl:text>
 </xsl:when>
 <xsl:when test="$SS = '{\`u}'">
 <xsl:text>\`{u}</xsl:text>
 </xsl:when>
 <xsl:when test="$SS = '{\^u}'">
```

```
 <xsl:text>\^{u}</xsl:text>
 </xsl:when>
 <xsl:when test="$SS = '|'">
 <xsl:text>$|$</xsl:text>
 </xsl:when>
 <xsl:when test="$SS = '_'">
 <xsl:text>\_</xsl:text>
 </xsl:when>
 <xsl:otherwise><xsl:value-of select="$SS"/>

 </xsl:otherwise>
</xsl:choose>

<xsl:text></xsl:text>

<xsl:call-template name="LaTeXChar">

<xsl:with-param name="i" select="$i - 1"/>

<xsl:with-param name="l" select="$l"/>

</xsl:call-template> </xsl:if>

</xsl:template>
<!-- End of LaTeXChar template -->



<!-- Bibliography Processing -->

<xsl:template match="Bibliograhy">

<xsl:text>

%==============================Biblioraphy
\begin{thebibliography}{99}</xsl:text>
<xsl:apply-templates select="Citation"/>
<xsl:text>
\end{thebibliography}
</xsl:text> </xsl:template>

<!-- End of Bibliography Processing -->

<!-- Citation Processing -->

<xsl:template match="Citation"> <xsl:text>
\bibitem{</xsl:text>


<xsl:value-of select="Label"/> <xsl:text>} </xsl:text>
<xsl:value-of select="Authors"/> <xsl:text>, \textit{ </xsl:text>
<xsl:value-of select="Title"/> <xsl:text> }, </xsl:text>
<xsl:value-of select="normalize-space(Description/text())"/>
<xsl:text>, </xsl:text> <xsl:value-of select="Description/Date"/>
<xsl:text>.</xsl:text>
```

```
</xsl:template>

<!-- Enumerates Process -->

<xsl:template match="Enumerates"> <xsl:text>
\begin{enumerate}</xsl:text>
<xsl:for-each select="Item"> <xsl:text> \item </xsl:text>
<xsl:value-of select="."/> </xsl:for-each> <xsl:text>
\end{enumerate}
</xsl:text> </xsl:template>

<!--Enumerates Process -->

<!-- Citations Process -->

<xsl:template match="Cite"> <xsl:text>
\cite{</xsl:text><xsl:value-of select="."/><xsl:text>} </xsl:text>
</xsl:template> <!--Citations Process -->

<!-- Figures Process (suppose that we use an eps format which is
commonly done) -->

<xsl:template match="Figure">
<xsl:text>
\begin{center}
\mbox{\epsfig{file=</xsl:text> <xsl:value-of select="path"/>
<xsl:text>}}
\end{center}
\begin{center}{\bf </xsl:text>
<xsl:value-of select="label"/> <xsl:text>: </xsl:text>
<xsl:value-of select="Title"/> <xsl:text>}\end{center} </xsl:text>
</xsl:template>

<!--Figuress Process-->

<!-- Items Process -->

<xsl:template match="Items"> <xsl:text>
\begin{itemize}</xsl:text>
<xsl:for-each select="Item"> <xsl:text> \item </xsl:text>
<xsl:value-of select="."/> </xsl:for-each> <xsl:text>
\end{itemize}
</xsl:text> </xsl:template>

<!--Items Process -->

</xsl:stylesheet>

<!-- End of the StyleSheet -->
```

B.3 Feuille de style générique pour le filtrage des documents

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

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
<!--
    Author: Tayeb Lemlouma
    January 2002(c).
-->
<xsl:template match="/">
<xsl:element name="xsl:stylesheet" namespace="http://www.w3.org/1999/XSL/Transform">
<xsl:attribute  name="version">1.0</xsl:attribute>
<xsl:text>
</xsl:text>
<xsl:comment>
    Services (SMIL, etc.) automatic adaptation.
    Author: Tayeb Lemlouma
    Email adress:Tayeb.Lemlouma@inrialpes.fr
    January 2002(c).
</xsl:comment>
<!--
 ======================================================================
 ==                          Copie du contenu                        ==
 ======================================================================
-->
<xsl:text>
</xsl:text>
<xsl:element name="xsl:output">
<xsl:attribute name="omit-xml-declaration">yes</xsl:attribute>
</xsl:element>
<xsl:text>
</xsl:text>
<xsl:element name="xsl:template">
<xsl:attribute name="match">*</xsl:attribute>
<xsl:text>
</xsl:text>
<xsl:element name="xsl:copy">
<xsl:text>
</xsl:text>
 <xsl:element name="xsl:apply-templates">
  <xsl:attribute name="select">@*</xsl:attribute>
 </xsl:element> <!-- xsl;apply-templates select @* -->
<xsl:text>
</xsl:text>
 <xsl:element name="xsl:apply-templates"></xsl:element>
<xsl:text>
</xsl:text>
</xsl:element> <!-- xsl:copy -->
<xsl:text>
</xsl:text>
</xsl:element><!-- xsl:template match * -->
<xsl:text>
</xsl:text>
```

```
<xsl:element name="xsl:template">
<xsl:attribute name="match">@*</xsl:attribute>
<xsl:text>
</xsl:text>
<xsl:element name="xsl:copy">
</xsl:element><!-- copy -->
<xsl:text>
</xsl:text>
</xsl:element><!-- template match @* -->
<xsl:text>
</xsl:text>
<!--
 ========================================================================
 ==                            Fin de copie                            ==
 ========================================================================
-->
<!--
 ========================================================================
 ==                    Adapter selon le profil
 ========================================================================
-->
<xsl:apply-templates select="ClientProfile/Service/ServiceComponent"/>
<!-- [@support='no'] -->

<xsl:text>
</xsl:text>
</xsl:element>
<!-- xsl: stylesheet-->
</xsl:template>
<xsl:template match="ServiceComponent">
<!--
[@support='no'] -->
    <xsl:variable name="E">
    <xsl:value-of select="@tagName" />
    </xsl:variable>
    <!-- Category: <xsl:value-of select="$E" /> -->
    <xsl:choose>
        <xsl:when test="@support = 'no'">
        <!--
        ========================================================================
        ==                Filtrer un element et son contenu                   ==
        ========================================================================
        -->
        <xsl:text>
</xsl:text>
        <xsl:element name="xsl:template">
        <xsl:attribute name="match"><xsl:value-of select="$E"/></xsl:attribute>
        <xsl:text>
</xsl:text>
        </xsl:element>
        </xsl:when>

        <xsl:when test="@support = 'noTag'">
        <!--
        ========================================================================
        ==                          Filtrer un element                        ==
```

```
        ========================================================================
        -->
        <xsl:text>
</xsl:text>
        <xsl:element name="xsl:template">
        <xsl:attribute name="match"><xsl:value-of select="$E"/></xsl:attribute>
        <xsl:text>
</xsl:text>
                <xsl:element name="xsl:apply-templates"></xsl:element>
                <xsl:text>
</xsl:text>
                </xsl:element>
        </xsl:when>
        <xsl:otherwise></xsl:otherwise>
    </xsl:choose>
</xsl:template>
</xsl:stylesheet>
```

Annexe C

Le Repository des profils UPS

Contenu

C.1 Le repository des profils

Le repository utilisé par NAC comporte cent dix huit profils de terminaux [72] écrits en UPS [77] et incluent une multitude de terminaux récents. Dans ce qui suit nous donnons deux exemples de profils du repository. Les profils correspondent à deux terminaux de types : mobile modèle SE-T310R201 et mobile modèle P-GD88 respectivement.

C.1.1 Profil UPS (T310R201)

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:prf="http://www.wapforum.org/profiles/UAPROF/ccppschema-20010330#"
xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:ccpp="http://www.w3.org/2000/07/04-ccpp#"
xmlns:neg="http://www.inrialpes.fr/opera/people/Tayeb.Lemlouma/
NegotiationSchema/ClientProfileSchema-03012002#">

<!--Universal Profiling Schema(UPS),
Tayeb.Lemlouma@inrialpes.fr,
INRIA, France, 2003.-->

<rdf:Description rdf:ID="ClientProfile">
<ccpp:component>
<rdf:Description rdf:ID="HardwarePlatform">
<neg:DeviceName>T310R201</neg:DeviceName>
<neg:DisplayPixel>101x80</neg:DisplayPixel>
<neg:DisplayChar>15x6</neg:DisplayChar>
<neg:ColorBit>8</neg:ColorBit>
<neg:DeviceVendor>Sony Ericsson Mobile Communications</neg:DeviceVendor>
<neg:PixelStretch>1/1</neg:PixelStretch>
</rdf:Description>
</ccpp:component>
<ccpp:component>
```

```
<rdf:Description rdf:ID="SoftwarePlatform">
<neg:ProtocolSecurity>WTLS class 1/2/3/signText</neg:ProtocolSecurity>
<neg:SystemCommunication>
<rdf:Bag>
<rdf:li>GPRS</rdf:li>
<rdf:li>CSD</rdf:li>
</rdf:Bag>
</neg:SystemCommunication>
</rdf:Description>
</ccpp:component>
<ccpp:component>
<rdf:Description rdf:ID="BrowserUA">
<neg:PlayerName>Sony Ericsson</neg:PlayerName>
<neg:TransferEncoding>
<rdf:Bag>
<rdf:li>base64</rdf:li>
</rdf:Bag>
</neg:TransferEncoding>
<neg:OnlySupportedResources>
<rdf:Bag>
<rdf:li>text</rdf:li>
<rdf:li>audio</rdf:li>
<rdf:li>wml</rdf:li>
<rdf:li>mms</rdf:li>
</rdf:Bag>
</neg:OnlySupportedResources>
<neg:OnlySupportedMimeTypes>
<rdf:Bag>
<rdf:li>application/vnd.wap.wmlc</rdf:li>
<rdf:li>application/vnd.wap.wbxml</rdf:li>
<rdf:li>application/vnd.wap.wmlscriptc</rdf:li>
<rdf:li>application/vnd.wap.multipart.mixed</rdf:li>
<rdf:li>text/x-vCard</rdf:li>
<rdf:li>text/x-vCalendar</rdf:li>
<rdf:li>text/x-vMel</rdf:li>
<rdf:li>text/x-eMelody</rdf:li>
<rdf:li>image/vnd.wap.wbmp</rdf:li>
<rdf:li>text/x-iMelody</rdf:li>
<rdf:li>image/gif</rdf:li>
<rdf:li>image/jpeg</rdf:li>
<rdf:li>application/vnd.wap.sic</rdf:li>
<rdf:li>application/vnd.wap.slc</rdf:li>
<rdf:li>application/vnd.wap.coc</rdf:li>
<rdf:li>application/vnd.wap.sia</rdf:li>
<rdf:li>application/vnd.wap.wtls-ca-certificate</rdf:li>
<rdf:li>application/vnd.wap.xhtml+xml</rdf:li>
<rdf:li>application/vnd.eri.thm</rdf:li>
<rdf:li>application/xhtml+xml</rdf:li>
<rdf:li>application/x-wap-prov.browser-settings</rdf:li>
<rdf:li>application/vnd.wap.connectivity-wbxml</rdf:li>
<rdf:li>application/vnd.oma.drm.message</rdf:li>
<rdf:li>application/vnd.mophun.application</rdf:li>
<rdf:li>application/vnd.mophun.certificate</rdf:li>
<rdf:li>application/smil</rdf:li>
<rdf:li>application/x-sms</rdf:li>
<rdf:li>audio/amr</rdf:li>
```

```
<rdf:li>audio/midi</rdf:li>
<rdf:li>audio/mid</rdf:li>
<rdf:li>audio/x-amr</rdf:li>
<rdf:li>audio/x-midi</rdf:li>
<rdf:li>text/plain</rdf:li>
<rdf:li>application/vnd.wap.mms-message</rdf:li>
<rdf:li>application/vnd.sem.mms.protected</rdf:li>
<rdf:li>application/vnd.wap.multipart.alternative</rdf:li>
<rdf:li>application/vnd.wap.multipart.related</rdf:li>
<rdf:li>application/vnd.3gpp.sms</rdf:li>
</rdf:Bag>
</neg:OnlySupportedMimeTypes>
<neg:OnlySupportedCharset>
<rdf:Bag>
<rdf:li>US-ASCII</rdf:li>
<rdf:li>ISO-8859-1</rdf:li>
<rdf:li>UTF-8</rdf:li>
<rdf:li>ISO-10646-UCS-2</rdf:li>
<rdf:li>ISO-8859-2</rdf:li>
<rdf:li>UTF-7</rdf:li>
<rdf:li>UTF-16</rdf:li>
<rdf:li>KOI8-R</rdf:li>
<rdf:li>windows-1251</rdf:li>
</rdf:Bag>
</neg:OnlySupportedCharset>
</rdf:Description>
</ccpp:component>
<ccpp:component>
<rdf:Description rdf:about="wml">
<neg:ResourceType>wml</neg:ResourceType>
<neg:CardMaxSize>3000</neg:CardMaxSize>
<neg:Version rdf:ID="wml">
<rdf:Bag>
<rdf:li>1.2.1/June 2000</rdf:li>
<rdf:li>1.1</rdf:li>
</rdf:Bag>
</neg:Version>
<neg:SupportedFunctionalities>
<rdf:Bag>
<rdf:li>tables</rdf:li>
</rdf:Bag>
</neg:SupportedFunctionalities>
<neg:NonSupportedFunctionalities>
<rdf:Bag>
<rdf:li>frames</rdf:li>
</rdf:Bag>
</neg:NonSupportedFunctionalities>
<neg:Version rdf:ID="wmlScript">
<rdf:Bag>
<rdf:li>1.2.1/June 2000</rdf:li>
<rdf:li>1.1</rdf:li>
</rdf:Bag>
</neg:Version>
<neg:SupportedScriptLibraries>
<rdf:Bag>
<rdf:li>Lang</rdf:li>
```

```
<rdf:li>Float</rdf:li>
<rdf:li>String</rdf:li>
<rdf:li>URL</rdf:li>
<rdf:li>WMLBrowser</rdf:li>
<rdf:li>Dialogs</rdf:li>
</rdf:Bag>
</neg:SupportedScriptLibraries>
<neg:WapClass>C</neg:WapClass>
<neg:WapPushMaxSize>1500</neg:WapPushMaxSize>
<neg:SupportedWtaiLibraries>
<rdf:Bag>
<rdf:li>WTA.Public.makeCall</rdf:li>
<rdf:li>WTA.Public.sendDTMF</rdf:li>
<rdf:li>WTA.Public.addPBEntry</rdf:li>
</rdf:Bag>
</neg:SupportedWtaiLibraries>
</rdf:Description>
</ccpp:component>
<ccpp:component>
<rdf:Description rdf:about="mms">
<neg:MmsImageMaxResolution>160x120</neg:MmsImageMaxResolution>
<neg:Version rdf:ID="mms">
<rdf:Bag>
<rdf:li>1.0</rdf:li>
</rdf:Bag>
</neg:Version>
</rdf:Description>
</ccpp:component>
</rdf:Description>
</rdf:RDF>
```

C.1.2 Profil UPS (GD88)

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:prf="http://www.wapforum.org/profiles/UAPROF/ccppschema-20020710#"
xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:ccpp="http://www.w3.org/2000/07/04-ccpp#"
xmlns:neg="http://www.inrialpes.fr/opera/people/Tayeb.Lemlouma/
NegotiationSchema/ClientProfileSchema-03012002#">

<!--Universal Profiling Schema(UPS),
Tayeb.Lemlouma@inrialpes.fr,
INRIA, France, 2003.-->

<rdf:Description rdf:ID="ClientProfile">
<ccpp:component>
<rdf:Description rdf:ID="HardwarePlatform">
<neg:DeviceName>GD88</neg:DeviceName>
<neg:DisplayPixel>132x176</neg:DisplayPixel>
<neg:DisplayChar>16x8</neg:DisplayChar>
<neg:ColorBit>16</neg:ColorBit>
<neg:DeviceVendor>Panasonic</neg:DeviceVendor>
<neg:PixelStretch>1/1</neg:PixelStretch>
</rdf:Description>
</ccpp:component>
```

```
<ccpp:component>
<rdf:Description rdf:ID="SoftwarePlatform">
<neg:ProtocolSecurity>
<rdf:Bag>
<rdf:li>WTLS-1</rdf:li>
<rdf:li>WTLS-2</rdf:li>
</rdf:Bag>
</neg:ProtocolSecurity>
<neg:SystemCommunication>
<rdf:Bag>
<rdf:li>OneWaySMS</rdf:li>
<rdf:li>CSD</rdf:li>
<rdf:li>GPRS</rdf:li>
</rdf:Bag>
</neg:SystemCommunication>
</rdf:Description>
</ccpp:component>
<ccpp:component>
<rdf:Description rdf:ID="BrowserUA">
<neg:PlayerName>Panasonic</neg:PlayerName>
<neg:TransferEncoding>
<rdf:Bag>
<rdf:li>base64</rdf:li>
</rdf:Bag>
</neg:TransferEncoding>
<neg:OnlySupportedResources>
<rdf:Bag>
<rdf:li>text</rdf:li>
<rdf:li>audio</rdf:li>
<rdf:li>wml</rdf:li>
<rdf:li>mms</rdf:li>
</rdf:Bag>
</neg:OnlySupportedResources>
<neg:OnlySupportedMimeTypes>
<rdf:Bag>
<rdf:li>text/plain</rdf:li>
<rdf:li>text/css</rdf:li>
<rdf:li>text/x-server-parsed-html</rdf:li>
<rdf:li>text/vnd.wap.wml</rdf:li>
<rdf:li>text/vnd.wap.wmlscript</rdf:li>
<rdf:li>image/gif</rdf:li>
<rdf:li>image/png</rdf:li>
<rdf:li>image/jpeg</rdf:li>
<rdf:li>image/vnd.wap.wbmp</rdf:li>
<rdf:li>image/bmp</rdf:li>
<rdf:li>audio/midi</rdf:li>
<rdf:li>audio/mid</rdf:li>
<rdf:li>audio/imelody</rdf:li>
<rdf:li>audio/amr</rdf:li>
<rdf:li>audio/sp-midi</rdf:li>
<rdf:li>application/x-pmd</rdf:li>
<rdf:li>application/vnd.wap.wmlc</rdf:li>
<rdf:li>application/vnd.wap.wbxml</rdf:li>
<rdf:li>application/vnd.wap.wmlscriptc</rdf:li>
<rdf:li>application/vnd.wap.multipart.mixed</rdf:li>
<rdf:li>application/vnd.wap.wtls-ca-certificate</rdf:li>
```

```
<rdf:li>application/wml+xml</rdf:li>
<rdf:li>application/xhtml+xml</rdf:li>
<rdf:li>application/smil</rdf:li>
<rdf:li>multipart/mixed</rdf:li>
<rdf:li>application/vnd.wap.multipart.related</rdf:li>
</rdf:Bag>
</neg:OnlySupportedMimeTypes>
<neg:OnlySupportedCharset>
<rdf:Bag>
<rdf:li/>
<rdf:li>ISO-8859-1</rdf:li>
<rdf:li>UTF-8</rdf:li>
<rdf:li>US-ASCII</rdf:li>
</rdf:Bag>
</neg:OnlySupportedCharset>
</rdf:Description>
</ccpp:component>
<ccpp:component>
<rdf:Description rdf:about="wml">
<neg:ResourceType>wml</neg:ResourceType>
<neg:CardMaxSize>12000</neg:CardMaxSize>
<neg:Version rdf:ID="wml">
<rdf:Bag/>
</neg:Version>
<neg:Version rdf:ID="wap">
<rdf:Bag>
<rdf:li>1.2.1</rdf:li>
</rdf:Bag>
</neg:Version>
<neg:SupportedFunctionalities>
<rdf:Bag>
<rdf:li>tables</rdf:li>
</rdf:Bag>
</neg:SupportedFunctionalities>
<neg:NonSupportedFunctionalities>
<rdf:Bag>
<rdf:li>frames</rdf:li>
</rdf:Bag>
</neg:NonSupportedFunctionalities>
<neg:Version rdf:ID="wmlScript">
<rdf:Bag/>
</neg:Version>
<neg:SupportedScriptLibraries>
<rdf:Bag>
<rdf:li>Lang</rdf:li>
<rdf:li>Float</rdf:li>
<rdf:li>String</rdf:li>
<rdf:li>URL</rdf:li>
<rdf:li>WMLBrowser</rdf:li>
<rdf:li>Dialogs</rdf:li>
</rdf:Bag>
</neg:SupportedScriptLibraries>
<neg:WapClass>C</neg:WapClass>
<neg:WapPushMaxSize>12000</neg:WapPushMaxSize>
<neg:SupportedWtaiLibraries>
<rdf:Bag>
```

```
<rdf:li>WTA.Public.makeCall</rdf:li>
<rdf:li>WTA.Public.sendDTMF</rdf:li>
<rdf:li>WTA.Public.addPBEntry</rdf:li>
</rdf:Bag>
</neg:SupportedWtaiLibraries>
</rdf:Description>
</ccpp:component>
<ccpp:component>
<rdf:Description rdf:about="mms">
<neg:MmsImageMaxResolution>132x134</neg:MmsImageMaxResolution>
<neg:Version rdf:ID="mms">
<rdf:Bag>
<rdf:li>1.0</rdf:li>
</rdf:Bag>
</neg:Version>
</rdf:Description>
</ccpp:component>
</rdf:Description>
</rdf:RDF>
```

Annexe D

Transformation de vocabulaire de description

Contenu

D.1 Transformation UAProf vers UPS

```
<?xml version='1.0'?> <!-- UAPROF to UPS Author: Tayeb LEMLOUMA
E-Mail:Tayeb.Lemlouma@inrialpes.fr June 2003 -->

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0" xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:ccpp="http://www.w3.org/2000/07/04-ccpp#"
xmlns:neg="http://www.inrialpes.fr/opera/people/Tayeb.Lemlouma/NegotiationSchema/
ClientProfileSchema-03012002#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:prf="http://www.wapforum.org/profiles/UAPROF/ccppschema-20010430#"
>

<xsl:output method="xml" indent="yes"/>

<xsl:template match="/">

    <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:neg="http://www.inrialpes.fr/opera/people/Tayeb.Lemlouma/NegotiationSchema/
    ClientProfileSchema-03012002#"
    xmlns:ccpp="http://www.w3.org/2000/07/04-ccpp#">
        <xsl:apply-templates select="rdf:RDF/rdf:Description"/>
    </rdf:RDF>

</xsl:template>

<xsl:template match="rdf:RDF/rdf:Description">
    <xsl:comment >Universal Profiling Schema(UPS), Tayeb.Lemlouma@inrialpes.fr,
    INRIA, France, 2003.</xsl:comment>
    <rdf:Description rdf:ID="ClientProfile">

            <ccpp:component>
```

```
    <rdf:Description rdf:ID="HardwarePlatform">
        <xsl:apply-templates select="prf:component/rdf:Description
        [@rdf:ID='HardwarePlatform']/prf:Model"/>
        <xsl:apply-templates select="prf:component/rdf:Description
        [@rdf:ID='HardwarePlatform']/prf:ScreenSize"/>
        <xsl:apply-templates select="prf:component/rdf:Description
        [@rdf:ID='HardwarePlatform']/prf:ScreenSizeChar"/>
        <xsl:apply-templates select="prf:component/rdf:Description
        [@rdf:ID='HardwarePlatform']/prf:BitsPerPixel"/>
        <xsl:apply-templates select="prf:component/rdf:Description
        [@rdf:ID='HardwarePlatform']/prf:Vendor"/>
        <xsl:apply-templates select="prf:component/rdf:Description
        [@rdf:ID='HardwarePlatform']/prf:CPU"/>
        <xsl:apply-templates select="prf:component/rdf:Description
        [@rdf:ID='HardwarePlatform']/prf:PixelAspectRatio"/>
    </rdf:Description>
</ccpp:component>

<ccpp:component>
    <rdf:Description rdf:ID="SoftwarePlatform">
        <xsl:apply-templates select="prf:component/rdf:Description
        [@rdf:ID='SoftwarePlatform']/prf:OSName"/>
        <xsl:apply-templates select="prf:component/rdf:Description
        [@rdf:ID='SoftwarePlatform']/prf:OSVersion"/>
        <xsl:apply-templates select="prf:component/rdf:Description
        [@rdf:ID='SoftwarePlatform']/prf:OSVendor"/>
        <xsl:apply-templates select="prf:component/rdf:Description
        [@rdf:ID='SoftwarePlatform']/prf:RecipientAppAgent"/>
        <xsl:apply-templates select="prf:component/rdf:Description
        [@rdf:ID='NetworkCharacteristics']/prf:SecuritySupport"/>
        <xsl:apply-templates select="prf:component/rdf:Description
        [@rdf:ID='NetworkCharacteristics']/prf:SupportedBearers"/>
        <xsl:apply-templates select="prf:component/rdf:Description
        [@rdf:ID='NetworkCharacteristics']/prf:SupportedBearer"/>
        </rdf:Description>
</ccpp:component>

<ccpp:component>
    <rdf:Description rdf:ID="BrowserUA">
        <xsl:apply-templates select="prf:component/rdf:Description
        [@rdf:ID='BrowserUA']/prf:BrowserName"/>
        <xsl:apply-templates select="prf:component/rdf:Description
        [@rdf:ID='BrowserUA']/prf:BrowserVersion"/>
        <xsl:apply-templates select="prf:component/rdf:Description
        [@rdf:ID='SoftwarePlatform']/prf:CcppAccept-Language"/>
        <xsl:apply-templates select="prf:component/rdf:Description
        [@rdf:ID='BrowserUA']/prf:CcppAccept-Language"/>
        <xsl:apply-templates select="prf:component/rdf:Description
        [@rdf:ID='BrowserUA']/prf:CcppAccept-Encoding"/>
        <xsl:apply-templates select="prf:component/rdf:Description
        [@rdf:ID='SoftwarePlatform']/prf:CcppAccept-Encoding"/>
        <neg:OnlySupportedResources>
            <rdf:Bag>
                <xsl:apply-templates select="prf:component/
                rdf:Description
                [@rdf:ID='HardwarePlatform']/prf:TextInputCapable
```

```
                [normalize-space(.)='Yes']"/>
                <xsl:apply-templates select="prf:component/
                rdf:Description
                [@rdf:ID='HardwarePlatform']/prf:SoundOutputCapable
                [normalize-space(.)='Yes']"/>
                <xsl:apply-templates select="prf:component/
                rdf:Description
                [@rdf:ID='SoftwarePlatform']/prf:JavaEnabled
                [normalize-space(.)='Yes']"/>
                <xsl:apply-templates select="prf:component/
                rdf:Description
                [@rdf:ID='BrowserUA']/prf:XhtmlVersion"/>
                <xsl:for-each select="prf:component/
                rdf:Description
                [@rdf:ID='WapCharacteristics']">
                    <rdf:li>wml</rdf:li>
                </xsl:for-each>
                <xsl:for-each select="prf:component/
                rdf:Description
                [@rdf:ID='MMSCharacteristics']">
                    <rdf:li>mms</rdf:li>
                </xsl:for-each>
                <xsl:for-each select="prf:component/
                rdf:Description
                [@rdf:ID='MmsCharacteristics']">
                    <rdf:li>mms</rdf:li>
                </xsl:for-each>

            </rdf:Bag>
        </neg:OnlySupportedResources>
        <neg:OnlySupportedMimeTypes>
            <rdf:Bag>
            <xsl:variable name="B">
                <xsl:value-of select="normalize-space(prf:component/
                rdf:Description[@rdf:ID='BrowserUA']/prf:CcppAccept/
                rdf:Bag/rdf:li[position()=1])"/>
            </xsl:variable>
            <xsl:for-each select="prf:component/rdf:Description
            [@rdf:ID='BrowserUA']/prf:CcppAccept/rdf:Bag/rdf:li">
                <rdf:li><xsl:value-of select="."/></rdf:li>
            </xsl:for-each>
            <!--
                Some uaprof profiles include the MIME Types in the
                software platform
            -->
            <xsl:for-each select="prf:component/rdf:Description
            [@rdf:ID='SoftwarePlatform']/prf:CcppAccept/rdf:Bag/rdf:li">
                <rdf:li><xsl:value-of select="."/></rdf:li>
            </xsl:for-each>

            <!--
                Insert only non already inserted charset (from BowserUA or
                SoftwarePlatform component) to avoid repetitions of elements
                This can cause an execution delay due to the number of
                iterations that equanls to:
                cardinal_of(prf:CcppAccept) X cardinal_of(prf:MmsCcppAccept)
```

```
-->
<xsl:choose>
<xsl:when test="$B!=''">
<xsl:for-each select="prf:component/rdf:Description
[@rdf:ID='MMSCharacteristics']/prf:MmsCcppAccept/rdf:Bag/rdf:li">
    <xsl:variable name="C">
        <xsl:value-of select="."/>
    </xsl:variable>
    <xsl:variable name="CFound">
        <xsl:for-each select="../../../../../
        prf:component/rdf:Description
        [@rdf:ID='BrowserUA']/prf:CcppAccept/rdf:Bag/rdf:li">
            <xsl:variable name="C2">
                <xsl:value-of select="."/>
            </xsl:variable>
            <xsl:if test="$C = $C2">
            The element is already inserted</xsl:if>
        </xsl:for-each>
    </xsl:variable>
    <xsl:if test="$CFound = ''">
        <rdf:li><xsl:value-of select="."/></rdf:li>
    </xsl:if>
</xsl:for-each>
<xsl:for-each select="prf:component/rdf:Description
[@rdf:ID='MmsCharacteristics']/prf:MmsCcppAccept/rdf:Bag/rdf:li">
    <xsl:variable name="C">
        <xsl:value-of select="."/>
    </xsl:variable>
    <xsl:variable name="CFound">
        <xsl:for-each select="../../../../../
        prf:component/rdf:Description
        [@rdf:ID='BrowserUA']/prf:CcppAccept/rdf:Bag/rdf:li">
            <xsl:variable name="C2">
                <xsl:value-of select="."/>
            </xsl:variable>
            <xsl:if test="$C = $C2">
            The element is already inserted</xsl:if>
        </xsl:for-each>
    </xsl:variable>
    <xsl:if test="$CFound = ''">
        <rdf:li><xsl:value-of select="."/></rdf:li>
    </xsl:if>
</xsl:for-each>
</xsl:when>
<xsl:otherwise>
<xsl:for-each select="prf:component/rdf:Description
[@rdf:ID='MMSCharacteristics']/prf:MmsCcppAccept/rdf:Bag/rdf:li">
    <xsl:variable name="C">
        <xsl:value-of select="."/>
    </xsl:variable>
    <xsl:variable name="CFound">
        <xsl:for-each select="../../../../../
        prf:component/rdf:Description
        [@rdf:ID='SoftwarePlatform']/prf:CcppAccept/rdf:Bag/rdf:li">
            <xsl:variable name="C2">
                <xsl:value-of select="."/>
```

```
                    </xsl:variable>
                    <xsl:if test="$C = $C2">
                    The element is already inserted</xsl:if>
                </xsl:for-each>
            </xsl:variable>
            <xsl:if test="$CFound = ''">
                <rdf:li><xsl:value-of select="."/></rdf:li>
            </xsl:if>
        </xsl:for-each>
        <xsl:for-each select="prf:component/rdf:Description
        [@rdf:ID='MmsCharacteristics']/prf:MmsCcppAccept/rdf:Bag/rdf:li">
            <xsl:variable name="C">
                <xsl:value-of select="."/>
            </xsl:variable>
            <xsl:variable name="CFound">
                <xsl:for-each select="../../../../../
                prf:component/rdf:Description
                [@rdf:ID='SoftwarePlatform']/prf:CcppAccept/rdf:Bag/rdf:li">
                    <xsl:variable name="C2">
                        <xsl:value-of select="."/>
                    </xsl:variable>
                    <xsl:if test="$C = $C2">
                    The element is already inserted</xsl:if>
                </xsl:for-each>
            </xsl:variable>
            <xsl:if test="$CFound = ''">
                <rdf:li><xsl:value-of select="."/></rdf:li>
            </xsl:if>
        </xsl:for-each>
        </xsl:otherwise>

        </xsl:choose>
        </rdf:Bag>
    </neg:OnlySupportedMimeTypes>

    <!-- Include the OnlySupportedCharset only when it is not empty-->

        <xsl:variable name="E1">
            <xsl:value-of select="normalize-space
            (prf:component/rdf:Description
            [@rdf:ID='BrowserUA']/prf:CcppAccept-Charset/
            rdf:Bag/rdf:li[position()=1])"/>
        </xsl:variable>
        <xsl:variable name="E2">
            <xsl:value-of select="normalize-space
            (prf:component/rdf:Description
            [@rdf:ID='BrowserUA']/prf:CcppAccept-Charset)"/>
        </xsl:variable>
        <xsl:variable name="E3">
            <xsl:value-of select="normalize-space
            (prf:component/rdf:Description
            [@rdf:ID='MMSCharacteristics']/
            prf:MmsCcppAccept-Charset/rdf:Bag/
            rdf:li[position()=1])"/>
        </xsl:variable>
        <xsl:variable name="E4">
```

```
                <xsl:value-of select="normalize-space
                (prf:component/rdf:Description
                [@rdf:ID='MmsCharacteristics']/
                prf:MmsCcppAccept-Charset/rdf:Bag/
                rdf:li[position()=1])"/>
            </xsl:variable>
            <xsl:variable name="E5">
                <xsl:value-of select="normalize-space
                (prf:component/rdf:Description
                [@rdf:ID='SoftwarePlatform']/
                prf:CcppAccept-Charset/rdf:Bag/
                rdf:li[position()=1])"/>
            </xsl:variable>

        <xsl:if test="$E1!='' or $E2!='' or $E3!='' or $E4!='' or $E5!=''">

        <neg:OnlySupportedCharset>
            <rdf:Bag>

            <!-- Some uaprof versions give this as a simple element or
            a Bag element, UPS consider this always as a rdf:Bag-->
            <xsl:variable name="W">
                <xsl:value-of select="normalize-space
                (prf:component/rdf:Description
                [@rdf:ID='BrowserUA']/prf:CcppAccept-Charset/
                rdf:Bag/rdf:li[position()=1])"/>
            </xsl:variable>
            <xsl:variable name="W2">
                <xsl:value-of select="normalize-space
                (prf:component/rdf:Description
                [@rdf:ID='BrowserUA']/prf:CcppAccept-Charset)"/>
            </xsl:variable>

            <xsl:choose>
                <xsl:when test="$W!=''">
                    <xsl:for-each select="prf:component/rdf:Description
                    [@rdf:ID='BrowserUA']/
                    prf:CcppAccept-Charset/rdf:Bag/rdf:li">
                        <rdf:li><xsl:value-of select="."/></rdf:li>
                    </xsl:for-each>
                </xsl:when>
                <xsl:when test="$W2!=''">
                    <rdf:li><xsl:value-of select="prf:component/rdf:Description
                    [@rdf:ID='BrowserUA']/prf:CcppAccept-Charset"/></rdf:li>
                </xsl:when>
            </xsl:choose>

            <!--
                Insert only non already inserted charset to avoid
                repetitions of elements. This can cause an execution
                delay due to the number of iterations that
                equanls to:
                cardinal_of(prf:CcppAccept-Charset) X
                cardinal_of(prf:MmsCcppAccept-Charset)
            -->
            <xsl:for-each select="prf:component/rdf:Description
```

```
        [@rdf:ID='MMSCharacteristics']/
        prf:MmsCcppAccept-Charset/rdf:Bag/rdf:li">
            <xsl:variable name="C">
                <xsl:value-of select="."/>
            </xsl:variable>
            <xsl:variable name="CFound">
                <xsl:for-each select="../../../../../
                prf:component/rdf:Description
                [@rdf:ID='BrowserUA']/
                prf:CcppAccept-Charset/rdf:Bag/rdf:li">
                    <xsl:variable name="C2">
                        <xsl:value-of select="."/>
                    </xsl:variable>
                    <xsl:if test="$C = $C2">
                    The element is already inserted</xsl:if>
                </xsl:for-each>
            </xsl:variable>
            <xsl:if test="$CFound = ''">
                <rdf:li><xsl:value-of select="."/></rdf:li>
            </xsl:if>
        </xsl:for-each>
        <xsl:for-each select="prf:component/rdf:Description
        [@rdf:ID='MmsCharacteristics']/
        prf:MmsCcppAccept-Charset/rdf:Bag/rdf:li">
            <xsl:variable name="C">
                <xsl:value-of select="."/>
            </xsl:variable>
            <xsl:variable name="CFound">
                <xsl:for-each select="../../../../../
                prf:component/rdf:Description
                [@rdf:ID='BrowserUA']/
                prf:CcppAccept-Charset/rdf:Bag/rdf:li">
                    <xsl:variable name="C2">
                        <xsl:value-of select="."/>
                    </xsl:variable>
                    <xsl:if test="$C = $C2">
                    The element is already inserted</xsl:if>
                </xsl:for-each>
            </xsl:variable>
            <xsl:if test="$CFound = ''">
                <rdf:li><xsl:value-of select="."/></rdf:li>
            </xsl:if>
        </xsl:for-each>
        <!--
            The software platform Charset (without verification with
            the other two previous elements!)
        -->
        <xsl:for-each select="prf:component/rdf:Description
        [@rdf:ID='SoftwarePlatform']/prf:CcppAccept-Charset/
        rdf:Bag/rdf:li">
            <rdf:li><xsl:value-of select="."/></rdf:li>
        </xsl:for-each>
        </rdf:Bag>
    </neg:OnlySupportedCharset>

    </xsl:if>
```

```
                </rdf:Description>
            </ccpp:component>
            <xsl:apply-templates select="prf:component/rdf:Description
            [@rdf:ID='WapCharacteristics']"/>
            <xsl:apply-templates select="prf:component/rdf:Description
            [@rdf:ID='MMSCharacteristics']"/>
            <xsl:apply-templates select="prf:component/rdf:Description
            [@rdf:ID='MmsCharacteristics']"/>
            <xsl:apply-templates select="prf:component/rdf:Description
            [@rdf:ID='SoftwarePlatform']/
            prf:JavaPlatform"/>
            <xsl:for-each select="prf:component/rdf:Description
            [@rdf:ID='BrowserUA']/prf:XhtmlVersion">
                <ccpp:component>
                    <rdf:Description rdf:about="xhtml">
                    <neg:Version rdf:ID="xhtml">
                        <rdf:Bag>
                        <rdf:li><xsl:value-of select="."/></rdf:li>
                        </rdf:Bag>
                    </neg:Version>
                    </rdf:Description>
                </ccpp:component>
            </xsl:for-each>

    </rdf:Description>
</xsl:template>

<xsl:template
match="prf:component/rdf:Description
[@rdf:ID='BrowserUA']/prf:CcppAccept-Encoding">
    <neg:TransferEncoding>
        <rdf:Bag>
        <xsl:for-each select="rdf:Bag/rdf:li">
            <rdf:li><xsl:value-of select="."/></rdf:li>
        </xsl:for-each>
        </rdf:Bag>
    </neg:TransferEncoding>
</xsl:template>

<!-- Some uaprof versions gives this in the SoftwarePlatform
component --> <xsl:template
match="prf:component/rdf:Description
[@rdf:ID='SoftwarePlatform']/prf:CcppAccept-Encoding">
    <neg:TransferEncoding>
        <rdf:Bag>
        <xsl:for-each select="rdf:Bag/rdf:li">
            <rdf:li><xsl:value-of select="."/></rdf:li>
        </xsl:for-each>
        </rdf:Bag>
    </neg:TransferEncoding>
</xsl:template>

<xsl:template
match="prf:component/rdf:Description
[@rdf:ID='SoftwarePlatform']/prf:JavaPlatform">
```

```
    <ccpp:component>
        <rdf:Description rdf:about="java">
        <neg:Platform>
            <rdf:Bag>
            <xsl:for-each select="rdf:Bag/rdf:li">
                <rdf:li><xsl:value-of select="."/></rdf:li>
            </xsl:for-each>
            </rdf:Bag>
        </neg:Platform>
        <xsl:apply-templates select="../prf:JVMVersion"/>
        </rdf:Description>
    </ccpp:component>
</xsl:template>

<xsl:template match="prf:JVMVersion">
        <neg:Version rdf:ID="jvm">
            <rdf:Bag>
            <rdf:li><xsl:value-of select="."/></rdf:li>
            </rdf:Bag>
        </neg:Version>
</xsl:template>

<xsl:template
match="prf:component/rdf:Description[@rdf:ID='WapCharacteristics']">
    <ccpp:component>
        <rdf:Description rdf:about="wml">

        <neg:ResourceType>wml</neg:ResourceType>

        <xsl:apply-templates select="prf:WmlDeckSize"/>
        <xsl:apply-templates select="prf:WmlVersion"/>
        <xsl:apply-templates select="prf:WapVersion"/>

        <xsl:variable name="V1">
            <xsl:value-of select="normalize-space(../../
            prf:component/rdf:Description
            [@rdf:ID='BrowserUA']/prf:TablesCapable)"/>
        </xsl:variable>
        <xsl:variable name="V2">
            <xsl:value-of select="normalize-space(../../
            prf:component/rdf:Description
            [@rdf:ID='BrowserUA']/prf:FramesCapable)"/>
        </xsl:variable>
        <xsl:if test="$V1 = 'Yes' or $V2 ='Yes'">
            <neg:SupportedFunctionalities>
                <rdf:Bag>
                    <xsl:if test="$V1 = 'Yes'">
                        <rdf:li>tables</rdf:li>
                    </xsl:if>
                    <xsl:if test="$V2 = 'Yes'">
                        <rdf:li>frames</rdf:li>
                    </xsl:if>
                </rdf:Bag>
            </neg:SupportedFunctionalities>
        </xsl:if>
        <xsl:if test="$V1 = 'No' or $V2 ='No'">
```

```
            <neg:NonSupportedFunctionalities>
                <rdf:Bag>
                    <xsl:if test="$V1 = 'No'">
                        <rdf:li>tables</rdf:li>
                    </xsl:if>
                    <xsl:if test="$V2 = 'No'">
                        <rdf:li>frames</rdf:li>
                    </xsl:if>
                </rdf:Bag>
            </neg:NonSupportedFunctionalities>
        </xsl:if>

        <xsl:apply-templates select="prf:WmlScriptVersion"/>
        <xsl:apply-templates select="prf:WmlScriptLibraries"/>
        <xsl:apply-templates select="prf:WapSupportedApplications"/>
        <xsl:apply-templates select="prf:WapDeviceClass"/>
        <xsl:apply-templates select="prf:WapPushMsgSize"/>
        <xsl:apply-templates select="../../prf:component/rdf:Description
        [@rdf:ID='PushCharacteristics']/prf:Push-MsgSize"/>
        <xsl:apply-templates select="prf:WtaiLibraries"/>

        </rdf:Description>
    </ccpp:component>
</xsl:template>

<xsl:template match="prf:WapDeviceClass">
    <neg:WapClass><xsl:value-of select="."/></neg:WapClass>
</xsl:template>

<xsl:template match="prf:WapPushMsgSize">
    <neg:WapPushMaxSize><xsl:value-of select="."/></neg:WapPushMaxSize>
</xsl:template>

<xsl:template match="prf:WmlScriptVersion">
    <neg:Version rdf:ID="wmlScript">
        <rdf:Bag>
        <xsl:for-each select="rdf:Bag/rdf:li">
            <rdf:li><xsl:value-of select="."/></rdf:li>
        </xsl:for-each>
        </rdf:Bag>
    </neg:Version>
</xsl:template>

<xsl:template match="prf:WapSupportedApplications">
    <neg:SupportedApplications>
        <rdf:Bag>
        <xsl:for-each select="rdf:Bag/rdf:li">
            <rdf:li><xsl:value-of select="."/></rdf:li>
        </xsl:for-each>
        </rdf:Bag>
    </neg:SupportedApplications>
</xsl:template>

<xsl:template match="prf:WmlScriptLibraries">
    <neg:SupportedScriptLibraries>
        <rdf:Bag>
```

```
        <xsl:for-each select="rdf:Bag/rdf:li">
            <rdf:li><xsl:value-of select="."/></rdf:li>
        </xsl:for-each>
        </rdf:Bag>
    </neg:SupportedScriptLibraries>
</xsl:template>

<xsl:template match="prf:WtaiLibraries">
        <neg:SupportedWtaiLibraries>
            <!-- Some uaprof versions give this as a simple element or a
            Bag element, UPS consider this always as a rdf:Bag-->
            <xsl:variable name="W">
                <xsl:value-of select="normalize-space
                (rdf:Bag/rdf:li[position()=1])"/>
            </xsl:variable>
            <rdf:Bag>
            <xsl:choose>
                <xsl:when test="$W!=''">
                    <xsl:for-each select="rdf:Bag/rdf:li">
                        <rdf:li><xsl:value-of select="."/></rdf:li>
                    </xsl:for-each>
                </xsl:when>
                <xsl:otherwise>
                    <rdf:li><xsl:value-of select="."/></rdf:li>
                </xsl:otherwise>
            </xsl:choose>
            </rdf:Bag>
        </neg:SupportedWtaiLibraries>
</xsl:template>

<xsl:template
match="prf:component/rdf:Description
[@rdf:ID='PushCharacteristics']/prf:Push-MsgSize">
    <neg:WapPushMaxSize><xsl:value-of select="."/></neg:WapPushMaxSize>
</xsl:template>

<xsl:template match="prf:WapDeviceClass">
    <neg:WapClass><xsl:value-of select="."/></neg:WapClass>
</xsl:template>

<xsl:template match="prf:WapPushMsgSize">
    <neg:WapPushMaxSize><xsl:value-of select="."/></neg:WapPushMaxSize>
</xsl:template>

<xsl:template
match="prf:component/rdf:Description
[@rdf:ID='PushCharacteristics']/prf:Push-MsgSize">
    <neg:WapPushMaxSize><xsl:value-of select="."/></neg:WapPushMaxSize>
</xsl:template>

<xsl:template match="prf:WmlScriptVersion">
        <neg:Version rdf:ID="wmlScript">
            <rdf:Bag>
            <xsl:for-each select="rdf:Bag/rdf:li">
                <rdf:li><xsl:value-of select="."/></rdf:li>
            </xsl:for-each>
```

```
            </rdf:Bag>
        </neg:Version>
</xsl:template>

<xsl:template match="prf:WmlScriptLibraries">
        <neg:SupportedScriptLibraries>
            <rdf:Bag>
            <xsl:for-each select="rdf:Bag/rdf:li">
                <rdf:li><xsl:value-of select="."/></rdf:li>
            </xsl:for-each>
            </rdf:Bag>
        </neg:SupportedScriptLibraries>
</xsl:template>

<xsl:template match="prf:WmlDeckSize">
    <neg:CardMaxSize><xsl:value-of select="."/></neg:CardMaxSize>
</xsl:template>

<xsl:template match="prf:WmlVersion">
        <neg:Version rdf:ID="wml">
            <rdf:Bag>
            <xsl:for-each select="rdf:Bag/rdf:li">
                <rdf:li><xsl:value-of select="."/></rdf:li>
            </xsl:for-each>
            </rdf:Bag>
        </neg:Version>
</xsl:template>

<xsl:template match="prf:WapVersion">
        <neg:Version rdf:ID="wap">
            <rdf:Bag>
                <rdf:li><xsl:value-of select="."/></rdf:li>
            </rdf:Bag>
        </neg:Version>
</xsl:template>

<xsl:template
match="prf:component/rdf:Description[@rdf:ID='MMSCharacteristics']">
    <ccpp:component>
        <rdf:Description rdf:about="mms">
        <neg:MmsImageMaxResolution>
        <xsl:value-of select="prf:MmsMaxImageResolution"/>
        </neg:MmsImageMaxResolution>
        <neg:Version rdf:ID="mms">
            <rdf:Bag>
            <xsl:for-each select="prf:MmsVersion">
                <rdf:li><xsl:value-of select="."/></rdf:li>
            </xsl:for-each>
            </rdf:Bag>
        </neg:Version>
        </rdf:Description>
    </ccpp:component>
</xsl:template>

<xsl:template
match="prf:component/rdf:Description[@rdf:ID='MmsCharacteristics']">
```

```
    <ccpp:component>
        <rdf:Description rdf:about="mms">
        <neg:MmsImageMaxResolution>
        <xsl:value-of select="prf:MmsMaxImageResolution"/>
        </neg:MmsImageMaxResolution>
        <neg:Version rdf:ID="mms">
            <rdf:Bag>
            <xsl:for-each select="prf:MmsVersion">
                <rdf:li><xsl:value-of select="."/></rdf:li>
            </xsl:for-each>
            </rdf:Bag>
        </neg:Version>
        </rdf:Description>
    </ccpp:component>
</xsl:template>

<xsl:template
match="prf:component/rdf:Description[@rdf:ID='HardwarePlatform']/
prf:TextInputCapable[normalize-space(.)='Yes']">
    <rdf:li>text</rdf:li>
</xsl:template>

<xsl:template
match="prf:component/rdf:Description[@rdf:ID='HardwarePlatform']/
prf:SoundOutputCapable[normalize-space(.)='Yes']">
    <rdf:li>audio</rdf:li>
</xsl:template>

<xsl:template
match="prf:component/rdf:Description[@rdf:ID='SoftwarePlatform']/
prf:JavaEnabled[normalize-space(.)='Yes']">
    <rdf:li>java</rdf:li>
</xsl:template>

<xsl:template
match="prf:component/rdf:Description
[@rdf:ID='BrowserUA']/prf:XhtmlVersion">
    <rdf:li>xhtml</rdf:li>
</xsl:template>

<xsl:template
match="prf:component/rdf:Description
[@rdf:ID='HardwarePlatform']/prf:Model">
<neg:DeviceName><xsl:value-of select="."/></neg:DeviceName>
</xsl:template>

<xsl:template
match="prf:component/rdf:Description
[@rdf:ID='HardwarePlatform']/prf:ScreenSize">
<neg:DisplayPixel><xsl:value-of select="."/></neg:DisplayPixel>
</xsl:template>

<xsl:template
match="prf:component/rdf:Description
[@rdf:ID='HardwarePlatform']/prf:ScreenSizeChar">
<neg:DisplayChar><xsl:value-of select="."/></neg:DisplayChar>
```

```
</xsl:template>

<xsl:template
match="prf:component/rdf:Description
[@rdf:ID='HardwarePlatform']/prf:BitsPerPixel">
<neg:ColorBit><xsl:value-of select="."/></neg:ColorBit>
</xsl:template>

<xsl:template
match="prf:component/rdf:Description
[@rdf:ID='HardwarePlatform']/prf:Vendor">
<neg:DeviceVendor><xsl:value-of select="."/></neg:DeviceVendor>
</xsl:template>

<xsl:template
match="prf:component/rdf:Description
[@rdf:ID='HardwarePlatform']/prf:CPU">
<neg:Memory><xsl:value-of select="."/></neg:Memory>
</xsl:template>

<xsl:template
match="prf:component/rdf:Description
[@rdf:ID='HardwarePlatform']/prf:PixelAspectRatio">
    <xsl:variable name="S">
        <xsl:value-of select="."/>
    </xsl:variable>
    <neg:PixelStretch><xsl:value-of select="substring-after($S,'x')"/>
    /<xsl:value-of select="substring-before($S,'x')"/></neg:PixelStretch>
</xsl:template>

<xsl:template
match="prf:component/rdf:Description
[@rdf:ID='SoftwarePlatform']/prf:OSName">
    <neg:PlatformName><xsl:value-of select="."/></neg:PlatformName>
</xsl:template>

<xsl:template
match="prf:component/rdf:Description
[@rdf:ID='SoftwarePlatform']/prf:RecipientAppAgent">
    <neg:PlatformName><xsl:value-of select="."/></neg:PlatformName>
</xsl:template>

<xsl:template
match="prf:component/rdf:Description
[@rdf:ID='SoftwarePlatform']/prf:OSVersion">
    <neg:PlatformVersion><xsl:value-of select="."/></neg:PlatformVersion>
</xsl:template>

<xsl:template
match="prf:component/rdf:Description
[@rdf:ID='SoftwarePlatform']/prf:OSVendor">
    <neg:PlatformVendor><xsl:value-of select="."/></neg:PlatformVendor>
</xsl:template>

<xsl:template
match="prf:component/rdf:Description
```

```
[@rdf:ID='NetworkCharacteristics']/prf:SecuritySupport">
    <neg:ProtocolSecurity>
    <rdf:Bag>
    <!-- Some uaprof versions give this as a simple element or a Bag element,
    UPS consider this always as a rdf:Bag-->
    <xsl:variable name="W">
            <xsl:value-of select="normalize-space
            (rdf:Bag/rdf:li[position()=1])"/>
    </xsl:variable>
    <xsl:choose>
        <xsl:when test="$W!=''">
        <xsl:for-each select="rdf:Bag/rdf:li">
            <rdf:li><xsl:value-of select="."/></rdf:li>
        </xsl:for-each>
        </xsl:when>
        <xsl:otherwise>
            <rdf:li><xsl:value-of select="."/></rdf:li>
        </xsl:otherwise>
    </xsl:choose>
    </rdf:Bag>
    </neg:ProtocolSecurity>
</xsl:template>

<xsl:template
match="prf:component/rdf:Description[@rdf:ID='NetworkCharacteristics']/
prf:SupportedBearers">
    <neg:SystemCommunication>
        <rdf:Bag>
        <xsl:for-each select="rdf:Bag/rdf:li">
            <rdf:li><xsl:value-of select="."/></rdf:li>
        </xsl:for-each>
        </rdf:Bag>
    </neg:SystemCommunication>
</xsl:template>

<xsl:template
match="prf:component/rdf:Description[@rdf:ID='NetworkCharacteristics']/
prf:SupportedBearer">
    <neg:SystemCommunication>
        <rdf:Bag>
        <xsl:for-each select="rdf:Bag/rdf:li">
            <rdf:li><xsl:value-of select="."/></rdf:li>
        </xsl:for-each>
        </rdf:Bag>
    </neg:SystemCommunication>
</xsl:template>

<xsl:template
match="prf:component/rdf:Description[@rdf:ID='SoftwarePlatform']/
prf:CcppAccept-Language">
    <neg:Language>
        <rdf:Bag>
        <xsl:for-each select="rdf:Bag/rdf:li">
            <rdf:li><xsl:value-of select="."/></rdf:li>
        </xsl:for-each>
        <xsl:for-each select="rdf:Seq/rdf:li">
```

```
            <rdf:li><xsl:value-of select="."/></rdf:li>
        </xsl:for-each>
        </rdf:Bag>
    </neg:Language>
</xsl:template>

<xsl:template
match="prf:component/rdf:Description[@rdf:ID='BrowserUA']/
prf:CcppAccept-Language">
    <neg:Language>
        <rdf:Bag>
        <xsl:for-each select="rdf:Bag/rdf:li">
            <rdf:li><xsl:value-of select="."/></rdf:li>
        </xsl:for-each>
        <xsl:for-each select="rdf:Seq/rdf:li">
            <rdf:li><xsl:value-of select="."/></rdf:li>
        </xsl:for-each>
        </rdf:Bag>
    </neg:Language>
</xsl:template>

<xsl:template
match="prf:component/rdf:Description[@rdf:ID='BrowserUA']/
prf:BrowserName">
<neg:PlayerName>
    <xsl:value-of select="."/>
</neg:PlayerName>

</xsl:template> <xsl:template
match="prf:component/rdf:Description[@rdf:ID='BrowserUA']/
prf:BrowserVersion">
<neg:PlayerName>
    <xsl:value-of select="."/>
</neg:PlayerName> </xsl:template>


</xsl:stylesheet>
```

Bibliographie

[1] A. K. Dey et Abowd G. D. Understanding and Using Context - Personal and Ubiquitous Computing Journal, 5(1), 2001, p. 4-7.

[2] Allen J., Maintaining knowledge about temporal intervals, Communications of the ACM, 26(11), 1983, p. 832-843.

[3] Apache SOAP, v2.3.1, http ://ws.apache.org/soap/

[4] Balabanovic M., An Adaptive Web Page Recommendation Service, Proceeding of the First International Conference en Autonomous Agents, 1997, p. 378-385.

[5] Barbir A., Chen R., Hofmann M., Orman H. et Penno R., An Architecture for Open Pluggable Edge Services (OPES), Internet Draft, IETF Network Working Group, http ://www.ietf.org/internet-drafts/draft-ietf-opes-architecture-04.txt, 11 Décembre 2002.

[6] Bekaert J., Balakireva L., Hochstenbach P. et Van de Sompel H., Using MPEG-21 DIP and NISO OpenURL for the Dynamic Dissemination of Complex Digital Objects in the Los Alamos National Laboratory Digital Library, D-Lib Magazine, 10(2), Février 2004. http ://www.dlib.org/dlib/february04/bekaert/02bekaert.html

[7] Berners-Lee T., Fielding R. et Frystyk H. Hypertext Transfer Protocol - HTTP/1.0, RFC 1945, HTTP Working Group, Mai 1996.

[8] Bharghavan V., Kang-Won L., Songwu L., Sungwon H., Jin-Ru L. et Dwyer D., The TIMELY Adaptive Resource Management Architecture. IEEE Personal Communications, 5(4), Août 1998, p. 20-31.

[9] Blakowski G. et Steinmetz R., A media Synchronisation Survey : Reference Model, Specification, and Case Studies, IEEE Journal Of Selected Ares In Communication, 14(1), janvier 1996, p. 5-34.

[10] Boll S., Klas W. et Wandel J., A Cross-Media Adaptation Strategy for Multimedia Presentations, Proceedings of the seventh ACM international conference on Multimedia, Orlando, Florida, USA, 1999, p. 37-46.

[11] Boll S., Klas W., ZYX - A Semantic Model For Multimedia Documents and Presentations, Proceedings of the 8th IFIP Conference on Data Semantics (DS-8), Rotorua, Nouvelle-Zélande, 5-8 Janvier 1999.

[12] Boll S., ZYX : Towards Flexible Multimedia Document Models for Reuse and Adaptation, Thèse de doctorat, Université de Vienne, Autriche, 2001, http ://mmit.informatik.uni-oldenburg.de/pubs/dissertation/

[13] Bormans, J., Hill, K., MPEG-21 Overview (v.5), /IEC JTC1/SC29/WG11/N5231, Shanghai, Octobre 2002.

[14] Borning A., Lin R. et Marriot K., Constraint-based Document Layout for the Web, Multimedia Systems, 8(3), 2000, p. 177-189.

[15] Borning A., Lin R. et Marriott K., Constraints for the Web, Proceedings of the fifth ACM international conference on Multimedia, Novembre 1997, Seattle, USA, p. 173-18.

[16] Brown PJ., The Stick-e document : a framework for creating context-aware applications. Proceedings of Electronic Publishing'96, Palo Alto, California, USA, 22-26 Septembre, p. 259-272.

[17] Bulterman D. C. A. et Ayars J., The SMIL 2.0 Content Control Modules, http ://www.w3.org/TR/smil20/smil-content.html, W3C.

[18] Bulterman D. C. A., van Rossum G., van Liere R., A structure for transportable, dynamic multimedia documents. Proceedings of the Summer 1991 USENIX Conference, Nashville, TN, Juin 1991, p. 137-155.

[19] Bulterman D. C. A., User-Centered Abstractions for Adaptive Hypermedia Presentations, CWI (Centrum voor Wiskunde en Iformatica), Proceedings of the sixth ACM interbational conference of Multimedia, Bristol, UK, Septembre 1998, p. 247-256.

[20] Carcone L., Formatage spatial dans un environnement d'édition/présentation de documents multimédia, Mémoire CNAM, Conservatoire National des Arts et Métiers, Grenoble, décembre 1997.

[21] CC/PP : Structure and Vocabularies Test Suite, http ://www.w3.org/2003/07/ccpp-SV-PR/test-suite-20030723/, W3C, Juillet 2003

[22] Chen H., Krishnamuthy A., Little T. D. C. et Venkatesch D., A Scalable Video-on-Demand Service for the Provision of VCR-like Functions, Proceedings ICMCS'95, Washington DC, USA, Mai 1995, p. 65-72.

[23] Chen L., Xie X., Ma W., Zhang H. et Zhou H. Image Adaptation Based on Attention Model for Small-Form-Factor Device. The 9th International Conference

on MultiMedia Modeling (MMM'03), Taiwan, Janvier 7-10, 2003, p. 421-442.

[24] Chen M., Kandlur D. D. et Yu P.S., Support for Fully Interactive Playout in a Disk-array-based Video Server, Proceedings ACM Multimedia'94, Octobre 1994, San Fransisco, USA, p. 391-398.

[25] Consensus Project, RIML Layout Definition, 3G Mobile Context Sensitive Adaptability - User Friendly Mobile Work Place for Seamless Enterprise Applications, Consensus membre du W3C DI WG, https ://www.consensus-online.org/publicdocs/20031021-RIML-layout.pdf, Octobre 2003.

[26] Corner M. D., Noble B. D. et Wasserman K. M. Fugue : time scales of adaptation in mobile video. Proceedings of the SPIE Multimedia Computing and Networking Conference, San Jose, CA, Janvier 2001.

[27] Crocker D. H., Standard for the format of ARPA Internet Text Messages, RFC 822, Août 13, 1982.

[28] Dalal M., Feiner S., McKeown K., Pan S., Zhou M. X., Hollerer T., Shaw J., Feng Y. et Fromer J., Negotiation for automated generation of temporal multimedia presentations, dans ACM Multimedia conference, Boston, MA, US, 1996, p. p. 55-64.

[29] Decker S., Mitra P. et Melnik S., Framework for the Semantic Web : an RDF Tutorial, IEEE Internet Computing, 4(6), 2000, p. 68-73.

[30] Dey AK, Abowd GD. Towards a better understanding of context and context-awareness. CHI'2000 Workshop on the What, Who, Where, When, and How of Context- Awareness, The Hague, Netherlands, Avril 2000.

[31] Elson J. et Cerpa A., Internet Content Adaptation Protocol (ICAP), ICAP Specification, ICAP RFC 3507,http ://www.ietf.org/rfc/rfc3507.txt, IETF Network Working Group, Avril 2003.

[32] Engelbart D. et English W. K., A Research Center for Augmenting Human Intellect, AFIPS Conference Proceedings of the 1968 Fall Joint Computer Conference, Vol. 33, San Francisco, CA, 1968, p. 395-410.

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

[34] Euzenat J., Layaïda N. et Diaz V., A semantic framework for multimedia document adaptation, International Joint Conference on Artificial Intelligence IJCAI 2003, Août 2003, Acapulco, Mexico, p. 9-16.

[35] Fielding R., Gettys J., Mogul J., Frystyk H., Masinter L., Leach P., Berners-Lee T., Hypertext Transfer Protocol - HTTP/1.1, RFC 2616, HTTP Working Group,

Juin 1999.

[36] Franks J. et Luotonen A., Byte ranges - Formal spec proposal, 17 Mai, 1995. http ://lists.w3.org/Archives/Public/uri/1995May/0020.html

[37] Freed N. et Borenstein N., Multipurpose Internet Mail Extensions (MIME) Part One : Format of Internet Message Bodies, RFC 2045, Network Working Group, Novembre 1996.

[38] Freed N. et Borenstein N., Multipurpose Internet Mail Extensions (MIME) Part Two : Media Types, RFC 2046, Network Working Group, Novembre 1996.

[39] Freed N., Klensin J. et Postel J., Multipurpose Internet Mail Extensions (MIME) Part Four : Registration Procedures, RFC 2048, Network Working Group, Novembre 1996.

[40] Freed N. et Borenstein N., Multipurpose Internet Mail Extensions (MIME) Part Five : Conformance Criteria and Examples, RFC 2049, Network Working Group, Novembre 1996.

[41] Frystyk N. H., Leach P. et Scott L. HTTP Extension Framework. Internet Draft, http ://www.w3.org/Protocols/HTTP/ietf-http-ext/draftfrystyk- http-extensions-03.txt, 15 Mars, 1999.

[42] Gimson R., Device Independence Principles, W3C Working Group Note, 01 Septembre 2003.

[43] Graham Klyne et al, Composite Capability/Preference Profiles (CC/PP) : Structure and Vocabularies, http ://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/, W3C Recommendation, 15 January 2004.

[44] Gronbaek K. et H. T. Randall, Design issues for a dexter-based hypermedia system, Communications of the ACM, 37(2), Fevrier 1994, p. 41-49.

[45] Groupe de travail MMI : Multimodal Interaction Working Group, http ://www.w3.org/2002/mmi/, Activité "Multimodal Interaction", W3C

[46] GStreamer. Gstreamer : open source multimedia framework. http ://www.gstreamer.net.

[47] Halasz F. et Schwartz M., The dexter hypertext reference model, Dans Proceedings of the Hypertext Workshop, National Institute of Standards and Technology, NIST Special Publication, Gaithersburg, Md, January 1990, p. 95-133.

[48] Hanrahan R., Merrick R., Wong C., Wasmund M., Lewis R., Lemlouma T. et Finkelstein S. R., Authoring Techniques for Device Independence",

http ://www.w3.org/TR/2003/WD-di-atdi-20031106/, W3C Working Group Note, 18 Février 2004, W3C.

[49] Hardman L., Bulterman D.C.A. et Rossum G.v., The amsterdam hypermedia model : Adding time and context to the dexter model, Communications de l'ACM, 37(2), 1994, p. 50-63.

[50] Hoi K. K., Lee D. L. et Xu J., Document Visualization on Small Displays, Mobile Data Management, Melbourne, Australia, 2003, p. 262-278.

[51] Hollfelder S., Kraiss A. et Rakow T. C., A Client-Controlled Adaptation Framework for Multimedia Database Systems, Dans R. Steinmetz et L.C. Wolf, editeurs, Proceedings IDMS'97, Darmstadt, Springer 1997, 10-12 Septembre, Allemagne, p. 397-409.

[52] Holtman K. et Mutz A., HTTP Remote Variant Selection Algorithm - RVSA/1.0, RFC 2296, Network Working Group, Mars 1998.

[53] Holtman K. et Mutz A. Transparent Content Negotiation in HTTP. RFC 2295, Network Working Group, Mars 1998.

[54] InfoRover Application, http ://www.pendragon-software.com/

[55] Inouye J., Cen S., Pu C. et Walpole J., System Support for Mobile Multimedia Applications, Proceedings of the 7th NOSSDAV, St. Louis, Missouri, USA, Mai 1997, p. 143-154.

[56] InstaVid, Repository dynamique de vidéos clips musicaux, http ://www.instavid.com.

[57] Internet Content Adaptation Protocol (ICAP) Forum, http ://www.i-cap.org/.

[58] Internet Engineering Task Force (IETF), http ://www.ietf.org.

[59] Jan G. et Dave B. RDF Test Cases, W3C Proposed Recommendation, http ://www.w3.org/TR/rdf-testcases/, 15 Decembre 2003.

[60] Java Technology, http ://java.sun.com/, Sun Microsystems, Inc.

[61] Jourdan M., Layaida N., Roisin C., Sabry-Ismail L. et Tardif L., Madeus, An Authoring Environment for Interactive Multimedia Documents, Proceedings of the ACM Multimedia 98. 1998, p. 267-272.

[62] Kate W. T., Deunhouwer P. et Clout R., Timesheets - Integrating Timing in XML, Proceedings WWW9 Workshop : Multimedia on the Web, 15 Mai 2000, Amsterdam, Hollande.

[63] KAON Ontology and Semantic Web Infrastructure, http ://kaon.semanticweb.org/ontologies
.

[64] Kidd T., Electronic Journal Usage Statistics in Practice, Serials : The journal for the serials community, 15(1), 2002, p. 11-17.

[65] Kim M. et Song J., Multimedia Documents with Elastic Time, dans Proceedings : Third ACM International Conference on Multimedia, 5-9 Novembre 1995, San Francisco, California, USA, p. 143-154.

[66] Koenen, R., From MPEG-1 to MPEG-21 : Creating an Interoperable Multimedia Infrastructure, ISO/IEC JTC1/SC29/WG11 N4518, Pattaya, Décembre 2001.

[67] Krishnamurthy B., Mogul J.C. et Kristol D. M., Key Differences between HTTP/1.0 and HTTP/1.1, Computer Networks : The International Journal of Computer and Telecommunications Networking, 31(11-16), Mai 1999, New York, NY, USA, p. 1737-1751.

[68] Layaïda N., Madeus : Système d'édition et de présentation de documents structurés multimédia, Thèse en Informatique, Université Joseph Fourier, Juin 1997.

[69] Layaïda N. et Lemlouma T., NAC : An Architecture for Multimedia Content Adaptation for Mobile Devices, ERCIM (The European Research Consortium for Informatics and Mathematics), ERCIM News No. 54, Juillet 2003.

[70] Lemlouma T. et Layaïda N., Context-Aware Adaptation for Mobile Devices, IEEE International Conference on Mobile Data Management, Berkeley, California, USA, 19-22 Janvier 2004, IEEE, p. 106-111.

[71] Lemlouma T. et Layaïda N., Encoding Multimedia Presentation for User Preferences and Limited Environments, IEEE International Conference on Multimedia & Expo (ICME), Baltimore, Maryland, USA, 6-9 Juillet 2003, p. 165-168.

[72] Lemlouma T., UPS Client Repository, http ://wam.inrialpes.fr/people/lemlouma/-MULTIMEDIA/UPS-Client-Repository/ups-client-repository.html

[73] Lemlouma T. et Layaïda N., Media Resources Adaptation for Limited Devices, ICCC/IFIP Seventh International Conference on Electronic Publishing, Guimarães, Portugal, 25-28 Juin 2003. p. 209-218.

[74] Lemlouma T. et Layaïda N., Adapted Content Delivery for Different Contexts, SAINT 2003 Conference, Orlando, Florida, USA, 27-31 Janvier, 2003, IEEE, p. 190-197.

[75] Lemlouma T. et Layaïda N., SMIL Content Adaptation for Embedded Devices, SMIL Europe 2003, Paris, 12-14 Février, 2003.

[76] Lemlouma T. et Layaïda N., The Negotiation of Multimedia Content Services in Heterogeneous Environments, The 8th International Conference on Multimedia Modeling Proceedings (MMM2001), CWI, Amsterdam, The Netherlands, 5-7 Novembre 2001. p. 187-206.

[77] Lemlouma T. et Layaïda N., Universal Profiling for Content Negotiation and Adaptation in Heterogeneous Environments, W3C Workshop on Delivery Context, W3C/INRIA Sophia-Antipolis, 4-5 Mars 2002, France.

[78] Lemlouma T. et Layaïda N., Content Adaptation and Generation Principles for Heterogeneous Clients, W3C Workshop on Device Independent Authoring Techniques, SAP University, St. Leon-Rot, 25-26 Septembre 2002, Allemagne.

[79] Lenglet R., Hagimont D., Layaïda N., Evaluation of Dynamic Adaptation of Video Streaming using Mobile Agents, Projet Sirac, Projet OPERA, INRIA Rhône-Alpes, Montbonnot Saint-Martin, France.

[80] Li C. S., Mohan R. et Smith J. R., Multimedia content description in the Info-Pyramid. IEEE Proc, Int. Conf. Acoust., Speech, Signal Processing (ICASSP). Special session on Signal Processing in Modern Multimedia Standards. Seattle, WA, USA, 12-15 Mai 1998.

[81] Maioli C., Sola S., Vitali F., Versioning for Distributed Hypertext Systems, Proceedings Hypermedia '94, Pretoria, South Africa, Mars 1994

[82] Masuda H., Yasutomi D. et Nakagawa H., How to Transform Tables in HTML for Displaying on Mobile Terminals, Workshop on Automatic Paraphrasing : Theories and Applications, Sixth Natural Language Processing Pacific Rim Symposium (NLPRS2001), National Center of Sciences, Novembre 2001, Tokyo, Japon, p. 29-36.

[83] Microsoft Commercial Application :WebTV, http ://www.webtv.com/

[84] Mobile Aware, Adaptation Solution for Explicit Layout, Mobile Aware membre du W3C DI WG, http ://www.mobileaware.com/standards/index.html, Novembre 2003.

[85] Moore K., MIME (Multipurpose Internet Mail Extensions) Part Three : Message Header Extensions for Non-ASCII Text, RFC 2047, Network Working Group, Novembre 1996.

[86] Moser F., Kraib A. et Klas W., L/MRP : A Buffer Management Strategy for Interactive Continuos Data Flows in a Multimedia DBMS, Proceedings VLDB 1995, 1995, Morgan Kaufmann, USA, p. 275-286.

[87] MPEG MDS Group, MPEG-21 Multimedia Framework, Part 7 : Digital Item Adaptation (Final Committee Draft), ISO/MPEG N5845, Juillet 2003.

[88] Open Mobile Alliance, OMA, http ://www.openmobilealliance.org/

[89] Open Mobile Alliance, OMA, Wireless Application Protocol (WAP) 2.0 specifications, http ://www.wapforum.org/what/technical.htm
.

[90] Pinheiro M., Lima J., Edelweiss N., Layaïda N., Lemlouma T., An Open E-Learning Authoring Environment, Spezielles Seminar im WS E-Learning 2002/2003, Université de Johann Wolfgang Geothe, Frankfurt Am Main. Allemagne.

[91] Pascoe J. Adding generic contextual capabilities to wearable computers. In : Proceedings of 2nd International Symposium on Wearable Computers, 1998, p. 92-99.

[92] Pihkala K., Vierinen J., et Vuorimaa P., Content Customization Using Device Profiles, Proceedings of the 2nd Intl. Workshop on Intelligent Multimedia Computing and Networking, 8-12 Mars 2002, Durham, North Carolina, USA, p. 1029-1032.

[93] Pocket Info. http ://www.pocketinfo.nl/

[94] Postel J. B., Simple Mail Transfer Protocol, RFC 821, IETF, Août 1982.

[95] Postel J., Media Type Registration Procedure, RFC 1590, IETF, Network Working Group, Novembre 1996.

[96] QuiP Version 2.2.1 , a W3C XQuery Prototype, http ://developer.softwareag.com/tamino/quip/

[97] Raggett D., Le Hors A. et Jacobs I., HTML 4.0 Specification, W3C Recommendation, http ://www.w3.org/TR/1998/REC-html40-19980424/, 24 Avril, 1998.

[98] Raghavan S. V. et Tripathi S. K., Networked Multimedia Systems : Concepts, architecture and design, Prentice-Hall, 07458 New Jersey, 1998.

[99] RDF Vocabulary Description Language 1.0 : RDF Schema, W3C Proposed Recommendation, http ://www.w3.org/TR/rdf-schema/, 15 December 2003.

[100] Reynolds J. et Postel J., Assigned Numbers, RFC 1700, IETF, Network Working Group, Octobre 1994.

[101] Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation, http ://www.w3.org/TR/1999/REC-rdf-syntax-19990222/, 22 Fevrier 1999.

[102] Rodden T., Cheverst K., Davies K. Dix A., Exploiting Context in HCI Design for Mobile Systems, Workshop on Human Computer Interaction with Mobile Devices, 21-22 Mai 1998.

[103] Rutledge L., Hardman L., Ossenbruggen J. V. et Bulterman D. C. A., Implementing Adaptability in the Standard Reference Model for Intelligent Multimedia. Proceeding of Multimedia Modeling 98, Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland, Octobre 1998, p. 12-19.

[104] Salton G. et McGill M. J., An Introduction to Modern Information Retrieval, McGraw-Hill, New York, 1983.

[105] Stéphane B., Transformations de documents structurés : une combinaison des approches déclaratives et automatiques, Thèse en Informatique, Université Joseph Fourier, Décembre 1998.

[106] Stéphane H. Maes et Raman T. V., Position paper for the W3C/WAP Workshop on the Multi-modal Web, W3C/WAP, Shangri-La Hotel, Hong Kong, 5-6 September 2000.

[107] SVG Working Group, Scalable Vector Graphics (SVG) 1.0 Specification. W3C Recommendation, http ://www.w3.org/TR/SVG, 14 Janvier, 2003.

[108] Synchronized Multimedia Working Group du W3C, Synchronized Multimedia Integration Language (SMIL) 1.0 Specification, W3C Recommendation, http ://www.w3.org/TR/REC-smil/, 15 Juin, 1998.

[109] Synchronized Multimedia Working Group du W3C, Synchronized Multimedia Integration Language (SMIL 2.0), W3C Recommendation, http ://www.w3.org/TR/smil20/, 07 Août, 2001.

[110] Smith J. R., Mohan R. et Li C. S., Scalable Multimedia Delivery for Pervasive Computing, Proceedings Seventh ACM International Conference on Multimedia (Part 1), 30 Octobre 4 Novembre, 1999, Orlando, Florida, USA, p. 131-140

[111] Schilit B. et Theimer M. Disseminating Active Map Information to Mobile Hosts. IEEE Network, 8(5). 1994. p. 22-32.

[112] Schilit B, Adams N, Want R. Context-aware computing applications. In : First International Workshop on Mobile Computing Systems and Applications, 1994, p. 85-90.

[113] Slein J., Vitali F., Whitehead E. et Durand D., Requirements for Distributed Authoring and Versioning on the World Wide Web, ACM Standard View, 1(5), 1997, p. 17-24.

[114] SOAP Version 1.2 Part 1 : Messaging Framework, W3C Recommendation , http ://www.w3.org/TR/2003/REC-soap12-part1-20030624/, 24 Juin 2003.

[115] Stefik M., Foster G., Bobrow D.G., Kahn K., Lanning S. et Suchman L., Beyond the Chalkboard : Computer Support for Collaboration and Problem Solveing in

Meeting, Communications of the ACM, 30(1), janvier 1987, p. 32-47.

[116] The User Agent Profile (UAProf), http ://www.openmobilealliance.org/.

[117] Third Generation Partnership Project (3GPP), Multimedia Messaging Services (MMS), http ://www.3gpp2.org/, Stage 1 Requirements, 30 October 2002.

[118] Toivonen S., Profile-Based Adaptability in the Semantic Web, ERCIM News No. 51, Octobre 2002.

[119] Travail en progression avec le groupe de travail DI : Device Independence Working Group, http ://www.w3.org/2001/di/, Activité "Device Independence", W3C.

[120] Vannevar Bush, As We May Think, The Atlantic Monthly, 176(1), Juillet 1945, p. 101-108.

[121] Vannevar Bush. Memex revisited. In J.M. Nyce and P. Kahn, editors, From Memex to Hypertext : Vannevar Bush and the Mind's Machine, Academic Press, San Diego, CA, USA, 1991, p. 197-216.

[122] Villard L., Roisin C. et Layaïda N., An XML-Based Multimedia Document Processing Model For Content Adaptation, Projet Opera, INRIA Rhône-Alpes, Digital Documents and Electronic publishing (DDEP00), Septembre 2000, p. 104-119.

[123] Vitali F., Durand D., Using versioning to support collaboration on the WWW, World Wide Web Journal, 1(1), O'Reilly, 1994, p. 37-50.

[124] Vitali F., Versioning hypermedia, ACM Computing Surveys (CSUR) 31(4), 1999.

[125] Volantis, Additional Authoring Techniques for Device Independence, Volantis Member Contribution for "Authoring Techniques for Device Independence", Volantis membre du W3C DI WG, Novembre 2003.

[126] Vuorimaa P., Teirikangas J. et Vierinen J., Ubiquitous Multimedia Services with XML, dans Proceedings First International Conference Universal Access in Human Computer Interaction, 2001, 5-10 Août, New Orleans, Louisiana, USA, p. 742-746.

[127] W3C, SMIL 2.0 Basic Profile and Scalability Framework, http ://www.w3.org/TR/smil20/smil-basic.html, W3C Recommendation, Août 2001.

[128] W3C Semantic Web Activity, http ://www.w3.org/2001/sw/.

[129] Ward A, Jones A, Hopper A. A new location technique for the active office. IEEE Personal Communications, Octobre 1997, 4(5), p. 42-47.

[130] Wood R., Fankhauser G., Kreula R. et Monni E., QoS Mapping for MultiMedia Applications in a Wireless ATM, Proceedings of ACTS Mobile Summit '97, Aalborg, Denmark, 7-10 Octobre 1997.

[131] XHTML 1.0 : The Extensible HyperText Markup Language XHTML 1.0. W3C Recommendation, http ://www.w3.org/TR/xhtml1/, 26 Janvier 2000, revised 1 Août 2002.

[132] Xie W., Sun H., Cao Y. et Trivedi K., Optimal Webserver Session Timeout Settings for Web Users, Proceedings of the Computer Measurement Group Conference, Décembre 2002, Reno, Nevada, USA, p. 799-820.

[133] XML Path Language (XPath), James Clark et Steve DeRose, Recommandation W3C, http ://www.w3.org/TR/xpath, 16 November 1999.

[134] XQuery 1.0 : An XML Query Language. W3C Working Draft, http ://www.W3.org/TR/xquery/, 22 Août 2003

[135] XSL Transformations (XSLT) Version 1.0, W3C Recommendation, http ://www.w3.org/TR/xslt/, 16 Novembre 1999

[136] Yin J., Alvisi L., Dahlin M. et Iyengar A., Engineering web cache consistency, ACM Transactions on Internet Technology (TOIT), Août 2002, New York - NY, USA, 2(3), p. 224 - 259.

Résumé

Les progrès technologiques récents ont permis l'apparition d'une grande variété de nouveaux moyens pour accéder et utiliser l'information multimédia du Web en tout lieu et à tout moment. L'hétérogénéité des appareils d'accès s'est accompagné une évolution importante de l'information disponible sur le réseau. Aujourd'hui, on trouve une multitude de formats complexes avec de nouvelles fonctionnalités. Ces formats s'appuient sur de nouveaux modèles de documents qui intègrent une structure logique, spatiale, temporelle et une dimension hypermédia.

Face à cette évolution, il est nécessaire de concevoir des systèmes qui permettent l'accès et l'utilisation de l'information sous une forme qui corresponde aux contraintes imposées par l'environnement. Cette thèse a pour objectif de contribuer à l'adaptation et à la négociation des contenus en considérant les limitations des utilisateurs et les contraintes de leur environnement. Nous présentons une architecture flexible appelée NAC qui permet de définir les composants qui interviennent dans la négociation et l'adaptation de contenu, et qui décrit comment ces composants sont organisés.

NAC permet plusieurs types d'adaptation : une adaptation structurelle, une adaptation sémantique et une adaptation des ressources médias. Ces types d'adaptation sont basés sur les différents contextes des clients. Nous proposons également un modèle de description de contextes UPS, un protocole de négociation et un ensemble de techniques d'adaptation. Les concepts de l'architecture NAC ont contribué aux travaux de standardisation du consortium W3C, en particulier au cadre de travail CC/PP et aux travaux sur l'indépendance des terminaux. Cette étude est complétée par une évaluation de performances qui démontre que le système proposé est exploitable dans la pratique.

Mots clés : multimédia, adaptation de contenu, négociation de contenu, accès universel, profils, environnements hétérogènes, mobiles.

Abstract

In the last few years, new devices such as palm computers, smart phones, pocket PCs became common components of the computing infrastructure. These devices allow multimedia information to be used on the Web at any time and anywhere. At the same time, the content of the Web has known an important revolution. Today, the Web includes continuous medias such as video, audio and 3D animations. The content is created in several formats with new functionalities. Usually, these formats are based on many structural dimensions : logical, spatial, temporal and hypermedia.

In order to ensure universal access to Web content, with respect to the constraints of the current environment, it is necessary to design new systems that enable content delivery in different contexts. The objective of our work is to resolve the problems related to content adaptation and negotiation based on the limitations of the target devices and the heterogeneous environment. We propose a flexible architecture called NAC that includes different components for content negotiation and adaptation, and ensures an efficient framework in which these components cooperate and exchange negotiation-based information in order to reach the objective of the universal access.

NAC allows several kinds of adaptations to be applied : structural adaptation, semantic adaptation and media resources adaptation. These adaptations satisfy different contexts of the clients. We also propose a description model of the environment context : UPS, a negotiation protocol and a rich collection of adaptation techniques. NAC concepts have contributed in W3C standardization efforts, in particular in the CC/PP framework and in work on Device Independence. This work includes performance evaluations in order to show the usability of our system in a practical framework.

Keywords : multimedia, content adaptation, content negotiation, universal access, profiles, heterogeneous environments, mobiles.