Conception d'une plate-forme de services ubiquitaires intégrant des interfaces multimodales distribuées

Florent Chuffart

Université de Caen Basse-Normandie
Ecole doctorale SIMEM
UFR de Sciences

Conception d'une plate-forme de services ubiquitaires intégrant des interfaces multimodales distribuées

THÈSE

présentée et soutenue publiquement le 5 décembre 2007

pour l'obtention du

Doctorat de l'université de Caen

spécialité informatique

arrêté du 07 août 2006

par

Florent Chuffart

Composition du jury

Rapporteurs :	Patrick Girard, Professeur	ENSMA / LISI, Poitiers
	Laurence Nigay, Professeur	Université Joseph Fourier, Grenoble
Examinateur :	Bertrand David, Professeur	École Centrale, Lyon
Directeur :	Patrice Enjalbert, Professeur	Université de Caen Basse-Normandie
Co-directeurs :	Lionel Courval	France Telecom - Orange Labs, Caen
	Jacques Madelaine, MC	Université de Caen Basse-Normandie

Mis en page avec la classe thloria.

Remerciements

Je remercie Patrice Enjalbert, Lionel Courval et Jacques Madelaine pour avoir accepté de diriger cette thèse, Laurence Nigay et Patrick Girard pour avoir accepté de rapporter cette thèse ainsi que Bertrand David pour avoir accepté de faire partie de mon jury.

Je remercie également Yvan R. et à travers lui les contributeurs du PRP TEP pour leur collaboration sans faille ; Eric C. et à travers lui la *team* Aurora, pour leur constante implication dans le projet européen ; et plus généralement les personnes qui m'ont permis de mener à bien mes travaux de recherche.

Je remercie l'équipe MMI du centre recherche du groupe France Telecom, en y adjoignant Tiphaine M., Christophe C., Agnès F. et Cécile N. pour les moments conviviaux qu'ils m'ont permis de partager. Je remercie particulièrement Daniel C. pour sa vision éclairée du monde vocal, Jean S., Pascal L., Julien Z et Thomas L. pour leur apport concernant les réseaux et la téléphonie IP, Guillaume D. et Alexandre F. pour leur contribution en terme de serveurs vocaux, Jocelyn M., Hubert S. et Alain V. respectivement pour leurs connaissances approfondies du réseau commuté, de la messagerie unifiée et des systèmes et réseaux. Je remercie aussi Marie-Adélaïde F., Philippe F., Vinh T. pour eux ainsi qu'Antoine B. avec qui j'ai partagé ma dernière année de doctorat, à qui je souhaite du bonheur et pour qui je laisse apparaître cette citation :

> *Les titres des tableaux ne sont pas des explications et les tableaux ne sont pas des illustrations des titres.*
> René Magritte.

Je remercie Bruno C. et les personnes de l'équipe DoDoLa, pour l'accueil qu'ils m'ont réservé, Khaldoun Z. pour sa connaissance approfondie du domaine , John for his correct English ; Hélène R. qui m'a permis d'imprimer ces pages ; Simon L., Cyril B. Léonard D., Céline B., Céline C., Hugo P., Guillaume B. et Matthieu B. pour leur réconfort dans les moments de doute.

Sans oublier Jean D., Jean-François P., †Dominique D., Christophe H. et Martine L., pour avoir touché mon esprit ; Manue du haut de ces 17 ans ; Carole P. pour son écoute et car :

> *Les paroles restent. Les écrits ne restent pas.*
> Jacques Lacan.

Je remercie mes amis d'enfance et mes amis en Bray pour les parfums qu'ils donnent à ma vie.

Je dédie cette thèse à mes parents, mon frère, ma famille.

Sommaire

Introduction

Contexte

Mon travail de thèse s'est déroulé dans le centre de recherche de France Telecom (Orange Labs) de Caen, dans le cadre d'un contrat CIFRE[1]. L'encadrement de ma thèse a été assuré par le GREYC[2]. Mon travail adresse les problématiques de la multimodalité et de l'interaction web. Il tend à fournir une plate-forme de service permettant l'intégration du mode vocal dans les pages web. Les sections qui suivent introduisent mon travail de thèse, elles présentent les motivations, la contribution et l'organisation de mon travail, ainsi que les perspectives offertes par ces trois années de recherche.

Le réseau ubiquitaire

Si l'on observe l'évolution des réseaux de télécommunication on s'aperçoit qu'elle se superpose à l'évolution des réseaux de transport. Le 27 février 2007, le ministre délégué à l'Aménagement du Territoire a signé un accord national pour la couverture GSM[3] des axes de transport prioritaires avec l'ARCEP[4], l'Assemblée des Départements de France, l'Association des Maires de France, Orange France, Bouygues Telecom, SFR, la SNCF et RFF[5]. Les opérateurs devront achever la couverture des autoroutes, des routes sur lesquelles le trafic est supérieur à 5 000 véhicules par jour en moyenne, ainsi que des axes reliant au sein de chaque département la préfecture aux sous-préfectures d'ici fin 2008 et fin 2009 (figure 1) [9]. Cette volonté commune d'accompagner la personne dans ses déplacements, en lui fournissant à la fois une infrastructure et des moyens d'y accéder, témoigne de l'importance des enjeux liés à l'exploitation du réseau GSM.

Parallèlement, l'évolution de près de 800% du nombre d'abonnements internet haut débit en France depuis 4 ans (figure 2), a pour effet d'augmenter la quantité d'information et la vitesse des échanges. Les contenus se dématérialisent, sont partagés, diffusés en flux. De nouvelles technologies émergent, permettant de répartir la charge des machines à travers le réseau. Les échanges se font de pair à pair[6] ; les services de VoIP[7] reproduisent à bas coût les fonctionnalités des services de téléphonie. La définition d'une infrastructure adaptée à ces nouvelles possibilités devient un réel enjeu pour les acteurs du domaine.

[1] CIFRE - Convention Industrielle de formation par la Recherche.

[2] GREYC - Groupe de Recherche en Informatique, Image, Automatique et Instrumentation de Caen (UMR 6072).

[3] GSM - Global System for Mobile communications, réseau de téléphonie mobile.

[4] ARCEP - Autorité de Régulation des Communications Électroniques et des Postes.

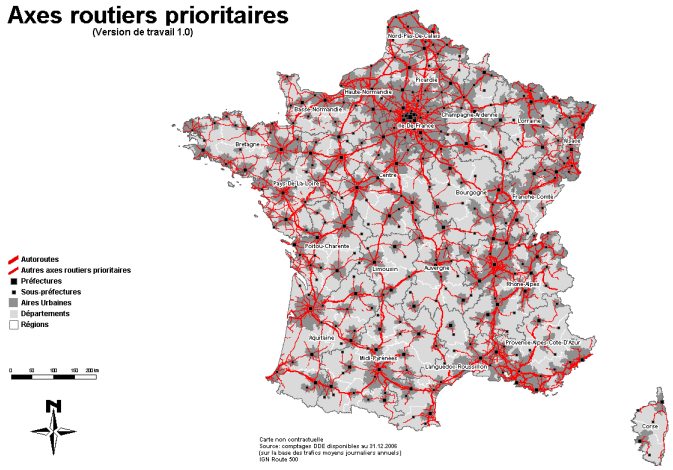
[5] RFF - Réseau Ferré de France, organisme en charge de la gestion des infrastructures ferroviaires.

[6] P2P - peer-to-peer, échange d'égal à égal à travers le réseau.

[7] VoIP - Voice over IP, service de téléphonie utilisant le réseau internet.

Le dimensionnement et la topologie des réseaux de télécommunication évoluent. De la même manière que l'évolution des infrastructures de transport a donné lieu à des phénomènes migratoires, l'évolution des réseaux de télécommunication modifie les échanges entre les nœuds de ce maillage, modifiant en même temps la nature des services. Les services deviennent accessibles non seulement à distance mais également en situation de mobilité. Les services personnalisés s'adaptent aux terminaux et aux utilisateurs. On parle alors du *réseau ubiquitaire* [33].

Dans sa présentation du 1er février 2006 chez Google [34], Dave Raggett[8] expose sa vision du réseau ubiquitaire. Le web devient une vaste plate-forme de services accessibles n'importe quand, n'importe où, depuis n'importe quel dispositif. Qu'il soit en situation de handicap, travailleur nomade ou dans un cadre résidentiel, l'utilisateur dispose d'une interface adaptée au contexte. Pour Dave Raggett, cette mutation passe par *l'intégration du mode vocal dans les interfaces web.*



L'observation du réseau routier Français révèle de façon flagrante un maillage reliant les principales agglomérations. Les axes routiers et les agglomérations concentrent la majeure partie des personnes. La couverture de cet espace présume de l'étendue du réseau ubiquitaire.

FIG. 1 – Axes routiers prioritaires en terme de déploiement de la téléphonie mobile en France (source ARCEP).

Le nombre d'abonnement internet haut débit en France a augmenté de près de 800% depuis 4 ans. Cette augmentation modifie à la fois la nature des échanges sur internet (contenus multimédia) et la manière dont les personnes utilisent le réseau (connexion permanente).

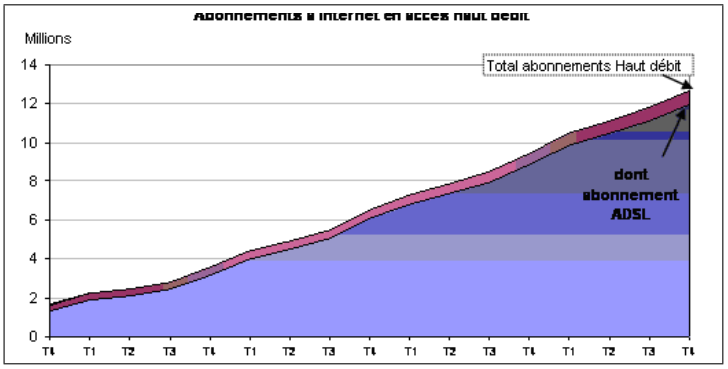


FIG. 2 – Évolution trimestrielle du nombre d'abonnement internet haut débit entre 2002 et 2006 en France (source ARCEP).

[8]Dave Raggett travaille à la standardisation du réseau ubiquitaire pour le W3C.

Les terminaux communicants

Les moyens d'accéder à ces réseaux se multiplient, ne se limitant pas aux traditionnels ordinateurs personnels, mais gagnant les téléphones mobiles, les équipements automobiles, les gadgets hi-tech et autres appareils dédiés (figure 3). Les *terminaux communicants* permettent non seulement d'échanger avec nos proches mais également d'accéder à notre patrimoine numérique et d'interagir avec notre environnement. Les navigateurs web ne sont plus le privilège des machines de bureau mais sont intégrés dans des appareils dédiés, les rendant tour à tour outils de communication, ou compagnons de notre vie numérique. Des objets communicants fleurissent sur nos bureaux, nous permettant, par exemple, de signaler notre état de présence à travers le réseau [Kranz *et al.*, 2006]. L'interaction devient alors mixte [Coutrix et Nigay, 2006], ancrées dans le monde réel et dans le monde virtuel.

Les terminaux communicants sont des nœuds du réseau ubiquitaire. Ils permettent d'accéder au réseau. La figure 3 montre que les terminaux communicants sont multiples et hétérogènes. Ces dispositifs sont équipés de modules communicants (NFC[9] bluetooth, WiFi, GPRS[10], UMTS[11]) assurant la connexion au réseau ubiquitaire. Cette hétérogénéité et cette connectivité étendent les possibilités d'usage des terminaux communicants.

FIG. 3 – Panel non exhaustif des terminaux communicants du XXIéme siécle. (de gauche à droite) (en haut) PDA *Qtek S100*, téléphone mobile *Sony Erickson T630*, ordinateur portable *Apple MacBook*, (en bas) Lapin communicant *Violet Nabaztag*, Dispositif GPS *Tom Tom One*, Baladeur multimédia *Appel iPod*, Console de jeu portable *Nintendo DS* et manette de jeu *Nintendo Wii* intégrant la modalité gestuelle.

Naturellement, la miniaturisation des terminaux mobiles induit une modification de leurs

[9]NFC - Near Field Communication, protocole permettant la lecture d'étiquettes interactives s'apparentant à des codes barre mais basé sur un technologie RFID (RFID - Radio-frequency identification), standard ISO depuis le 8 décembre 2003, poussé par le NFC-Forum.

[10]GPRS - General Packet Radio Service, téléphonie 2G, permet un accès internet bas débit mobile.

[11]UMTS - Universal Mobile Telecommunications System, téléphonie 3G, permet un accès internet haut débit mobile.

interfaces et de leurs performances. D'un côté, les dispositifs mobiles embarquent des périphériques traditionnels atrophiés voir amputés. La taille d'un écran d'ordinateur de bureau est de l'ordre de 19 pouces contre 2,8 pouces sur un PDA[12]. Le clavier d'un ordinateur portable possède 85 touches tandis qu'un téléphone mobile en possède tout au plus une vingtaine (figure 3). De l'autre, cette gamme de nouveaux *media* intègrent de nouvelles modalités, telles que le geste ou la voix, jusqu'alors peu exploités dans ce contexte d'utilisation. Les interfaces des services ne sont plus seulement graphiques mais deviennent à la fois vocales, gestuelles et tactiles. On parle alors d'*interfaces multimodales*.

L'interaction multimodale distribuée

Constatant d'une part, la possession par l'utilisateur d'un ou plusieurs dispositifs mobiles, certes innovants mais au demeurant limités et d'autre part, la présence dans l'environnement d'objets communicants, aux capacités d'interaction nouvelles, nous proposons de connecter l'ensemble de ces terminaux à une même session interactive. Nous définissons ainsi le paradigme d'*interaction multimodale distribuée*.

L'intérêt de ces services est multiple. Tout d'abord, il fournit à l'utilisateur des services aux interfaces riches, intégrant plusieurs modes de communication, notamment la voix. En effet, si chacun des terminaux, pris séparément, possède une interface simple et limitée, l'ensemble de ces terminaux permet l'utilisation conjointe des différents modes qu'ils intègrent. Ensuite, l'interaction multimodale distribuée est particulièrement adaptée au travail nomade, à la consommation de loisirs numériques et à l'accessibilité des contenus en situation de handicap. Le travailleur nomade ira chercher, dans son environnement, les périphériques dont il ne dispose pas sur lui. Les contenus multimédia seront diffusés sur les interfaces les plus adaptées à leurs natures. Les modalités inexploitables en situation de handicap se verront substituées par d'autres, plus adaptées au contexte d'interaction. Enfin, ce paradigme exploite les terminaux communicants actuels. Il laisse l'utilisateur dans son environnement d'origine, il ne nécessite pas d'investissement en matériel coûteux et intrusif ; la mise en situation est immédiate.

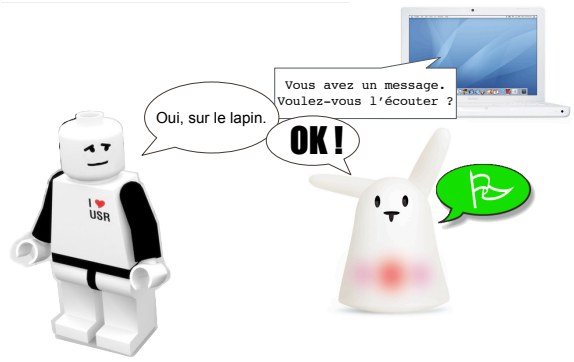


FIG. 4 – Exemple d'interaction multimodale distribuée : l'unité centrale de l'utilisateur le notifie de l'arrivée d'un nouveau message ; l'utilisateur se voit proposé une vocalisation de ce message ; il accepte et précise que le message doit être joué sur son objet communicant favori.

Ce paradigme permet d'adresser de nouveaux services et contribue à la construction du réseau ubiquitaire. Les *services ubiquitaires*, « accessibles par tous, n'importe où, sur tout dispositif communicant » utilisent l'environnement de l'utilisateur pour être rendus de manière visuelle ou sonore. L'intégration du mode vocal dans les interfaces web, plus généralement l'utilisation de la parole dans les interfaces multimodales, se révèle être un élément majeur de notre problé-

[12]PDA - Personnal Digital Assitant (Newton, Palm Pilot, Pocket PC), terminal mobile utilisant la métaphore du bureau et possédant éventuellement une connectivité réseau (GSM, WiFi) [Buisson et Jestin, 2001].

matique. En effet, tandis que l'on considère une classe hétérogène de terminaux communicants aux organes interactifs limités voire amputés, la parole permet de compenser ces manques. De plus, l'interaction vocale permet d'interagir à distance avec des interfaces distribuées dans l'environnement de l'utilisateur, permettant ainsi de limiter les charges liées à la maintenance de périphériques fragiles et coûteux. Ainsi, un simple micro protégé par une structure adaptée offre des capacités d'interactions suffisantes à la réalisation de tâches possédant différents degrés de complexité (commandes vocales, tris et filtrages dans de jeux de données...). Ensuite, l'utilisation conjointe de la parole et des modes d'interactions plus conventionnels (clavier, souris) permet au concepteur de services d'offrir à l'utilisateur la possibilité de choisir le mode d'interaction qui lui convient le mieux. Cette utilisation conjointe permet également la réalisation de tâches plus complexes : par exemple la définition par le pointeur de l'objet d'une commande vocale. Ces tâches complexes possèdent différents niveaux d'abstraction : la tâche et l'objet de la tâche. L'utilisation de la parole permet de limiter le nombre d'indirections nécessaires à la définition de ces différentes abstractions : plutôt que de définir séquentiellement l'objet de la tâche puis de définir la tâche à réaliser, l'utilisation de la parole permet de concrétiser de manière simultanée ces différentes abstractions. Enfin, la parole peut être traitée par les terminaux téléphoniques : classe de terminaux dont l'usage est particulièrement bien assimilé par la population et dont l'omniprésence laisse présumer de l'impact lié à l'exploitation de l'interaction multimodale distribuée. Afin d'illustrer ces nouvelles possibilités, observons un service particulier : « le kiosque multimodal ».

Cas concret : mKiosque

Plaçons nous dans un contexte d'utilisation particulier. Un utilisateur possédant un téléphone mobile se trouve chez son disquaire. Le disquaire met à la disposition de ses clients un point d'écoute permettant de consulter l'actualité musicale de la semaine. Cette borne est constituée d'un écran tactile sur lequel sont affichées les pochettes des albums. En utilisant l'écran tactile, l'utilisateur navigue au travers des livrets numérisés des albums ainsi présentés. Il peut consulter des articles de presse relatifs à ces albums. L'interface propose à l'utilisateur d'écouter des extraits de ces albums.

Le kiosque étant dépourvu de dispositif d'écoute, l'interface suggère à l'utilisateur de coupler le dispositif et son téléphone mobile. Une fois les deux appareils synchronisés, le téléphone mobile de l'utilisateur devient le dispositif d'écoute du service. L'utilisateur voit sur l'écran les informations (pochette de l'album, titre du morceau état d'avancement de la lecture du titre) relatives au titre qu'il écoute sur son mobile (en utilisant les oreillettes).

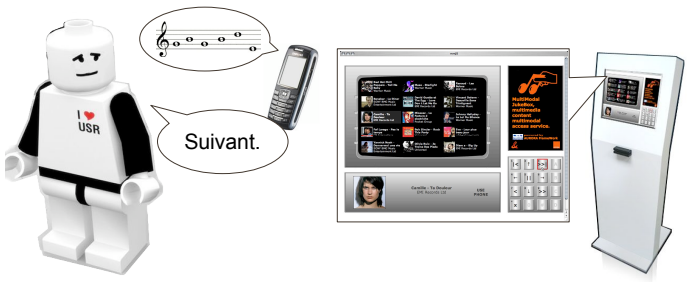


FIG. 5 – mKiosque, une borne dépourvue de haut-parleur et de microphone va pouvoir rendre compte d'un service de consultation de contenu multimédia intégrant la modalité vocale comme moyen de contrôle !

Le mobile devient également un dispositif de contrôle du service. En effet, l'utilisateur a la possibilité de contrôler le flux audio non seulement avec l'interface graphique en utilisant les boutons prévus à cet effet (lecture, pause, suivant, précédent, avance rapide et retour rapide)

mais également en utilisant les commandes vocales (« lecture », « pause »...) capturées par le microphone de son terminal mobile ou les touches de ce même téléphone. Ainsi un kiosque dépourvu de haut-parleur et de microphone va pouvoir rendre compte d'un service de consultation de contenu multimédia intégrant la modalité vocale comme moyen de contrôle (figure 5) ! Outre l'intérêt pour le disquaire qui n'a plus à sa charge la maintenance des dispositifs d'écoute, un tel service se réfère aux repères de l'utilisateur par l'intermédiaire de son propre matériel. Cet exemple simple montre les possibilités d'interaction offertes par la multimodalité distribuée.

Au cours de notre étude, nous illustrons le concept d'interaction multimodale distribuée au travers de trois applications. La première est une interface web permettant de superviser un système d'authentification utilisant la biométrie vocale des individus. La seconde est un simulateur de distributeur de billets intégrant une assistance vocale adaptée aux utilisateurs non ou malvoyants. La troisième est un démonstrateur permettant de prendre le contrôle d'une application web instanciée sur un terminal web classique depuis un terminal téléphonique. Chacune de ces applications illustre l'apport des interfaces multimodales distribuées en terme d'accessibilité, de domotique[13] ou de nomadisme et montre de façon incrémentale les facettes de notre travail.

Contributions

Nos applications de démonstration reposent sur notre modèle d'architecture distribuée *UbiArch*. Ce modèle s'inscrit dans la continuité des travaux de la communauté et respecte les principes fondamentaux des modèles conceptuels de Seeheim et Arch [Pfaff, 1985, SIGCHI, 1992]. Il assure la mise en œuvre de services web interactifs utilisant les terminaux téléphoniques et les clients web de l'utilisateur de manière simultanée. Nous apportons un soin particulier à définir l'architecture sous-tendant UbiArch. Nous identifions les composantes logicielles de cette architecture et implémentons la plate-forme *mVIP*, composante clef permettant l'initiation d'une session interactive distribuée à travers le web.

Notre architecture constitue une réponse aux recommandations de l'« ubiquitous web domain » [44]. Nous assemblons les dialectes du domaine autour d'un canevas, assurant ainsi l'accès au web en situation de mobilité, de handicap ou dans un contexte résidentiel. Notre travail trouve ses applications dans le domaine de l'accessibilité, en proposant l'intégration de modalités telles que la voix afin d'assurer un rendu du service adapté à chacun. Nous adressons le domaine de la mobilité et du travail nomade en proposant des interfaces distribuées dans l'espace public et rendues interactives par l'intermédiaire du téléphone mobile de l'utilisateur. Nous illustrons l'utilisation de notre plate-forme dans chacun de ces domaines au travers des démonstrateurs proposant des solutions d'interactions adaptées à l'utilisateur.

Nos choix d'implémentation intègrent les standards de la voix sur IP et du web. Ainsi, nous contribuons à l'adoption de standards ouverts. Nous assurons du même coup l'intégration de notre technologie dans les infrastructures existantes ; qu'elles soient web ou téléphoniques. Nous définissons des outils, destinés au concepteur de services web. Nous lui permettons ainsi d'intégrer les fonctionnalités du réseau téléphonique dans ses interfaces. Ces outils prennent la forme de couples objet graphique (*widgets*) et stratégie de dialogue (*diaget* [Depaulis *et al.*, 2006]).

[13] Domotique - automatisation des activités domestiques.

Nous contribuons ainsi à la construction du réseau ubiquitaire.

Notre modèle d'architecture organise l'environnement de l'utilisateur en tenant compte de sa présence sur le réseau. Il étend les principes de l'espace de conception *pipeline* [Nigay, 1994] en intégrant le cheminement des flux média à travers le réseau. Ceci nous permet de partager la responsabilité des infrastructures de communication entre les différents acteurs du service et de définir une distribution des composantes logicielles à travers le réseau en assurant une répartition de la charge adaptée à l'infrastructure de communication. Nous contribuons de cette manière à intégrer la sécurité des individus et de l'infrastructure au plus tôt dans le processus de conception.

En résumé, notre travail de recherche contribue de manière significative à la définition des interfaces web de demain [Coutaz, 1998, Beaudouin-Lafon, 2004]. Notre travail met en œuvre une architecture de service s'appuyant sur les standards du web. L'architecture permet la définition d'interfaces distribuées dans l'environnement de l'utilisateur. Notre architecture repose sur un composant clef qui permet de réunir dans une même session interactive des clients web et des terminaux téléphoniques commutés ou IP. L'utilisation de notre technologie se voit facilitée par la définition d'une boîte à outils destinée au concepteur d'applications web et permettant à ce dernier d'étendre les capacités des interfaces web classiques.

Organisation du mémoire

Afin d'adresser ces nouveaux besoins, nous étudions les processus communicationnels mis en jeu lors de l'interaction entre un utilisateur et un service (chapitre 1). Nous identifions le VoiceXML comme un langage assurant la définition de stratégies de dialogues convergentes selon le modèle projectif du dialogue de Vernant [Vernant, 1992]. VoiceXML est un langage à balises permettant de décrire des stratégies de dialogues. VoiceXML intègre initialement (versions 1.0, 2.0 et 2.1) le concept de mode (vocal et DTMF[14]). Il est utilisé pour définir des services accessibles depuis un téléphone tels que les répondeurs, mais il permet également de construire un scénario d'attente et de mise en relation... La version 3.0 de ce dialecte XML tend à intégrer d'autres modalités telles que la vidéo comme modalité de présentation (sortie) ou le stylet comme modalité de contrôle (entrée). La section 1.4 montre comment VoiceXML propose des structures de contrôle du dialogue articulées autour d'un canevas de requêtes et de réponses.

Les fondements du web intègrent également les concepts de requête et de réponse. Cependant, le web et plus généralement l'ordinateur personnel se sont construit principalement autour de la modalité graphique. L'étude des *IHM* a pour principal but de fournir une solution logicielle permettant à un utilisateur d'interagir avec un ordinateur animé par un programme. Dans l'article « Designing interaction, not interface », Michel Beaudouin-Lafon [Beaudouin-Lafon, 2004] révèle trois paradigmes d'interaction homme machine : la machine comme un outil (*computer-as-tool*), qui étend les capacités humaines à travers la machine ; la machine comme un partenaire (*computer-as-partner*), qui se voit déléguer des tâches à travers un dialogue évolué ; la machine comme un medium (*computer-as-medium*), qui est un moyen de médiatiser la communication entre humains en vue d'un travail collaboratif. La section 1.1 montre comment notre problématique adresse ces trois paradigmes. Cette section pose les fondements de notre architecture de services et ancre notre problématique dans le domaine des IHM.

En introduction, nous avons identifié le web et les réseaux de téléphonie comme supports

[14]DTMF - Dual Tone Modulaltion Frequency, signalisation associée à chacune des touches du téléphone et véhiculée sur les lignes téléphoniques dans les fréquences audibles.

de notre architecture de service. Ainsi, nous étudions ces deux infrastructures de communication (section 1.3). La première, du point de vue du W3C[15], qui promeut la compatibilité des technologies du web ; la seconde, du point de vue de l'IETF[16] qui participe à l'élaboration de standards pour internet, notamment les RFCs[17] ayant trait à la voix sur IP. Nous étudions ici de manière fine les infrastructures de VoIP. Néanmoins, l'utilisation de passerelles VoIP nous permet d'affirmer que le traitement de notre problématique englobe les architectures de téléphonie traditionnelle (RTC[18] et GSM) sans perte de généralité ni impact sur la scalabilté de notre solution.

Par la suite, nous dressons l'inventaire des briques logicielles assurant le déploiement de services sur cette infrastructure (section 1.6). Nous constatons que certaines plates-formes du marché, dîtes « clients légers », autorisent le déploiement de services uniquement vocaux sur la plupart des terminaux téléphoniques, tandis que d'autres, dîtes « clients lourds », rendent compte de services exploitant le mode vocal et graphique de terminaux gourmands, coûteux et peu répandus, mais qu'aucune de ces plates-formes ne permet de connecter un terminal téléphonique à bas coût et un client web dans une même session interactive. Dès lors, il convient de définir une architecture de service adaptée.

Le chapitre 2 présente notre architecture de service. Elle s'articule autour de notre modèle d'architecture UbiArch. La section 2.2 recense les éléments qui composent l'environnement de l'utilisateur. Cet espace intègre les terminaux et des nœuds du réseau ubiquitaire, il est structuré autour de sphères d'interaction définies par rapport à l'utilisateur. Les interactions entre l'utilisateur et le service sont modélisées par des flux qui transitent sur le réseau. À ce titre, le modèle pipe-line [Nigay, 1994] rend compte de l'environnement de l'utilisateur. Les sphères d'interactions représentent des sous-ensembles du réseau ubiquitaire impliquant à différents niveaux la responsabilité de l'utilisateur, du fournisseur de service et de l'exploitant de l'infrastructure.

Dans la section 2.3, nous construisons notre modèle UbiArch à partir des modèles d'interaction homme machine de la littérature [Pfaff, 1985, SIGCHI, 1992] et de l'évolution des technologies web depuis le début des années 90 à nos jours. Notre modèle d'architecture homme machine réunit les terminaux du réseau ubiquitaire dans une même session interactive. Nous avons apporté un soin particulier à définir un modèle implémentationnel adapté à l'infrastructure existante et à l'usage qui en est fait. Le noyau fonctionnel d'UbiArch est modélisé par un agrégat de services web distribués dans le réseau. Un script CGI[19] est en charge de construire les interacteurs qui composent l'interface du service à partir des objets métiers du noyau fonctionnel. Les interacteurs sont distribués sur les différents terminaux de l'utilisateur et interagissent selon une logique de dialogue globale assurée par un maillage d'agents PAC[20] [Coutaz, 1987].

Dans l'optique d'un déploiement à court terme de notre technologie, nous concentrons nos efforts sur l'intégration du mode vocal dans les pages web. Cependant, tout au long de notre étude, nous gardons en tête la possibilité d'intégrer d'autres modes de communication. Ce soucis de généralisation, propre à un travail de recherche, se concrétise par l'intégration du mode

[15]W3C - World Wide Web Consortium, consortium fondé en octobre 1994 dont la gestion est assurée conjointement par le Massachusetts Institute of Technology (MIT) aux États-Unis, le European Research Consortium for Informatics and Mathematics (ERCIM) en Europe (auparavant l'Institut national de recherche en informatique et en automatique français (INRIA)) et l'Université Keio au Japon [41].

[16]IETF - Internet Engineering Task Force, groupe informel, international, ouvert à tout individu [17].

[17]RFC - Request For Comments, peu de RFC sont des standards, mais tous les standards d'internet sont enregistrés en tant que RFC.

[18]RTC - Réseau Téléphonique Commuté.

[19]CGI - Common Gateway Interface.

[20]PAC - Présentation Abstraction Contrôle.

gestuel dans le cadre d'un projet collaboratif de recherche. Nous avons démontré l'intégration dans notre architecture d'un module de gestuelle, développé par Philips Innovation Lab Philips [Quaedvlieg *et al.*, 2006, Quaedvlieg et de Wildt, 2006], lors de la restitution du projet européen ITEA[21] Aurora[22] du cluster EUREKA[23] en juin 2006. Nous avons également identifié le composant clef de notre architecture permettant cette interopérabilité et nous l'avons soumis à deux demandes de brevet[24].

Le contexte industriel de notre travail de recherche nous impose de nous inscrire dans la continuité des efforts de normalisation des acteurs du domaine, d'y participer [Chuffart *et al.*, 2006] et d'apporter un soin particulier quant à l'intégration des standards. Cette ligne de conduite permet d'anticiper les évolutions des terminaux actuels. Elle assure la compatibilité ascendante des infrastructures et l'interopérabilité entre les différents systèmes existants et à venir, éléments essentiels des problématiques de pré-production.

Ainsi, le chapitre 3 de notre étude, présente en détail l'implémentation de notre plate-forme de service : mVIP. La plate-forme mVIP assure l'interopérabilité entre les réseaux internet et le réseau téléphonique, elle est l'acteur majeur de notre architecture ubiquitaire. Notre approche étend les fonctionnalités des serveurs vocaux interactifs (SVI) classiques en leur adjoignant une interface de contrôle de type service web. De fait, mVIP est en mesure de fournir aux concepteurs d'interfaces web un ensemble de widgets[25] facilement intégrables aux interfaces web et permettant d'intégrer les fonctionnalités DTMF et vocales des téléphones. Dès lors, l'utilisateur est en mesure d'utiliser son téléphone pour interagir avec le service web.

Afin d'obtenir une adhésion massive à notre technologie, nous avons développé un mécanisme de « snippet »[26] HTML permettant d'intégrer le téléphone de l'utilisateur dans les interfaces web existantes. Afin d'illustrer la puissance de notre mécanisme de snippet, nous l'avons utilisé dans le logiciel de présentation « Slidy HTML »[27]. En y ajoutant une unique ligne de code HTML, nous avons étendu le brillant outil web qu'est slidy, en lui adjoignant une télécommande outil ubiquitaire, où le téléphone de l'utilisateur pilote la présentation.

Enfin, le *quatrième chapitre* illustre les possibilités qu'offre notre plate-forme à travers trois services [11]. Le premier a été réalisé à la fois dans le cadre du projet ITEA Aurora, et plus particulièrement grâce à une collaboration étroite avec la société italienne Intesi, acteur européen du domaine de la sécurité des sytèmes et de la biométrie [Van Gool, 2004], et dans le cadre d'un contrat de recherche entre France Telecom et l'IDIAP [28]. Chacun de ces partenaires a contribué à la définition d'une architecture de test en fournissant une plate-forme de biométrie vocale. Dans ce contexte, notre travail était de démontrer l'intégration, dans notre architecture, d'une « brique logicielle » de biométrie. Ainsi, nous avons défini un service de supervision des utilisateurs d'un parc de machine. Notre démonstrateur exploite les fonctionnalités des plates-formes de biométrie précédemment citées. Il offre : aux administrateurs système, une interface de gestion des empreintes vocales des utilisateurs ; aux développeurs de services web, la possi-

[21] ITEA - Information Technology for European Advancement [19].
[22] ITEA Aurora - projet ITEA 03005 [20].
[23] EUREKA - organisme européen qui finance et coordonne des projets de recherche et de développement [14].
[24] Demande de brevet 06055900 le 29 juin 2006 et demande 07 59234 le 22 novembre 2007 auprés de l'INPI.
[25] Widget - composant d'interface graphique.
[26] Snippet - quelques lignes de code source réutilisables.
[27] HTML Slidy - *Slide Shows in XHTML*, dévellopé par Dave Raggett, fondé sur S5 de Eric Meyer, il utilise exclusivement des standards du web (XHTML, CSS et JavaScript) [32].
[28] IDIAP - l'Institut Dalle Molle d'Intelligence Artificielle Perceptive, est un centre de recherche semi-privé basé à Martigny (CH) [16].

bilité d'utiliser la biométrie comme moyen d'authentification ; aux utilisateurs, l'usage de leur téléphone comme capteur biométrique. La première section du chapitre 4 décrit donc cette mise en œuvre, le service et l'architecture qui le sous-tend.

La difficulté et du même coup l'intérêt, de cet exercice réside dans l'indépendance de notre architecture vis-à-vis de la plate-forme biométrique vocale et de l'annuaire utilisé. Cet effort d'abstraction assure l'adaptation à moindre coût de notre service sur d'autres systèmes d'information, et surtout, nous permet d'envisager l'intégration d'autres biométries telles que la biométrie de la main ou du visage [Doublet *et al.*, 2006]. Ce service a fait l'objet d'une démonstration de l'architecture ITEA Aurora lors de la revue finale du projet européen en Juin 2006. Il a également été présenté les 13 et 14 octobre 2005, lors du sixième symposium annuel d'ITEA à Helsinki.

Le second service, mATM[29] a trait au domaine bancaire. Il a été réalisé dans le cadre des travaux du LATEMS[30]. Le LATEMS matérialise l'accord de partenariat passé entre le centre de recherche du groupe France Telecom, l'université de Caen et l'ENSI Caen[31]. Il bénéficie de la synergie des compétences de la recherche industrielle et publique et cristalise autour de plusieurs thématiques : les cartes à puce avec et sans contact ; la sécurité des échanges ; la reconnaissance biométrique ; l'évolution des usages ; la multiplicité des terminaux ; les réseaux monétiques.

FIG. 6 – mATM. Notre simulateur est composé d'un écran tactile (1), d'un lecteur de carte de crédit (2) et d'un tag NFC (3) permettant d'identifier le dispositif physique (la borne). L'interface graphique mATM est une interface web instanciée dans un navigateur. Elle reprend les caractéristiques d'un distributeur de billet. Elle est constituée de huit sélecteurs (4) disposés de part et d'autre d'un écran de sélection (5). Deux modules (6, 7) sont chargés de simuler la sortie des billets et du ticket. Un dernier module (8) reproduit le pavé numérique permettant d'interagir avec l'automate. Le tag NFC permet permet d'inclure le téléphone de l'utilisateur dans le session web. Le téléphone, disposant d'un lecture NFC, lit les informations concernant le service, déclenche un appel vers la plate-forme mVIP et intègre la session web. L'utilisateur peut alors interagir avec le distributeur en utilisant sa voix ou les touches DTMF de son téléphone.

mATM adresse la plupart de ces thématiques. Il met en œuvre un simulateur de distributeur automatique de billets (figure 6). Ce service s'adresse particulièrement à un public non ou malvoyant. En effet, en plus d'offrir un contrôle déporté sur le mobile de l'utilisateur, les oreillettes du téléphone nous permettent d'inclure dans le service un lecteur d'écran adapté à l'utilisateur. La seconde section du quatrième chapitre présente l'architecture mise en œuvre ainsi qu'une

[29] mATM - multimodal Automatic Teller Machine, Ditributeur de billets innovant [11].
[30] LATEMS - Laboratoire de Transactions Électroniques, de Monétique et de Sécurité.
[31] ENSI Caen - École Nationale Supérieure d'Ingénieurs de Caen.

description fine du scénario du simulateur. La figure 6 présente le simulateur et son interface graphique et la disposition des différents composants interactifs. La figure 7 illustre le dialogue entre l'utilisateur et le système. Ce simulateur a été l'objet de plusieurs présentations, en particulier, en juin 2006, à Issy-les-Moulineaux, il a été présenté au groupe France Telecom dans le cadre du salon d'été des résultats de la recherche du groupe. Ce simulateur a été adapté afin de rendre compte du service mKiosque[32] qui introduit cette thèse. Cette version a été présentée : le 3 octobre 2006, à Deauville, lors du septième Rendez-vous des Systèmes d'Information, Solution Internet, et Stratégies Innovantes [35] ; les 5 et 6 octobre 2006, à Paris lors du symposium annuel ITEA 2 ; en décembre 2006, au groupe et aux partenaires du groupe dans le cadre du salon d'hiver des résultats de la recherche de France Telecom. Actuellement, ce simulateur fait l'objet de présentations aux partenaires du groupe dans le cadre des « jardins de l'innovation » et son exposition permanente « Magasin Nouvelle Génération ».

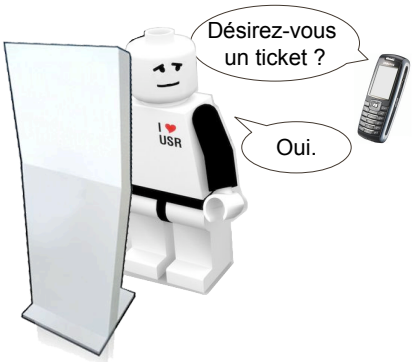
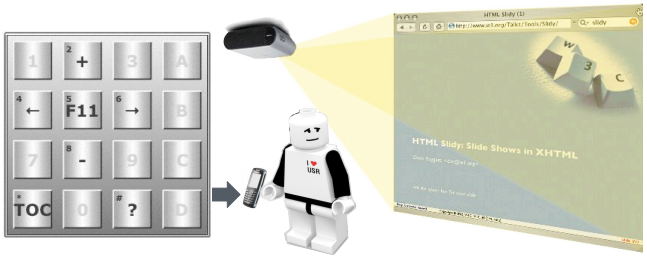


FIG. 7 – mATM, vocalisation d'écran par le téléphone mobile et utilisation de la voix pour interagir avec le service.

Le dernier démonstrateur illustre l'extension des services existants grâce à notre mécanisme de snippet. En octobre 2005, nous avons spécifié une architecture de service permettant de piloter une présentation (slide-show) depuis un téléphone mobile [Chuffart *et al.*, 2005]. Nous avons observé un vif intérêt de la part de le communauté pour notre architecture fondé sur les standards du web et les terminaux existants. Par la suite, nous avons implémenté ce démonstrateur en étendant la capacité d'interaction du logiciel de présentation Slidy.

FIG. 8 – Contrôle de Slidy depuis un téléphone mobile, . Les DTMF 2 et 8 modifient la taille de la police, 4 et 6 assure la navigation, 5 active le mode plein écran, * affiche la table des matières et # affiche un résumé de ces contrôles.



Slidy permet de réaliser des présentations grâces aux standards du web, le contenu des transparents est décrit en HTML, la charte graphique est factorisée dans une feuille de style CSS et le contrôle de la présentation s'effectue depuis le clavier et la souris par l'intermédiaire du gestionnaire d'événements du navigateur. Les événements pris en compte sont les suivants : se déplacer

[32] mKiosque - Kiosque multimodal, Service de diffusion de contenus multimédia innovant [11].

au slide suivant ou précédent grâce aux flèches droite et gauche du clavier ; une pression sur la touche C affiche la table des matières du document ; la touche F11 active ou désactive le mode plein écran ; les touches + et - modifient la taille de la police. La figure 8 montre l'association que nous avons réalisée entre les touches du téléphone et les événements de Slidy. mSlidy[33] démontre comment l'utilisation des standards du web permettent l'extension rapide des interfaces web existantes. Nous avons utilisé notre plate-forme pour étendre l'application Slidy et pour montrer la facilité avec laquelle le concepteur d'interface va pouvoir prototyper des services intégrant notre technologie.

Le mémoire se termine par une conclusion présentant le bilan, les limites et les perspectives de notre travail.

Hormis ce paragraphe, le reste de mon mémoire est écrit à la première personne du pluriel. Ce « nous » inclut les personnes qui m'ont dirigé, celles qui ont participé à mes travaux ainsi que vous, lecteur qui parcourez ces pages.

[33] mSlidy - Extensioin multimodale de l'application web Slidy HTML [11].

Chapitre 1

Fondements du réseau ubiquitaire

> *... une voix lisait le texte imprimé... Elle possédait un grand registre de tons, se faisait doctorale pour les ouvrages de philosophie, sèche pour les mathématiques, tendre pour les romans d'amour, grasse pour les recettes de cuisine.*
> Barjavel, « Ravage », 1943.

Le réseau ubiquitaire offre des possibilités d'utilisation élargie pour des usages en situation de mobilité de handicap ou de domotique. Cette valeur ajoutée passe par l'utilisation de plusieurs modes de communication venant compléter ou enrichir les interfaces existantes. De la *multimodalité* à l'*interaction homme machine* en passant par les *IHM*, la compréhension des enjeux du domaine passe par son étude préalable. La définition d'un ensemble de termes, de mécanismes et de concepts, l'étude des protocoles, langages et composantes logicielles sont les prérequis nécessaires à une description plus fine des modèles sous tendant la conception d'une plate-forme de services ubiquitaires. Ce chapitre présente les langages, protocoles et plates-formes utilisées au cours de nos travaux de recherche.

Dans la section 1.1 nous définissons les différents concepts d'interaction homme machine dont nous nous servons pour élaborer notre modèle. Nous différencions l'interaction, qui a pour but la réalisation de la tâche, du dialogue homme machine qui reconnaît à la machine des compétences langagières et le fait qu'elle coopère à la tâche. La section 1.2 réalise un tour d'horizon des modèles conceptuels d'architecture utiles à notre étude. Dans la section 1.3 nous explicitons comment le web est né et comment il est passé en vingt ans du web documentaire au web des services. Dans la section 1.4 nous réalisons un tour d'horizon des technologies web permettant la navigation vocale sur le web. Nous y découvrons des structures de contrôle permettant la définition de dialogues évolués respectant le modèle projectif du dialogue de Vernant [Vernant, 1992]. Dans la section 1.5 nous énumérons les différents langages permettant le transport et le contrôle des flux média à travers internet. Ce sont ces protocoles qui ont permis l'essort récent de la téléphonie sur IP reproduisant à bas coût les fonctionnalités de la téléphonie classique. Enfin, dans la section 1.6, nous décrivons les différents outils dont nous disposons pour réaliser notre plate-forme de services.

Parce que le réseau ubiquitaire repose sur la définition de standards, ce chapitre pose les fondements de notre travail, il explicite les différents protocoles et dialectes qui font le réseau ubiquitaire. Les différentes sections décrivent les fonctionnalités de ces formalismes et préfigurent leur assemblage décrit dans les chapitres suivants.

1.1 Interaction, interfaces et dialogue homme machine

Le domaine de l'interaction homme machine a pour principal but de fournir une interface rendant compte pour l'utilisateur du modèle de l'application et de ces fonctionnalités. Michel Beaudouin-Lafon [Beaudouin-Lafon, 2004] révèle trois paradigmes d'interaction homme machine :

- la machine comme un outil (*computer-as-tool*) qui étend les capacités humaines à travers la machine ;
- la machine comme un partenaire (*computer-as-partner*) qui se voit déléguer des tâches au travers un dialogue évolué ;
- la machine comme un medium (*computer-as-medium*) qui est un moyen de médiatiser la communication entre humains en vue d'un travail collaboratif.

Notre problématique adresse chacun de ces trois paradigmes. En effet, avec l'avènement du web, nous assistons à une « redocumentarisation du monde » [Pédauque et Sèdes, 2007]. Le contenu des bibliothèques deviennent accessibles en ligne. Ces documents sont composites, multimédias et intégrent à la fois du texte, des photos, des cartes mais aussi des flux audio et vidéo [Salaün, 2006]. Le web documentaire devient le web des services. Pour assurer le rendu de ces services, les interfaces web intègrent des composants interactifs. Ces composants constituent des outils qui, assemblés de manière cohérentes forment un dispositif de souscription adapté. Enfin, l'aide à la navigation ou la recherche de documents est assuré par des agents disposant de compétences langagières [Sabouret, 2002, Loisel *et al.*, 2005] et prenant la forme d'avatars 3D [Breton *et al.*, 2006].

Tandis que la définition d'outils relève du domaine des IHM, la définition de dialogues adresse clairement les problématiques de l'intelligence artificielle et des agents conversationnels.

Interaction web Le web s'est construit autour du protocole HTTP et de du langage HTML[34]. Le premier assure le transfert de documents sur le web. Le second permet de construire un maillage de documents répartis sur le web. La section 1.3.1 décrit ces deux standards et montre comment la simplicité et l'efficacité d'HTTP et le caractère déclaratif de HTML ont permis, à la fois au web de connaître un essor rapide, et en même temps, de supporter la charge provoquée par son expansion. Historiquement, les terminaux du web sont les ordinateurs personnels. De fait, le web s'est également construit autour de la modalité graphique. Dans les années 70, le centre de recherche Xerox de Palo Alto met au point la souris et l'interface graphique (*GUI*[35]) pour remplacer les traditionnelles consoles (*CLI*[36]) des ordinateurs. Le concept repris par Macintosh est aujourd'hui incontournable, on le retrouve sur les principaux systèmes d'exploitation de nos ordinateurs personnels.

Dans la section 1.3.1 nous montrons comment les standards du web assurent la définition d'outils interactifs permettant la navigation et l'interaction avec des services web.

Comme nous le voyions en introduction, la miniaturisation des terminaux mobiles a provoqué à la fois une modification des interfaces et l'introduction de nouveaux *media*, utilisant des *modalités* jusqu'alors peu exploitées dans les interfaces traditionnelles (vocales, gestuelles, tactiles).

[34] HTML - HyperTexte MetaLanguage [Rec. HTML, 1999].
[35] GUI - Graphic User Interface, aussi appelé WIMP pour Window Incon Menu Pointer.
[36] CLI - Command Line Interface.

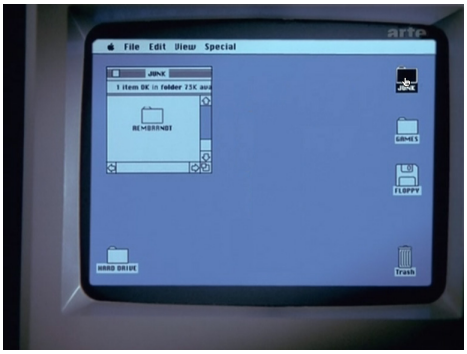


FIG. 1.1 – L'interface graphique selon Apple (*Les Pirates de la Silicon Valley*, Martyn Burke, 1999).
L'utilisateur dispose d'une *souris* permettant de déplacer le *curseur* et d'interagir avec des *icônes* et des *fenêtres*. L'interface graphique utilise les métaphores du bureau, des dossiers, des fichiers. Cette interface permet une approche plus conviviale fondée sur la référence aux *prés-requis socioculturels* de l'utilisateur.

Interaction vocale En 1991, l'ouvrage [Haton *et al.*, 1991] traitant de « la reconnaissance vocale » définit l'interaction vocale comme : *l'interaction entre un utilisateur et un système complexe par le biais d'un dialogue évolué, assurant la saisie de données pour des tâches de tri, d'inspection, de contrôle de qualité et plus généralement d'interaction avec des bases de données et permettant la commande de machines.* L'interaction vocale adresse donc, les trois paradigmes précédemment identifiés : le partenaire au travers d'un dialogue évolué, l'outil par un jeu de commandes vocales destinées à la réalisation d'un tâche et le médium permettant la navigation dans un grand ensemble de données.

Les technologies vocales recouvrent plusieurs activités complémentaires : la synthèse vocale, consistant à produire un signal sonore ; la reconnaissance vocale, consistant à faire interpréter par une machine une commande vocale produite par le locuteur afin de réaliser une action précise ; l'enregistrement vocal, consistant à acquérir et stocker un signal afin de le stocker ou de le transmettre à une machine ; la vérification du locuteur, activité dans laquelle on cherche à authentifier le locuteur par l'analyse de son empreinte vocale.

L'utilisation de la voix pour communiquer avec une machine présente plusieurs avantages. Elle s'impose dans le cas où les autres modalités de communication de l'utilisateur sont occupés. Par exemple dans le cockpit d'un avion de chasse, un pilote pourra utiliser sa voix pour marquer une position [Bouchet et Nigay, 2004]. Dans le cadre de l'accessibilité et de la prise en compte du handicap, le mode vocal permet de rendre accessible, aux personnes non-voyantes, les écrans d'ordinateur ou le contenu des bibliothèques [Lebert, 2001]. Dans le cadre de la réalisation de tâches sensibles, on peut utiliser le mode vocal pour confirmer une action ou pour authentifier l'utilisateur. Pour les utilisateurs non spécialistes d'un système, le mode vocal offre une assistance guidant l'utilisateur dans sa démarche. Enfin, pour l'utilisateur souhaitant accéder à distance à un système, le mode vocal permet d'interagir à distance de manière directe, dans une maison par exemple, ou de manière médiatisée, par un accès téléphonique.

Dans le section 1.4, nous voyons comment VoiceXML permet d'assembler des structures de contrôle de dialogue.

Interaction multimodale L'*interaction multimodale* se caractérise par l'utilisation de plusieurs *modes*, comme par exemple la voix, le geste ou le mode graphique, pour accéder à un service, utiliser un logiciel. En 1980, Bolt [Bolt, 1980] met en évidence l'existence de cette interaction et illustre ce phénomène avec l'instruction « Put-that-there »[37]. Dans cet exemple, le mode verbal référence l'objet (ça) et le lieu (ici) et l'action (mettre). Les déictiques (désignation) référençant l'objet et le lieu se voient complétés par un second mode permettant de rendre

[37] « met çà ici ».

compte de la disposition des objets dans l'espace.

Telles que les définissent Yacine Bellik, Daniel Teil, Laurence Nigay et Joëlle Coutaz au début des années 90 [Bellik et Teil, 1992, Nigay et Coutaz, 1993], les modes font références aux sens humains et véhiculent les intentions de l'utilisateur. Une application multimodale fait intervenir plusieurs modes.

Dans le section 3.4 nous spécifions une composante logicielle permettant d'appréhender la problématique de la coopération des modes.

Mode, modalité et média Les *modes* de communication font référence aux sens et aux moyens d'expression humains : le mode graphique fait référence à la vue ; le mode tactile fait référence au toucher ; le mode sonore fait référence à l'ouïe ; les modes gustatifs et odorants font référence au goût et à l'odorat. À ces références, s'ajoute la sensibilité somesthésique[38] qui définit la perception que l'individu possède de son corps. La prise en compte de la conscience de lui-même de l'utilisateur permet de prototyper des prothèses manipulables par la pensée [Millán *et al.*, 2007]. Le dispositif analyse l'activité électrique des terminaisons nerveuses.

Une *modalité* est une forme particulière d'un mode de communication. Une photo, un graphique et une carte sont trois modalités du mode graphique.

Considérons un *media* comme un dispositif physique avec lequel interagit l'utilisateur. Un magnétophone est un média, au même titre qu'un téléphone, un microphone, ou une caméra. Définissons un *système multimodal* comme étant capable, d'intégrer plusieurs modes de communication, et par extension plusieurs modalités. L'objet du système multimodal est l'interaction, C'est ce qui différencie un système multimédia (diffusion) d'un système multimodal (interaction) [Coutaz et Caelen, 1991]. Le système multimodal distingue ces modalités d'entrée captées par des dispositifs de capture, et ces modalités de sortie véhiculant l'information structurée [Truillet *et al.*, 2000] sur des dispositifs de lecture permettant par exemple « d'afficher » des images tactiles [Lécuyer *et al.*, 2004, 23].

Interaction verbale et dialogue homme machine L'interaction verbale se situe donc dans le paradigme « computer as tool », les commandes verbales permettent la manipulation d'une interface. Le dialogue homme machine adresse dans le paradigme « computer as a partner » [Villaseñor-Pineda, 1999] c'est-à-dire qu'il reconnaît à la machine des compétences langagières et qu'elle coopère à la tâche. L'interaction verbale reste centrée sur l'interaction et a pour but la réalisation de la tâche. Le dialogue homme machine est un champ d'exploration de l'intelligence artificielle. Il confère aux agents logiciels des intentions, des émotions [Ochs *et al.*, 2005]. Sa mise en œuvre repose à la fois sur une représentation formelle des connaissances et sur le raisonnement d'agents rationnels utilisant des modèles de planification pour résoudre le problème de la coordination d'actions. L'étude du dialogue homme machine nous révèle une adaptation forte du dialogue au métier et au but que le concepteur de service cherche à adresser. Par exemple, la modélisation d'un service d'information voyageur en dialogue naturel nécessite une représentation des concepts d'espace, de lieu, de chemin une définition des buts du dialogue et la spécification d'une ensemble de méthodes permettant de l'atteindre [Bazin *et al.*, 2006].

Dans la section 1.4 nous montrons comment VoiceXML assure la conception d'outils permettant l'interaction verbale. Nous montrons également comment les structures VoiceXML permettent la définition de stratégies de dialogue selon le modèle projectif du dialogue de Vernant [Vernant, 1992].

[38] Somesthésie - (n.f.) conscience du corps [Berube, 1991]. source : [18].

1.2 Aspects conceptuels en IHM

La conception d'une plate-forme de services ubiquitaires soulève deux principaux problèmes. Tout d'abord, les services ubiquitaires donnent lieu à la réalisation de tâches qui exploitent les infrastructures de communication existantes. De ce fait, les interfaces de ces services doivent exploiter les systèmes d'informations, faire appel aux primitives du système et utiliser les protocoles prédéfinis tout en tenant compte de la pérennité des infrastructures de communication. Ensuite, les services se doivent d'être accessibles par des utilisateurs. De ce fait, l'interface fait référence aux prérequis socioculturels de l'utilisateur. Les interfaces utilisent un langage sémiologique permettant de notifier à l'utilisateur les possibilités offertes par le service. L'interface suggère à l'utilisateur le fonctionnement du service, les moyens offerts pour y accéder, l'enchaînement des différentes commandes dans le dessein de rendre compte du service. Ainsi, on distingue le modèle conceptuel de l'architecture de service – dont le but est d'offrir à l'utilisateur une interface en cohérence avec les services qu'elle offre – de l'architecture d'implémentation qui assure une intégration adaptée à l'infrastructure de communication cible. L'IHM se trouvant aux confins du monde de l'utilisateur et du monde du système [Nigay, 2006], la prise en compte des aspects conceptuels et implémentationnels est nécessaire à la résolution de notre problématique.

La littérature nous révèle plusieurs modèles conceptuels d'architecture logicielle d'applications interactives. La section suivante présente les modèles d'architecture nécessaires à la compréhension de notre étude. Tandis que le modèle de Seeheim [Pfaff, 1985] et le modèle Arch [SIGCHI, 1992] sont purement conceptuels, le modèle MVC [Krasner et Pope, 1988] trouve son origine dans l'utilisation de langage de programmation objet. Le modèle PAC [Coutaz, 1987], quant à lui, est orienté agent : dans PAC, une hiérarchie d'agents est en charge d'assurer l'interface entre le système et l'utilisateur. Les patrons PAC et MVC se distinguent par l'absence pour MVC de la couche de médiation en charge de la communication entre la partie métier de l'application et sa partie interface. On peut noter une variante de MVC : M2VC, laquelle étend MVC en y ajoutant une seconde facette modèle assurant le rôle de cette couche de médiation. De ce fait, M2VC rend compte du patron PAC. Enfin, citons les modèle hybrides PAC-Amodeus [Nigay, 1994] et H4[Depaulis *et al.*, 2006] qui combinent les architectures conceptuelles Arch et des hiérarchies d'agents en charge de rendre compte des différents niveaux d'abstraction du modèle Arch. C'est d'ailleurs pour cette raison qu'ils sont qualifiés de modèles hybrides.

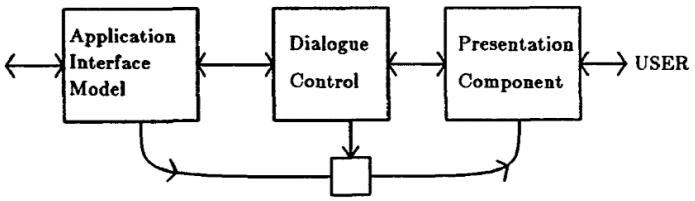


FIG. 1.2 – Modèle de Seeheim [Pfaff, 1985].

Modèle de Seeheim Le modèle de Seeheim [Pfaff, 1985] est un modèle conceptuel de système interactif. Il a été introduit en 1983 lors de l'atelier organisé par Eurographics à Seeheim (Allemagne) sur le thème « système de gestion d'interface utilisateur ». Le modèle de Seeheim décompose l'architecture logicielle en trois couches (figure 1.2) :

1. la couche de *présentation* est la composante lexicale du modèle, elle capture les commandes

de l'utilisateur et lui assure un retour sur l'état du système ;

2. le *contrôleur de dialogue* analyse la structure syntaxique des échanges, elle assure le dialogue entre ses couches adjacentes ;
3. l'*interface applicative*, composante sémantique du modèle, assure la communication entre le noyau fonctionnel du système et l'interface.

Le modèle de Seeheim a la particularité de séparer l'interface interactive de l'application et le cœur de l'application qui intègre les composantes métier du système. La figure 1.2 présente ces trois couches et leur organisation. Il est à noter que la couche sémantique ne communique pas avec la couche lexicale. Dans sa version originale, le modèle de Seeheim intègre une quatrième composante en charge de la délégation sémantique. Ce composant assure la remontée des informations sémantiques au niveau de la couche de présentation du modèle. C'est cette composante qui fournira à l'interface utilisateur les éléments propres au domaine de l'application. En effet ces informations métiers ne peuvent se trouver que dans le noyau fonctionnel de l'application puisque qu'elle sont propre au domaine concerné.

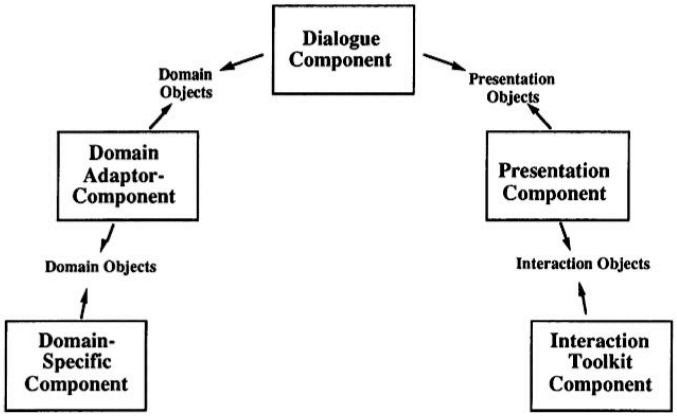


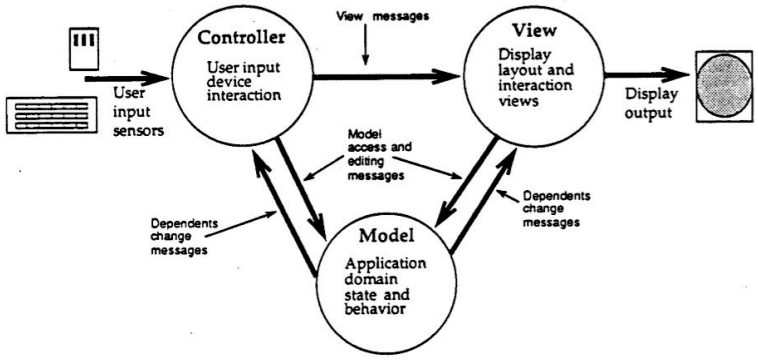
FIG. 1.3 – Modèle Arch [SIGCHI, 1992].

Le modèle Arch Le modèle Arch est un modèle conceptuel d'interaction homme machine. Il a été défini en 1991 par le « CHI'91 UIMS Tool Developers' Workshop » [SIGCHI, 1992]. Il reprend les termes introduits par le modèle de Seeheim. Contrairement au modèle de Seeheim, le modèle Arch adresse l'ensemble de l'application. En effet, ce modèle intègre le noyau fonctionnel de l'application dans le modèle d'architecture. Ainsi, le modèle Arch est composé de 5 éléments :

1. le *noyau fonctionnel* (la *composante métier*), qui conceptualise le cœur du système est en charge de fournir au système global les objets métiers ;
2. l'adaptateur sémantique va formater les objets métiers de manière à les rendre compatible avec la couche syntaxique ;
3. la composante de dialogue (pour reprendre la terminologie de Seeheim) garde les fonctionnalités décrites dans Seeheim, elle assure la communication entre les objets métiers, et les objets interactions gérés par la couche lexicale du modèle. La composante de dialogue contrôle les accès au noyau fonctionnel de l'application selon une stratégie de dialogue ;
4. la quatrième couche correspond à la partie logique de la présentation de Seeheim, elle assure la communication avec le contrôleur de dialogue ;

5. la cinquième couche correspond à la partie physique de la présentation de Seeheim, elle est perceptible par l'utilisateur, c'est une bibliothèque de primitives haut niveau permettant la manipulation de boîtes de dialogue et autres composants interactifs évolués en charge de capter les actions de l'utilisateur et de lui fournir un retour sémantique des actions réalisées. L'utilisation d'une boîte à outils permet de prédéfinir et de réutiliser un ensemble de composants. Elle assure le développement rapide d'applications interactives.

Patron « Modèle Vue Contrôleur » Le patron MVC[39] fut utilisé dans le système Smalltalk pour implémenter des interfaces graphiques[Krasner et Pope, 1988]. Il est adapté à la gestion d'objets communicants de manière événementielle. Le modèle constitue la partie métier du système. Il intégre la composante sémantique du système. Il se distingue de la vue qui assure le rendu du modèle, rendant le modèle perceptible par l'utilisateur sur les périphériques de sortie. Le contrôleur modifie le modèle, et assure la communication entre le modèle, ses vues associées et les périphériques d'entrée. Chacune des vues est associée à un seul contrôleur, possédant un seul modèle. En revanche, le modèle peut posséder plusieurs vues et plusieurs contrôleurs. La figure 1.4 représente ce patron et ses composantes.



Le patron de conception MVC permet de modéliser le système comme un agent autonome interactif composé de trois objets :
- le *Modèle* représente l'état interne de l'agent ;
- le *Contrôleur* est en charge de modifier l'état interne de l'agent ;
- la *Vue* fournit une représentation de l'état interne de l'agent.

FIG. 1.4 – patron MVC [Krasner et Pope, 1988] et ses trois composantes.

MVC dans les applications web : MVC2 Dans le cadre de l'interaction web, la répartition à travers le réseau du client et du serveur à longtemps imposé un accès séquentiel aux applications web. La séquentialité de l'interaction provient de la nature client serveur du protocole HTTP. La séquence d'interaction est généralement la suivante : requête utilisateur, traitement de la requête, construction de la réponse, envoie de la réponse. Les trois dernières étapes sont exécutées sur le serveur web. De nombreux systèmes de développement d'applications web utilise le patron MVC pour réaliser cette séquence [37, 30]. Le traitement de la requête côté serveur par le contrôleur central et unique (MVC2 [37]) donne lieu à la mise à jour ou l'interrogation du modèle : par exemple une base de données, un annuaire ou un serveur de messagerie. Suite à ce traitement le serveur construit la réponse qui sera fournie à l'utilisateur : la vue. La vue prend la forme d'un document, généralement HTML, construit à partir des informations du modèle. La vue ainsi construite est retournée au client qui l'instancie. On retrouve dans cette échange la délégation sémantique du modèle de Seeheim (figure 1.2). En effet, les informations propres au domaine

[39] MVC - Model View Controller, patron de conception permettant de modéliser le système comme un agent autonome interactif.

sont encapsulées dans la structure de document transmis. Dans les versions 0.9 et 1.0 de notre modèle d'architecture UbiArch nous tenons compte de cette séquentialité de l'interaction et de cette délégation sémantique (sections 2.3.1 et 2.3.2 pages 56 et 57).

Si cette vision macroscopique fonctionne très bien dans le cadre des applications web classiques, elle se heurte à la complexité des applications internet riches (RIA[40]). En effet, dans le cadre des RIA, la décomposition atomique de l'interface et surtout les facultés qu'ont les composantes de l'interface à exploiter le protocole HTTP, modifie la donne. Chaque composant de l'interface graphique va pouvoir chercher des informations propres au domaine sur le réseau sans pour autant avoir à reconstruire l'ensemble de la vue. Une fois récupérées par le client web, ces informations vont être traitées, mises en forme et vont être à l'origine des modifications partielles de l'interface. On observe ici un changement de nature de l'interaction. Ce changement de nature de l'interaction s'accompagne d'un changement de nature des données échangées. Si dans le cadre des applications web classiques, le formatage des données échangées sur le réseau intègre une structure tenant compte de la mise en forme future du document ; dans le cadre des RIA, les informations échangées sont uniquement propres au domaine. Ce sont les composantes de l'interfaces qui vont traduire cette information et adapter leur apparence et leur comportement en fonction. Ainsi, dans ce contexte, la vue est déplacée côté client qui n'est plus un simple formateur, mais devient interactif. Nous détaillons les technologies sous-jacente au fonctionnement des RIA dans la section 1.3.2 page 28. Afin de centraliser sur client web le composant en charge d'effectuer les échanges sur le réseau, notre modèle d'architecture web UbiArch dans sa version 2.0 (section 2.3.3 page 59) intègre une hiérarchie d'agents. C'est l'élément racine de cette hiérarchie qui prend à sa charge les communications sur le réseau, c'est lui qui priorise certains échanges par rapport à d'autres, effectue des mises en cache sur le navigateur et anticipe les requêtes à venir lorsqu'il n'est pas trop sollicité par le reste de l'interface. Notre modèle d'architecture UbiArch 2.0 rend ainsi compte de ce nouveau type d'applications web et optimise leur fonctionnement.

Modèle et Agents PAC Le modèle PAC (Présentation Abstraction Contrôle) [Coutaz, 1987, Coutaz, 1997] permet une implémentation agent du modèle de Seeheim assurant un retour sémantique immédiat. Les agents PAC sont constitués de trois facettes (figure 1.5) :

- la facette *Abstraction* contient la sémantique propre à l'agent afin d'assurer un retour sémantique immédiat ;
- la facette de *Présentation* est en charge de ce retour sémantique immédiat mais aussi du retour sémantique provoqué par le changement d'état du système global ;
- la facette *Contrôle* est en charge de la communication entre les deux couches précédemment évoquées, mais aussi de la communication entre les agents PAC dans le but d'assurer le retour sémantique global.

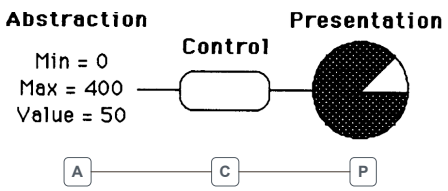


Fig. 1.5 – Modèle PAC [Coutaz, 1987].

[40]RIA - Rich Internet Applications.

Les agents PAC s'exécutent sur des threads distincts. La communication entre agents PAC s'effectue par échanges de messages. La connexion des interfaces de Contrôle des agents PAC s'effectue de manière hiérarchique. La propagation hiérarchique des messages entre les agents PAC assure la répartition du retour sémantique sur l'ensemble des agents concernés[41]. La figure 1.6 illustre la hiérarchie PAC. À gauche, on remarque l'emboîtement des niveaux de présentation et la répartition des informations sémantiques dans la hiérarchie PAC.

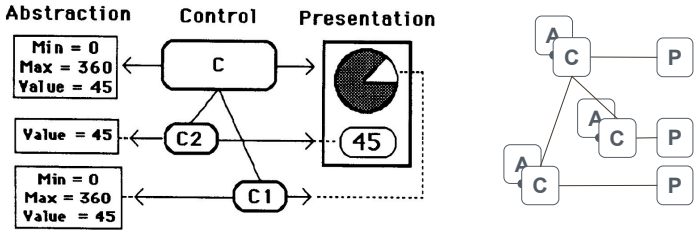


Fig. 1.6 – Hiérarchie PAC [Coutaz, 1987].

Modèles Hybrides Le modèle hybride PAC-Amodeus (figure 1.7) combine le modèle Arch et le modèle PAC. En utilisant une hiérarchie d'agents PAC pour rendre compte des fonctionnalités du contrôleur de dialogue, le modèle PAC-Amodeus combine les atouts du modèle PAC – à savoir : être en mesure de fournir un retour sémantique immédiat à l'utilisateur grâce à une segmentation des éléments de l'interface utilisateur – et la structure laminaire du modèle Arch permettant de traiter les échanges entre l'utilisateur et le système. Initialement, le modèle PAC-Amodeus a été conçu pour rendre compte de l'interaction multimodale. La structure hiérarchique du maillage d'agents PAC permet l'agrégation des événements utilisateur (modalités d'entrée) et la distribution des notifications du système sur les différents éléments qui composent l'interface, selon des modalités perceptibles par l'utilisateur (modalité de sortie).

Tout comme PAC-Amodeus, le modèle H4 (figure 1.8) est un modèle hybride. H4 intègre quatre hiérarchies pour rendre compte des couches du modèle Arch. Le contrôleur de dialogue se décompose, lui aussi, en une hiérarchie permettant une modélisation simple d'un dialogue structuré. Le modèle H4 se destine à la conception d'applications de CAO. À ce titre, le modèle exploite finement les primitives bas niveau de la carte graphique. Cette particularité liée au domaine auquel s'adresse le modèle a poussé les auteurs du modèle à réhabiliter la délégation sémantique présente dans le modèle de Seeheim. Ainsi, dans H4, l'adaptateur de noyau fonctionnel attaque directement l'adaptateur de présentation sans passer par la couche de dialogue. Nous verrons comment les versions 0.9 et 1.0 de notre modèle UbiArch reprennent ce principe de délégation sémantique et sont donc comparables au modèle H4. Les raisons qui nous pousse, nous aussi, à exploiter la *petite boîte* de la figure 1.2 trouvent leurs origines dans la spécificité du domaine abordé : le web et plus particulièrement le protocole HTTP.

Synthèse sur les aspects conceptuels en IHM Cette section réalise un tour d'horizon des modèles conceptuels d'architecture utiles à notre étude. Nous fondons notre modèle implémentationnel d'architecture sur les modèles conceptuels décrits dans cette section et au regard des réflexions citées. Dans le chapitre suivant, nous décrivons notre modèle d'architecture UbiArch.

[41] IntrosPAC - outil graphique permettant de voir transiter les messages de proche en proche de la hiérarchie PAC[Lachenal et Coutaz, 2003].

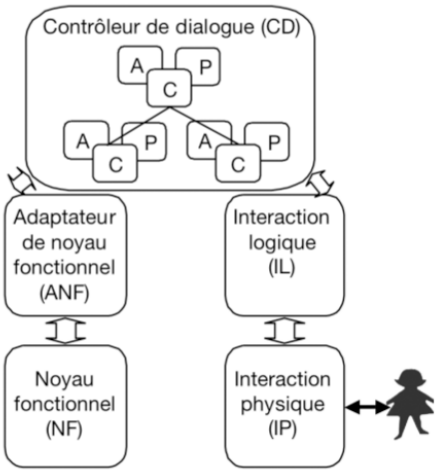


FIG. 1.7 – Modèle PAC-Amodeus [Nigay, 1994].

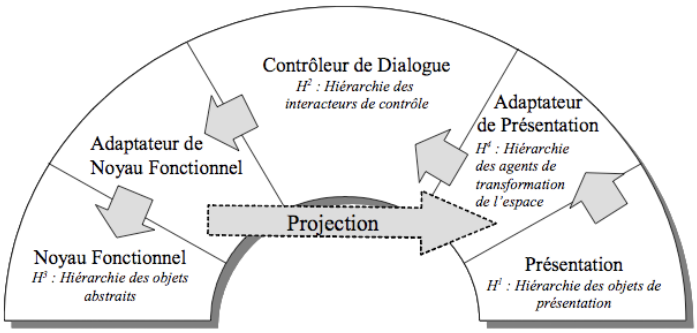


FIG. 1.8 – Modèle H4 [Depaulis *et al.*, 2006].

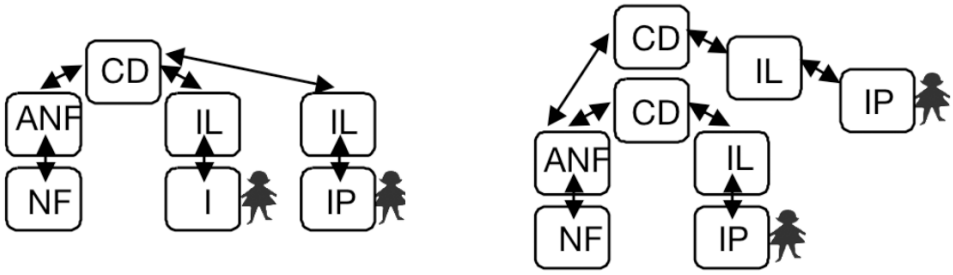


FIG. 1.9 – Mécanisme de branche du modèle Arch [Nigay, 2006].

UbiArch est destiné à la conception de services ubiquitaires intégrant des interfaces multimodales distribuées. Dans l'optique de traiter la problématique de la multimodalité, UbiArch reprend les principes fondamentaux du modèle PAC-Amodeus : il est fondé sur le modèle Arch et intègre une hiérarchie d'agent PAC pour modéliser la composante de dialogue. Contrairement à PAC-Amodeus, UbiArch est une architecture implémentationnel dédiée au web et en mesure de prendre en compte l'interaction vocale. De plus, UbiArch hérite des propriétés de Arch et notamment du mécanisme de branche. Le mécanisme de branche de Arch permet de dupliquer ces différents niveaux d'abstraction (figure 1.9). Le mécanisme de branche a été utilisé dans le cadre du modèle d'architecture Clover [Laurillau, 2002] afin de rendre compte de travail collaboratif assisté par ordinateur ; les utilisateurs sont répartis dans le réseau et communiquent au travers un espace de travail médiatisé par une machine. Avec UbiArch, nous utilisons cette propriété afin de rendre compte de la distribution de l'interface dans l'environnement de l'utilisateur. Cependant la compréhension de notre étude passe par un ensemble de prérequis concernant les technologies web. C'est l'objet des sections qui termine ce chapitre.

1.3 Infrastructure du réseau ubiquitaire, du web au « web 2.0 »

Parce que le web s'est construit autour de consensus entre les différents acteurs du domaine, nous explicitons l'ensemble des technologies qui ont permis son expansion. Cette partie décrit les langages et protocoles du web ubiquitaire. Promu principalement par deux instances de normalisation : le *W3C* et l'*IETF*, le web ubiquitaire est défini par un ensemble de technologies rendues cohérentes par des consensus industriels et l'établissement de standards. Cette partie présente les principaux acteurs de cette normalisation ainsi que les travaux de ces différents groupes de travail.

Point de vue du W3C Le W3C est le principal acteur de la normalisation d'internet. Il a été fondé par Tim Burners-Lee en octobre 1994. Le W3C regroupe, à travers le monde, des industriels (IBM, Cisco, Nuance, Voxeo...), des instituts (MIT, ERCIM, INRIA...) des universités (Keio University). Les activités du W3C sont regroupées par domaines. *W3C Ubiquitous Web Domain* est le domaine adressé par notre problématique. Il regroupe quatre activités : *Mobile Web Activity*, traitant du déploiement d'internet sur les terminaux mobiles ; *Multimodal Interaction Activity* traitant de la multimodalité ; *Voice Browser Activity* traitant de l'accès vocal à internet et *Ubitiquitous Web Applications Activity* traitant des applications du réseau ubiquitaire. Ce groupe de travail produit depuis 2001 des recommandations en terme d'ingénierie XML, d'architecture de plate-forme dans le but de construire le réseau ubiquitaire.

Point de vue de l'IETF L'IETF est un consortium industriel qui développe et promeut des standards pour internet. Il a été fondé en janvier 1986 par des chercheurs du gouvernement des État Unis. En octobre 86, des partenaires industriels rejoignent le consortium et depuis l'IETF est devenu un consortium ouvert à tous les contributeurs. L'IETF fonctionne principalement par mail et produit principalement les *RFC*[42]. L'IETF est organisé par groupes de travail. Le groupe de travail concerné par notre étude est le *Network Working Group*. C'est ce groupe de travail qui a notamment spécifié, en 1989, le protocole *TCP/IP*[43] sur lequel repose internet.

[42] RFC - Request for Comments.

[43] TCP/IP - Transmission Control Protocol / Internet Protocol, séparation de la couche de transport et de la couche applicative du modèle OSI [RFC 1122, 1989].

1.3.1 HTTP, HTML et les URL : les fondements du web

Inventé dans les années 60 aux États Unis, internet représente un maillage entre plusieurs ordinateurs reliés entre-eux sans contrôle central. Il faut attendre les années 90 pour qu'internet devienne un outil de publication documentaire à l'échelle mondiale. Des années 1990 à nos jours internet et ses utilisateurs ont intégré le concept de service en même temps que celui de paiement. Aujourd'hui, le « web 2.0 » intègre ces concepts dans le processus d'interaction.

HTTP [44] est un protocole client-serveur (figure 1.10) développé pour le réseau internet. Spécifié par Tim Berners-Lee en 1990 dans sa version 0.9, il devient 1.0 en mai 1996 [RFC 1945, 1996] et 1.1 en janvier 1997 [RFC 2068, 1997]. La définition de la version 1.1 se verra complétée en juin 1999 [RFC 2616, 1999]. Les évolutions successives du protocole HTTP ont permis au web de supporter la charge relative à son expansion. Ce protocole repose sur une architecture client serveur, il repose sur un ensemble exhaustif de méthodes[45] dont les plus utilisées sont GET et POST. La méthode GET demande une ressource, la méthode POST envoie un ensemble de variables à une ressource. Les méthodes HTTP permettent de manipuler les ressources désignées par les *URL*[46]. Chaque requête donne lieu à une réponse codifiée. Les diagrammes de séquence de la figure 1.10 montrent comment le client et le serveur communiquent.

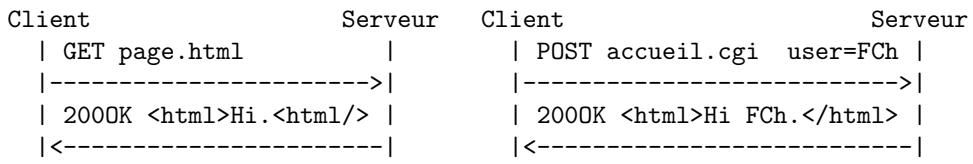


FIG. 1.10 – Diagrammes de séquence de l'échange client serveur HTTP. Les deux méthodes de ce protocole les plus souvent utilisées sont les méthodes *GET* (à gauche) et *POST* (à droite). Ces deux méthodes permettent respectivement de demander une ressource désignée par son *URL* (*page.html*, *accueil.cgi*) et d'envoyer un ensemble de variables (*user=Florent*). La requête donne lieu à une réponse codifiée (*200OK* pour succès) et à l'échange de données structurées.

HTML À l'origine inventé pour permettre l'échange d'information entre les chercheurs, le protocole HTTP tire sa richesse du métalangage *HTML*. Le HTML est un langage de balisage permettant de structurer un document. Le langage HTML permet de décrire la hiérarchie d'un document et d'y insérer des pointeurs vers d'autres documents grâce à la manipulation d'ancres et d'URL. On parle alors de liens *hypertextes* et d'*hyperdocuments*. Le parcours d'un document hypertexte n'est pas que séquentiel. Le lecteur a la possibilité de parcourir l'information d'ancres en ancres, à la manière de références pointant vers des ressources [Turbout, 2002]. Notons que l'espace des URL est un sous ensemble de l'espace des *URI*[47]. Dès 1990, les trois briques HTTP, HTML et les URL préfiguraient le web tel un espace documentaire.

[44] HTTP - HyperText Transfer Protocol.
[45] Méthodes HTTP : GET POST HEAD OPTIONS CONNECT TRACE PUT DELETE
[46] URL - Uniform Resource locators [RFC 1738, 1994].
[47] URI - Uniform Resource Identifiers [RFC 2396, 1998, Rec. URI URL URN, 2001].

```
<html>
  <head>
    <title>Exemple de code HTML</title>
  </head>
  <body>
    <h1 id="header">Exemple de Code HTML</h1>
    <h2>Les paragraphes</h2>
      <p>Le HTML permet de structurer l'information.
         Il manipule les concepts de titre, sous-titre,
         paragraphe, liste...
      <p>Notez qu'en HTML les balises ne doivent pas
         obligatoirement être fermées. Ce langage est
         destiné à être écrit par des humains ou
         généré par des machines.
    <h2>Les ancres et les listes</h2>
      <ul>Voici une liste de liens :
        <li><a href="http://wikipedia.org">Wikipédia</a>
        <li><a href="http://www.info.unicaen.fr">Site web
            du département d'informatique de l'université
            de Caen</a>
        <li><a href="mailto:fchuffar@info.unicaen.fr">
            Écrivez-moi !</a>
      </ul>
  </body>
</html>
```

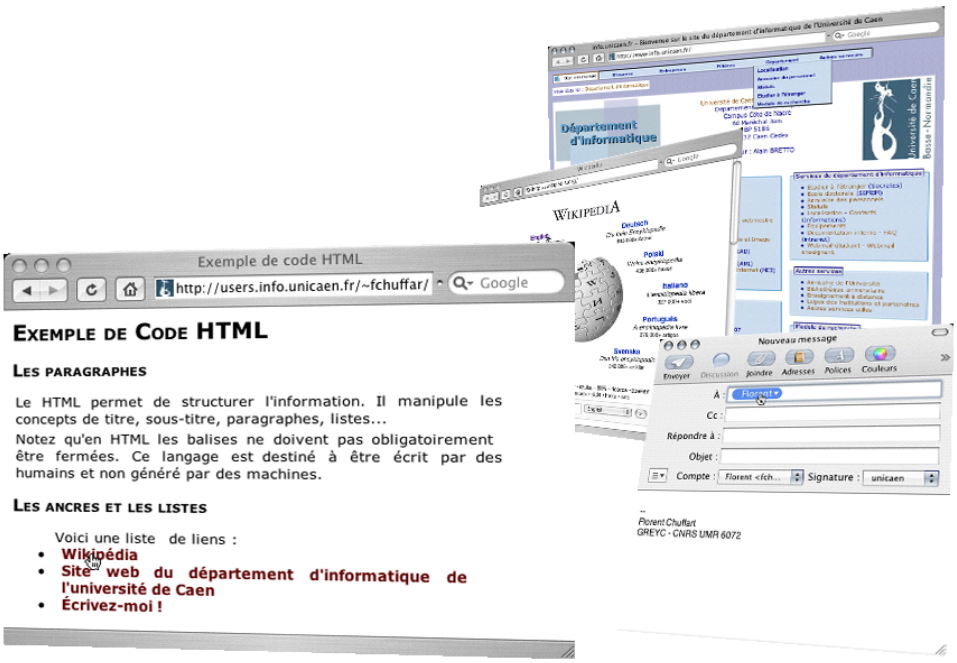


FIG. 1.11 – Exemple de code HTML. Le document HTML possède une entête contenant des informations sur son contenu (métadonnées), un corps contenant de l'information structurée et des liens hypertextes. L'information est structuré par ses titres (*h1*, *h2*), ces paragraphes (*p*), ces listes (*ul*), ces items (*li*) et ces ancres *a*. Les ancres permettent l'accès à des ressources hétérogènes. Dans l'exemple, les ancres pointent vers deux documents composites et le dernier ouvre le client de messagerie de l'utilisateur.

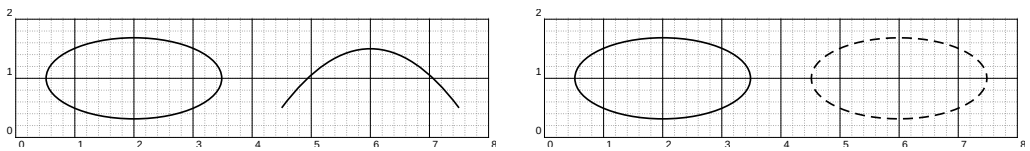
1.3.2 XHTML, DOM, CSS et javaScript : les briques du web 2.0

L'augmentation des débits, la connexion permanente au réseau internet et l'évolution des navigateurs s'accompagnent d'une évolution des techniques présentes sur internet. Cette section explicite les techniques qui font le succès du *Web 2.0*.

XHTML Le métalangage *XHTML*[48] [Rec. XHTML, 2002] reprend les structures manipulées par HTML à ceci près qu'il respecte la *DTD*[49] de *XML*[50]. La rigidité de XHTML implique sa génération par des processus informatiques. En effet le document XHTML n'est pas destiné à être écrit par un humain (il y a le HTML pour cela !). Par sa « XML validité », le document XHTML est plus facilement parsé par les navigateurs ou les robots. Cette remarque prend tout son sens sur les terminaux mobiles qui ne disposent pas de beaucoup de ressources.

DOM [51] [Rec. DOM, 1998] est une représentation globale du document XML. Le DOM peut être représenté par un arbre et offre la possibilité de naviguer dans un document par ses nœuds. Il autorise de naviguer vers les nœuds parents, frères et fils. Souvent opposé à *SAX*[52] qui est moins coûteux en temps et en mémoire pour un « parsing » en une passe, DOM se révèle être plus efficace pour des accès répétés et non séquentiels aux éléments du document. *DOM* est une instance du document regroupant les données, leur représentation et leur comportement.

CSS [53] [Rec. CSS, 2006] est un langage permettant de décrire une feuille de style à appliquer sur un document XML. La figure 1.12 illustre simplement le concept de style. La feuille de style CSS permet de séparer les données du document XML de leur représentation. CSS manipule le DOM prend. En effet, la feuille de style CSS décrit les attributs de style d'un ensemble de nœuds référencés par leur classe. C'est au moment du parsing du DOM que le processeur va surcharger les attributs des nœuds par leur valeurs décrites dans la feuille de style. Dans sa version 2.1, CSS [Rec. CSS, 2006] prend en compte les médias cibles (navigateur classique , lecteur d'écran, lecteur braille, terminal mobile...) afin d'assurer l'accessibilité du document (figure 1.13). Le CSS permet la séparation des données et leur représentation. La figure 1.13 présente un exemple de code CSS et montre comment les attributs de style traitent des classes particulières de média. La figure 1.14 montrent une utilisation avancée des feuilles de style CSS. Chacune de ces interfaces structure strictement le même contenu HTML grâce à des feuilles de styles CSS différentes.



(a) Un style, deux figures. (b) une figure deux styles.

FIG. 1.12 – Illustration du concept de style. Le graphique 1.12(a) présente deux figures : une ellipse et une parabole, toutes deux dessinées dans le même style : trait plein. Le graphique 1.12(b) présente une ellipse dessinée avec deux styles différents : trait plein et pointillés.

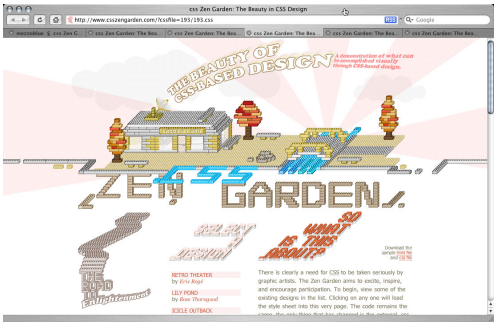
[48] XHTML - Extensible HyperText Markup Language.
[49] DTD - Document Type Definition, syntaxe du document.
[50] XML - eXtensible Markup Language [Rec. XML, 2006].
[51] DOM - Document Object Model.
[52] SAX - Simple API for XML.
[53] CSS - Cascading Style Sheets.

```
/* Choix de la police ***************************/
a, p, ul, li { font-family: Bitstream Vera Sans; }

/* codification des erreurs en fontion du media */
@media screen { .erreur { color: red;    } }
@media aural  { .erreur { volume: loud; } }

/* arriére plan de la div header ****************/
#header{ background-image: url('logo.gif'); }
```

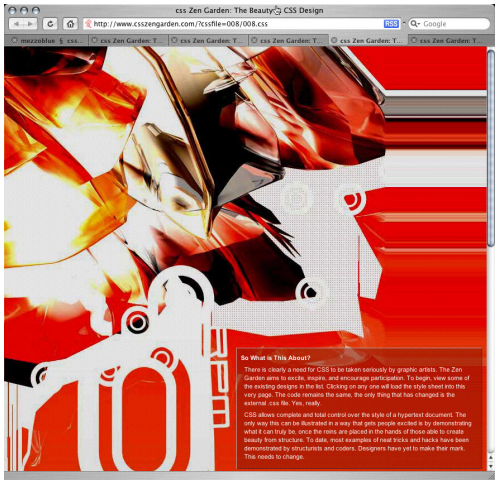
FIG. 1.13 – Exemple de feuille de style CSS. Le premier bloc fait référence à l'ensemble des balises *a*, *p*, *ul* et *li*. Il définit la police à utiliser pour représenter ces éléments comme étant *Bitstream Vera Sans*. Le second bloc illustre l'utilisation des fonctionnalité CSS 2.1.

Les éléments de la classe *erreur* sont définis de la couleur rouge pour le média graphique, et d'un volume « fort » pour leur rendu respectivement sur écran et par un dispositif de synthèse vocale. Enfin, le dernier bloc définit l'arrière plan de la section identifiée par *header*.

(a) prét-à-porter Minz Meyer, Germany.



(b) Leggo My Ego Jon Tan, United Kingdom



(c) RPM Bruno Cunha, United States.

(d) 45 RPM Thomas Michaud, United States

FIG. 1.14 – CSS Zen Garden, utilisation avancée des feuilles de styles [12]. CSS Zen Garden est un exercice de style montrant la puissance des feuilles de style. Outre l'image fond, les feuilles de style CSS sont en mesure de redéfinir les lettrines, le colonage ou encore la police, la mise en page et l'emplacement des menus, le tout avec le même contenu HTML.

ECMAScript *JavaScript* est le nom de l'implémentation du langage de script *ECMAScript* [Spec. ECMA, 1999] par Netscape Communications Corporation et maintenant par la Mozilla Foundation [24]. Il existe d'autres implémentations d'ECMAScript, notamment *JScript* par Microsoft. ECMAScript est un langage prototypé inventé par Brendan Eich (Netscape). La principale utilisation de ECMAScript est le traitement d'informations dans les applications web côté client (voir la figure 1.10). ECMAScript permet la manipulation du DOM [RFC 4329, 2006]. En effet, le navigateur possède une instance ECMAScript du DOM du service. Les nœuds du DOM sont des objets ECMAScript et possèdent des méthodes permettant d'accéder, de modifier et d'effectuer des traitements sur les éléments du DOM. La réponse du serveur permet ainsi de modifier une partie du document affiché sans recharger totalement la page. L'objet *XmlHttpRequest* a récemment rendu populaire l'utilisation de JavaScript. Cet objet permet d'effectuer des requêtes HTTP, de traiter les réponses et de modifier le DOM en conséquence. Le format des réponses obtenues est de plusieurs natures. Il peut être au format texte, reprendre les syntaxes XML ou *JSON*[54]. Cet objet est la pierre angulaire de la technologie *AJAX*[55] et du *web 2.0*. Le web 2.0 est donc un terme technique désignant le bon usage des technologies XHTML, DOM, CSS et JavaScript, mais c'est aussi un terme marketing désignant les nouveaux usages de l'internet. Un exemple d'application typique du web 2.0 est webmail « gmail » développé par Google. Cette technologie permet de développer des applications hautement interactives sans déploiement spécifique sur les machines des utilisateurs, hormis le navigateur web.

1.3.3 Synthèse sur le web

L'évolution des technologies web depuis 20 ans a provoqué un profond bouleversement de son utilisation. Le réseau internet est devenu une plate-forme de service et d'échange répartie à l'échelle mondiale. La construction et l'expansion du web sont principalement fondées sur les technologies que nous venons de décrire. C'est l'adoption par les industriels, les fournisseurs de services et les éditeurs de logiciels qui a permis de rendre interopérable cette architecture répartie. L'évolution de ces standards a permis au web de supporter la charge relative à son expansion. Ainsi, dans la section 2.3 nous construisons notre modèle d'architecture, UbiArch, dans le but d'adresser cette plate-forme. Notre modèle s'articule autour des modèles d'interaction de la littérature mais aussi autour des langages et protocoles qui font le réseau ubiquitaire.

Les services web initialement accessibles depuis nos « boîtes grises[56] » nous entourent. Une multitude de terminaux hétérogènes y accède grâce à des interfaces graphiques et tactiles utilisant la métaphore du bureau [Buisson et Jestin, 2001] mais aussi depuis des interfaces dédiées utilisant le mode vocal. L'évolution de la téléphonie, l'explosion des offres téléphoniques : fixes ou mobiles, IP ou RTC justifie la prise en compte du mode vocal dans les interfaces web. On voit (entend ? !) apparaître des robots téléphoniques en charge d'effectuer des campagnes de publicités ou de sondages, les stations de carburants, les gares proposent aux usagers des automates parlant, nos messageries vocales ressemblent de plus en plus à nos boîtes mail. Dans le section suivante, nous étudions les structures de contrôle qui permettent au système de manipuler le mode vocal. Par la suite nous détaillerons les protocoles permettant d'établir des flux média à travers le net.

[54] JSON - JavaScript Object Notation, est un format de structure de données générique. Il utilise la notation des objets JavaScript pour transmettre de l'information structurée. C'est le seul point commun qu'il a avec le langage JavaScript [46].

[55] AJAX - Asynchronous JavaScript And XML.

[56] les ordinateurs personnels [Coutaz, 1998]

1.4 La navigation vocale sur le web

« Il existe sur la planète beaucoup plus de téléphones que d'ordinateurs ». C'est sur ce constat que débute l'ouvrage de José Rouillard intitulé « VoiceXML » [José, 2004]. Cette section explicite l'ingénierie XML rendant possible l'accès à internet par le mode vocal.

VoiceXML Le langage *VoiceXML* [Rec. VXML, 2004] est un dialecte XML [Rec. XML, 2006] spécifiant un dialogue homme machine vocal. VoiceXML permet donc de décrire un scénario de services vocaux. De la même manière que la visualisation d'une page web est l'interprétation d'un script HTML par un navigateur web, un service vocal VoiceXML est l'interprétation d'un script VoiceXML par un navigateur VoiceXML (figure 1.15).

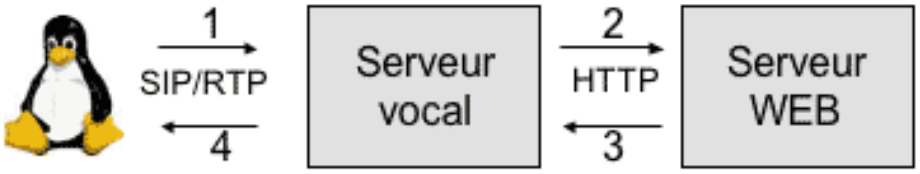


FIG. 1.15 – Fonctionnement d'un serveur vocal. Architecture mise en place dans le cadre des travaux pratiques autour de VoiceXML à l'université de Caen. L'objectif de l'architecture est de fournir aux étudiants du département d'informatique de l'Université de Caen un espace de développement de services vocaux. Ces services vocaux sont accessibles en voix sur IP. Cette figure montre les différentes étapes de la navigation vocale : (1) l'utilisateur p1gu1 appelle le service en utilisant l'adresse *sip :p1gu1@serveur.vocal :5060*. Le serveur vocal prend l'appel ; (2) en utilisant l'adresse d'appel (variable called_id), le serveur vocal exécute la requête HTTP suivante : *GET http ://user.info.unicaen.fr/˜p1gu1/vxml/index.vxml* ;(3) le serveur web fournit ainsi le script VoiceXML au serveur vocal ; le serveur vocal charge enfin le script VoiceXML et (4) effectue le rendu vocal du service demandé.

Le VoiceXML permet de manipuler des ressources de reconnaissance vocale, de reconnaissance de DTMF, de synthèse vocale, de lecture de fichiers audio, d'enregistrement de séquences audio et des ressources de mise en relation (aboutement) d'appels téléphoniques. La figure 1.4 décrit un service très simple faisant intervenir les fonctionnalités de lecture de fichiers audio et de synthèse vocale de VoiceXML.

FIG. 1.16 – Exemple simple d'utilisation de VoiceXML. L'interprétation de ce script déclenche la lecture du fichier *jingle.wav*, ensuite le texte *Hello World !* est synthétisé puis joué dans le combiné de l'utilisateur et enfin la session se termine par la déconnexion du service.

```
<vxml version="2.0">
  <form>
    <block>
      <audio src="jingle.wav"/>
      <prompt>Hello World !</prompt>
      <disconnect/>
    </block>
  </form>
</vxml>
```

Le VoiceXML permet de décrire un scénario de dialogue selon le modèle projectif du dialogue de Vernant (figure 1.17). Les tours de parole conduisent l'utilisateur selon un canevas structuré

par le langage. Le but à atteindre est défini par le concepteur de l'application. Il peut par exemple être la détermination des lieux de départ et d'arrivée de l'utilisateur d'un système de navigation [Bazin *et al.*, 2006]. Le dialogue est fortement contraint par l'enchaînement des séquences qui composent le script VoiceXML. Le langage VoiceXML permet la manipulation des divergences par rapport au but à atteindre. Il traite les dépassements de délai par rapport à l'attente d'un événement utilisateur et est capable de reformuler ses requêtes et de proposer des sous dialogues alternatifs.

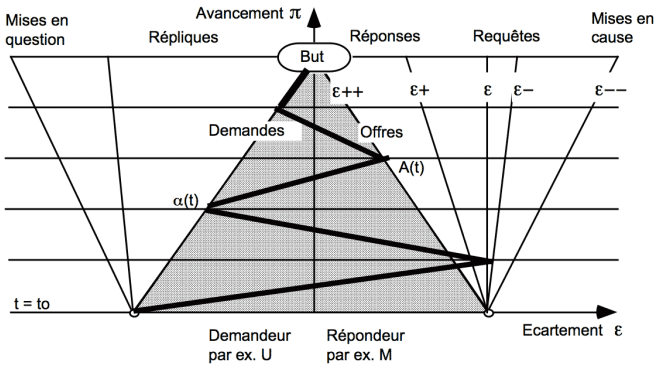


FIG. 1.17 – Modèle projectif du dialogue [Vernant, 1992]. « *Dans le cas d'un dialogue réussi, la convergence du dialogue au cours du temps est conduite par le but à atteindre. Ce schéma - qui représente un Échange - montre les axes convergents (Demandes, Offres) et les axes divergents (Mises en question, Mises en cause, Répliques et Requêtes) du dialogue´* » [Caelen et Villaseñor-Pineda, 1997].

VoiceXML manipule des nœuds `form` connectés les uns aux autres par l'intermédiaire des nœuds `goto` (figure 1.18). Un nœud `goto` déplace le focus de l'application vers l'URI décrite par l'attribut `next`. Le nœud `menu` de VoiceXML capture et interprète les commandes vocales ou DTMF de l'utilisateur et modifie la logique applicative du service en conséquence.

FIG. 1.18 – Exemple d'utilisation des balises *goto* et *menu* de VoiceXML. L'utilisateur arrive sur le portail des services vocaux. Le portail lui souhaite la bienvenue. Le focus est ensuite porté sur le menu principal. Le service invite l'utilisateur à choisir le service auquel il souhaite accéder. Le choix d'un des deux services disponibles déplace le focus vers l'ancre spécifique du fichier *service.vxml*. Notez également que la sélection du service peut se faire par commande vocale. Les mots à prononcer se trouvent à l'intérieur de la balise *choice* (Ici : *météo* ou *horoscope*).

```
<vxml version="2.0">
  <form id="accueil">
    <block>
      <prompt>Bienvenue sur le portail
        des services vocaux</prompt>
      <goto next="#menu"/>
    </block>
  </form>
  <menu id="menu">
    <prompt>Tapez 1 pour accéder à la
      météo, Tapez 2 (...)</prompt>
    <choice dtmf="1" next="serv.vxml#meteo">
      météo</choice>
    <choice dtmf="2" next="serv.vxml#horosc">
      horoscope</choice>
  </menu>
</vxml>
```

VoiceXML permet également l'enregistrement de fichiers audio. Cette fonctionnalité de VoiceXML est accessible par le biais de la balise `record`. La figure 1.19 décrit l'utilisation de cette fonctionnalité.

FIG. 1.19 – Exemple d'utilisation de la balise *record* de VoiceXML. L'utilisateur est mis en relation avec un répondeur. Le service l'invite à laisser un message vocal. Le message audio est affecté dans la variable *msg*. L'utilisateur a la possibilité d'écouter le message qu'il vient de déposer.

```
<vxml version="2.0">
  <form>
    <record  name="msg" type="audio/x-wav">
      <prompt>Laisser un message après le bip.</prompt>
    </record>
    <block>
      <prompt>Votre message est :
        <audio expr="msg"/>.</prompt>
    </block>
  </form>
</vxml>
</vxml>
```

Enfin, VoiceXML permet de transférer, de router des appels vers d'autres services ou vers des postes téléphoniques en utilisant la balise `transfer`. VoiceXML propose des patrons simples permettant la manipulation d'une session téléphonique et le rendu de services vocaux. Un service VoiceXML s'apparente à un ensemble de formulaires. Chacun de ces formulaires attend des évènements utilisateurs. Une fois ces formulaires remplis, leur soumission déplace le focus vers la suite du service, conformément aux instructions contenues dans le formulaire. Les événements attendus par le système sont de type `DTMF` ou `voice`. La figure 1.20 décrit un cas d'usage de cette fonctionnalité.

SRGS Les grammaires *SRGS*[57] permettent de décrire les entrées `DTMF` ou `voice` attendues par le système. Les grammaires SRGS sont manipulées par les structures de contrôle de VoiceXML. La DTD de VoiceXML inclut celle de SRGS. La figure 1.21 présente deux exemples de grammaires SRGS. Le premier exemple rend compte de l'utilisation d'une grammaire SRGS évoluée utilisant le mode DTMF. La seconde grammaire le mode `voice`.

SSML Outre les structures qui contrôlent les interruptions de l'utilisateur, VoiceXML permet de manipuler des ressources de synthèse vocale. *SSML*[58] est un dialecte de XML destiné à décrire les paramètres utilisés par synthèses vocales. La DTD de VoiceXML inclut celle de SSML. Le SSML contrôle les aspects de la parole tels que la prononciation, la prosodie, le volume, le débit, la langue. Ces aspects sont regroupés sous trois thèmes : la structure du document, le style et la description des contenus. Le premier thème, la structure, regroupe les balises `p` (paragraphe), `s` (phrase) ainsi que les éléments nécessaires à la bonne prononciation d'un texte. Les éléments de style sont la voix (`voice`), l'accentuation (`emphasis`), la pause (`break`) et la prosodie (`prosody`). Enfin, les éléments descriptifs (`audio`, `mark` et `desc`) permettent entre autre de proposer une alternative au texte à synthétiser pour les personnes mal-entendantes. Cette utilisation s'apparente à celle de l'attribut `alt` de la balise `img` de HTML pour les personnes mal-voyantes. La figure 1.22 présente un exemple de SSML. Notez tout de même que les synthèses

[57] SRGS - Speech Recognition Grammar Specification [Rec. SRGS, 2004].
[58] SSML - Speech Synthesis Markup Language [Rec. SSML, 2004].

```
<vxml version="2.0">
  <form>
    <block>
      <prompt>Vous allez être mis en relation avec (...).</prompt>
    </block>
    <transfer dest="tel:+33-6********">
      <prompt>Tapez étoile pour annuler la mise en relation.</prompt>
      <grammar src="annuler.srgs"/>
      <filled>
        <prompt> Au revoir.</prompt><disconnect></filled>
    </transfer>
  </form>
</vxml>
```

FIG. 1.20 – Exemple d'utilisation des balises *transfert* et *grammar* de VoiceXML. L'interprétation du second script permet la mise en relation de l'utilisateur avec un service de transfert d'appel. Il est notifié de la mise en relation avec son correspondant. Notez que l'utilisateur a la possibilité d'annuler le transfert en tapant *étoile*. Cette fonctionnalité est rendue possible par l'emploi de la balise *grammar* et de l'utilisation de la grammaire externe *annuler.srgs*. Une fois l'action validant les conditions décrites dans la grammaire du fichier *annuler.srgs* réalisée, le focus se déplace dans la branche *filled* du DOM (section 1.3.2) du service VoiceXML. Ceci provoque la déconnexion du service.

```
<grammar version="1.0" mode="dtmf" root="password">
  <rule id="digit">
    <one-of> <item>0 </item> <item>1 </item>...  <item>9</item> </one-of>
  </rule>
  <rule id="password" scope="public">
    <item>
      <item repeat="4"><ruleref uri="#digit"/></item>
      <item repeat="1"># </item>
    </item>
  </rule>
</grammar>

<grammar version="1.0" mode="voice" root="cmdVoc">
  <rule id="cmdVoc" scope="public">
    <one-of> <item>supprimer </item> <item>archiver </item> </one-of>
  </rule>
</grammar>
```

FIG. 1.21 – Deux exemples de grammaire SRGS. Premier exemple, la grammaire *password* compose deux grammaires simples. La première grammaire simple est la grammaire *digit* qui correspond à l'ensemble des chiffres compris entre *0* à *9*. La grammaire évoluée *password* compose quatre éléments de la grammaire *digit* et l'élément *dièse*. L'utilisation de cette grammaire dans un service VoiceXML permet à l'utilisateur de saisir son code secret composé de quatre chiffres et de valider sa saisie en tapant dièse. Second exemple, la grammaire *cmdVoc* offre à l'utilisateur la possibilité d'interagir avec un système en utilisant des commandes vocales *supprimer* ou *archiver*.

vocales ont chacune leur implémentation de SSML et que par conséquent, l'écriture du service VoiceXML est fortement contrainte par les briques logicielles utilisées. D'une manière générale, cette remarque est valable pour l'ensemble des subtilités de VoiceXML.

```
<vxml version="2.0">
  <menu id="menu">
    <prompt><voice name="LAURA">Press key one to choose
                                English.</voice></prompt>
    <prompt><voice name="CLAIRE">Tapez 2 pour choisir le
                                Français.</voice></prompt>
    <choice dtmf="1" next="#En"/>
    <choice dtmf="2" next="#Fr"/>
  </menu>
</vxml>
```

FIG. 1.22 – Exemple d'utilisation des fonctionnalités multilingues de SSML. Cet exemple propose une utilisation de VoiceXML (et donc SSML) dans le cadre d'un service multilingue. L'utilisateur est mis en relation avec un service multilingue qui propose en anglais puis en français de choisir la langue à utiliser pour la suite service. Le formalisme SSML permet de définir la voix à utiliser pour synthétiser un prompt.Le premier *prompt* utilisera une voix anglaise afin d'assurer un rendu correct de la phrase, le second utilisera une voix française.

CCXML Autant SRGS et SSML permettent de manipuler la session interactive avec l'utilisateur, autant CCXML permet de contrôler la session téléphonique VoiceXML. *CCXML*[59] [Draft CCXML, 2007] permet de contrôler cette session téléphonique. CCXML rend obsolète l'usage du tag `transfer` de VoiceXML. Le tag `transfer` du VoiceXML va induire deux actions CCXML : `send` et `hold`. Le CCXML permet donc de décrire des services de transfert d'appel, mise en attente, gestion de présences (figure 1.23). La DTD de CCXML englobe celle de *SCXML*[60] [Draft SCXML, 2005]. SCXML permet une abstraction du contrôle des applications.

```
<ccxml version="1.0">
  <eventhandler>
    <transition event="connection.CONNECTION_ALERTING" name="evt">
      <if cond="evt.callerid == '+33-6********'">
        <accept/> <else/> <reject/>
      </if>
    </transition>
  </eventhandler>
</ccxml>
```

FIG. 1.23 – Exemple d'utilisation de CCXML. Cette exemple présente une structure de contrôle CCXML permettant le filtrage d'appel en fonction du numéro appelant.

[59] CCXML - Voice Browser Call Control XML.
[60] SCXML - State Chart XML (SCXML) : State Machine Notation for Control Abstraction.

Conclusion sur VoiceXML VoiceXML manipule des structures de contrôle permettant de construire un canevas autour duquel le concepteurs de services va pouvoir articuler des stratégies de dialogue. Dans la section 2.3 nous verrons comment le VoiceXML rend compte du contrôle du dialogue de notre modèle d'architecture UbiArch.

Mais VoiceXML n'est pas le seul dialecte XML intégrant le mode vocal. Contrairement à VoiceXML réservé à l'élaboration de services purement vocaux, SALT et X+V sont destinés à intégrer le mode vocal dans des interfaces web. Tous deux sont des dialectes XML rendant possible le développement d'applications web utilisant en entrée la reconnaissance vocale et en sortie la synthèse vocale. Chacun de ces deux langages possède ses propres balises pour spécifier les traitements de reconnaissance, et de synthèse de parole [Georgescu et Christopher, 2005]. Néanmoins, ces deux dialectes possèdent des différences significatives.

SALT A l'initiative du *SALT forum* [36] , le langage *SALT*[61] a pour objectif d'ajouter des fonctions de reconnaissance et de synthèse vocale ainsi que navigation par DTMF à un application web. SALT utilise le formalisme SRGS pour décrire les évènements attendus par le système. SALT utilise SSML pour contrôler la synthèse vocale. Contrairement à VoiceXML et bien qu'il le permette, SALT n'est pas destiné au développement de portails purement vocaux. SALT est un langage qui permet la navigation multimodale dans une page web. SALT traite deux classes particulières d'interfaces : les interfaces multimodales, où une page web est augmentée avec le mode vocal et les interfaces multi-canals ou plastiques, où une application web unique peut prendre différentes interfaces adaptées aux caractéristiques du terminal de l'utilisateur[David Thevenin, 2000, Chevrin, 2006]. SALT permet la réutilisation du code existant. En effet, l'enrichissement vocal se fait par ajout d'éléments au DOM de HTML et de *SVG*[62]. La figure 1.24 présente un exemple d'utilisation de SALT dans le cadre du remplissage d'un formulaire par interaction vocale.

Le 31 juillet 2002, le SALT forum a soumis les spécifications de SALT aux deux groupes de travail : Multimodal Interaction Activity et Voice Browser Activity du W3C. Cette contribution permet de définir certains aspects de l'évolution de VoiceXML [42]. Le 5 avril 2006, Microsoft communique par voie de presse [2] le support de VoiceXML par *Speech Server 2007* rendant obsolète l'utilisation de SALT.

X+V *X+V*[63] [Note X+V, 2001, Spec. X+V, 2004, Spec.X+V mobile, 2005], est un dialecte de XML. X+V permet la description d'application intégrant le mode vocal dans des pages XHTML. X+V intègre les recommandations du W3C concernant XHTML et VoiceXML. Comme VoiceXML, X+V suscite l'intérêt concernant l'interaction vocale homme-machine. Contrairement au VoiceXML, X+V utilise à la fois l'interaction web et l'interaction vocale. Comme SALT, il offre la possibilité aux développeurs d'enrichir une application web d'éléments vocaux en entrée et en sortie. X+V intègre *XML event* [Rec. XML event, 2007] pour faire communiquer le DOM XHTML et le DOM VoiceXML. La figure figure1.25 présente un exemple d'XML event.

[61] SALT - Speech Application Language Tags [Spec. SALT, 2002].

[62] SVG - Scalable Vector Graphics, SVG est un dialecte de XML permettant la descrition d'objet vectoriel [Rec. SVG, 2003].

[63] X+V - XTHML + VoiceXML.

```
<html xmlns:salt="http://www.saltforum.org/2002/SALT">
  <head>
    <object id="k-tags" VIEWASTEXT/>
  </head>
  <body>
    <p>Pressez le bouton et prononcez le nom de votre ville de départ.</a>
    <?import namespace="salt" implementation="#k-tags"/>
    <input name="txtBoxCity" type="text" />
    <input name="buttonCityListen" value="Listen" type="button"
           onClick="listenCity.Start();" />
    <!--Speech Application Language Tags -->
    <salt:listen id="listenCity">
      <salt:grammar name="city" src="./city.grxml" />
      <salt:bind targetelement="txtBoxCity" value="//city" />
    </salt:listen>
  </body>
</html>
```

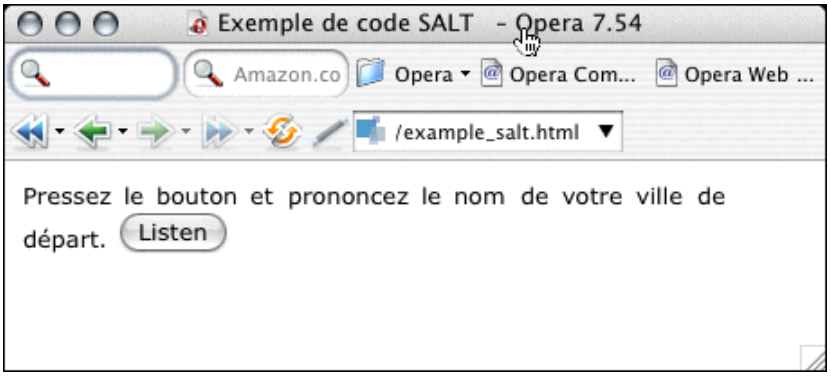


FIG. 1.24 – Exemple de code SALT dans un scénario *push-to-talk*.L'utilisateur presse le bouton *Listen* pour lancer la reconnaissance vocale. La grammaire *city* est activée. Le résultat de la reconnaissance vocale est associé au champ d'entrée adjacent.[Bachelet, 2004]

```
    <catch ev:event="nomatch">
    <listener event="click" targetid="#button1" handler="#clicker"/>.
    <onevent ev:event="enterforward">
        <go href="/url"/>
    </onevent>
    <script ev:event="activate" type="application/x-javascript">
        doactivate(event);
    </script>
```

FIG. 1.25 – Quatre exemples de code XML event. Le premier exemple va capturer les événements *nomatch* qui correspondent à une entrée utilisateur non attendue. Le second exemple associe le *handler #clicker* à l'événement *click* associé à l'élément *#button1*. Le troisième exemple capture l'événement *enterforward* et lui associe la redirection du navigateur sur l'URL *url*. Enfin, le dernier exemple capture l'évenement *activate* et lui associe la fonction ECMAScript *doactivate()* en lui transmettant le paramètre *event* qui contient les propriétés de l'événement.

Synthèse sur la navigation vocale sur le web VoiceXML manipule des structures de contrôle permettant d'adresser les principales activités de l'interaction vocale. Tandis que VoiceXML est purement orienté voix (et vidéo pour la version 3.0), SALT et X+V permettent l'intégration du mode vocal dans des interfaces web. Microsoft rend obsolète SALT par l'intégration de VoiceXML dans son « Spee Server 2007 ». X+V est entièrement fondé sur des langages standardisés.

Les morceaux de code VoiceXML utilisés dans une application X+V peuvent être réutilisés à l'intérieur d'une application VoiceXML autonome. En conséquence, des applications en X+V peuvent être codées en parties, avec des experts en programmation vocale développant des éléments vocaux et des experts dans la programmation web pour les aspect HTML. Ce dernier point fait la force de X+V mais également sa faiblesse. En effet, la séparation de ces deux modes d'interaction rend difficile le maintien de la cohérence globale de l'application.

Nous identifions donc le XHTML et le VoiceXML comme deux langages permettant de rendre compte de notre problématique. Dans le section 3.2, nous spécifions une plate-forme qui intègre ces deux langages de manière à réaliser des services mêlant voix et navigation web.

1.5 La diffusion de flux média sur le net

Après avoir pris connaissances des technologies manipulant le mode vocal, nous analysons les moyens de diffusion de flux média à travers le réseau IP. Cette section fait l'inventaire des protocoles permettant cette diffusion. L'essor de la VoIP repose sur l'ensemble de ces protocoles.

RTP, RTCP *RTP*[64] est un protocole permettant la diffusion - « streaming » - de contenus audio et vidéo à travers le réseau internet. Il est défini en 1996 par la RFC 1889 puis en 2003 par la RFC 3550. RTP est généralement transporté par UDP[65]. De fait, RTP est unidirectionnel. RTP utilise des ports UDP dynamiques, par conséquent, il rend difficile la configuration des équipements de sécurisation du réseau (firewall, passerelle, NAT[66]). Il est souvent nécessaire d'utiliser le protocole STUN[67] [RFC 3489, 2003] qui permet de déterminer l'adresse publique (celle de la passerelle) avec laquelle les destinataires verront arriver les paquets. Le protocole RTP est utilisé de paire avec le protocole *RTCP*[68]. RTCP ne transporte pas les données mais des informations sur le flux RTP. RTCP offre un retour d'information sur la qualité du service fourni (*QoS*[69]). Ce retour permet l'ajustement du flux RTP afin de s'assurer du bon rendu du flux streamé. Le protocole RTP n'est pas destiné à être utilisé seul. En effet, à la session RTP précède la négociation des codecs, des adresses IP et des ports des machines cibles - « peer ». Pour effectuer cette tâche, il existe plusieurs protocoles de signalisation et de contrôle.

RTSP Le protocole client serveur *RTSP*[70] est défini en 1998 par la RFC 2326 [RFC 2326, 1998]. Rendu populaire par le logiciel *VideoLAN* [39] développé à l'origine par les élèves de l'École Normale Supérieure de Paris puis par la communauté *Open Source* [Basset, 2003]. Le protocole RTSP a pour but de contrôler un serveur de streaming distant grâce à ses méthodes. Les méthodes du

[64] RTP - Real-time Transport Protocol [RFC 1889, 1996, RFC 3550, 2003].
[65] UDP - User Datagram Protocol.
[66] Network Address Translators
[67] STUN - Simple Traversal of UDP through NATs.
[68] RTCP - RTP Control Protocol[RFC 3550, 2003].
[69] QoS - Quality of Service.
[70] RTSP - Real Time Streaming Protocol.

protocole RTSP utilise la métaphore du magnétoscope (lecture, pause, enregistrement...). Mais RTSP n'est pas le seul protocole permettant de négocier une session RTP.

SIP, H323 Les protocoles P2P *SIP* [71] et *H323*[72] sont des protocoles de signalisation permettant de définir une session RTP. Ces protocoles sont utilisées pour véhiculer la voix au travers des réseaux IP (VoIP). Tandis que H323 [21] est une adaptation au réseau IP des principes de la téléphonie, SIP est directement pensé IP. Encouragé au niveau industriel par Cisco [40] et rendu populaire par l'IP PBX[73] open source *Asterisk* [10] il existe plusieurs implémentations du protocole SIP [31, 27]. La figure 1.26 montre l'initiation de la session RTP en SIP. La souplesse du protocole SIP est l'objet de nombreuses attaques [Nassar *et al.*, 2006]. La sécurisation du protocole SIP est un enjeu majeur de la VoIP. Des initiatives de chiffrement des messages *SDP*[74] ouvrent des voies dans ce domaine [20].

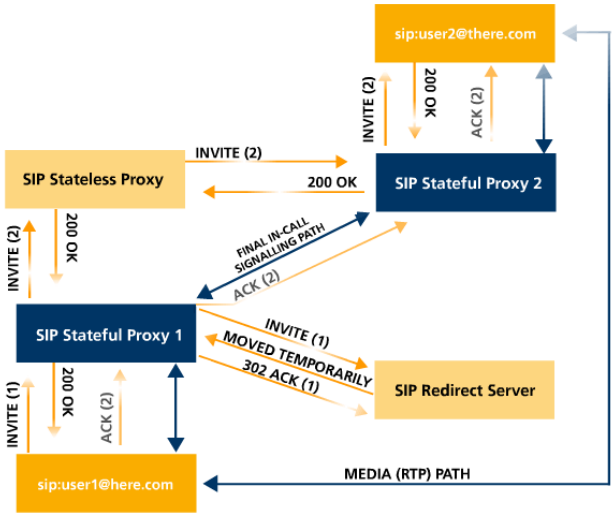


FIG. 1.26 – Distinction des flux de signalisation et des flux média lors de l'initialisation d'une session RTP avec SIP. On distingue clairement le « triangle » formé par la centralisation des flux de signalisation et les flux média point à point. On remarque que le flux RTP est point à point ce qui repartit la charge applicative liée à la gestion du flux média sur les extrémités de la chaîne. Les équipements intermédiaires ne gèrent que l'enregistrement, les droits d'accès, la facturation...

SDP La protocole *SDP* permet de décrire les paramètres d'initialisation du flux média. Le protocole SDP permet de négocier les codecs, les adresses IP et les ports à utiliser pour une session multimédia donnée. Le protocole SDP est utilisé dans le corps des messages SIP.

MRCP et SpeechSC Les protocoles *MRCP*[75] et *SpeechSC*[76] ont vocation à contrôler les ressources de reconnaissance vocale, de synthèse vocale, les systèmes de reconnaissance du locuteur et d'enregistrement de flux audio. Ce sont, comme SIP, des protocoles de signalisation par opposition au RTP qui est un protocole de diffusion de flux média. Le protocole MRCP ressemble beaucoup au protocole RTSP (lecture, pause, enregistrement...) mais il apporte plus

[71] SIP - Session Initiate Protocol [RFC 2543, 1999, RFC 3261, 2002].
[72] standard de l'International Telecommunication Union (ITU)
[73] IP PBX - IP Private Branch eXchange, équivalent IP des centrales téléphoniques analogiques PABX.
[74] SDP - Session Description Protocol [RFC 4566, 2006].
[75] MRCP - Media Resource Control Protocol [RFC 4463, 2006].
[76] SpeechSC - Speech Service Control, Version 2 de MRCP [Spec. SpeechSC, 2005].

de fonctionnalités (échange de grammaires de reconnaissance vocale, échange de textes à synthétiser...) Le protocole MRCP tend a fédérer les nombreuses API propriétaires des industriels du domaine des services vocaux[RFC 4313, 2005].

Conclusion sur les flux média L'étude des flux média sur les réseaux IP nous a permis d'identifier des protocoles de signalisation (SIP) et de contrôle (MRCP, RTSP) permettant la construction d'un réseau de flux média (RTP). Nous avons pu identifier les mécanismes permettant de construire un maillage point à point de flux média (figure 1.27). Dans le cadre de notre étude, ces points sont : des terminaux ou des agents logiciels délivrant un service. L'utilisation de l'architecture SIP (proxy registrar...) permet de vérifier l'authentification, la présence des points du réseau P2P[77].

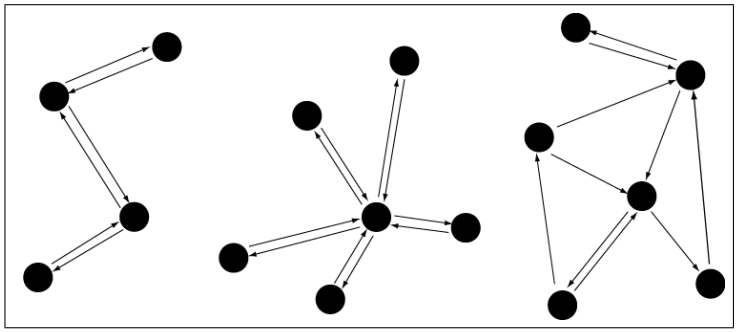


FIG. 1.27 – Trois patrons (de gauche à droite : chaîné, centralisé et mesh) pour la construction d'un maillage RTP [28]. La séparation de la couche de transport et de la couche applicative du modèle *OSI* permet de définir un réseau mesh de flux RTP. Le contrôle de session SIP permet de construire le réseau P2P RTP selon plusieurs motifs. Les points du maillage sont des agents SIP.

1.6 Briques logicielles existantes

Afin de définir une infrastructure permettant d'offrir les services relatifs à notre problématique, nous effectuons un tour d'horizon des plates-formes dont nous disposons dans le laboratoire. Nous identifions deux plates-formes, nous décrivons leurs fonctionnalités et nous voyons comment elles sont complémentaires. La première, la Plate-forme Multimédia XML, propose une architecture de service 3-tiers permettant la mutualisation de services vocaux dans le réseau. Le point fort de la PMX est sa souplesse qui permet d'offrir aux PME[78] des solutions sur mesure. La seconde, la plate-forme VIP permet le déploiement d'une infrastructure réseau robuste, hautement scalable, destinées à prendre un très grand nombre d'appels vers les services vocaux des grand comptes. Certes, il existe de nombreuses architectures abstraites, des outils pour le prototypage et des espaces de conception destinés à la réalisation de services multimodaux[43, 15] mais le fort ancrage opérationnel de notre problématique nous a poussé à nous tourner vers les outils du groupe France Telecom déjà en production dans le réseau de l'opérateur de service.

[77] UA - User Agent.
[78] PME - Petites et Moyennes Entreprises

Plate-forme Multimédia XML La plate-forme PMX[79] est une architecture 3-tiers qui combine un serveur vocal (SMX), des ressources de reconnaissance et de synthèse vocales, un module de prétraitement et un serveur de pages connecté à une application distante (figure 1.28).

Le SMX est un patron qui combine un client de synthèse, un client de reconnaissance vocale, un frontal de téléphonie. Ces trois modules s'articulent autour de l'OpenVXI, implémentation open source du dialecte VoiceXML.

L'architecture 3-tiers permet d'interroger des applications distantes, hébergées chez les clients. Le client peut ainsi rendre une partie de son système d'information accessible depuis le réseau téléphonique, tout en en conservant le contrôle.

Le module de prétraitement permet de compiler des dictionnaires. Ces dictionnaires sont utilisés au moment de la construction des pages VoiceXML pour assurer un rendu correct du texte à synthétiser. Les dictionnaires utilisent un langage « pseudo phonétique ». Ainsi le groupe nominal « *le boulevard du général Weygand* » sera traduit en « *le boulevard du général végan* » grâce à l'entrée *Weygand = [végan]* du dictionnaire des noms propres.

Les clients de synthèse et de reconnaissance vocale intègrent le protocole MRCP ce qui assure la compatibilité de l'architecture avec la plupart des synthèses industrielles du marché.

Le frontal de la plate-forme intègre de nombreux protocoles de téléphonie IP ou RTC.

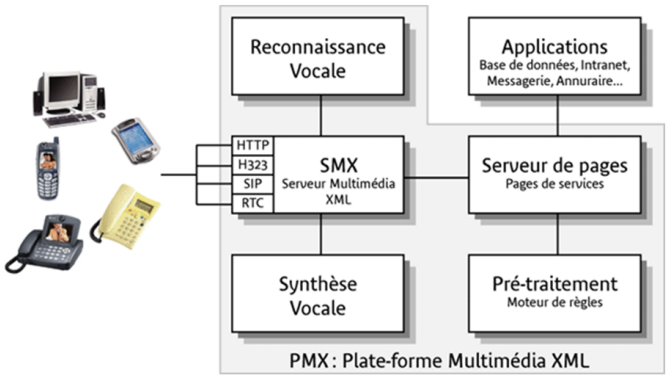


FIG. 1.28 – Architecture PMX [Loqué *et al.*, 2004].

La PMX a été développée dans le cadre du projet collaboratif Volcan, suite à l'appel à projets 2003 du Comité Régional Imagerie et Technologies de l'Information et de la Communication (CRITIC). Le projet Volcan réunit le laboratoire de recherche GREYC de l'Université de Caen ainsi que les partenaires industriels France Telecom, Twisto (groupe Kéolis) et Ingelis. L'objectif du projet Volcan est de concevoir et développer des services avancés sur la base de la PMX. Dès le début de l'année 2004, Volcan a donné lieu à l'expérimentation Timéo[80].

Timéo est un service d'accès téléphonique à l'information voyageur du réseau de transport en commun Twisto. Grâce à son téléphone, l'utilisateur est mis en relation avec un serveur vocal. Le serveur vocal accueille l'utilisateur avec le prompt suivant : *« Bienvenue sur l'information vocale du réseau Twisto. À tout moment, tapez sur la touche étoile pour revenir à l'accueil. Saisissez le code de votre arrêt, suivi de la touche dièse de votre téléphone. »*. En indiquant les informations demandées l'utilisateur obtient les horaires de passage de son bus et ceux du suivant à partir de la géolocalisation temps réel des éléments du réseau de transport en commun.

[79] PMX - Plate-forme Multimédia XML.
[80] Timéo - service d'accés vocal à l'informations voyageur du réseau de tramways et de bus Twisto (Caen).

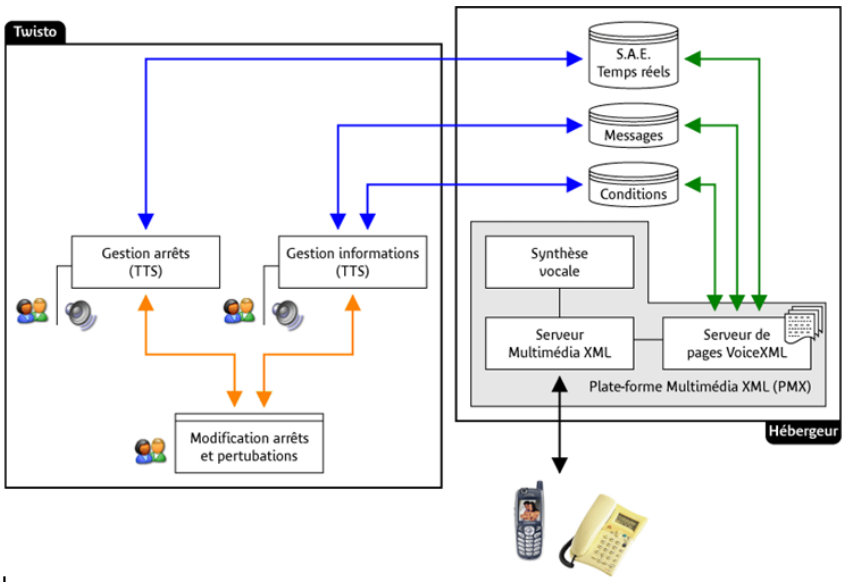


FIG. 1.29 – Service Timéo [Loqué *et al.*, 2004].

Le service a reçu un vif engouement de la part des utilisateurs du réseau de transport. Nous avons notamment constaté des pics d'accès au service lors des perturbations du réseau de transport. Dès les premiers résultats, le groupe Keolis a envisagé d'étendre ce premier service de vocalisation des horaires de passage aux arrêts sur plusieurs grandes villes françaises.

Dans le cadre de l'expérimentation « Caen ville NFC », nous avons étendu ce service en offrant un accès direct aux horaires par l'intermédiaire d'un tag NFC. Disposé à côté des horaires papiers, le tag NFC permet de déclencher automatiquement l'appel vers le service Timéo et de transmettre les codes de l'arrêt et du bus souhaité.

Enfin, dans le cadre du projet Volcan, nous avons spécifié une stratégie de dialogue permettant de définir le lieu de départ et le lieu d'arrivé de l'utilisateur afin de lui proposer le moyen de transport le plus adapté [Bazin *et al.*, 2006]. Si théoriquement, la stratégie de dialogue minimise le nombre d'interactions entre l'utilisateur et le système, le déploiement et la mise en service de ce type d'algorithme se heurte aux capacités techniques limitées des serveurs de reconnaissance vocale mis à notre disposition.

Plate-forme VIP Conçue et développée par les équipes de Lannion du centre de recherche de France Telelcom, pour les besoins du service du 12 vocal automatique, puis expérimentée avec un trafic de 60 000 appels par jour, la plate-forme VIP[81] a rapidement suscité l'intérêt des exploitants et des unités d'affaires du groupe [Bai *et al.*, 2005a].

Son architecture est répartie comme le sont maintenant la plupart des serveurs vocaux interactifs du marché : séparation des fonctions *front-end*, des fonctions *back-end*, des ressources du réseau de téléphonie et des ressources de technologies vocales (Reconnaissance vocale ou ASR et synthèse vocale ou TTS) [Bai *et al.*, 2005b].

La plate-forme VIP est pilotée par le bus propriétaire basé sur le protocole Voice Interaction Protocol [Ferrieux *et al.*, 2002]. Il a l'avantage de reposer sur un principe très simple d'échanges d'informations de type texte à travers TCP/IP, d'où sa performance, sa robustesse (pas de

[81] VIP - Voice Interactive Protocol.

couches à traverser ni de conversions) et sa simplicité d'évolution pour intégrer une nouvelle fonction ou un nouveau composant. Il en découle un moindre coût en développement et mise au point.

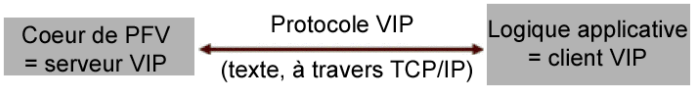


FIG. 1.30 – Protocole VIP.

Un langage de script propriétaire appelé LSD[82] a été conçu pour piloter les ressources de la plate-forme VIP. Ce script est un TCL auquel une fonction de synchronisme a été ajoutée pour les besoins de l'application du 12 vocal. Il est générique et utilisable pour tout service à développer. Dès le début 2005, l'intégration de l'interpréteur VoiceXML de PMX apparaît dans les road-map de VIP, pour les besoins du service QuiDonc de PagesJaunes. Nous avons participé activement à l'intégration du patron SMX dans l'architecture mVIP. Nous avons également participé à la migration du service QuiDonc sur la plate-forme VIP ainsi modifiée. Aujourd'hui, la plate-forme VIP dans sa version VoiceXML est la plate-forme vocale du groupe France Telecom, elle suit les recommandation du W3C et anticipe les évolutions de la version 3.0 du dialecte XML. Dans la section 3.2 nous spécifions les évolutions de VIP permettant la mise en œuvre de services web intégrant le mode vocal.

Synthèse Vocale La synthèse vocale, ou TTS pour « Text-To-Speech », désigne une technologie permettant de construire un signal audio à partir d'un texte.

Il existe plusieurs techniques de synthèse vocale. La synthèse à formant est une technique de synthèse qui n'utilise aucune voix réelle mais uniquement un modèle mathématique pour créer la parole. Cette technique a pour principal avantage de ne pas générer de droit sur les voix. Autre avantage, les modèles de voix sont décrits par des fonctions mathématiques, par conséquent la taille des modèles reste petite ce qui permet d'intégrer cette technologie dans des terminaux aux capacités de stockage limitées [Cotto, 1992]. Cette technique est mise en œuvre par le logiciel eSpeak [13] sous licence GNU. Une seconde technique de synthèse vocale et la synthèse par concaténation de diphones. Les diphones sont des morceaux de signal audio caractérisant des demi syllabes. Les synthèses vocales utilisant les diphones produisent des séquences audio au rendu très synthétique. Cependant, le rendu peut être amélioré en utilisant des fonctions de lissage du signal pour reproduire par exemple la prosodie et les intonations des voix humaines [22]. La dernière technique de synthèse audio utilise des unités longues. Les unités longues sont les syllabes, des mots ou même des ensembles de mots. La concaténation de ces unités longues, utilisée conjointement avec des fonctions de lissage produit des séquences audio au réalisme proche de la voix humaine.

On observe l'usage des technologies de synthèse vocale sur le web notamment dans les podcasts (blogs audio). Par exemple, le site d'information Agoravox[83] propose une vocalisation de l'ensemble de ces articles. L'utilisation qui est faite de cette technologie prend tout son sens dans le cadre de l'accessibilité des contenus aux personnes non ou mal-voyantes. Cependant, on s'aperçoit très vite de limites liées de cet usage. En effet, la voix de synthèse, même très réaliste, ne retient pas l'attention de l'auditeur et l'écoute complète d'un article et bien plus coûteuse en

[82] LSD - Langage de spécifications des dialogues.
[83] http ://www.agoravox.fr

temps que le parcours rapide du texte écrit. Ce dernier point peut être contourné par l'utilisation de dispositifs permettant de modifier le débit du flux audio [Asakawa *et al.*, 2003]. Néanmoins cette utilisation de la synthèse reste innovante.

InicipBlog, « le journal audio de lectures à voix haute »[84], propose dans son billet du 29 janvier 2005, une utilisation plus appropriée de cette technologie. La synthèse est utilisée pour introduire les titres, les chapitres et autres éléments structurels du texte. La lecture du contenu du texte reste à la charge d'un acteur. Notons ici l'utilisation d'une synthèse par diphones. Le côtés « artificiel » de cette technologie assure une rupture claire dans la lecture du texte et semble parfaitement appropriée à cette usage.

Dans le cadre de notre travail de recherche nous utilisons principalement les synthèses vocales MRCP Elan Speech du groupe Acapela et SynthServer du groupe France Telecom. Ces synthèses par unité longue offrent un panel de voix multilingues, réalistes et personnalisables, adaptées au services vocaux modernes. Nous utilisons des voix françaises et anglaises afin d'assurer un rendu bilingue de nos démonstrations.

Reconnaissance vocale La reconnaissance vocale, ou ASR pour « Automatic Speech Recognition », désigne une technologie qui permet d'analyser un signal audio afin d'en extraire des mots selon un modèle de reconnaissance. Les systèmes de reconnaissance vocale récents combine une analyse cepstrale du signal et une analyse des chaînes de Markov cachées. Les modèles de reconnaissances vocales peuvent être dynamiques et multilocuteurs, c'est à dire qu'ils ne nécessitent pas d'apprentissage préalable des mots prononçables mais utilisent un modèle global de la langue. Ce type de reconnaissance vocale nécessite le stockage d'un modèle volumineux et consommateur de ressources. A cette technologie s'oppose la reconnaissance par apprentissage préalable, où l'ensemble des éléments prononçables et initialement vide et où le modèle de reconnaissance vocal s'enrichit au fur et à mesure que l'utilisateur y ajoute des éléments.

Dans le cadre de notre travail de recherche, nous utilisons principalement la reconnaissance vocale teliSpeech Recognizer de Telisma. La plate-forme teliSpeech offre une solution de reconnaissance vocalex multilingue, multilocuteur, mutualisée dont les fonctionnalités sont rendues accessibles par l'intermédiaire du protocole MRCP.

Vers l'exploitation du réseau ubiquitaire En quinze ans, le web documentaire est devenu le web des services. L'évolution des technologies web et la refonte des réseaux de télécommunication ont provoqué un profond bouleversement de son utilisation. Internet est passé d'une base de données documentaire répartie à une plate-forme de service à l'échelle mondiale. Ces services ne concernent pas seulement les utilisateurs d'ordinateurs mais sont accessibles en situation de mobilité, de handicap ou dans un contexte résidentiel. Les moyen d'accès au web se sont multipliés et aujourd'hui le web adresse une classe hétérogène de terminaux.

La simplicité de HTTP (question réponse) fait le succès du réseau internet. Si cette simplicité, HTTP la partage avec SIP, ces deux protocoles diffèrent par la topologie des réseaux qu'ils engendrent. En effet, tandis que HTTP repose sur une architecture client serveur, SIP est un protocole point à point. La prise en compte du mode vocal passe par la construction d'un maillage de flux média (RTP). L'étude des flux média sur les réseaux IP nous a permis d'identifier des protocoles de signalisation (SIP) et de contrôle (MRCP, RTSP) permettant la construction d'un réseau de flux média.

Cet état des lieux va nous permettre de définir notre modèle d'architecture de manière adaptées aux spécificités du réseau ubiquitaire. Ceci nous permettra d'adresser le réseau internet

[84]http ://www.incipitblog.com 29 janvier 2005 Florent Latrive

comme une plate-forme de service. Ainsi nous définissons une plate-forme de service qui utilise l'infrastructure existante pour à la fois définir de nouveaux services et en même temps étendre les services existants. La prise en compte des protocoles et formalismes précédemment décrits va nous permettre d'appréhender la problématique de la voix sur IP et de reproduire à bas coût les services de téléphonie. Nous pensons ainsi favoriser l'essor de notre technologie en s'appuyant des services au modes de facturations indépendant de la durée des appels.

Dans la section 2.2 nous modélisons l'environnement de l'utilisateur en y intégrant ces terminaux et en tenant compte de la topologie des réseaux. Dans la section 2.3 nous définissons notre modèle d'architecture UbiArch. UbiArch intégre l'ensemble de ces terminaux dans une même session interactive.

Dans la section 3.2 nous spécifions notre plate-forme de service mVIP sur les bases de la plate-forme VIP dans sa version VoiceXML. mVIP permet la mise en œuvre des services identifiés comme relevant de notre problématique.

Chapitre 2

Représentation de l'environnement de l'utilisateur et modélisation de l'architecture de services

Trois Mondes, M. C. Escher

Ce chapitre aborde deux thèmes clefs de notre problématique : l'environnement de l'utilisateur et l'interaction entre l'utilisateur et le système. Ainsi, nous analysons la composition de l'environnement de l'utilisateur en tenant compte de sa présence sur le réseau. Nous observons une classe hétérogène de terminaux connectés au réseau ubiquitaire. Nous mettons en évidence l'existence d'espaces dont la responsabilité et les coûts engendrés par la mise en place des infrastructures de services relèvent des différents acteurs du domaine : de l'utilisateur jusqu'au fournisseur de service en passant par les responsables de la maintenance de l'infrastructure. Les sphères d'interaction prennent en compte cette segmentation et propose une représentation abstraite du réseau manipulable par les outils de l'administrateur. Fort de cette analyse, nous construisons notre modèle d'architecture UbiArch dans le respect des principes fondamentaux des modèles de Seeheim [Pfaff, 1985] et Arch [SIGCHI, 1992].

2.1 Vers une représentation de l'environnement utilisateur

L'utilisateur est placé au centre d'un environnement communicant. Proche de lui se trouve les terminaux communicants. Que ce soit les téléphones fixes ou mobiles, IP ou commutés ; les ordinateurs personnels portables ou de bureau ; les dispositifs audiovisuels ou multimédia, l'ensemble de ces terminaux sont connectés à différentes infrastructures. Ces infrastructures, elles aussi sont multiples. On énumère par exemple les réseaux de téléphonie, commutés, mobiles ou de troisième génération ou IP ; les réseaux informatiques privés, publiques sans fil, maillés locaux ou internet.

Afin de structurer cette espace, nous définissons les *sphères d'interaction* par proximité par rapport à l'utilisateur. Les sphères d'interaction structurent l'environnement de l'utilisateur en y incluant le réseau. Les architectures sous-tendant l'interaction multimodale distribuée sont réparties sur ces différents domaines. La maintenance de cette infrastructure donne lieu à un partage des responsabilités, de la prise en charge des coûts et du transit de l'information entre les différents organes. Sur le schéma de la figure 2.1 , la sphère 1 présente les dispositifs se trouvant sur l'utilisateur (téléphone, oreillettes). La sphère 2 regroupe les équipements proches de l'utilisateur. Les sphères 3 et 4 sont les sphères locales et extérieures. La sphère locale est le domaine du privé, la sphère extérieure est le domaine du service. Les responsabilités et les coûts liés à ces sphères diffèrent selon le contexte d'utilisation du service.

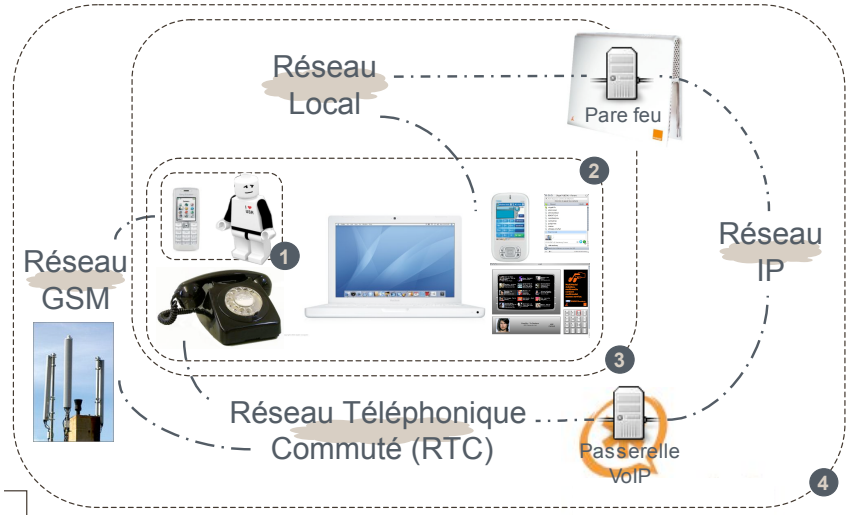


FIG. 2.1 – Sphères d'interaction et flux modal.

Si la figure 2.1 met en évidence la segmentation des réseaux de communication, elle montre également la perméabilité des différents segments. Les *flux modaux* transitent dans et entre les sphères d'interaction. La perméabilité des réseaux est rendue possible par la mise en place et la maintenance d'équipements spécifiques. Les routeurs - modem ADSL - Wifi des opérateurs assurent la communication entre le réseau privé de l'utilisateur ou de l'entreprise et internet. Cette passerelle donne lieu à la souscription à un abonnement, en l'échange de quoi, l'opérateur assure l'acheminement du flux modal sur internet de manière sécurisée. De la même manière, les passerelles VoIP assurent l'interopérabilité entre les réseaux de téléphonie classique et le réseau IP. En conséquence, l'acheminement du flux modal entre ces deux réseaux donne lieu au support, à la sécurisation et au contrôle d'organes essentielles du réseau ubiquitaire.

Du point de vue de l'utilisateur L'utilisateur dispose des interfaces réelles ou réalistes des terminaux de son environnement [Coutaz, 1998]. Dans le cadre de notre problématique, un exemple d'interface réelle est le clavier du téléphone tandis que le pavé de l'interface mATM (figure 6, page 10) est une interface réaliste qui simule les fonctionnalités de l'interface réelle. Ces interfaces sont autant de moyens permettant à l'utilisateur d'accéder aux services ubiquitaires.

Du point de vue du fournisseur de service Le *graphe d'interaction* définit les relations entre les terminaux de l'utilisateur et les ressources du réseau. Les terminaux utilisateur intègrent des capteurs et des effecteurs du système. Les sommets de ce graphe sont les terminaux et les ressources de la plate-forme. Les flux modaux cheminent le long des arêtes du graphe. Par choix, nous considérons deux classes d'éléments terminaux côté utilisateur : les clients web et les agents téléphoniques. Les téléphones et les agents SIP héritent de la seconde classe. Les navigateurs web et les objets communicants tels que les Nabaztags (figure 3 page 3) héritent de la première.

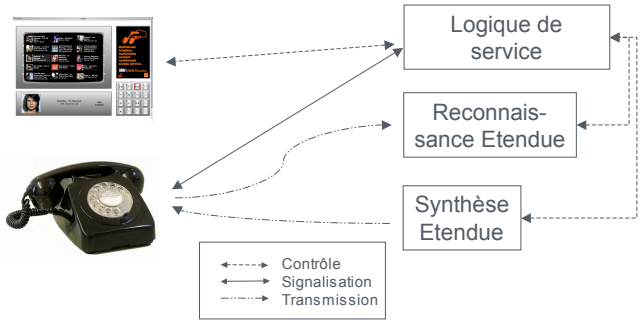


FIG. 2.2 – Graphe d'interaction.

Les relations qui lient les sommets de ce graphe recouvrent trois classes : les relations de signalisation, de contrôle et de transmission de flux média. Les *flux média* sont orientés. Ils représentent le signal envoyé par l'utilisateur (voix, DTMF) au dispositif d'analyse (flux entrant) et celui émis par le système (lecture, synthèse) à destination de l'utilisateur (flux sortant). Dans un contexte IP, ces flux se matérialisent sous forme de flux RTP, définis dans le section 1.5. La *signalisation* permet la négociation de l'établissement des flux média entre des sommets du graphe, le *contrôle* alimente la logique de service grâce à des messages adéquats, suivant la syntaxe définie par l'élément en charge de la logique de service.

2.2 Canevas intégrateur de la chaîne de traitements du flux

Afin d'intégrer les différents terminaux de l'utilisateur dans une même session interactive, nous analysons les différentes configurations d'interaction liées à notre problématique selon le canevas défini par le modèle pipe-line. Pipe-line modélise les échanges entre le système et l'utilisateur lors de l'interaction. Il permet de faire l'inventaire des objets communicants adressés par notre problématique. Nous étudions ce canevas dans trois situations : le pilotage d'un téléviseur depuis une télécommande, la navigation sur internet depuis un navigateur WIMP et l'interaction avec un service vocal. Chacune de ces configurations structure l'environnement de l'utilisateur. Ces structures sont propres à la situation dans laquelle se trouve l'utilisateur et des terminaux dont il dispose. L'étude de ces canevas nous permet de dégager un cadre général sur lequel viendront se greffer les différentes couches de notre modèle d'architecture distribué.

Modèle Pipe-line Le modèle Pipe-line est un canevas intégrateur des activités de l'utilisateur et du système. Le modèle Pipe-line définit par Laurence Nigay en 1994 [Nigay, 1994] représente l'interaction sous forme de flux (figure 2.3). Le flux d'entrée est orienté utilisateur → machine, il possède des propriétés physiques mesurables. le flux achemine ce signal jusqu'au cœur du système en traversant plusieurs couches de traitement. Une chaîne de traitements logiciels est en charge du transport et du traitement de ce flux. Le flux d'entrée modifie le système selon une logique propre au service rendu. Ce changement d'état provoque un changement de l'interface perçue par l'utilisateur. Nous utilisons le patron MVC (section 1.2 page 19) comme une métaphore permettant de représenter le cœur du système dans notre chaîne de traitements du flux média . En effet, si le patron MCV se situe sur le plan implémentationnel, le modèle pipe-line est clairement conceptuel.

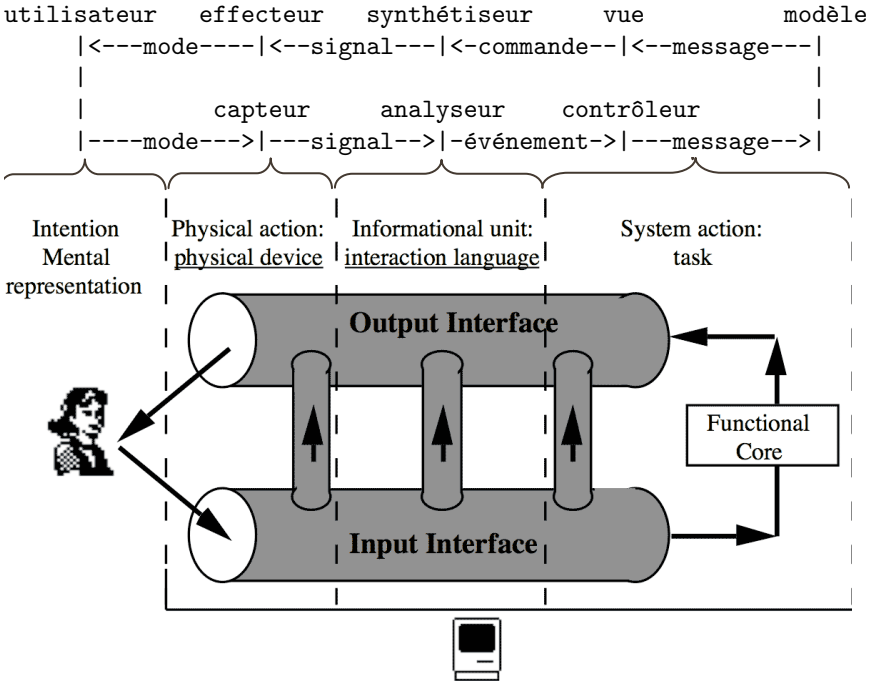


FIG. 2.3 – Modèle Pipe-line [Nigay, 1994].

Cas concret Prenons le cas du pilotage d'un téléviseur avec une télécommande et montrons comment notre modèle capte les situations de ce cas concret Les *capteurs* sont les touches de la télécommande, elles se trouvent à proximité ou dans les mains de l'utilisateur. Les *modes* sollicités sont la vue, l'ouïe et le touché. On apporte un soin particulier à définir la modalité utilisée dans le cadre de l'interaction avec la télécommande. En effet, les matériaux utilisées, la forme, la couleur sont autant de modalités qui relèvent de choix de conception (design) dont le but est de penser l'outil en terme d'usage (*ergonomie*). Le *signal* transmis par la télécommande est un flux, généralement infrarouge, code associé à chacune des touches. Ce signal est *analysé* au niveau du téléviseur. La pression d'une touche de la télécommande provoque la réaction du téléviseur en conséquence.

Le *modèle* lui, se situe à différents niveaux. Le modèle de l'outil se situe au niveau du téléviseur, il est en charge par exemple du contrôle du volume, du canal, du contraste. Le modèle du service se situe chez le fournisseur de service. En effet, la grille des programmes, le contrôle d'accès et la facturation sont à la charge du fournisseur du service. Les modifications apportées aux modèles sont répercutées sur les *effecteurs* du téléviseur : l'écran de la télévision et ses enceintes. Ils se trouvent à proximité de l'utilisateur, de manière à ce qu'il puisse voir ou entendre l'information télévisée. L'augmentation du volume sonore de l'outil se répercute sur les *vues* du dispositif. Les vues modifient alors les propriétés des *synthétiseurs* du dispositif provoquant l'apparition d'une jauge sur l'écran et la variation de la puissance du signal audio.

La figure 2.4 présente le canevas de ce cas concret. Ce cas simple montre l'imbrication des niveaux de vue. On y retrouve la chaîne de processus permettant d'analyser le flux produit par la télécommande et celle en charge de l'adaptation des éléments de présentation. On observe ici l'existence de plusieurs modèles : le modèle de l'outil et le modèle du service. Le modèle de l'outil se situe dans l'environnement proche de l'utilisateur, sa gestion est à la charge de l'utilisateur. Le modèle du service est mutualisé, il est distant et sa gestion est à la charge du fournisseur de services. Comme nous l'expliquions, le contrôleur et la vue sont les points d'entrée et de sortie du modèle de l'outil. Le modèle de l'outil est en charge de dialoguer avec le modèle du service. Cette pluralité des modèles avec leur distribution spécifique doit être prise en compte par notre modèle d'architecture afin de définir une architecture adaptée à la topologie des réseaux de communication. Afin de répondre à notre problématique (l'intégration de la modalité vocale dans les interface web) nous procédons à cette même analyse dans le cadre de l'interaction web et de l'interaction vocale.

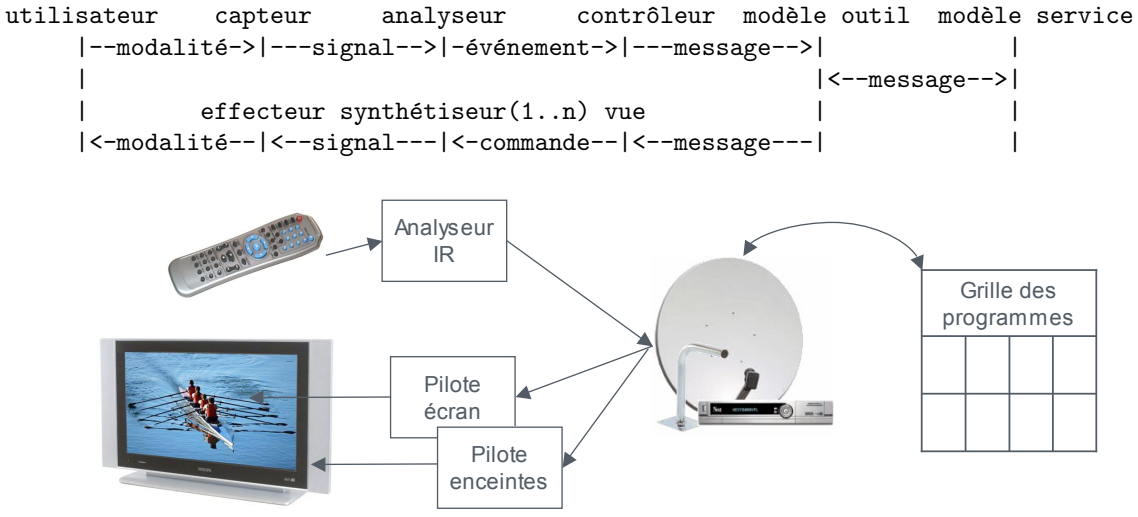


FIG. 2.4 – Mise en évidence de l'imbrication des vues et de la pluralité des modèles.

2.2.1 Interaction web

Dans le cadre d'une interaction web classique [85] *les capteurs* sont : la souris ou le « pavé traquant » [86] permettant d'interagir avec le curseur de la souris ; le clavier et le pavé numérique souvent ne faisant qu'un. Les effecteurs sont l'écran et les enceintes. La gestion de ces organes (analyse et synthèse) est à la charge du système d'exploitation du dispositif qui relaie l'information au navigateur web.

Le *modèle du document* communique avec le moteur de rendu (synthétiseur) du navigateur. Par exemple le moteur *Gecko*, pour Firefox [1], est en charge du rendu graphique du DOM (section 1.3.2) HTML. Il est également en charge de son rendu audio (lecture de fichiers midi, signal audio). Un navigateur web inclut également des plugins[87] en charge de l'évaluation de sous-arbres et de feuilles du DOM. On note le plugin Adode SVG Viewer [4] en charge du rendu vectoriel d'objets graphiques ou Macromédia Flash [3] permettant en plus une gestion en entrée et en sortie des flux multimédia audio et vidéo. Les plugins vidéos Apple QuickTime [5] ou Microsoft Windows Media Player [7] intègrent également les versions récentes de nos navigateurs. Chacun de ces plugins sont autant de synthétiseurs, voire de capteur dans le cas du plugin *Flash*, s'insérant dans la chaîne de traitements du flux modal.

Le navigateur web reçoit des instructions provenant du clavier et la souris (capteurs). Le signal capturé est analysé par l'interface graphique du dispositif (serveur X de UNIX), puis acheminé au client web. Les événements ainsi transmis, provoque la notification du modèle du document. *Dans une même chaîne de traitements, il existe plusieurs modules en charge de la reconnaissance des événements.* Par exemple, les navigateurs Opéra 0.9 et Firefox 2 possèdent leur propre module d'analyse du signal. Ce module se plaçant entre le serveur graphique et le gestionnaire d'événements du navigateur. Ces deux navigateurs permettent la prise en charge du mode gestuel. La navigation gestuelle permet de piloter le navigateur web en procédant à l'analyse des mouvements de la souris. Par exemple, en mode « navigation gestuelle » le déplacement de la souris vers la gauche permet de revenir à la page précédente, action équivalente à l'activation de l'icône du navigateur symbolisant une flèche allant de droite à gauche. Ces deux navigateurs gèrent de manière différente ce mode. L'un nativement, l'autre grâce à l'utilisation d'un plugin.

L'architecture client serveur du web nécessite une répartition du modèle dans le réseau. En effet, tandis que le modèle du document se situe au niveau du client web. Le modèle du service est distant. Il intègre le contrôle d'accès, la gestion des contenus et assure la logique de service. Ces organes peuvent d'ailleurs être eux-même répartis dans le réseau. Le modèle s'en trouve donc segmenté en fonction des propriétés de chacun de ces modules. Par exemple, pour un site marchand, un annuaire sera en charge d'assurer sa propre logique de service pour gérer les utilisateurs, tandis qu'un gestionnaire de base de données sera en charge de gérer un stock de produits, enfin une solution sécurisée de paiement assurera la transaction financière. L'agrégation de ces organes permet la construction du modèle du document qui est envoyé au client web en charge de l'instancier. La figure 2.5 représente les différents acteurs de la chaîne de traitements du flux modal et leur localisation dans l'environnement de l'utilisateur et dans le réseau.

[85] WIMP - Windows Icon Menu Pointer, interface comprenant une interface graphique, un clavier, une souris sur un ordinateur de bureau, ou portable.

[86] Trackpad - dispositif physique permettant d'interagir avec le curseur de la souris en faisant glisser son doigt sur une surface sensible.

[87] Plugin - en informatique, le terme anglais plugin (ou plug-in, se prononçant /plœgin/, du verbe to plug in qui signifie brancher), est employé pour désigner un programme qui interagit avec un logiciel principal, appelé programme hôte, pour lui apporter de nouvelles fonctionnalités [46].

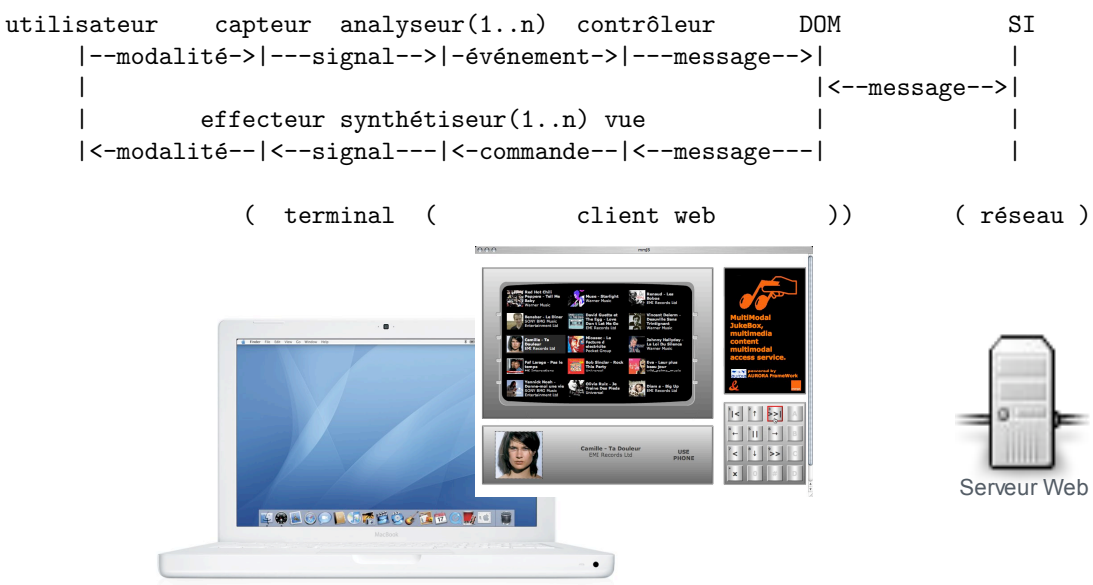


FIG. 2.5 – modèle d'architecture web et répartition des acteurs.

L'observation des acteurs de la chaîne de traitements du signal modal dans le cas de l'interaction web révèle la multiplication des analyseurs et des synthétiseurs. Elle nous révèle également la localisation du modèle de l'outil (DOM) et de ces organes interactifs sur le terminal de l'utilisateur. Enfin, dans le cas de l'interaction web on observe une segmentation du modèle propre à l'architecture client serveur du web. L'interaction entre un utilisateur et un serveur vocal reprend ce schéma, mais se différencie par son architecture client léger. En effet, dan le cadre de l'interaction vocale, le navigateur VoiceXML se situe dans le réseau, tandis que dans le cadre de l'interaction web, le navigateur est sur le terminal.

2.2.2 Interaction vocale

Dans le cadre d'une interaction téléphonique classique, mettant en relation un téléphone fixe ou mobile avec un serveur vocal, *les capteurs* sont : le microphone du téléphone ou du micro-casque ; les touches du téléphone. Les effecteurs sont : l'écouteur ; le haut-parleur ; l'oreillette du téléphone. Ils sont gérés localement par le terminal qui est capable de recevoir et d'envoyer un signal audio. Le terminal est connecté au réseau téléphonique. Ainsi, l'utilisateur accède à distance, en situation de mobilité ou dans un contexte résidentiel, à des services vocaux. L'utilisateur navigue dans des menus interactifs. Il déclenche vocalement ou par pression des touches de son téléphone, des actions distantes. Ces actions donnent lieu à un retour audio révélant la pertinence de l'exploration du service. Le signal ainsi capturé, est envoyé au serveur vocal interactif (SVI). Le SVI joue le rôle d'interlocuteur.

L'architecture téléphonique déporte le traitement du signal et la logique du service du terminal vers le réseau. En effet, contrairement à l'interaction web, l'organe assurant la logique de l'outil ne se situe pas sur le terminal mais dans le réseau. L'architecture 3-tiers de notre SVI d'étude (VIP) distribue la chaîne de traitements du flux modal dans le réseau. Le module d'analyse du signal est en charge de l'extraction des événements à partir du signal. Ces événements vont permettre d'alimenter une logique de service.

Analyseur Dans la partie de la chaîne située en amont du modèle, on distingue deux classes d'interruption provenant de l'utilisateur : des interruptions prenant la forme d'un signal audio vocal, analysées par des dispositifs de traitements de la parole et des interruptions sous forme d'événements DTMF permettant à l'utilisateur d'interagir avec le système grâce aux touches de son téléphone.

DTMF Dans ce cas d'interaction, nous distinguons deux modes : le vocal et les DTMF. Chacun des ces deux modes donne lieu à une analyse du signal particulière. Dans le cas de DTMF, l'analyse propre à la définition des DTMF (figure 2.6) est réalisée par extraction des harmoniques. En effet, une analyse du signal par calcul de la transformée de Fourier permet d'extraire les deux fréquences qui composent le code DTMF. La lecture dans le tableau de la figure 2.6 permet de retrouver le code correspondant.

	1209 Hz	1336 Hz	1477 Hz	1633 Hz
697 Hz	1	2	3	A
770 Hz	4	5	6	B
852 Hz	7	8	9	C
941 Hz	*	0	#	D

FIG. 2.6 – Les codes DTMF (Dual-tone multi-frequency) sont les combinaisons de fréquences utilisées pour la téléphonie moderne (c'est-à-dire pas à impulsions). Ces codes ont permis la création de services vocaux. Techniquement, les touches de téléphones correspondent à un couple de deux fréquences audibles qui sont jouées simultanément. Ces fréquences peuvent être reconnues par des appareils électroniques et sont utilisées pour réaliser des serveurs vocaux. On peut constater que contrairement aux terminaux habituels, on a une colonne supplémentaire à droite, avec des touches de A à D : celles-ci étaient utilisées par l'armée américaine pour représenter la priorité d'une communication.[46].

Reconnaissance vocale Dans le cas de la reconnaissance vocale, l'analyseur procède également à une extraction des harmoniques du signal. Le cepstre[88] obtenu s'avère plus compliqué à interpréter. La complexité de la voix humaine comparée aux seize codes DTMF est sans équivoque. En conséquence, les techniques de reconnaissance vocale modernes couplent à l'analyse des harmoniques, une recherche des chaînes cachées de Markov[89]. Cette seconde analyse nécessite un modèle de reconnaissance propre à la logique de service, il est constitué des mots à reconnaître.

Par exemple, le système de reconnaissance de la parole du service de renseignement 118 710[8] reprend cette technologie. Dans ce service, le signal audio est transmis au système qui l'analyse en fonction du modèle de reconnaissance spécifique. Les mots reconnus sont alors transmis au système sous forme d'événements. Le système utilise deux modèles de reconnaissance vocale pour rendre compte de la navigation (« *suivant.* », « *précédent* »...) et de la recherche parmi la liste des abonnés au service téléphonique (« *Antoine Durand.* ») ou des villes (« *Paris.* »). Le premier modèle est dit « mots isolés », le second, est un modèle TGV[90].

[88] Cepstre - anagramme de spectre.
[89] Analyse probabiliste, également utilisée dans le cadre du décodage du génome.
[90] TGV - Très Grand Vocabulaire, modèle de reconnaissance vocale

DSR [91] La décomposition des modules de reconnaissance vocale distribuée dans notre modèle illustre le concept de signal et d'événement. Comparée au module de reconnaissance classique, l'architecture DSR exploite différemment les capacités du terminal et celles du réseau. Le module DSR effectue deux analyses du signal. Une première analyse extrait le cepstre (analyse cepstrale[92]) du signal (grande taille) au niveau du terminal ; ce dernier transmet les vecteurs de petite taille sur le réseau. Dés lors, une seconde analyse, markovienne celle-ci, extrait les éléments du modèle (de grande taille) contenus dans le signal originel. La figure 2.7 présente la répartition des acteurs de la reconnaissance vocale distribuée. On distingue le signal décomposé par analyse de Fourier et l'événement, produit par analyse de chaînes cachées de Markov. Entre les deux se situe le cepstre qui, du point de vue de Fourier est un événement et du point de vue de Markov est un signal. Suite à ces deux analyses, le signal est transmis au modéle du service vocal sous forme d'un mot reconnu avec un facteur de confiance relative à l'analyse. Cette technique permet de limiter la consommation de bande passante. La mise en œuvre de ce modéle est l'objet du projet AURORA [Pearce *et al.*, 2000, 29]. Notons que ce projet AURORA est différent du projet ITEA Aurora auquel nous avons pris part.

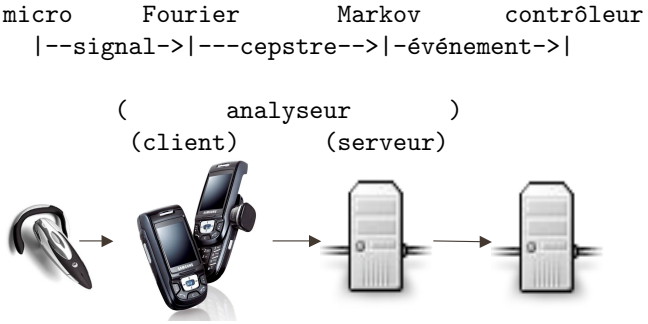


FIG. 2.7 – Modèle pipe-line et reconnaissance vocale distribuée.

Modèle À chacune des actions de l'utilisateur correspond un changement d'état du système. Ce changement d'état a pour conséquence de modifier les parties visibles (ou audibles) du système. En effet, le modèle mis à jour répercute les modifications sur les différentes vues associés. Ces modifications prennent la forme de notifications intégrant le contenu structuré permettant d'adapter la vue en conséquence. Selon le huitième mythe de Sharon Oviatt [Oviatt, 1999] il est illusoire de chercher à traiter les différentes modalités de manière uniforme. Pour cette raison, chacune des chaînes de traitements, en entrée comme en sortie, donne lieu à une implémentation spécifique, adaptée aux particularités des dispositifs physiques, des spécifications des constructeurs et de l'utilisation qui est faite du système. Cette spécialisation est valable également pour une même modalité. Cela signifie qu'une modalité donnée sera gérée de manière différente en fonction des capteurs et effecteurs déployés. Elle varie selon la qualité des micros utilisés ; des performances des dispositifs de traitements du signal mis en jeu et du contexte d'utilisation. Par exemple, une interaction en environnement bruité sera différente d'un interaction dans un milieu calme propice à une reconnaissance fine de la parole.

Cette architecture MVC (Model View Controller) permet de définir des outils spécifiques dont les interfaces sont réparties sur les terminaux de l'environnement de l'utilisateur. Au cours

[91] DSR - Distributed Speech Recognition.
[92] Cepstrale - anagramme de spectral.

de notre étude, nous apportons un soin particulier à rendre accessible ces outils depuis des interfaces web. Cette intégration passe par la définition de messages assurant à la fois le contrôle du modèle de l'outil et la visualisation de son état dans l'interface web. Pour l'utilisateur, la manipulation de ces outils passe par l'élaboration d'un dialogue multimodal évolué.

Synthétiseur La synthèse vocale est aussi effectuée par des organes distants de l'utilisateur. La solution vocale joue aussi le rôle d'interlocuteur. Elle est en charge de la synthèse du flux audio à partir d'un catalogue de voix et d'extraits sonores jouant tour à tour des séquences appropriées, interagissant avec l'utilisateur, le faisant patienter et dynamisant le dialogue.

Le modèle du service vocal combine à la fois, la logique de la navigation et la logique du dialogue. En effet, les interfaces vocales suivent la logique d'un dialogue dirigé selon la nature du service. Dans ce cas également, on distingue le modèle du document et le modèle du service. Dans notre modèle, ces deux aspects sont distincts. Le modèle du service est distant, réparti chez les différents acteurs du service. Dans le cas de l'accès vocal à un service bancaire, la partie métier du service (information sur les comptes, les portefeuilles, le suivi des opérations) est gérée par l'établissement bancaire. Dans le cas d'un accès vocal à l'information voyageur d'un réseau de bus, la géo-localisation et le trajet des bus est stocké chez l'exploitant du réseau de transport. Le modèle du document est une interface du système d'information distant. Cette architecture 3-tiers assure la confidentialité des données et le contrôle des accès au système d'information des différents acteurs du service. La gestion du modèle du document, attribuée au navigateur dans le cadre de l'interaction web, et à la charge du serveur vocal interactif dans ce cas d'interaction vocale.

La figure 2.8 présente notre modèle d'interaction dans le cadre de l'interaction entre un utilisateur et un SVI. Outre la multiplication des modules d'analyse et de synthèse déjà constatée lors de l'étude des interactions web, on remarque une segmentation différente des acteurs de la chaîne de traitements du flux modal. Dans ce cas de figure, le navigateur web se trouve dans le réseau. Le téléphone intègre les capteurs et les effecteurs ainsi que des fonctionnalités permettant de transmettre les flux média aux différents organes en charge de leur traitement dans le réseau.

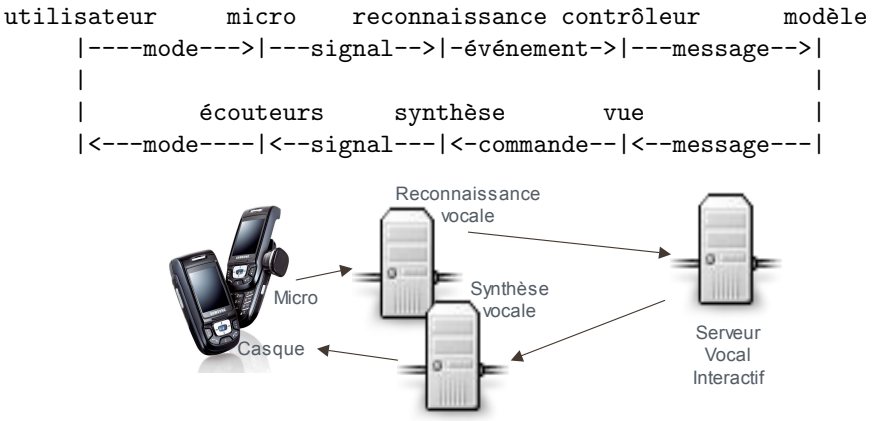


FIG. 2.8 – Modèle pipe-line dans le cadre de l'interaction vocale.

2.2.3 Synthèse sur l'intégration des flux média

L'interaction multimodale distribuée utilise les multiples terminaux se trouvant dans l'environnement de l'utilisateur afin de les utiliser de manière complémentaire. Par exemple, les interfaces graphiques des ordinateurs équipés de navigateur web se voient complétées par les interfaces vocales des téléphones. Ainsi, un service multimodal dans un tel contexte sera rendu de manière simultanée sur l'ordinateur grâce à une interface web et sur le téléphone au travers une interface vocale.

Notre problématique nous mène à intégrer la modalité vocale dans les interfaces web. Pour ce faire, il convient de combiner les deux canevas précédemment évoqués. De plus, notre approche consiste à fournir aux développeurs web un service leur permettant d'inclure les terminaux téléphoniques de l'utilisateur dans une même session. Cette approche orientée service ajoute une contrainte supplémentaire relative à la définition d'un modèle d'architecture multimodale distribuée.

Cette approche révèle un degré de difficulté supérieur lié à la distribution du modèle de l'outil. Notre modèle d'architecture se doit de prendre en compte cette distribution. Nous utilisons ce canevas intégrateur lors de la définition de notre modèle d'architecture distribuée comme une combinaison d'interaction web et d'interaction vocale. Les sections qui suivent décrivent nos modèles d'interaction et leur combinaison.

2.3 Modèle d'architecture web

Cette section présente la manière dont nous construisons notre modèle d'architecture baptisé UbiArch pour « Architecture Ubiquitaire ». UbiArch modélise l'interaction web entre un utilisateur et une interface WIMP[93], mais aussi entre un utilisateur et un serveur vocal. L'objectif de UbiArch est de définir une architecture logicielle qui intègre les terminaux de l'utilisateur dans une même session web. Nous définissons UbiArch par analyse et construction : analyse de l'évolution d'internet depuis le début des années 90, construction du modèle implémentationnel UbiArch dans le respect des principes fondamentaux des modèles conceptuels de Seeheim et Arch. La construction du modèle donne lieu à l'incrémentation de son numéro de version. UbiArch 0.9 fait référence à la version 0.9 du protocole HTTP sur laquelle repose sa composante de dialogue. Les versions 1.0 et 2.0 font référence au « web 2.0 ». La version 1.0 d'UbiArch caractérise l'évolution du web côté serveur. La version 2.0 caractérise la plus récente évolution côté client d'internet, généralement étiquetée de la signalétique 2.0.

2.3.1 UbiArch 0.9

Dans cette section nous construisons notre modèle implémentationnel d'interaction web à partir de la définition d'internet par Tim Berners-Lee (section 1.3.1), au début des années 90. UbiArch 0.9 s'articule autour du protocole HTTP et de sa logique de question/réponse. UbiArch 0.9. Dans cette section nous observons comment UbiArch 0.9 respecte les principes fondamentaux du modèle conceptuel d'interaction de Seeheim. Ensuite nous explicitons les limites de ce modèle et nous proposons de le faire évoluer.

UbiArch 0.9 Notre modèle UbiArch 0.9 possède chacune des couches de Seeheim. La composante de dialogue d'UbiArch 0.9 repose sur la méthode GET du protocole HTTP. Ce choix permet l'identification claire des couches du modèle de Seeheim. La figure 2.9 superpose notre modèle implémentationnel et les composantes du modèle de Seeheim. Le couple requête/réponse assure le contrôle du dialogue. Pour cette raison nous définissons le client et le serveur HTTP comme des agents en charge du contrôle du dialogue. La ressource demandée, l'hyperdocument, constitue une projection du « noyau fonctionnel » que serait internet ; ce dernier est considéré ici, comme un maillage d'hyperdocuments. La présentation du document est assurée par le moteur de rendu du navigateur web. La délégation sémantique est assurée par l'acheminement du DOM encapsulé dans la réponse HTTP jusqu'au moteur de rendu. Une fois le document rendu visuellement, graphiquement, il présente à l'utilisateur un ensemble d'interacteurs, les hyperliens, permettant de continuer le processus d'interaction.

UbiArch permet donc la navigation dans un maillage de liens hypertextes. Si UbiArch 0.9 rend compte de la navigation web à travers des hyperdocuments HTML, l'évolution du web au cours des années 90 nous révèle une forte augmentation des portails, annuaires et autres services en ligne. L'interaction web intégre le concept de session. Le serveur prend alors en charge le maintien de la session et stocke les objets entre deux requêtes. Cette mutation s'accompagne de la génération dynamique des contenus souvent stockés dans des bases de données ou sur des serveurs distants. Les interfaces web permettent de manipuler un grand nombre d'objets, de les trier. L'utilisateur n'est plus seulement au bout de la chaîne de production de contenus, il l'intègre et devient auteur, le modérateur ou l'éditeur de ces documents numériques [Pédauque, 2006].

[93] WIMP - Acronyme anglais pour « Windows, Icons, Menus and Pointing device », (fenêtres, icônes, menus et dispositif de pointage), le paradigme WIMP présente des bases fonctionnelles d'une interface graphique informatique.

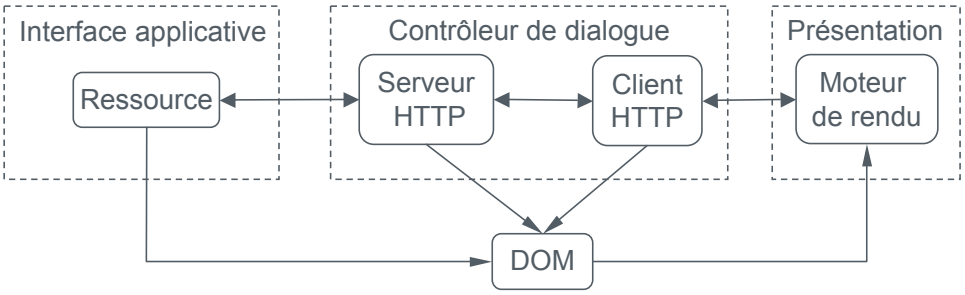


FIG. 2.9 – UbiArch 0.9.

Ceci nous conduit à reconsidérer le rôle du noyau fonctionnel et de l'interface applicative. Le premier devient distribué et la seconde devient dynamique.

La première évolution de notre modèle consiste à y intégrer le noyau fonctionnel de l'application et par conséquent de reprendre un des principes fondamentaux du modèle Arch, présenté section suivante.

2.3.2 UbiArch 1.0

Cette section décrit la première évolution de notre modèle UbiArch. L'objectif de cette évolution est de prendre en compte la génération dynamique de contenus côté serveur et le maintien d'une session entre le client et le serveur. UbiArch 1.0 caractérise le passage du web documentaire au « web des services ». Cet aspect nécessite l'intégration du noyau fonctionnel dans le modèle d'architecture.

UbiArch 1.0 L'intégration du noyau fonctionnel dans UbiArch 1.0 caractérise l'évolution du « web documentaire » vers le « web des services ». Ceci a pour principal effet la distribution du noyau fonctionnel. Nous spécifions le noyau fonctionnel comme un ensemble de composants de type service web. Les services web sont organisés de façon hiérarchique. L'éclatement du cœur de l'application permet la distribution de la logique de service dans le réseau, assurant du même coup, la réutilisation des modules fonctionnels, en charge de tâches spécifiques (authentification, calcul réparti, module de recherche...) et l'agrégation de contenus (fils RSS, podcasts, vidéocasts) côté serveur. L'élément racine de la hiérarchie est l'adaptateur sémantique, il est en charge de la génération du DOM, conformément à la stratégie de dialogue adoptée.

Stratégie de dialogue UbiArch 1.0 permet la définition d'une stratégie de dialogue adaptée au métier que le service cherche à traiter. La littérature nous renseigne sur différentes stratégies de dialogues adaptées au contexte. Par exemple, les auteurs de l'article [Depaulis *et al.*, 2006], définissent une stratégie de dialogue, modélisée par un réseau de Pétri, destinée à des applications de conception technique [94]. Dans le cadre de notre travail de recherche [Bazin *et al.*, 2006] nous avons mis en œuvre une stratégie de dialogue fondé sur la manipulation d'un hypergraphe [Berge, 1987] et permettant, par le biais d'un dialogue interactif d'identifier le lieu de départ et d'arrivée d'un utilisateur cherchant son chemin.

[94] CAO - Conception Assitée par Ordinateur

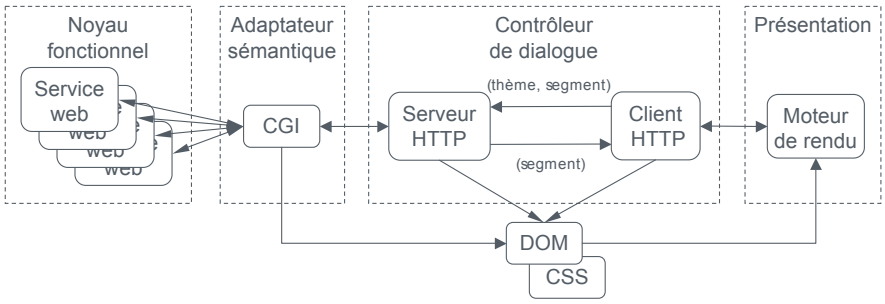


FIG. 2.10 – UbiArch 1.0.

Afin d'illustrer le concept de stratégie de dialogue et d'identifier sa place dans notre modèle UbiArch 1.0, nous définissons ici une stratégie de dialogue permettant la manipulation de listes. L'interfaçage des services web avec les bases de données donne lieu à la manipulation de listes : listes de billets dans le cas des blogs, liste d'articles ou de livres dans le cas des bibliothèques numériques, liste de personnes dans les annuaires, liste de messages dans les forums, liste d'articles dans les magasins en ligne, liste de documents multimédia sur les podcasts ou les vidéocasts, liste de mails sur les webmails, liste de liens dans les moteurs de recherche. Ces listes sont organisées sous forme de files thématiques rendues, accessibles par un menu. Nous définissons formellement cette logique de dialogue à partir des caractéristiques de la liste : la cardinalité, le nombre total d'éléments de la liste ; la ségmentation, le nombre d'élément maximum de la liste à afficher ; la progression, le segment courant. À partir d'une thématique donnée notre contrôleur de dialogue retournera la première partie des éléments de la thématique dont la cardinalité sera en accord avec la segmentation par défaut du contrôleur. Afin de pouvoir progresser dans cette liste, l'utilisateur peut fournir au contrôleur de dialogue le numéro du segment souhaité par l'intermédiaire de la barre de navigation (figure 2.11). Notons ici que la thématique courante est une variable de session du serveur.

FIG. 2.11 – La barre de navigation du moteur de recherche Google permet la navigation dans une liste de lien trier selon les *page rank*. Ce score dépend des mots clefs entrés par l'utilisateur. Nous utilisons une barre de navigation similaire pour naviguer dans la liste des utilisateurs des ressources informatiques du laboratoire. Cette liste est construite en fonction des requêtes LDAP de l'utilisateur (Démonstrateur permettant la gestion d'un système biométrique, section 4.1.2).

Session sur le serveur Le serveur web est en charge du maintien de la session web. Cette dernière assure la survie des objets côté serveur, cantonne l'interprétation du DOM à la présen-

tation des données. Dans la session sont stockés la progression de l'utilisateur dans le service et le contexte de l'interaction. L'utilisation de session permet de définir des stratégies de dialogue plus évoluées que la simple demande de ressource définie dans notre précédent modèle. Cette architecture particulièrement adaptée aux clients légers implique une augmentation de la charge du serveur, liée au maintien de la session. Il conviendra donc de minimiser l'utilisation de la session sur le serveur.

Factorisation des attributs de style sur le client Parallèlement, les attributs de la présentation, relatifs au style, sont factorisés. L'organisation sous forme de classes des éléments qui composent l'interface permet une spécialisation non plus des éléments eux-mêmes, mais de leur classe. Ce point assure la mutualisation de code et la réutilisation des attributs de style, la cohérence de la charte graphique dans le cas des interfaces web WIMP, ou de la charte sonore des interfaces vocales. Cela se traduit au niveau implémentationnel par l'implémentation des feuilles de styles CSS (section 1.3.2). Si les feuilles de styles assurent une mutualisation des attributs relatifs au rendu, l'implémentation qui en est faite dans nos navigateurs web traditionnels suggère que la fusion des données s'effectue au chargement du DOM. Cependant, ce point n'affecte pas notre modèle car une fois instancié côté client, le DOM reste statique dans cette version 1.0.

Le modèle UbiArch 1.0 assure la définition de services dynamiques, accessibles depuis des terminaux pouvant disposer de peu de ressources. Notre plate-forme vocale (navigateur VoiceXML) fonctionne sur ce modèle. En revanche, UbiArch n'exploite pas les fonctions avancées et les ressources des terminaux plus puissants. La navigation web se heurte au caractère global et non persistant de la présentation. En effet, à chaque tour de parole de la stratégie de dialogue, la présentation est complètement rechargée par le navigateur. Le web 2.0 se caractérise, entre autre, par ces nouvelles interfaces web interactives qui ne clignotent plus. En plus de cela, les interfaces web 2.0 offrent des possibilités d'interaction avancées, où les utilisateurs peuvent « glisser/déplacer » des objets graphiques redimmensionnables ou effectuer des « clicks droit » pour accéder à des menus contextuels. Nous définissons alors UbiArch 2.0, notre modèle d'architecture web qui rend compte de ces nouvelles interfaces.

2.3.3 UbiArch 2.0

Cette section décrit la troisième évolution d'UbiArch. Elle se caractérise par la décomposition de la présentation et la migration du contrôleur de dialogue côté client. La version 1.0 d'UbiArch souffrait de la (re)génération monolithique du DOM coté serveur. Nous proposons donc de superposer les nœuds du DOM en charge de présentation avec une hiérarchie d'agents PAC en charge du contrôle du dialogue.

Modèle UbiArch 2.0 UbiArch 2.0 intègre une hiérarchie d'agents PAC sur le client web. Cette hiérarchie est en charge du contrôle du dialogue. L'information sémantique fournie à l'agent PAC racine de la hiérarchie est stockée dans sa facette abstraction et est répercutée sur l'ensemble de la hiérarchie PAC. Chacun des agents stocke cette information sémantique et adapte sa facette présentation en conséquence. La présentation se matérialise par un nœud du DOM et la classe CSS à laquelle il fait référence. Cette évolution de notre modèle passe par un changement de nature des objets fournis par l'adaptateur. Dans UbiArch 1.0 l'objet retourné était le DOM de l'interface contenant la sémantique du noyau fonctionnel. Dans UbiArch 2.0, les objets sont uniquement sémantiques. Les aspects présentation étant complètement définis par le DOM et la CSS côté client. La figure 2.12 présente l'organisation des composants de UbiArch 2.0. La version 2.0 de notre modèle permet la définition de stratégies de dialogue plus évoluées

que HTTP. Néanmoins, les trames des protocoles haut niveau à définir sont encapsulées dans le protocole HTTP au moment de l'échange entre le serveur (CGI) et le l'agent PAC racine de la hiérarchie. HTTP est simplement utilisé pour le transport. Si la figure 2.12 ne laisse apparaître qu'une seule hiérachie, le client peut en intégrer plusieurs, permettant l'intégration de plusieurs services distants, par exemple, la diffusion d'informations concernant la météo, les fluctuations boursières ou les résultats sportifs.

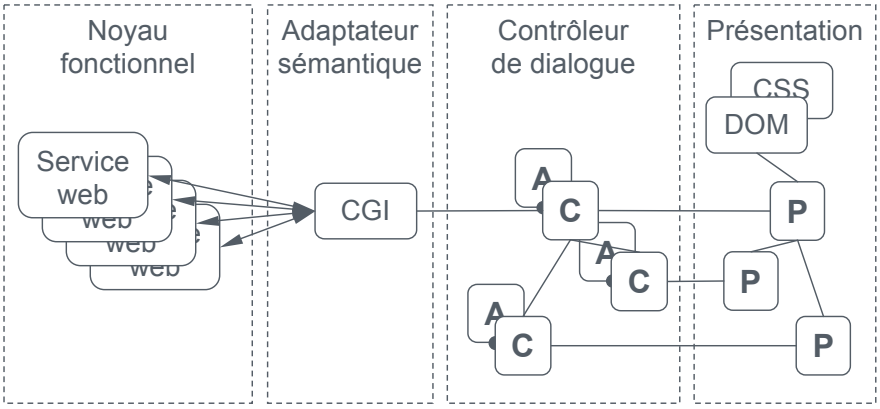


Fig. 2.12 – UbiArch 2.0.

Si on reprend l'exemple de notre stratégie de dialogue fondée sur le manipulation de listes, la facette abstraction de l'agent racine connaît le segment courant et ses composants, le nombre total de segments, le nombre total d'items et la taille maximale d'un segment. Elle obtient ces informations de la facette contrôle qui interroge l'adaptateur distant. Dès lors, les répercutions de cette connaissance sur la hiérarchie PAC, par le biais des facettes de contrôle, modifie le DOM en conséquence. L'implémentation du contrôleur de dialogue sur le client permet la gestion de plusieurs listes sur la même interface mais surtout, l'interaction avec chacune d'elles sans impliquer de modification, ni même d'accès sur les autres. On obtient ainsi, une interaction continue et non plus segmentée par la disparition, réapparition de l'interface web.

Synthèse sur le modèle d'architecture web Notre modèle d'architecture web permet à l'utilisateur de solliciter des ressources distantes. Il coordonne les ressources de manière adaptée à la topologie du web et permet au concepteur de services de mettre en œuvre des outils évolués. Notre modèle permet la définition d'applications web riches.

2.4 Modèle d'architecture pour l'interaction vocale

Cette section décrit notre modèle d'architecture vocale entre un utilisateur et un serveur vocal interactif (SVI). Comme nous l'expliquions dans le section 2.3.2, UbiArch 1.0 modélise l'interaction entre un utilisateur un service web et un utilisateur en mettant l'accent sur le rôle du serveur. Les raisons qui nous ont poussé à faire évoluer notre modèle vers une version 2.0 dans d'une interaction web, à savoir le rafraîchissement d'une partie de l'interface et la minimisation des quantités de données échangées à travers le réseau, ne s'applique pas à l'interaction vocale. Dans le cadre de l'interaction vocale, l'interaction est continue et temporelle. Le dialogue reste séquentiel mais la modalité utilisée dépend du temps. De plus l'architecture d'un serveur vocal place le client web dans le réseau, ne sont envoyés au téléphone (client léger) uniquement les flux média.

Notre modèle d'architecture vocale, permet la construction du service (agrégation de services web) par l'adaptateur sémantique. Le contrôle du dialogue et la présentation sont alors retournés sous forme d'un document VoiceXML (DOM VoiceXML). Cette logique de dialogue est en charge du traitement des flux média selon l'instance spécifique du DOM.

Noyau fonctionnel et adaptateur sémantique Cette approche rend également compte de l'interaction entre un utilisateur et un serveur vocal interactif. En effet, le SVI est un client web (section 1.4). À ce titre, il instancie un document (DOM VoiceXML) que lui fournit le serveur web. Le serveur web joue le même rôle que dans le cadre de l'interaction web sous-tendue par notre modèle UbiArch 1.0. Il agrège un ensemble de services web unitaires permettant d'authentifier l'utilisateur, de collecter des données... Il manipule donc les objets métiers. À partir de cette agrégation de services unitaires, le serveur web construit le service vocal sous la forme d'un document (DOM VoiceXML) manipulable par le client web. Ce document est alors transmit au client web qui instancie le DOM ainsi construit et assure la logique de service.

Contrôle du dialogue et présentation Comme pour l'interaction web, le protocole HTTP est en charge du transport du DOM. Cependant, le rôle du client web dépend de la nature du DOM instancié. Dans le cadre de l'interaction vocale, le DOM instancié est un DOM VoiceXML. À ce titre il intègre les structures de contrôle du dialogue identifiées dans la section 1.4. Ainsi le contrôle du dialogue est spécifié par un jeu d'énoncés permettant de décrire les attentes du système. Les événements utilisateur attendus sont décrits par des grammaires (SRGS) qui initialisent les capteurs et les analyseurs propres aux modalités attendues. Le rendu audio des énoncés est complètement décrit par les nœuds SSML intégrés au DOM VoiceXML. La séquence SSML est ainsi en charge du rendu audio (présentation).

Si le DOM HTML met l'accent sur la présentation, et délègue les aspects contrôle de dialogue au nœuds xForm, le DOM VoiceXML est orienté contrôle de dialogue et intègre les aspects présentation par l'intermédiaire des nœuds SSML.

Pourquoi UbiArch 1.0 est-il-adapté à l'interaction vocale ? Dans le cadre de l'interaction vocale, nous ne retrouvons pas les éléments qui ont motivé le passage à une architecture UbiArch 2.0. En effet, nous avons défini la version 2.0 de notre modèle, d'une part pour des contraintes liées au mode graphique (rafraîchissement des pages web dans la version 1.0 de UbiArch) et d'autre part pour maintenir une session côté client. Or, d'une part, la perception par l'utilisateur du chargement du DOM est beaucoup moins critique dans le cadre de l'interaction vocale, d'autant plus qu'une infrastructure et un dimensionnement réseau adapté pallie

complètement à cette limite, et d'autre part, les spécifications VoiceXML impose le maintien d'une session côté client. En effet, VoiceXML spécifie trois niveaux permettant de définir trois portées des objets manipulées par le client web [Rec. VXML, 2004].

1. Le niveau plate-forme, définit les objets dont la portée est globale à toutes les sessions de la plate-forme (dansle cadre d'une plate-forme mutualisée), cet espace contient par exemple les objets relatifs à la version du navigateur VoiceXML, relatifs aux ressources de synthése et de reconnaissance disponibles.
2. Le niveau session, définit la portée des variables propres à la session VoiceXML. L'identifiant de session, les messages d'erreur peuvent être définis avec ce scope et par conséquent être définit à l'initialisation de la session VoiceXML pour toute la durée de la session.
3. Enfin, le niveau local possède une portée qui est le sous arbre du DOM VoiceXML dans lequel les variables sont définies. C'est dans ce scope par exemple que l'ont défini la variable ECMA permettant de stocker le signal audio enregistré par le script utilisant le tag `RECORD`

Ainsi, avec une architecture UbiArch de type 1.0, il est possible de maintenir une session sur le client et de définir des services vocaux performants capables de fournir un retour sémantique immédiat à l'utilisateur.

La figure 2.13 décrit notre modèle d'architecture vocale fondé sur UbiArch 1.0. On y observe l'agrégation sémantique des services web côté serveur, le transport du DOM par le protocole HTTP, l'instanciation de la logique de dialogue sur le client ainsi que le contrôle de la présentation depuis le client web.

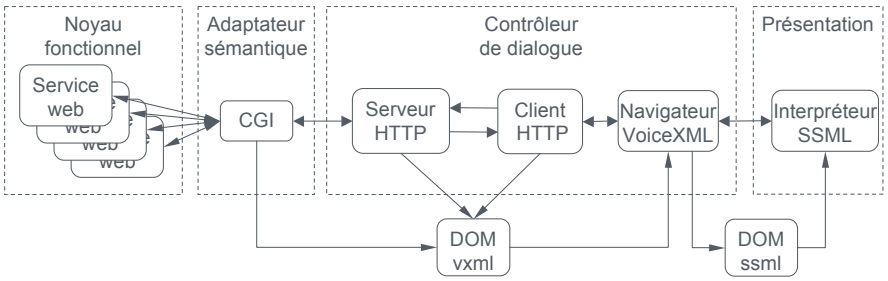


FIG. 2.13 – modèle d'architecture vocale fondé sur UbiArch 1.0.

Synthèse sur l'interaction vocale Notre modèle d'architecture vocale intègre une logique de dialogue tirant partie de la complexité et des ressources du web. Il permet de manipuler un ensemble de structures de contrôle du dialogue définies de manière formelle. Cette formalité nous permet de distinguer la définition du service vocal et les artefacts technologiques permettant d'accéder au téléphone de l'utilisateur. De plus le DOM VoiceXML (la structure inhérente à tout document VoiceXML) assure la manipulation formelle des propriétés du modèle projectif du dialogue de Vernant. En effet, nous veillerons à ce que les dialogues (section 3.5.3) que nous implémentons (section 4) convergent vers un but. Ce but étant par exemple le transfert d'une information (horaires de bus), l'exécution d'une tâche (interrogation d'un répondeur). Dans le cas d'un dialogue divergent, nous veillerons à utiliser des messages d'erreur significatifs permettant d'influer sur le déroulement du dialogue et dans le cas d'un échec, nous en avertirons systématiquement l'utilisateur.

2.5 Intégration du mode vocal dans les interfaces web

L'objectif de notre étude est de définir un service permettant d'interagir avec une interface web depuis le téléphone de l'utilisateur par le biais de ces fonctionnalités vocales et DTMF. Nous avons donc défini plusieurs versions d'UbiArch, notre modèle d'architecture web, en tenant compte des récentes évolutions du web. Ensuite, nous avons défini notre modèle d'architecture vocale, sur les principes de la version 1.0 de UbiArch, de manière adaptée aux particularités d'un serveur vocal interactif.

Ainsi nous définissons une architecture combinant ces deux modèles d'interaction. L'objectif est de fournir une couche de présentation distribuée sur plusieurs terminaux. Afin d'assurer la cohérence des différentes présentations, nous attachons l'élément racine de la hiérarchie PAC en charge du contrôle du dialogue dans UbiArch 2.0, au contrôleur de dialogue de notre modèle d'architecture vocal. Dans cette configuration la logique de service assurée par les scripts VoiceXML impacte non seulement le mode vocal du téléphone mais aussi les objets graphiques de l'interface web. Ainsi, l'interface web et l'interface vocale collaborent afin de produire un retour sémantique distribué, des changements d'état du service .

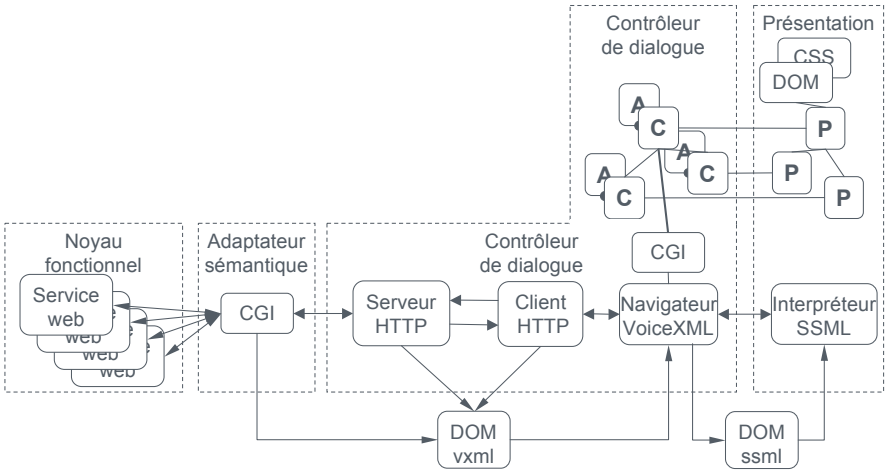


Fig. 2.14 – modèle d'architecture multimodale.

Cette architecture intègre le mode vocal dans les interfaces web, elle vient compléter l'architecture UbiArch 2.0 en lui adjoignant une hiérarchie PAC capable de collaborer avec une logique de dialogue propre au mode vocal. Ainsi les spécificités du mode vocal viennent compléter celles des dernières génération d'interfaces web. Sur la figure 2.14 nous détaillons cette intégration. Comme nous le précisons, cette figure ne laisse pas apparaître les autres éléments qui composent l'interface. Il faut garder en tête que plusieurs hiérarchies d'agent PAC peuvent cohabiter et collaborer dans une instance unique du DOM HTML. La hiérarchie d'agents visibles sur la figure 2.14 et les facettes *P*résentations (nœuds du DOM HTML) de ces agents, ne constituent qu'une petite partie de l'interface. La section 4.3 page 98 présente une mise en œuvre de plusieurs composants interactifs. Dans cette section nous utilisons cette architecture pour étendre les capacités d'interaction de l'application web Slidy en lui adjoignant des fonctionnalités de contrôle par la voix ou les DTMF. Plus généralement le chapitre 4 nous détaillons trois mises en œuvre de cette architecture ayant trait à divers domaines d'activité.

Synthèse sur l'architecture implémentationnel UbiArch est destiné à la conception de services ubiquitaires intégrant des interfaces multimodales distribuées. Dans l'optique de traiter la problématique de la multimodalité, UbiArch reprend les principes fondamentaux du modèle PAC-Amodeus : il est fondé sur le modèle Arch et intègre une hiérarchie d'agent PAC pour modéliser la composante de dialogue. Contrairement à PAC-Amodeus, UbiArch est une architecture implémentationnelle dédiée au web et en mesure de prendre en compte l'interaction vocale. De plus, UbiArch hérite des propriétés de Arch et notamment du mécanisme de branche. UbiArch utilise ce mécanisme de branche afin de rendre compte de la distribution de l'interface dans l'environnement de l'utilisateur. Enfin, UbiArch exploite les protocoles du web et de la VoIP (section 1.5 page 36) afin d'intégrer au mieux le mode vocal dans les interfaces web. Comme le modèle H4, UbiArch réhabilite la délégation sémantique du modèle de Seeheim, et ce, pour des contraintes liées au web.

Afin de réunir les interfaces web et vocales de l'utilisateur dans une même session, nous identifions un composant directement attaché au navigateur VoiceXML : le CGI du contrôleur de dialogue de la figure 2.14. Ce composant réunit le client web (outils) et la logique de dialogue évoluée (partenaire). Sur notre plate-forme d'étude VIP, ce composant n'existe pas. Le chapitre qui suit détaille les évolutions apportées à la plate-forme VIP afin de mettre en œuvre notre modèle. Nous détaillons également le mécanisme permettant cette communication entre les différentes instances de DOM, la composition des messages échangés, le formalisme des données transmises et la scénarisation des dialogues sous forme d'automates.

Chapitre 3

Mise en œuvre du modéle, spécification du composant mVIP

Dans le chapitre 2, nous avons inventorié et structuré les éléments qui composent l'environnement de l'utilisateur dans le but de réunir les terminaux, dans une même session interactive, à travers le réseau. Nous avons défini un modèle implémentationnel d'interaction web. D'un côté, notre modèle fondé sur les modèles conceptuels de la littérature, nous permet de rendre compte d'outils graphiques destinés à un usage ciblé et rendus interactifs selon une logique de dialogue. De l'autre, notre modèle utilise les standards du web ; ceci nous assure de cibler les principaux terminaux du marché et minimiser les efforts. Nous avons également défini un modèle implémentationnel d'architecture vocale fondé sur les mêmes principes fondamentaux que notre modèle d'architecture web. Notre modèle d'architecture vocale utilise les fonctionnalités téléphonique des terminaux. Ceci nous permet de cibler l'ensemble des terminaux téléphoniques, qu'ils soient fixes ou mobiles, commutés ou IP. Nous avons défini notre modèle d'architecture multimodale distribuée en combinant les deux modèles précédemment évoqués. Notre modèle d'architecture multimodale permet de définir des outils dont les interfaces exploitent à la fois les capacités d'interaction web et les fonctionnalités des téléphones. Nous avons identifié le composant clef permettant cette combinaison (CGI du contrôleur de dialogue de la figure 2.14 page 63, CGI du serveur vocal de la plate-forme mVIP figure 3.1).

L'exploitation de notre modèle implémentationnel UbiArch passe par son implémentation préalable. Le fort ancrage opérationnel de notre problématique de recherche nous a poussé à prendre en compte les protocoles et les formalismes décrits dans le premier chapitre (sections 1.3 à 1.5) lors de l'implémentation de notre modèle. Cette prise en compte nous permet dans premier temps de réutiliser les développements existants, qu'il s'agisse des services vocaux, des interfaces web ou des plates-formes du groupe France Telecom. Ensuite, l'utilisation de standards nous permet de minimiser les coûts de déploiement liés au passage à l'échelle des solutions proposées. En effet, les équipements industriels permettant le déploiement de notre architecture implémentent les standards du web et de la VoIP. Ainsi, le coût lié à l'exploitation des licences des piles industrielles [31] dès les étapes de prototypage se voit compensé par les économies réalisées lors de la migration des solutions proposées vers le réseau de production. Enfin, l'inscription de notre démarche au sein de différents consortiums [25, 41] nous permet d'anticiper les évolutions des standards, de participer à un effort de normalisation [Chuffart *et al.*, 2006] et de continuer d'être un acteur majeur du monde des services de télécommunications.

De plus notre démarche nous place clairement dans le domaine de l'innovation. Les solutions

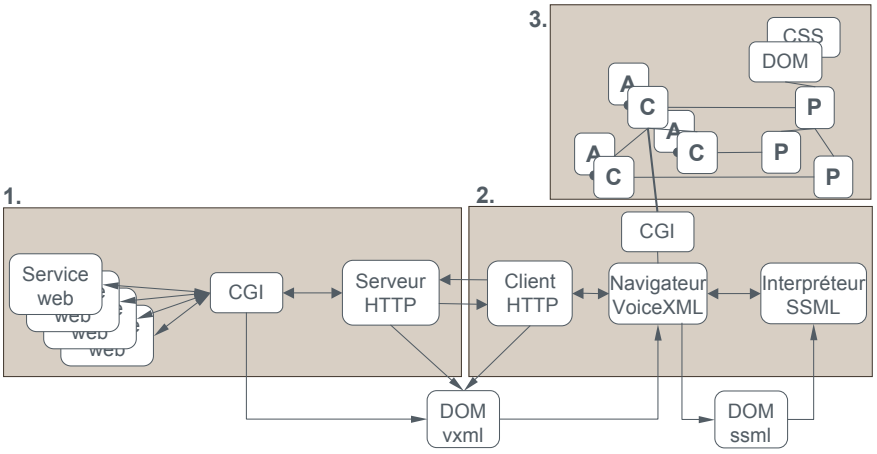


FIG. 3.1 – Architecture logicielles mVIP. Le modèle Ubiarch de la figure 2.14 se voit décomposé en trois composantes logicielles : la composante 1 est un serveur web classique en charge d'agréger les différents services web nécessaires à la construction du service, la composante 2 est le serveur multimodal en charge de l'intégration des commandes utilisateurs et de la mise à jour des interfaces distribuées, la composante 3 est le client web qui supporte le modèle de l'outil.

proposées permettent l'accès à un environnement communicant à partir du mobile dont dispose déjà une grande partie de la population ; elles permettent d'augmenter les capacités d'interaction des interfaces existantes (téléphone ou web) et d'offrir de nouveaux moyens de consommation de biens ou de services. Nous identifions des scénarios et des cas d'usage dans lesquels chacun peut se retrouver et y trouver un intérêt, qu'il s'agisse de l'utilisateur, du fournisseur de service ou de l'exploitant de l'infrastructure de communication. Finalement, le mécanisme de snippet décrit à la section 4.3 page 98 nous assure une migration rapide des interfaces web existantes vers notre solution ubiquitaire. Ce mécanisme laisse présumer, par là même, une adoption massive de la part de la communauté des concepteurs de services web.

Afin de mettre en œuvre ce modèle, ce chapitre spécifie l'implémentation de ce composant et son intégration dans les infrastructures existantes. Dans la section 3.2, nous décrivons comment nous avons fait évoluer le patron de conception PMX (section 1.6) afin de rendre compte de notre modèle d'architecture et comment nous avons intégré ce patron dans la plate-forme VIP (section 1.6) pour définir la plate-forme mVIP[95] et prétendre un déploiement de notre technologie sur le réseau de France Telecom. La plate-forme mVIP offre aux concepteurs d'interfaces web un jeu de services personnalisables destinés à l'intégration du mode vocal dans les pages web selon le modèle UbiArch.

La plate-forme mVIP reprend le principe des entrées/sorties de flux modal de la plate-forme PMX (figure 3.2(a)). Cependant sa conception étend ce concept à d'autres modes tels que la communication par proximité, la gestuelle, l'échange de texte court et plus généralement selon une logique événementielle. Dans le cadre d'une contribution autour de la visiophonie, nous avons eu l'occasion de prendre en charge la gestion la gestion de flux vidéos. La plate-forme mVIP nous a permis de rendre compte d'un service de messagerie vidéo. De cette approche,

[95]mVIP - VIP multimodale.

nous avons appris que la gestion de plusieurs flux média en entrée ou en sortie nécessite une prise en compte fine de la synchronisation des canaux et de la compression du flux. En effet, à l'heure actuelle les processus de conversion et le transport de flux vidéo implique une montée en charge de l'infrastructure de communication. La gestion des événements est rendue possible par l'intégration d'un frontal web (figure 3.2(b)). Le frontal web accède aux fonctionnalités de la plate-forme pouvant tour à tour modifier la progression de l'utilisateur dans le service mVIP ou être notifié de cette progression (figure 3.4).

(a) Architecure PMX (b) Architecture mVIP

FIG. 3.2 – Evolution de l'architecture PMX vers mVIP, permettant l'intégration du mode vocal dans les interfaces web.

Nous apportons un soin particulier à concevoir ce composant de manière à minimiser le couplage entre les objets, favorisant ainsi la réutilisation des classes dans le but de prévoir les évolutions futures de l'architecture et d'anticiper l'intégration des modalités des nouveaux terminaux communicants (section 3.2).

Ainsi, nous spécifions un client de reconnaissance étendu en charge de la fusion des informations provenant de différentes sources et de la génération des événements qui viendront alimenter la logique de dialogue du service mVIP. Ce client implémente le protocole MRCP (section 1.5) ce qui nous permet d'utiliser de multiples ressources réparties dans le réseau. Le comportement de ce composant suit une logique déclarative définie par une grammaire SRGS (section 1.4). Le caractère déclaratif de cette logique permet de réifier la logique applicative du composant et donc de la réutiliser dans d'autres services, de la dupliquer afin de l'adapter pour d'autres outils.

De la même manière, nous spécifions un client de synthèse étendu en charge de contrôler les différents effecteurs se trouvant dans l'environnement de l'utilisateur. Lui aussi implémente le protocole MRCP. Sa logique est elle aussi déclarative mais repose sur le langage SSML. Les scripts SSML destinés initialement à la description de prompts audio tendent à intégrer d'autres modes tels que la vidéo (section 1.4). Nous utilisons la balise `meta` afin de traiter l'élément racine de la hiérarchie d'agents en charge du contrôle de la présentation.

Afin de faciliter le développement rapide de maquettes ou de prototypes, nous avons prédéfini un ensemble d'outils mVIP exploitant les fonctionnalités de la plate-forme. Ces outils sont personnalisables et extensibles. Ils reposent sur les technologies web modernes et couramment employées par les concepteurs d'applications web. Les outils mVIP sont accessibles via HTTP. Il assurent la mise en relation d'un terminal téléphonique et d'un service mVIP et le contrôle de la session depuis le client web. La section 3.5 décrit les outils mVIP pré-définis et présente les principales fonctionnalités offertes par ces outils mVIP.

La réutilisation des outils existants assure le développement rapide de services ubiquitaires par des personnes familières des techniques de développement web ne développant pas couramment des services vocaux. De plus, les outils mVIP fournissent une approche graphique de VoiceXML. Les outils mVIP permettent de superviser, de tester les services vocaux qui

sont joués. La figure 3.3 organise les différentes composantes des outils mVIP. Les outils mVIP suivent le modèle UbiArch. Ils sont composés d'un scénario de dialogue décrit par le langage VoiceXML et d'un interacteur instancié sur le client web et en charge de manipuler le DOM HTML. L'interacteur et le scénario interagissent par l'intermédiaire des services web mVIP.

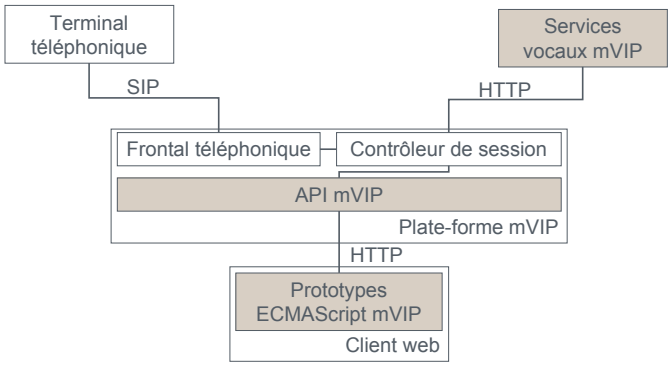


FIG. 3.3 – les outils mVIP.

Le scénario VoiceXML est un canevas manipulant les structures de contrôle du dialogue de VoiceXML. Le scénario VoiceXML est assimilable à un automate et représentable par un graphe [Ould Ahmed Limam, 2003]. Ceci assure une représentation commune aux différents acteurs impliqués dans la conception d'une interface. Le scénario permet par exemple de définir l'outil répondeur en charge de l'enregistrement et de la lecture de messages audio (figure 3.4). Le scénario constitue le noyau fonctionnel de l'outils mVIP. Son interprétation est l'élément racine de la hiérarchie en charge du contrôle du dialogue.

Les interacteurs mVIP sont des prototypes ECMA instanciés sur le client web. Ils sont en charge de la communication entre la logique de dialogue le DOM HTML instancié. Ils assurent le relais des notifications de changement d'état de l'outil (figure 3.4). Ces notifications permettent un retour sémantique relatif à la progression de l'utilisateur dans le scénario. L'interacteur mVIP utilise la pile HTTP du navigateur web pour communiquer avec les services web de la plate-forme mVIP. Dans la section 3.5.2 nous spécifions ce composant et décrivons son fonctionnement.

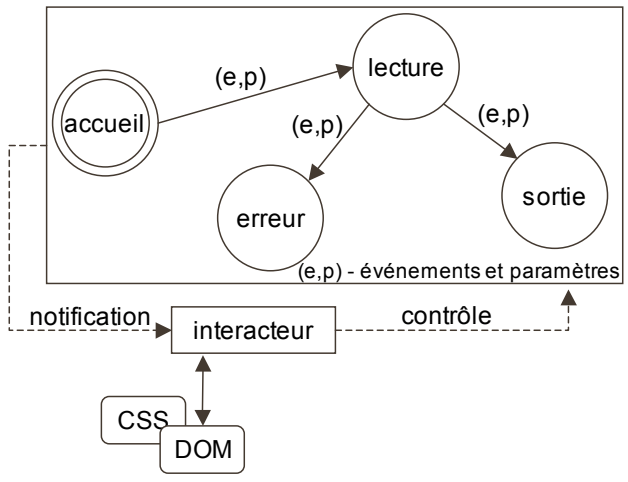


FIG. 3.4 – Scénario et interacteur mVIP. Le scénario mVIP est un automate d'états. L'état est une séquence VoiceXML. Le changement d'état est provoqué par le traitement des événements capturés par les périphériques de l'utilisateur. Le changement d'état provoque la notification des interacteurs mVIP et par conséquent assure un retour sémantique global sur l'ensemble de l'interface répartie. À l'inverse, les interacteurs contrôlent l'état de l'outil et donc la progression de l'utilisateur dans le scénario mVIP.

3.1 Fonctionnement de la plate-forme

L'initialisation de la plate-forme mVIP (démon UNIX) provoque l'initialisation de n contrôleurs de session. Historiquement n est égal à 30, ce qui correspond au nombre de lignes disponibles sur une T2[96]. Chaque contrôleur de session est en attente d'un appel au service web *Subscribe()*. L'appel au service web provoque la mise en relation d'un terminal téléphonique et d'un outil mVIP. Le numéro de téléphone ou l'URI SIP ainsi que l'URI du script VoiceXML sont passés en paramètres de l'invocation du service web. Ainsi, le contrôleur de session instancie la ressource de téléphonie adéquate (SIP ou RTC) et le DOM VoiceXML demandé. Le jeu du service VoiceXML provoque l'instanciation des ressources (synthèse, reconnaissance) nécessaires à la progression du script. L'OpenVXI[97] envoie des commandes à ses ressources et se met en attente d'événements extérieurs.

Gestion des événements en entrée Les *événements* sont des interruptions du système qui vont nourrir l'OpenVXI et entraîner la logique de dialogue VoiceXML. Les événements sont, par exemple, des interruptions vocales ou des « non événements » (dépassement de délai, échec de la reconnaissance vocale), des changements d'état de la synthèse vocale (fin de lecture), des événements téléphoniques (décroché, raccroché) ou des interactions web transmises par le service web *control()*. Ainsi, un signal audio est capturé par le micro du téléphone de l'utilisateur, transmis sous forme de flux RTP, analysé par la ressource de reconnaissance selon le modèle de reconnaissance fourni par l'OpenVXI et donne lieu à une interruption de l'automate OpenVXI dans sa logique de dialogue. Il en va de même pour la lecture de la fin d'une séquence, l'interruption par un événement web ou la déconnexion du terminal de l'utilisateur.

La nature des événements utilisateurs attendus par l'OpenVXI est décrite sous forme de *grammaires*. La grammaire possède un *mode*, la grammaire est associée à la ressource de reconnaissance correspondant à ce mode. Dans le cas d'un accès exclusif au système selon une modalité, c'est uniquement le client de reconnaissance associé à cette modalité qui est activé. De fait, les événements ne correspondant pas à la modalité attendue ne sont pas capturés et ne viennent pas modifier la logique de service de l'automate OpenVXI. Dans le cadre de l'utilisation de plusieurs ressources de reconnaissance simultanément (« Dîtes, tapez ou clickez sur 1 pour accéder à la météo. »), la détection d'un élément terminal d'une des grammaires provoque la notification de l'automate VoiceXML et l'inhibition des clients de reconnaissance. Dans le cas d'un accès au système par des modalités complémentaires (« put that there », grammaire de mode *mixte*) les différents clients spécifiques chargent les grammaires dédiées pour chacun des modes. Il transfèrent les événements unimodaux marqués temporellement à la ressource de reconnaissance multimodale en charge de l'intégration des différents événements selon leur temporalité.

Dans le cas d'une grammaire de mode web, le client web est notifié de la grammaire à charger. Le client est en charge de traduire cette grammaire en formulaire dans le cas d'une grammaire SRGS mais les événements attendus peuvent être directement décrits sous forme de formulaire. Un formulaire spécifie la structure des événements attendus. La validation de ce formulaire provoque la notification du système par le client web. Ainsi le client web transmet à l'automate OpenVXI les événements décrits par le formulaire. Notons que l'utilisation des valeurs *web* et

[96] T2 - autrement appelée PRI (Primary Rate Interface), une ligne T2 est une interface d'accès à un réseau ISDN.

[97] OpenVXI - implémentation opensource d'un interpréte VoiceXML]

mixte pour l'attribut *mode* du tag *grammar* de SRSG n'est pas pris en compte par la DTD actuelle de VoiceXML. Nous avons donc créé deux branches pour notre interpréteur VoiceXML. L'une étend la DTD de VoiceXML afin de prendre en compte ces tags. L'autre reste VoiceXML pure. La version VoiceXML pure ne perd pas pour autant ses capacités web. En effet, avec cette version, nous notifions le client web des chargements de grammaire *voice* et *DTMF*, le client web se charge du traitement de la grammaire SRGS.

Gestion des événements en sortie Le contrôle des ressources reprend la logique de service du VoiceXML. L'OpenVXI fait diffuser du contenu structuré, fait charger des grammaires et attend des interruptions pour rendre compte de sa logique de service. Dans le cadre de l'utilisation de plusieurs ressources de synthèse (vocale, avatar 3D ou notification web) le contrôleur de session est en charge d'affecter la diffusion du contenu structuré à la ressource de synthèse correspondante. Cette affectation dépend du choix de la langue utilisée, du mode utilisé, des préférences de l'utilisateur, des autorisations liées à l'abonnement au service mVIP. Dans le cas d'une notification web, nous avons utilisé la balise *meta* de SSML, ce qui nous permet de conserver inchangée la DTD de VoiceXML nous assurant une meilleur intégration dans les plates-formes VoiceXML existantes. La plate-forme mVIP est en mesure de garantir la séquentialité des événements en sortie. Ainsi, la gestion du *bargein*[98], le contrôle du flux audio et des événements web permet de déploiement de stratégies de dialogues alternant des retours sémantiques audios, des énoncés graphiques et des relances vocales [Horchani *et al.*, 2007]. C'est le déroulement linéaire du scénario VoiceXML qui assure cette séquentialité.

3.2 Architecture de la plate-forme mVIP

mVIP est la plate-forme ubiquitaire telle que nous la concevons dans ce mémoire. Elle caractérise les évolutions du patron PMX dérivé pour la multimodalité et intégré dans l'architecture VIP. La plate-forme *multimodale VIP* (mVIP) met en œuvre des services ubiquitaires sur les différents terminaux dont dispose l'utilisateur (téléphones, ordinateurs, objets communicants). mVIP fédère les ressources présentes dans l'environnement de l'utilisateur (terminaux) et sur le réseau (services). Les services deviennent accessibles de manière adaptée aux cas d'utilisation (domotique, mobilité, accessibilité).

La plate-forme mVIP est un *démon UNIX*[99] pouvant être exécuté par exemple sur une machine distante connectée à internet ou sur un *routeur modem ADSL Wifi* dans un réseau privé. Dès lors, mVIP dispose des ressources (SIP, MRCP, RTSP) identifiées et accessibles sur le réseau. mVIP peut également être contrôlée à distance grâce à son API et un jeu de services web[100].

Patron PMX La plate-forme mVIP intégre le patron PMX (section 1.6) adapté pour la multimodalité. Le terme patron (pattern[101]) est utilisé à bon escient dans le mesure ou le patron PMX représente le nœud du réseau ubiquitaire en charge de la manipulation du flux média et par conséquent de la configuration du maillage de flux média à travers le réseau. Le patron PMX (figure 3.5) organise le maillage des flux média selon une logique de dialogue et d'une

[98] Bargein - interruption de l'énoncé vocal du système par l'utilisateur.)

[99] Démon UNIX - programme résident, UNIX daemon.

[100] Web Service - le service web est un programme informatique permettant la communication et l'échange de données en utilisant le protocole de transport HTTP est un protocole de description adapté (JSON, XML).

[101] Pattern - concept destiné à résoudre les problèmes récurrents suivant le paradigme objet

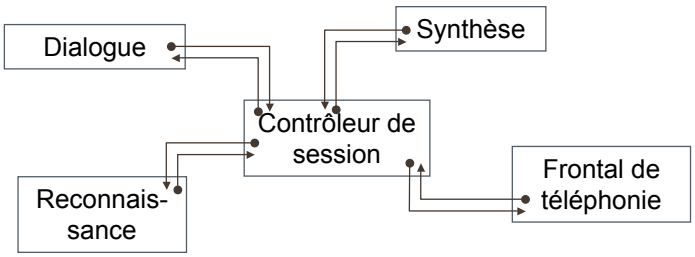
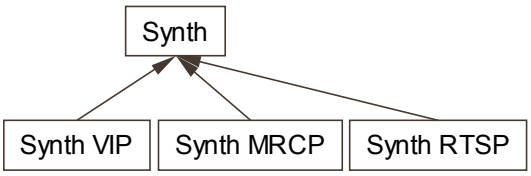
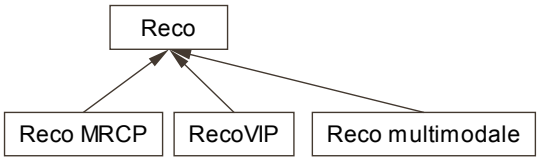


FIG. 3.5 – Patron PMX.

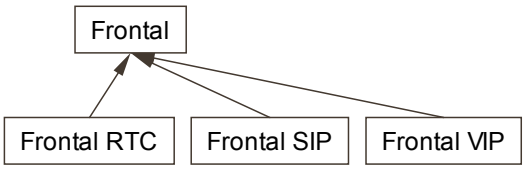
pile d'événements. Nous avons dérivé le patron PMX afin de l'intégrer dans l'architecture VIP. Dans la section 1.6 nous voyions que VIP était un protocole propriétaire destiné au pilotage de ressources de synthèse, de reconnaissance et de téléphonie à travers le réseau. Nous avons donc dérivé les ressources abstraites de PMX pour implémenter le protocole VIP comme nous l'avions déjà fait pour le MRCP (figure 3.6). Notons que l'organe de dialogue que nous utilisons (OpenVXI) dérive de l'organe de dialogue abstrait. Notre choix c'est porté sur VoiceXML par rapport à son statut au sein du W3C.



(a) Dérivation de l'organe de synthèse abstrait.



(b) Dérivation de l'organe de reconnaissance abstrait.



(c) Dérivation de l'organe abstrait de téléphonie.

FIG. 3.6 – Dérivation des organes PMX.

Ainsi, le cœur de la plate-forme mVIP (figure 3.7) dialogue en MRCP ou en VIP avec les ressources de synthèse et de reconnaissance vocale du laboratoire (Synthèse vocale *Synthserver*, reconnaissance vocale *Philsoft*) et ceux du marché (Elan Sayso, Nuance, Telispeech de Telisma). Il est accessible depuis le réseau RTC par l'intermédiaire des cartes *NMS Communicator* et depuis les réseaux VoIP grâce aux piles SIP et H323 RadVision. Il dispose d'un scénario VoiceXML accessibles via HTTP sur le réseau IP.

Nous avons défini les ressources de la plate-forme par leur classe abstraite, dans le but de réduire le couplage entre les objets et d'augmenter la réutilisabilité des objets que nous définissons. Nous définissons une ressource de *reconnaissance étendue* pour désigner une entité qui modifie son état en fonction d'un flux média entrant (figure 3.8(a)). Nous définissons une ressource de *synthèse étendue* pour désigner une entité qui produit un flux média en modifiant son

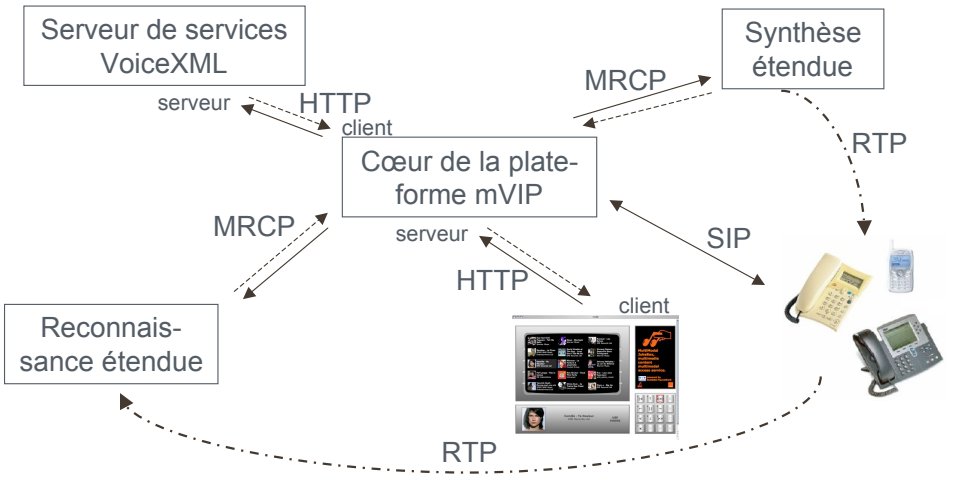
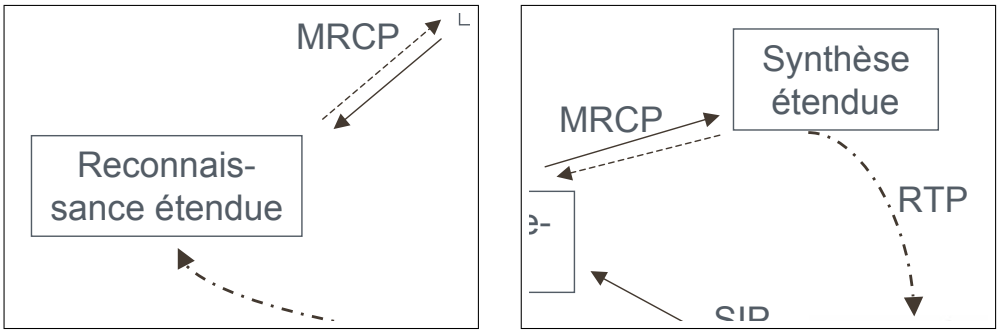


FIG. 3.7 – Macro architecture mVIP.

état (figure 3.8(b)). Ces deux composantes assurent la construction d'un maillage de flux RTP. La figure 3.8 isole ces deux composants. Dans la section suivante nous réalisons une description fine des composants du cœur de la plate forme mVIP.



(a) Ressource de reconnaissance étendue (b) Ressource de synthèse étendue

FIG. 3.8 – Gestion des flux média entrants et sortants.

3.3 Cœur de la plate-forme mVIP

Cette section décrit le cœur de la plate-forme mVIP (figure 3.9) ainsi que la façon dont il gère les flux média à travers le réseau. Le cœur de la plate-forme mVIP est constitué :

- d'un client de synthèse étendue permettant la synthèse vocale mais aussi la lecture de fichiers audio, vidéos ;
- d'un client de reconnaissance étendue permettant la reconnaissance vocale mais aussi enregistrement de flux audio, vidéos, système d'identification du locuteur ;
- d'un navigateur VoiceXML assurant la logique des services VoiceXML ;
- d'un frontal de téléphonie mettant en relation un terminal et la plate-forme ;
- d'un contrôleur de session en charge d'orchestrer l'ensemble des organes mVIP.

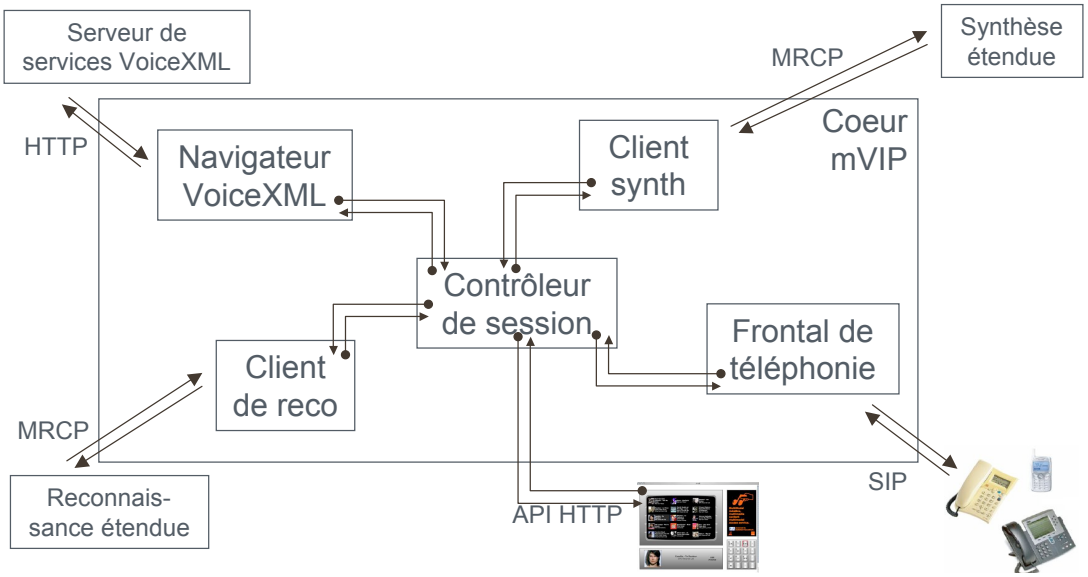


FIG. 3.9 – Cœur de la plate-forme mVIP.

Automate OpenVXI L'*OpenVXI* est un automate d'état qui se nourrit du *DOM VoiceXML*, lequel est récupéré sur le serveur d'application via un protocole internet (http, https, file,). L'automate d'état envoie des commandes aux ressources de téléphonie, synthèse et de reconnaissance étendue, puis se met en attente de la réception d'un événement d'une de ces dernières. Si rien ne se passe durant un laps de temps défini, un événement *timeout* est émis dans la pile d'événements. L'automate d'état se charge alors de réagir en conséquence.

Frontal de téléphonie Le frontal téléphonique désigne une abstraction des frontaux de téléphonie analogique ou IP. Par abus de notation, nous désignerons cette abstraction par « frontal SIP ». Le frontal de téléphonie est en charge l'établissement de l'appel entre le terminal de l'utilisateur et le service mVIP. C'est ce composant qui initie la session téléphonique, négocie l'établissement des flux média entre le terminal de l'utilisateur et les ressources de synthèse et de reconnaissance étendues. C'est également lui qui assure le maintien, la duplication et la redéfinition de ces flux dans le cas d'un transfert d'appel.

Contrôleur de session Les objets de la plate-forme dialoguent par l'intermédiaire du *contrôleur de session*. En terme de patrons [Gamma *et al.*, 1995], le contrôleur de session est un *médiateur*. Le médiateur assure la cohérence tout en minimisant le couplage entre plusieurs objets : les *collaborateurs*. Dans notre cas, le contrôleur de session assure la cohérence entre l'OpenVXI, les clients de synthèse et de reconnaissance étendues et le frontal de téléphonie. Le contrôleur de session accède à l'instance de son collaborateur par son interface abstraite. L'OpenVXI contrôle les autres collaborateurs par le contrôleur de session. Les ressources de téléphonie, de synthèse et de reconnaissances notifient le contrôleur de session de leurs changements d'état. Enfin, le contrôleur de session met à disposition les fonctionnalités de la plate-forme par l'intermédiaire de ces services web.

Gestion des flux RTP entrants En terme de patrons [Gamma *et al.*, 1995], les clients de synthèse et de reconnaissance étendues sont des *remote proxy*. Le remote proxy fournit localement

l'interface d'un objet distant. Ces deux clients nous fourniront donc les méthodes MRCP ou VIP des ressources distantes.

Afin de gérer plusieurs modes en entrée, nous devons instancier plusieurs clients de reconnaissance. Par exemple, la navigation dans une hiérarchie peut s'effectuer de manière vocale ou en utilisant les touches du téléphone. Nous implémentons donc notre client de reconnaissance étendue sous forme d'un tableau de reconnaissances, indexé par le mode supporté par la reconnaissance (Voice, DTMF, gesture). Ceci nous permet d'envoyer la grammaire de reconnaissance au bon client. La définition de deux grammaires au niveau du service provoque l'instanciation de deux clients de reconnaissances chargés de nourrir l'OpenVXI. Chacun des clients de reconnaissance étendue est connecté au module de reconnaissance adéquat. Le frontal de téléphonie est en charge de la négociation des flux RTP et de leur redirection vers les ressources de reconnaissance adéquates. C'est ce mécanisme qui assure la construction du mesh RTP pour les flux entrants sur le système mVIP.

À l'heure actuelle, dans la version déployée, la gestion des événements multimodaux s'effectue de manière exclusive. Cela signifie qu'un client de reconnaissance recevant une réponse signifiante par rapport à la grammaire chargée inhibera les autres clients de de reconnaissance précédemment solicités. Toutefois, dans la section 3.4 nous spécifions le fonctionnement d'un client de reconnaissance gérant des modes complémentaires.

Gestion des flux RTP sortants De la même manière, l'instanciation de plusieurs clients de synthèse permet la gestion de plusieurs modalités en sortie du système. Par exemple, dans le but d'assurer l'accessibilité au plus grand nombre, le rendu d'un service est effectué à la fois sur le canal audio et sur le canal graphique du système.

Pour réaliser cela, le client abstrait de synthèse étendue est implémenté sous forme d'un tableau de synthèses indexés selon leur modalité. Chacune de ces synthèses est en charge du contrôle d'une ressource de synthèse spécifique. Ces ressources de synthèse prennent la forme de synthèse vocale, lecture de fichiers audio, production de contenu textuel structuré ou encore contrôle d'un avatar 3D. Le frontal de téléphonie est en charge de la négociation des flux RTP sortants, leur redirection et leur duplication. Ce mécanisme assure la construction du mesh RTP pour les flux sortants du système mVIP Chacun des clients de synthèse étendue est connecté au module de synthèse adéquat. C'est également le client de synthèse qui alimente le service web des notifications de changement d'état de l'automate VoiceXML, Dans la section 3.5.3 nous explicitons comment le concepteur de services mVIP assure la notification du client web d'un changement d'état de l'outil mVIP.

3.4 Client de reconnaissance multimodale

Dans son article de référence « Ten Myths of Multimodal Interaction », Sharon Oviatt précise que bien que la complémentarité des modes soit un des enjeux les plus séduisants de l'interaction multimodale ce n'est pas le plus utilisé. Néanmoins, il existe des applications de cette complémentarité et nous devons aborder cet aspect. Ainsi l'attente d'un énoncé multimodal de la part de l'utilisateur se traduit par l'instanciation d'un client de reconnaissance multimodale par l'automate OpenVXI. Ce client est en charge du contrôle des clients de reconnaissance modale, il transmet à un client de reconnaissance dédié, la description des événements attendus. Les clients se mettent donc en attente de la reconnaissance d'événements modaux. La reconnaissance d'un événement modal provoque la notification du client de reconnaissance multimodale. Le client de reconnaissance multimodal empile ainsi les événements modaux et cherche à reconnaître un

terminal de la grammaire multimodale. Le client de reconnaissance multimodale hérite du client de reconnaissance abstrait. Par conséquent, il possède une interface le rendant accessible par le contrôleur de session (et donc par l'OpenVXI). Le client de reconnaissance multimodal instancie dynamiquement des clients de reconnaissance dédiés en fonction des grammaires transmises par l'OpenVXI. Le client de reconnaissance multimodale réalise la synthèse des événements tels que les détecte chacun des clients de reconnaissance modale, selon la combinaison logique définie par les propriétés CARE. Les propriétés *CARE*[102] adressent à la fois le problème de la complémentarité des modes et celui de la coopération entre agents. Elles combinent de façon logique les événements modaux provenant de différents dispositifs agents en charge d'analyser l'activité de l'utilisateur.

Le VoiceXML ne prévoit pas la définition de grammaires multimodales. Pour cette raison, nous avons étendu la DTD de VoiceXML en définissant des grammaires de mode *mixte*. La grammaire mixte intègre des grammaires de différents modes. Le chargement de cette grammaire par l'OpenVXI instancie un client de reconnaissance multimodale. Le client charge la grammaire multimodale, détecte les modes attendus par le système (voice et DTMF), instancie les clients de reconnaissance voice et DTMF en conséquence et leur transmet respectivement les grammaires *action* et *item*. Chacun des clients dédiés notifie le client de reconnaissance multimodal des résultats de son analyse.

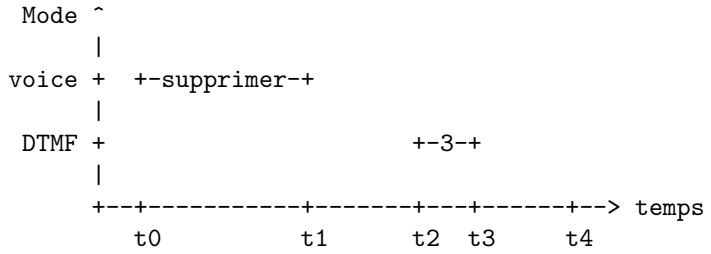


FIG. 3.10 – Exemple de prise en compte d'un énoncé multimodal

[102]CARE - Complementarity, Assignment, Redundancy, Equivalence [Coutaz *et al.*, 1995, Nigay et Coutaz, 1997].

Traitement des grammaires multimodales La figure 3.10 décrit la prise en compte d'un énoncé multimodal par le système. À la date *t1* (figure 3.10) le client de reconnaissance vocale notifie le client de reconnaissance multimodale de la reconnaissance de l'action *supprimer*, à la date *t3* le client de reconnaissance dtmf notifie le client de reconnaissance multimodale de la reconnaissance de l'item *3*. À la date *t4*, le client de reconnaissance multimodal détecte la fin l'analyse des événements utilisateurs et notifie l'automate OpenVXI de la détection de l'énoncé « supprimer l'item 3 ». *t0* et *t1* représentent respectivement le début et la fin de reconnaissance d'un énoncé vocal. *t2* et *t3* représentent respectivement le début et la fin de reconnaissance d'une saisie DTMF. *t4* est la date à la laquelle l'énoncé multimodal va être transmis à l'automate OpenVXI. Nous définissons le délai exprimé par la différence *t3* - *t1* comme étant l'*interEventTime*. Il exprime de temps d'attente entre deux événements appartenant au même énoncé. L'*interEventTime* est maximisé par la valeur de la variable *interEventTimeout*. De la même manière, nous définissons la différence *t3* - *t4* comme étant le *termTime*. Il exprime le temps d'attente entre la fin de l'énoncé et la vérification de l'appartenance de l'énoncé à l'ensemble des terminaux de la grammaire multimodale. Le *termTime* est maximisé par la valeur de la variable *termTimeout*.

```
<grammar version="1.0" mode="mixte" root="actionOnItem">
  <rule id="actionOnItem" scope="public">
    <item>
      <item repeat="1"><ruleref uri="action.srgs"/></item>
      <item repeat="1"><ruleref uri="item.srgs"/></item>
    </item>
  </rule>
</grammar>

<grammar version="1.0" mode="voice" root="action">
  <rule id="action" scope="private">
    <one-of> <item>supprimer </item> <item>archiver </item> </one-of>
  </rule>
</grammar>

<grammar version="1.0" mode="dtmf" root="item">
  <rule id="action" scope="private">
    <one-of> <item>1 </item> <item>2 </item> <item>3 </item> </one-of>
  </rule>
</grammar>
```

Fig. 3.11 – Exemple de grammaire multimodale

Notons que nous spécifions ici le fonctionnement d'une grammaire multimodale mais qu'elle n'a pas donné lieu à une implémentation, à une mise en œuvre industrielle. Notre démarche, certes de recherche, est somme toute industrielle et est guidée par les attentes de nos financeurs et, bien que proche de la problématique de l'utilisation de grammaires SRGS évoluées (figure 1.21 page 32) implémentés pour le service « Qui donc ? », cet aspect ne s'est pour l'instant pas révélé déterminant pour nos financeurs.

3.5 Boîte à outils mVIP

Le framework mVIP[103], met à la disposition du concepteur de services un ensemble d'outils permettant l'exploitation des ressources de la plate-forme depuis une interface web. Ces services définissent des outils composés de triplets : un scénario VoiceXML, un service web de mise en relation et de contrôle, un interacteur web exploitant les services web mVIP. Les deux services web que nous avons implémentés sur le contrôleur de session sont les services web *Subscribe()* et *Control*. Le premier permet d'initialiser une session mVIP (donc un outil) et de tenir informé le client des changements d'état de l'outil mVIP. Le second permet de modifier l'état de l'outil mVIP. Dans le section 3.5.1 nous décrivons finement les mécanismes mis en jeu par ces services web. Les interacteurs mVIP sont des objets javascript en charge de l'exploitation des services web mVIP, ils possèdent un ensemble de méthodes permettant le dialogue entre le client web et les service web mVIP. Pour ce faire, ils étendent l'objet XmlHttpRequest et ses méthodes pour faire communiquer le DOM instancié sur le client et la plate-forme mVIP. Ils mettent également à la disposition du concepteur de services web une API leur permettant de communiquer avec les autres éléments du DOM. Enfin le service VoiceXML décrit un scénario de dialogue type exploitant les structures de contrôle du langage VoiceXML. L'interacteur et le scénario sont destinés à être étendus par le concepteur d'interface web dans le but de les adapter à sa problématique. .

3.5.1 Les services web mVIP

Subscribe Le service web *subscribe* (souscrire) reprend le mécanisme de notification décrit dans le RFC 3265 [RFC 3265, 2002]. Nous adaptons ce mécanisme destiné à une architecture point à point, à l'architecture client serveur du protocole HTTP. Le web service subscribe permet de souscrire à un service mVIP. La souscription donne lieu à la mise en relation entre un terminal et un service mVIP. Son invocation initie une session sur la plate-forme mVIP, entre un terminal téléphonique et un service mVIP, et retourne une main courante sur la session mVIP[104]. L'entête de la requête contient la méthode invoquée, les URI du service et du terminal à mettre en relation et une clef permettant vérifier les autorisations relatifs à l'usage du service. Le corps de la requête contient les variables permettant d'initialiser le service mVIP. La réponse contient l'identifiant de la session. Les requêtes et les réponses sont de la forme suivante :

```
POST http://mVIP/ws/mVIP.cgi
Xmvip-method: SUBSCRIBE
Xmvip-service: record.vxml
Xmvip-terminal: sip:user@domain.net
[ Xmvip-key: MD5  ]
XmVIP-Id: MD5
Xmvip-expires: date
CRLF
[ message-body ]
```

```
HTTP/1.0 200 OK
XmVIP-Id: MD5
Xmvip-expires: date
```

FIG. 3.12 – Datagramme de l'appel au service web *subscribe*.

L'identifiant de session permet d'initialiser un flux http véhiculant les notifications du contrôleur de session. Ce flux correspond à la réponse à une requête HTTP GET sur l'API mVIP.

[103] Framework - espace de travail modulaire, ensemble de bibliothèques, d'outils et de conventions permettant le développement rapide d'applications.

[104] Handle de session - valeur numérique identifiant un objet informatique et permettant donc sa « manipulation » ou sa gestion [38].

L'acquittement est composé d'une entête contenant un code de retour, la date d'expiration de la souscription. Le corps de la réponse est un flux envoyé progressivement au client. Ce flux est composé de notifications (NOTIFY) sur l'état du contrôleur de session de la plate-forme et se terminant par un message de fin (*END*). La date d'expiration de la requête *subscribe* suggère la durée de l'abonnement. La date d'expiration de la réponse confirme la durée réelle de l'abonnement. La date d'expiration du flux HTTP correspond à la durée de vie du flux. Le renouvellement de l'abonnement, l'initialisation et le maintien du flux HTTP sont à la charge du client.

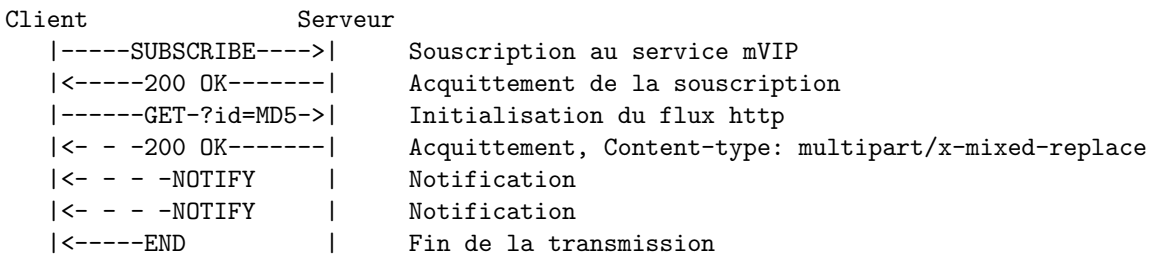


FIG. 3.13 – Diagramme de séquence de l'appel au service web *subscribe*.

Control Le web service *control* (contrôler) reprend le mécanisme de l'agent *Control* du modèle PAC[Coutaz, 1987]. Le web service control permet de contrôler une session mVIP par son API. La session mVIP est initialisée par le service web subscribe, au niveau du médiateur. Le service web *control* modifie le DOM du service VoiceXML, pouvant tour à tour modifier les paramètres des ressources de synthèse et de reconnaissance, la composition des messages d'erreur, et la logique VoiceXML de l'automate. Le service web *control* envoie des événements sur la pile de l'automate OpenVXI.

Pour réaliser cette notification, le web service *control* adresse une ressource de reconnaissance virtuelle matérialisée par un client de reconnaissance virtuel. Le web service *control* invoque les méthodes du client de reconnaissance virtuel. Le client de reconnaissance est virtuel en ce sens qu'il n'adresse pas de serveur de reconnaissance. L'entête de la requête contient la méthode invoquée, l'identifiant de la session mVIP initialisée par le service web *subscribe*.
L'entête de la requête (figure 3.14) contient les paramètres du service web. Le corps de la requête contient, si besoin, les variables qui caractérisent l'événement.

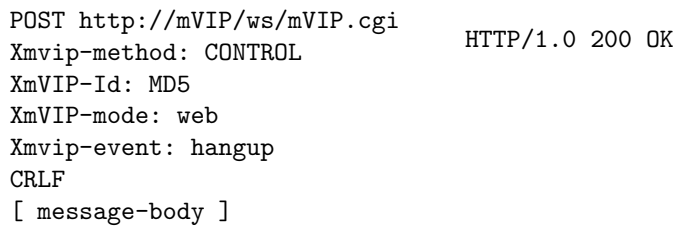


FIG. 3.14 – Datagramme de l'appel au service web *control*.

3.5.2 Les interacteurs mVIP

Les interacteurs mVIP sont des objets ECMA destinés à être instanciés sur le client web. Il sont en charge de la communication entre la plate-forme mVIP et le DOM HTML. Pour réaliser cela, ils étendent l'objet XmlHttpRequest, ce qui permet au DOM de dialoguer avec la session mVIP par le biais du protocole HTTP. Les interacteurs sont destinés à être adaptés par le concepteur d'interfaces web, dans le but de faire communiquer les composantes de son interface avec la logique de dialogue instanciée sur la plate-forme mVIP.

Subscriber Le prototype *Subscriber* (souscrivant à une session web mVIP) permet de simuler le comportement point à point entre un client et un serveur HTTP. Le *Subscriber* étend l'objet XmlHttpRequest cher aux utilisateurs de la technologie AJAX. Nous avons vu que cet objet intègre un pile HTTP permettant de créer des requêtes HTTP depuis une instance du DOM HTML. L'objet XmlHttpRequest génère la requête et traite la réponse HTTP selon cinq états :

0. création ;
1. initialisation ;
2. envoi de la requête ;
3. réception en cours ;
4. fin de la transmission.

À chacun de ces états, le prototype *Subscriber* associe une fonction de rappel (callback). Le comportement point à point attendu est obtenu par traitement de la réponse HTTP dans l'état 3. Le *Subscriber* possède les méthodes :

- subscribe() ;
- connect() ;
- unsubscribe().

La méthode *subscribe()* invoque le service web *subscribe* avec les paramètres *key*, *service*, *terminal*. En cas de succès, le terminal et le service sont en relation. Le *Subscriber* initialise ses attributs *id*, *expires* et appelle sa méthode *connect()*. En cas d'échec le *Subscriber* affecte la valeur 0 à son attribut *expires*. La méthode *connect()* invoque le service web *transmit* avec les paramètres *key* et *id*. Comme nous le précisions, le *Subscriber* traite la réponse en cours de réception. Dans l'état trois, il initialise les attributs *isConnected* à *true*. Puis il traite le corps de la réponse ligne par ligne au fur et à mesure de sa transmission. Le traitement du corps de la réponse provoque les changements d'état du *Subscriber* (voir les prototypes *observer, observed.*). En fin de transmission, le *Subscriber* affecte la valeur *false* à l'attribut *isConnected*, vérifie la période de validité de son abonnement. Si l'abonnement est toujours valide la fonction retourne un appel à la fonction *connect()*. Sinon, elle retourne *false*. La méthode *unsubscribe()* invoque le service web *subscribe* avec les paramètres *key*, *id* et le paramètre *expires* valant 0. Le contrôleur de session mVIP provoque la sortie du service VoiceXML, notifie ses collaborateurs la fin de la session. En cas de succès, le service web retourne une valeur *expires* valant 0. En cas d'échec le *Subscriber* affecte la valeur 0 à son attribut *expires*.

La figure 3.15 présente comment le Subscriber traite le flux HTTP en cours de chargement (*onload*). La fonction de callback *handleContent()* est appelée à chaque réception d'information sur le flux HTTP. L'information est traitée et répercutée sur les objets à l'écoute des modifications. La propagation de l'information est assurée par le mécanisme mis en jeu par les objets *Observer* et *Observed*.

```
function handleContent(event)
{
  var result = event.target.responseXML;
}

var xrequest = new XMLHttpRequest();
xrequest.multipart = true;
xrequest.open("POST","http://mVIP/ws/mVIP.cgi");
xrequest.onload = handleContent;
xrequest.send(null);
```

FIG. 3.15 – Mécanisme de « push HTTP ».

Observer, Observed Les prototypes *Observer* (observateur) et *Observed* (observé) implémentent le patron observateur/observé [Gamma *et al.*, 1995]. Ils assurent la propagation des changements d'état du modèle de l'outil mVIP dans la hiérarchie d'objets *Observer*/*Observed*.

L'objet *Observed* connaît ses observateurs, il les notifie de ses changements d'état. Il existe deux implémentations de ce mécanisme : l'observé notifie simplement les observateurs que son état a changé ou bien, l'observé notifie les observateurs des changements d'état. Dans le premier cas, une fois notifié du changement, les observateurs ont à leur charge de se mettre à jour en fonction de l'observé. Dans le second cas, la notification provoque la mise à jour de l'observateur. Le choix de l'une plutôt que l'autre des implémentations dépend du nombre d'observateurs et de la pertinence de leur mise à jour. Par exemple, des objets graphiques hors champs n'ont besoin de se mettre à jour uniquement lorsqu'ils réintègrent le champs. A contrario, la notification des changements d'état réduit le nombre des appels de fonction et permet à un observé d'être son propre observateur.

L'*Observer* possède les méthodes :

- attach() ;
- detach() ;
- notify() ;
- getState().

La méthode *attach()* ajoute la référence d'un observateur dans la liste des observateurs de l'objet. La méthode *detach()* l'enlève de cette liste. La méthode *notify()* déclenche pour chaque observateur la méthode *update()*. La méthode *getState()* est rendu accessible aux observateurs afin qu'il puissent se mettre à jour dans le cas d'une notification simple. L'*Observed* possède la méthode *update()* lui permettant de se mettre à jour en fonction de l'état de l'objet observé.

Le prototype *Subscriber* hérite des caractéristiques de *Observed*. En effet, le traitement du flux HTTP donne lieu à des changements d'états du *Subscriber* et à la notification des objets qui l'observent. Le prototype *Subscriber* hérite de *Observer* observateur. En effet, dans le cas l'appel à la méthode *unsubscribe()*, le *Subscriber* modifie son état en se notifiant des changements.

Controller Le prototype *Controller* (contrôleur) est en charge de modifier l'état de l'outil mVIP. Il transmet les événements capturés par le navigateur web qui sont significatifs pour le modèle de l'outil mVIP. Il utilise le service web *control* pour envoyer les événements au gestionnaire d'événements de la plate-forme mVIP. Dans la section 4.2 du chapitre suivant nous verrons comment le pavé numérique du simulateur de DAB transmet les clics utilisateur à la plate-forme mVIP et permet d'interagir avec le service VoiceXML.

Synthèse sur les interacteurs mVIP Les interacteurs mVIP composent les prototypes ECMA précédemment cités. Ils les organisent selon de diagramme de classe de la figure 3.16.

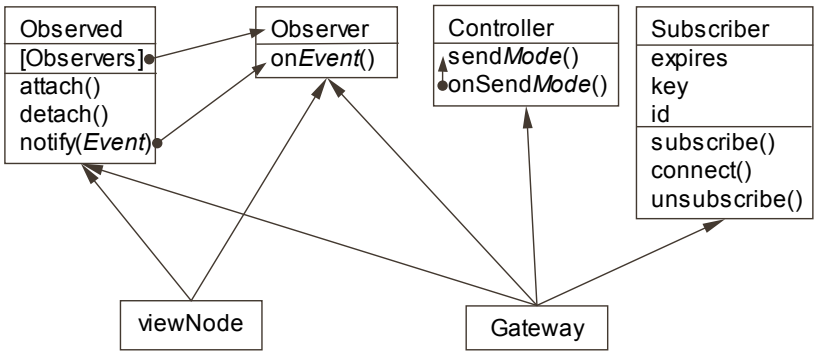


Fig. 3.16 – Organisation des objets ECMA.

On note deux interacteurs principaux : la *Gateway* en charge de communiquer avec la plate-forme mVIP et le *viewNode* en charge de modifier le DOM. Le *viewNode* hérite de la classe *Observed*. En effet, le *viewNode* est observé par la *Gateway* : ceci permet de remonter les événements du navigateur web vers la plate-forme mVIP. Le *viewNode* hérite également de la classe *Observer*. En effet, il observe la *Gateway* afin d'adapter la présentation aux changements d'état de l'outil mVIP.

Par conséquent la *Gateway* hérite elle aussi des classes *Observer* et *Observed*. La *Gateway* hérite également de la classe *Subscriber*. Ainsi la *Gateway* établit la communication entre le terminal téléphonique et la plate-forme mVIP, elle est tenue informée de l'évolution de la session par l'intermédiaire du mécanisme de streaming HTTP. Enfin, la *Gateway* hérite de la classe *Controller* afin d'injecter des événements dans la session mVIP. La section suivante présente le scénario et la séquence mVIP, dernier élément du triplet définissant un outils mVIP.

3.5.3 Scénario et séquence mVIP

L'étude de VoiceXML nous révèle un langage permettant de définir des outils. La plate-forme mVIP offre au concepteur un ensemble d'outils personnalisables. Cet outil prend la forme d'un *scénario* constitué de *séquences*. La séquence est élaborée selon la logique de dialogue attendue par le concepteur. Ainsi, le système produit un énoncé, attend la réaction de son interlocuteur et lit une nouvelle séquence en conséquence.

Ces séquences intègrent différentes briques données à titre d'exemple et dans le but d'être modifiées par le concepteur. Ces séquences regroupent plusieurs catégories :

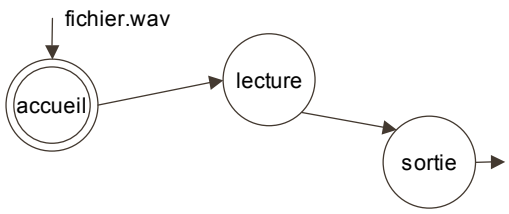
- des énoncés personnalisables intégrant des séquence de « synthèse », qui sont des descriptions SSML de textes à synthétiser, ils sont donnés pour être réutilisés et plus particulièrement pour être adaptés au service cible ; des « jingles », qui sont un ensemble de fichiers audio permettant de dynamiser le dialogue, notifier l'utilisateur de traitements de données et des « radio », qui sont des flux audio envoyés en continu à l'utilisateur, ces énoncés n'ont pas de fin, il sont notamment utilisés pour faire patienter l'utilisateur pendant un transfert d'appel ou un traitement de données particulièrement long ;
- des actions types reprenant la logique de service de VoiceXML, offrant au concepteur des cas d'utilisation avancés du langage VoiceXML : enregistrer, transférer, reconnaître, naviguer et facturer ;

– des redirections types destinées à être redéfinies et adaptées au contexte d'utilisation : le timeout est joué quand le délai d'attente d'un événement utilisateur est dépassé, les erreurs sont jouées quand l'utilisateur effectue une action non désirée, la sortie permet de quitter le dialogue en s'assurant de clôre les sous dialogues établis entre le système et l'utilisateur, enfin le retour au menu principal est une redirection nécessaire à l'ergonomie de l'interface vocale.

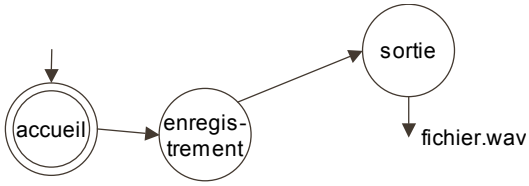
La notification de changement d'état du noyau fonctionnel s'effectue par l'insertion de marqueurs dans les séquences. Nous utilisons la balise `meta` de SSML dans ce but. La balise `meta` de SSML contient deux attributs : `name` et `content`. Ceci nous permet de transmettre des événements sémantiques propres à l'outils mVIP et de définir une interaction fine entre les différentes composantes de l'interaction. L'attribut content contient les paramètres à transmettre qui sont notés selon la notation JSON. Ce qui donne par exemple :

```
<meta name    = "send_event"
      content = "{event  : 'InitLangChoice',
                  params : {lang : 'en'} }"/>
```

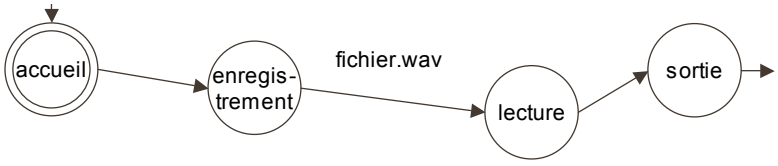
Comme vu en introduction de section, le scénario VoiceXML est assimilable à un automate d'états ; il possède un état initial et des états terminaux. L'article [Ould Ahmed Limam, 2003] met évidence l'intérêt de la définition de sous automates. L'utilisation de sous automates favorise la factorisation de code et la réutilisation des composantes existantes. La figure 3.17 montre comment l'arrangement du scénario « lecture » et « enregistrement » rend compte de l'outil « magnétophone ». Le scénario enregistrement (figure 3.17(a)) stocke le contenu prononcé par l'utilisateur dans un fichier et retourne un pointeur vers ce fichier. Le scénario lecture (figure 3.17(b)) lit un fichier audio passé en paramètre.



(a) Scénario de lecture



(b) Scenario d'enregistrement



(c) Scénario de l'outil magnétophone

FIG. 3.17 – Combinaison de scénario.

Chacune de ces séquences est destinée à être adaptée au service cible. Les séquences adaptées viennent enrichir la librairie de séquences types, assurant ainsi la réutilisation des développements passés. Le triplet énoncé, action, redirection constituent les tours de parole système du dialogue entre l'utilisateur et l'outil mVIP.

Dans la section 4.2 nous décrivons intégralement le scénario nous permettant de définir un distributeur de billets qui exploite les capacités d'interaction du téléphone de l'utilisateur. Dans la section 4.3 un unique triplet permet le pilotage d'un diaporama depuis le téléphone de l'utilisateur.

Conséquence de l'implémentation de la plate-forme mVIP

Initialement nous disposions d'une plate-forme VoiceXML d'expérimentation (PMX). Cette plate-forme a montré l'intérêt du public pour les services vocaux innovants (Timéo). De cette plate-forme nous avons extrait le patron permettant le croisement d'une logique de dialogue et d'un flux média. Nous disposons également d'une plate-forme vocale industrielle VIP ayant fait ses preuves en terme de montée en charge et de flexibilité. Nous avons réalisé l'intégration du patron PMX sur cette plate-forme. Nous avons également adjoint au contrôle téléphonique de la logique de dialogue, un contrôle web sous forme de services web, permettant ainsi l'extension des traditionnelles interfaces web grâce à de nouveaux moyens d'interaction et, réciproquement, le couplage de modalités graphiques faisant défaut aux services vocaux. Enfin, nous avons organisé les logiques de dialogue et les clients web dans un espace de conception destiné aux concepteurs d'applications web.

Nous avons ainsi construit la plate-forme mVIP, élément clef du réseau ubiquitaire. Elle est capable de contrôler et d'analyser les flux générés par les terminaux téléphoniques depuis de web (RTP, section 1.5). Son utilisation repose sur le langage déclaratif VoiceXML (section 1.4) et sur l'utilisation des technologies web (section 1.3.1). La plate-forme mVIP fournit aux concepteurs d'interfaces web des services (figure 3.3) permettant de situer l'utilisateur dans un environnement interactif.

Dans le chapitre suivant, nous décrivons trois démonstrateurs que nous avons réalisés au cours de notre étude. Ces démonstrateurs exploitent les fonctionnalités de la plate-forme mVIP et montrent pas à pas les possibilités qu'elle offre. Le premier démonstrateur montre comment les services mVIP apportent une solution simple aux administrateurs d'un système biométrique, leur permettant de superviser la phase d'enrôlement des utilisateurs. Le second est un simulateur de distributeur de billets. Le simulateur utilise une interface répartie à la fois réelle et réaliste. Nous décrivons avec soin le matériel utilisé et le scénario du simulateur. Le troisième démonstrateur montre comment nous avons défini un snippet de quelques lignes permettant d'étendre les capacités d'interaction des interfaces web existantes. Nous illustrons l'utilisation de notre snippet en étendant les capacités d'interaction de l'application web slidy HTML.

Chapitre 4

Services ubiquitaires intégrant des interfaces multimodales distribuées

Winston rapprocha de lui le phonoscript, souffla la poussière du microphone et mit ses lunettes.
George Orwell, « 1984 », 1949.

Extrait du film « 1984 », réalisé en 1984 par Michael Radford, inspiré du roman.

Ce chapitre met en œuvre les concepts précédemment énoncés à travers plusieurs démonstrateurs ayant trait au domaine bancaire, à la sécurité des systèmes, des contenus et des individus. Chacun de ces démonstrateurs explicite un point clef de notre travail. Le premier démonstrateur permet d'administrer un système d'authentification des utilisateurs par biométrie vocale. Il montre comment notre technologie va permettre le déploiement de services innovants intégrant le mode vocal. Il illustre notre technologie à travers un cas d'usage concret. Le second démonstrateur met en scène un utilisateur et un distributeur automatique de billets (DAB). Le DAB bancaire reprend les fonctionnalités d'un DAB classique dont les capacités d'interaction se voient enrichies grâce à une interaction téléphonique. Il illustre avec un scénario innovant, la mise en œuvre complète d'un service utilisant notre technologie. Enfin le dernier démonstrateur permet de prendre le contrôle d'un diaporama grâce à un téléphone. Il illustre comment notre technologie permet d'étendre à moindre coût les applications web existantes.

4.1 Interface d'administration d'un système d'authentification biométrique

Dans [Cabal, 2003], C. Cabal définit la biométrie comme étant « un système d'identification des individus utilisant des caractéristiques mesurables comme les empreintes digitales, l'iris, la rétine, la forme du visage, de la main, la voix, voire la démarche ou le système veineux ». La biométrie permet donc de trouver ou de vérifier l'identité d'une personne, à l'aide de ses caractéristiques physiques ou comportementales. Cette idée est l'application de notre comportement quotidien : on reconnaît les personnes grâce à leurs démarches, leurs visages, leurs voix au téléphone.

4.1.1 Les systèmes biométriques

Dans les systèmes traditionnels, on a recours à un élément extérieur à connaître ou à posséder pour vérifier l'identité d'une personne : un mot de passe, un identifiant, une clef ou une carte d'accès. Ces éléments comportent beaucoup de risques de vols et de fraudes : les utilisateurs choisissent souvent des mots de passe simples ou faisant partie d'un dictionnaire ; ils les enregistrent de manière non chiffrée fragilisant ainsi la sécurité des systèmes d'information.

La biométrie ajoute un niveau supplémentaire de sécurité en utilisant une caractéristique physiologique ou comportementale de la personne pour vérifier son identité. Elle peut remplacer entièrement les systèmes classiques, ou fonctionner en complément de ceux-ci.

Pour pouvoir utiliser une caractéristique physiologique ou comportementale comme caractéristique biométrique, il faut qu'elle soit universelle, que toute personne la possède ; distinctive, différente chez tout le monde ; permanente, invariante dans le temps et quantifiable. On notera de plus que si le coût initial d'un système biométrique est généralement plus élevé que la mise en place de mots de passe ou de cartes d'accès, les coûts engendrés par les oublis et pertes sont nuls dans le cadre de la biométrie [Jain *et al.*, 2004].

Caractéristiques des systèmes biométriques Un point important pour le déploiement d'un système biométrique est l'adaptation de la technique biométrique avec le service à sécuriser. Ainsi, le choix de la (ou des) donnée(s) biométrique(s) à utiliser dépend de l'application : en effet, chacune a des avantages et des inconvénients, et celles-ci peuvent convenir à certains cas et non à d'autres. Il n'existe pas de système idéal. Un des critères à considérer pour ce choix est tout d'abord la robustesse. En effet, les conditions d'utilisation de la biométrie impliquent une variation de l'empreinte biométrique. Ainsi, certaines biométries comme la voix seront plus sensibles au stress que d'autres. Un autre critère est la sensibilité aux conditions extérieures. On observera la luminosité pour la reconnaissance du visage, la propreté pour les empreintes digitales, le bruit pour la voix. Il est également important de prendre en compte les impressions des utilisateurs : la facilité d'utilisation et le caractère intrusif doivent être considérés [Catoire *et al.*, 2006].

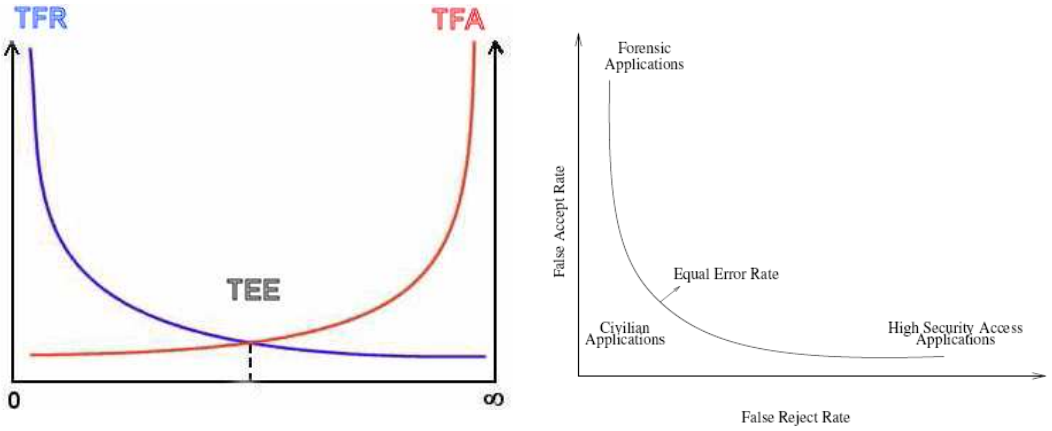
On notera enfin qu'en France, la CNIL[105] recommande l'usage de techniques peu intrusives, se basant sur des données ne laissant pas de traces, (contrairement aux empreintes digitales), et ne dévoilant pas la vie privée des personnes (une image de la rétine permet par exemple de détecter de l'hypertension ou le diabète chez un utilisateur) [Cabal, 2003]. Elle recommande par exemple l'usage de la biométrie vocale.

[105] CNIL - Commission Nationale de l'Informatique et des Libertés.

Fonctionnement d'un système biométrique Un système biométrique contient systématiquement quatre modules. Un capteur acquiert les données biométriques. Un second module extrait les paramètres utilisés à partir de l'information fournie par le capteur. Un troisième module compare l'empreinte extraite à l'empreinte de référence. Enfin, un quatrième module contient des renseignements sur les personnes, ainsi que leurs données acquises lors de la phase d'enrôlement[106]. On note que cette base de données peut faire partie intégrante du système ou se situer à l'extérieur (comme sur des cartes d'accès par exemple). Le module de comparaison va établir un score à partir des similarités et différences entre l'information contenue dans la base et l'information prélevée. Les données biométriques n'étant jamais totalement identiques, il va falloir introduire un *seuil* d'acceptation au-dessus duquel les personnes sont considérées comme acceptées, et en-dessous duquel elles sont refusées [Jain *et al.*, 2004]. Ce seuil est généralement déterminé de façon statistique.

Pour la *biométrie vocale*, le capteur est un micro, par exemple celui du téléphone. Les paramètres extraits sont les harmoniques du signal, comme pour la reconnaissance vocale. Contrairement à un module de reconnaissance vocale, le système de biométrie vocale cherche à être discriminant par rapport au locuteur.

Pour évaluer les performances d'un tel système, ainsi que pour fixer le seuil, on considère deux valeurs : le taux de fausses acceptations[107], et le taux de faux rejets[108]. Les fausses acceptations correspondent aux personnes acceptées à tort. Les faux rejets correspondent aux personnes rejetées à tort, alors qu'elles revendiquent la bonne identité.



(a) TFR et TFA en fonction du seuil choisi. (b) Modulation du seuil selon l'utilisation.

FIG. 4.1 – Evaluation d'un système biométrioque [6]. La figure 4.1(a) présente l'allure générale de ces courbes représentant de TFR et le TFA en fonction du seuil choisi. On remarque que ces variables sont liées. La figure 4.1(b) représente le TFA en fonction du TFR, elle permet de déterminer le seuil à choisir en fonction des attentes du système biométrique (confort ou sécurité).

Le seuil est à moduler en fonction de l'application : pour une application où la sécurité est très importante, on choisira un seuil élevé, qui permettra d'avoir peu de faux acceptés, mais augmentera le nombre de faux rejetés. Au contraire, si le coté rapide et la facilité d'utilisation

[106] Phase d'enrôlement : phase durant laquelle les données concernant un utilisateur sont extraites puis enregistrées avec ses caractéristiques (nom de la personne, identifiant...).
[107] TFA - Taux de Faux Acceptés, acceptés à tord.
[108] TFR - Taux de Faux Rejetés, rejetés à tord.

sont mis en avant, on abaissera le seuil. La valeur du seuil pour laquelle TFA = TFR est appelée TEE[109] (figure 4.1). Plus il est petit, plus le système est performant.

Les courbes de la figure 4.1 caractérisent tout système biométrique. Afin de prendre en compte les variations liées à l'inconstance de la donnée biométrique, l'acquisition de celle-ci durant l'enrôlement se fait à l'aide de plusieurs captures. Le système élimine les empreintes de mauvaise qualité, et construit le gabarit[110] avec celles significatives.

Pour finir, on ajoutera que les systèmes biométriques ont deux modes de fonctionnement : *authentification*, et *identification*. Le premier implique que l'utilisateur donne au système un identifiant. Le système vérifie la véracité des informations transmissent et accepte ou rejette l'utilisateur. Le second analyse les données qu'il capture et cherche à identifier l'utilisateur.

4.1.2 Service ubiquitaire d'administration des utilisateurs d'un système biométrique

Objectif La mise en place et la maintenance d'un système de biométrie vocale se heurte à trois dificultés. La première est inhérente à tous les systèmes de biométrie. Il concerne l'étape d'*enrôlement*. La collecte des empreintes biométriques d'une personne donne lieu à la réunion physique de l'administrateur du système et de la personne à enrôler ce qui implique la synchronisation d'au moins deux agendas. Pour cette raison, l'interface *ubiquitaire* d'administration des utilisateurs du système de biométrie vocale réuni ces acteurs sur le réseau et non pas physiquement ce qui diminue les contraintes de synchronisation en éliminant les paramètres géographiques.

La seconde difficulté concerne également toutes les biométries mais particulièrement la biométrie vocale. Elle traite de l'importance du *capteur*. Dans un contexte téléphonique, on note une grande diversité des terminaux (marques, modèles) et donc des capteurs. Dès lors, il est crucial d'effectuer le prélèvement des empreintes biométriques depuis le même terminal qui servira à authentifier l'utilisateur. L'interface ubiquitaire devient le point de convergence du réseaux IP et du réseau téléphonique, assurant la liaison entre la plate-forme biométrique et les capteurs biométriques.

Le dernier point, propre à la biométrie vocale, est lié au caractéristique physique et cognitive du mode vocal. L'interface *multimodale* d'administration des utilisateurs du système de biométrie vocale adjoint au mode vocal, le mode graphique. La complémentarité de ces deux modes dans un tel cas d'usage offre à l'utilisateur et à l'administrateur un support visuel rendant compte des différentes étapes propres à l'enrôlement.

Le service ubiquitaire d'administration des utilisateurs du système biométrique présenté fournit un ensemble de solutions propres à ce cas d'usage.

Scénario de test Notre système biométrique est relié à l'annuaire LDAP de l'institut cible. Ainsi, les informations sur les utilisateurs (nom, prénom, numéro de téléphone) sont collectés à travers le réseau. Pour chaque utilisateur, le système crée une entrée locale et stocke les empreintes biométriques. L'interface d'administration des utilisateurs du système biométrique offre un visuel de chacun des utilisateurs (figure 4.2), on y retrouve les informations de l'annuaire LDAP et l'état de cette entrée dans le système. Initialement, l'utilisateur ne possède aucune empreinte et, par conséquent, ne peut procéder à aucune authentification. L'interface inclut un lien actif permettant l'enrôlement de l'utilisateur. L'activation de ce lien provoque la mise

[109] TEE - Taux d'Égale Erreur

[110] Gabarit - (biométrie) empreinte de référence, donnée à partir de laquelle les comparaisons futures seront effectuées

en relation entre le terminal de l'utilisateur, identifié par le numéro de téléphone présent dans l'annuaire LDAP, et le service d'enrôlement. Notons, que l'utilisateur et l'administrateur ne se trouvent pas obligatoirement dans le même pièce. Ce détail se révèle important car l'empreinte biométrique dépend non seulement de l'utilisateur, mais aussi de son environnement (terminal, bruit ambiant, contexte d'utilisation).

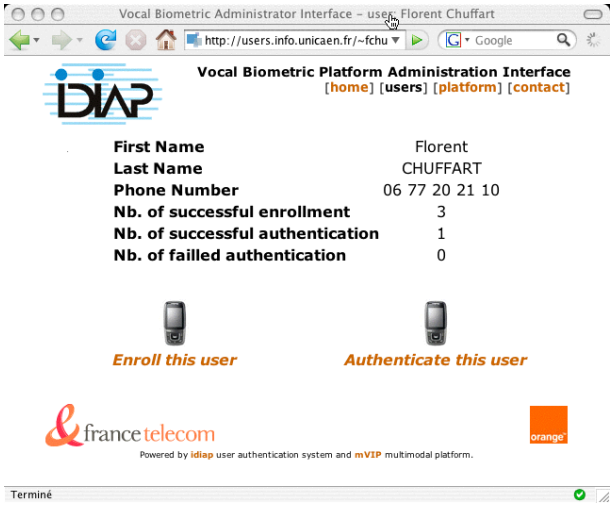


FIG. 4.2 – Interface d'administration des utilisateurs.

L'utilisateur est guidé par la voix de synthèse qui le notifie de chacune des étapes propres à l'enrôlement. L'administrateur et l'utilisateur suivent pas à pas les étapes du service d'enrôlement grâce à l'interface graphique (figures 4.3(a) et 4.3(c)). Dans le cas du système de notre service de biométrie vocale, l'enrôlement consiste en l'analyse de trois répétitions identiques d'une phrase secrète suffisamment longue, au moins sept syllabes.

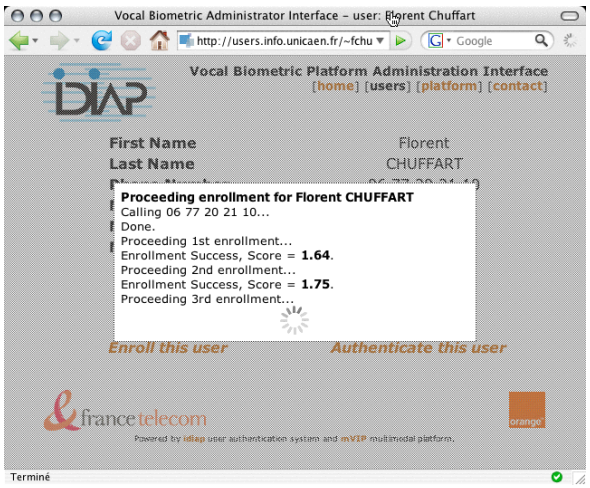
Une fois l'enrôlement réussi, l'interface offre la possibilité de tester le service d'authentification. En cliquant sur le lien actif *Authenticate this user* (figure 4.2), l'utilisateur est mis en relation avec le service d'authentification. Comme pour la phase d'enrôlement, l'utilisateur est guidé par la voix de synthèse. L'utilisateur et l'administrateur suivent les étapes d'authentification sur l'interface graphique(figure 4.3(b) et 4.3(d)).

Enfin, le système biométrique stocke localement le nombre de tentatives d'authentification réussies ou manquées. L'interface graphique rend compte de ces valeurs (figure 4.2).

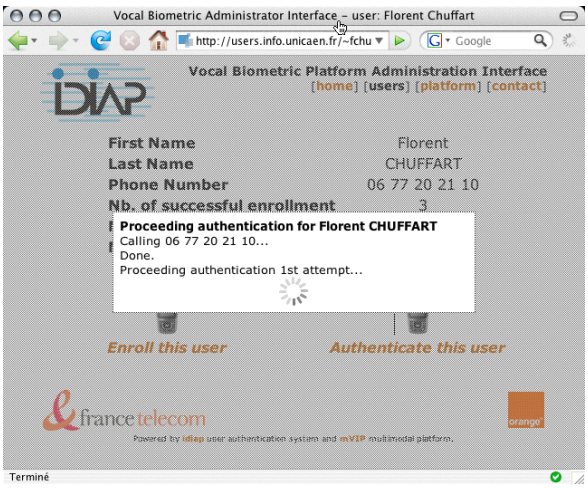
Architecture de test L'architecture de test (figure 4.4) est composée :

- d'un annuaire LDAP contenant les informations sur les personnes de l'institution cible, accessible depuis l'intranet de l'institution ;
- d'une plate-forme de biométrie vocale proposant un jeu de services web permettant d'exploiter les fonctionnalités de la plate-forme ;
- de la plate-forme de services ubiquitaires mVIP intégrant des fonctionnalités de téléphonie et son interface de service web précédemment décrite ;
- d'un serveur web en charge de l'intégration des différents services offerts par ces trois dernières composantes.

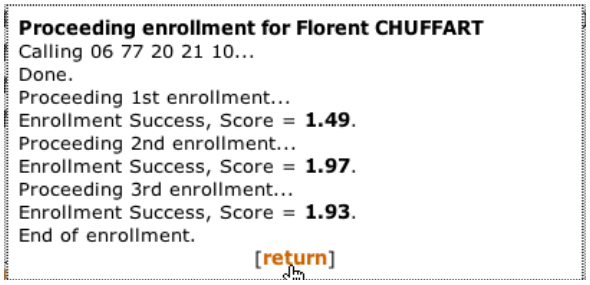
Nous appelons ce dernier composant le *Maestro* relativement à son rôle d'orchestration des services distribués dans le réseau.



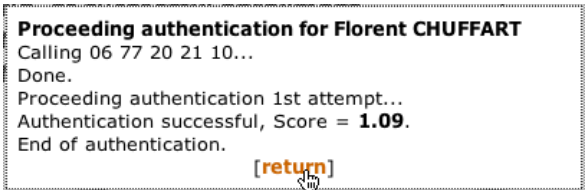
(a) Enrôlement.



(b) Authentification.



(c) Fin de l'enrôlement.



(d) Fin de l'authentification.

FIG. 4.3 – Assistance graphique de l'enrôlement et de l'authentification vocale. les figures 4.3(c) et 4.3(d) présentent respectivement les popups des figures 4.3(a) et 4.3(b) à la fin des processus d'enrôlement et d'authentification.

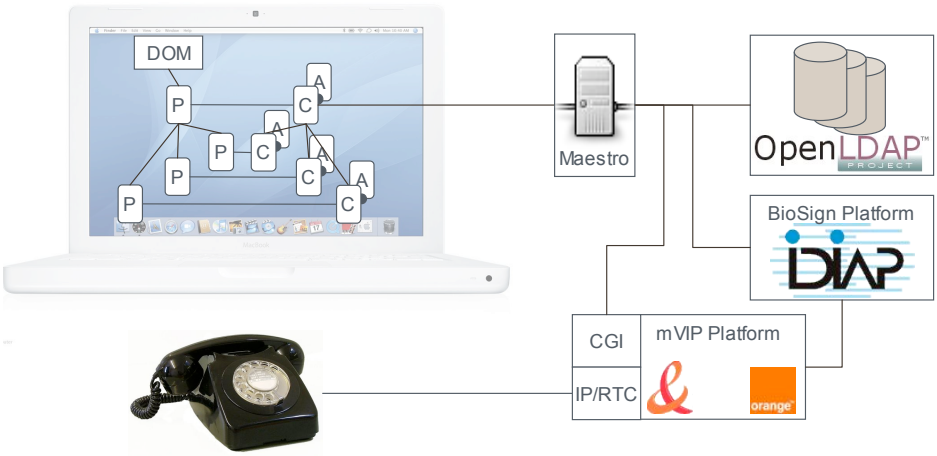


FIG. 4.4 – Architecture de test du module mBioSign.

Plate-forme BioSign La plate-forme BioSign dispose d'une base de données associant à un utilisateur un jeu d'empreintes vocales de référence. À chaque tentative d'authentification, ces empreintes de référence sont comparées à l'empreinte présentée par l'utilisateur. Cette comparaison donne lieu à un score, qui, comparé au seuil pré-défini, permet d'accepter ou de rejeter l'utilisateur.

Les fonctionnalités de la plate-forme BioSign sont rendues accessibles par cinq services web :

- *addUser(numTel)* ajoute un utilisateur dans la base BioSign, l'utilisateur est identifié par son numéro de téléphone *numTel*, il retourne *OK* en cas de succès ou *KO* en cas d'échec ;
- *delUser(numTel)* efface un utilisateur et ses empreintes existantes de la base BioSign, il retourne *OK* ou *KO* en fonction du succès de l'opération ;
- *getUserStatus(numTel)* retourne le statut de l'utilisateur, à savoir le nombre d'enrôlements valides pour l'utilisateur identifié par son numéro de téléphone, ou *KO* si l'utilisateur n'existe pas ;
- *enrollUser(numTel, wavFile)* tente d'ajouter l'empreinte de référence extraite de *wavFile*, à l'utilisateur identifié par son numéro de téléphone *numTel*, retourne *OK nbEmprValid* en cas de succès, où *nbEmprValid* représente de nombre d'empreintes valides relatif à l'utilisateur dans le base BioSign, retourne *KO* en cas d'échec ;
- *authenticateUser(numTel, wavFile)* tente d'authentifier à partir de l'empreinte vocale extraite de *wavFile*, l'utilisateur identifié par son numéro de téléphone *numTel*, retourne *OK* ou *KO* en fonction de succès de l'opération.

Annuaire OpenLDAP OpenLDAP est une implémentation libre du protocole LDAP[111], développée par The OpenLDAP Project [26]. OpenLDAP ne stocke pas les données directement, il utilise une bibliothèque tiers pour le faire. En revanche, il permet l'interrogation et la modification des services d'annuaire par l'intermédiaire du protocole LDAP. Notre annuaire OpenLDAP est celui déployé dans l'institution cible : il contient les identifiants, les coordonnées et les numéros de téléphone des utilisateurs des ressources informatiques de l'institution. Dans le cadre de notre architecture de test, un accès anonyme à l'annuaire permet d'obtenir les coordonnées et le numéro de téléphone des utilisateurs à partir de la clef *uid* qui correspond à l'identifiant de l'utilisateur sur le réseau de l'institution cible.

Apports du démonstrateur Le service ubiquitaire d'administration des utilisateurs du système biométrique offre une interface conviviale et fonctionnelle permettant de mettre en œuvre et de superviser les étapes clefs de l'enrôlement et de l'authentification biométrique. L'interface offre une approche graphique rendant compte d'une modalité vocale. Le support graphique de l'application offre un retour sémantique de l'état du système.

Ce démonstrateur a été présenté lors du symposium ITEA à Helsinki, les 13 et 14 octobre 2005. Il est le fruit de deux collaborations de France Telecom : la première avec le groupe Intesi dans le cadre du projet européen ITEA Aurora et la seconde avec l'institut de recherche suisse IDIAP dans le cadre d'un contrat de recherche. Il contribue de manière significative à l'intégration de la biométrie vocale dans les applications web.

[111] LDAP - Lightweight Directory Access Protocol [RFC 4510, 2006].

4.2 Service mATM : simulateur d'un automate bancaire

Dans le cadre de la collaboration entreprise entre France Telecom et l'Université de Caen Basse-Normandie sur la thématique de la sécurité, des cartes à puce et des nouveaux usages, nous avons pris en charge la mise en œuvre d'un service bancaire intégrant des interfaces innovantes. Nous avons choisi d'adresser la problématique de l'accessibilité.

Les personnes présentant un handicap visuel sont confrontées à l'utilisation de dispositifs utilisant des interfaces graphiques. Les actions quotidiennes telles qu'envoyer un SMS, réserver un billet de train, consulter une grille d'horaire, une carte des desserts ou des vins font appel au sens de la vue. Les lois sur l'accessibilité [Projet de loi sur l'égalité des droits, 2005] oblige les services de communication en ligne des services de l'Etat, des collectivités territoriales et des établissements publics à être accessibles aux personnes handicapées. Les fournisseurs de services proposent des alternatives, par exemple, les horaires de bus sont vocalisés par le téléphone de l'utilisateur dans le service Timéo (section 1.6). Des initiatives prises par les organismes de normalisation [45] tendent à permettre à tous d'accéder au web.

Cependant, nous constatons un manque dans ce domaine en ce qui concerne l'accès aux automates bancaires. Bien que la cinématique des distributeurs de billets soit relativement cadrée, on constate des divergences d'un établissement bancaire à l'autre ou encore entre les différents modèles des constructeurs de DAB. Une personne non-voyante est fidèle à un ensemble réduit de DAB. Les DAB sont préalablement identifiés par essais successifs ou par l'intermédiaire d'un tiers qui va décrire à l'utilisateur les actions proposées par l'automate. Afin d'illustrer la manière dont les services mVIP vont accroître l'autonomie des personnes handicapées, nous avons élaboré un simulateur offrant une assistance vocale à la consultation de services bancaires sur un automate.

Dans les sections qui suivent nous présentons le simulateur mATM mettant en situation une personne non voyante disposant d'un téléphone mobile et un automate bancaire équipé du service mATM. Nous détaillons l'interface du service et l'architecture qui la supporte. Ensuite nous commentons le scénario qui a permis de réaliser le simulateur.

4.2.1 Cas d'usage du service mATM

Le contexte de l'interaction se situe dans l'espace public. Un utilisateur non-voyant se trouve face à un distributeur de billets équipé du service mATM. Le téléphone de l'utilisateur est équipé d'un kit piéton ce qui permet une utilisation moins contrainte. L'utilisateur approche son téléphone près du lecteur de tag NFC destiné à initier le service (figure 4.6(a)). Grâce aux informations contenues dans la puce, le distributeur de billets initie un appel vers le téléphone de l'utilisateur. La figure 4.5 retranscrit le dialogue issue du cas d'usage qui suit.

À la prise d'appel, l'utilisateur est accueilli par un prompt audio lui indiquant qu'il se trouve devant un distributeur de billets et qu'il va être guidé vocalement pendant ses opérations. Ensuite l'utilisateur est invité à insérer sa carte dans le lecteur prévu cet effet (figure 4.6(b)). L'utilisateur est convié à taper son code secret sur l'automate (figure 4.6(d)). Une fois le code validé, l'écran présente les différents services accessibles. Un prompt audio énumère les choix et précise qu'ils sont accessibles depuis les touches du téléphone ou depuis le pavé numérique du DAB (figure 4.6(d)). Par exemple, l'utilisateur tape 2 sur son téléphone ce qui lui permet d'effectuer un retrait d'argent (figure 4.6(e)). L'écran affiche un ensemble de sommes prédéfinies allant de 20 à 120 euros et offre la possibilité pour l'utilisateur de saisir le montant du retrait (figure 4.6(f)). La voix de synthèse notifie l'utilisateur qu'en tapant 7 il pourra saisir le montant qu'il désire et qu'en tapant un chiffre compris entre 1 et 6 il pourra retirer une somme donnée.

L'utilisateur choisit de préciser lui même le montant. Tandis que l'écran affiche un formulaire précisant le montant désiré, la voix demande à l'utilisateur de saisir le montant et de valider la saisie en tapant dièse. L'utilisateur s'y emploie par l'intermédiaire des touches de son téléphone. Les billets sortent du distributeur. En même temps, l'interface graphique et l'interface vocale somment l'utilisateur de prendre ses billets. L'utilisateur raccroche. Un signal sonore est émis par le DAB en même temps qu'il affiche un pictogramme animé présentant la carte bancaire restituée à son titulaire. L'utilisateur range ses billets et sa carte.

Système : Bonjour, vous êtes en relation avec le service mATM, je vais vous accompagner tout au long de vos opérations. Veuillez insérer votre carte dans le lecteur prévu à cet effet.
Utilisateur : [insertion de la carte]
S : Merci. Veuillez entrez votre code confidentiel sur l'automate.
U : [sur l'automate] ****.
S : Merci. Tapez 1 pour effectuer un retrait d'argent, tapez 2 pour accéder à la synthèse de vos comptes, tapez 3 pour recharger votre compte mobile ou tapez étoile pour quitter le service.
U : [sur le téléphone] 1.
S : Pour précisez la somme désirée, tapez 7. Sinon, tapez 1 pour retirer 20 euros, tapez 2 pour retirer 40 euros...
U : [sur le téléphone] 7.
S : Saisissez la somme souhaitée et valider en tapant dièse.
U : [sur le téléphone] 7 0 #.
S : Désirez vous un ticket ?
U : non.
S : Veuillez prendre vos billets. ... Tapez 1 pour effectuer...
U : [raccrocher]
S : [alerte sonore sur l'automate]
U : [reprend sa carte.]

Fig. 4.5 – Dialogue entre l'utilisateur et le service mATM.

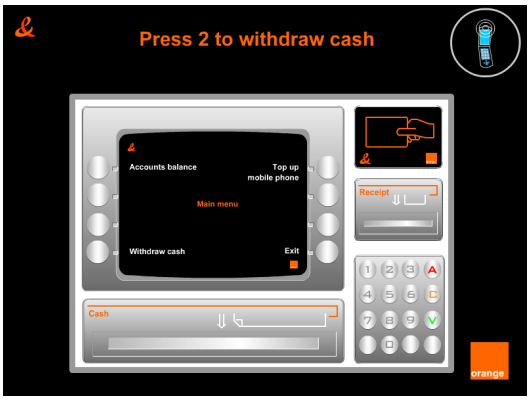
4.2.2 Interface et architecture mATM

L'interface du DAB (figure 4.7) reprend les composantes d'un DAB classique. Il est composé d'un lecteur de cartes bancaires, d'un lecteur de tags NFC et d'un composant piézoélectrique. Le pavé numérique, la fente à billets, la fente à reçu, et l'écran interactif sont simulés par une interface graphique se trouvant derrière un écran tactile. Le tout est incorporé à une borne permettant de disposer ces éléments à hauteur. Le lecteur NFC permet d'identifier le téléphone de l'utilisateur et de mettre en relation l'utilisateur et l'assistant vocal.

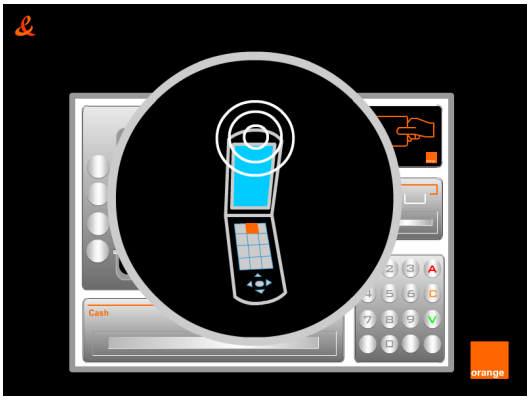
Les clients web communiquent avec la plate-forme mVIP grâce au réseau IP. Le téléphone mobile communique avec le frontal de téléphonie de la plate-forme mVIP. Le lecteur de cartes et le lecteur NFC sont interfacés avec l'objet gateway par le biais d'une applet Java intégrée à l'application web. L'objet gateway communique avec la plate-forme VIP en appelant les services web *subscribe()* et *control()*. La plate-forme mVIP distante instancie la logique de dialogue VoiceXML et effectue son rendu sur le téléphone de l'utilisateur et sur l'interface graphique. La plate-forme mVIP réalise la synthèse des événements provenant du téléphone de l'utilisateur et de l'interface et alimente la logique de dialogue VoiceXML. Elle répercute la progression de l'utilisateur dans le scénario VoiceXML sur les différentes interfaces.

(a) Activation par NFC.

(b) Saisie du code confidentiel.

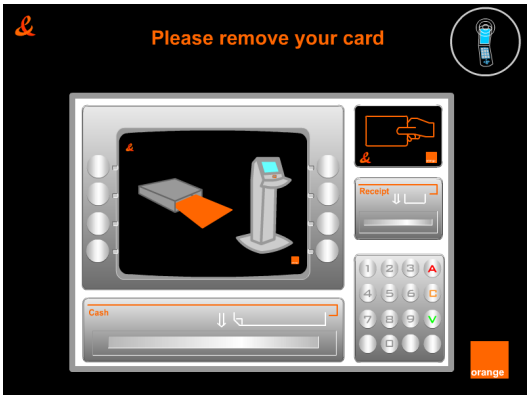


(c) Menu principal.



(d) Saisie DTMF.

(e) Choix du montant.



(f) Restitution de la carte.

FIG. 4.6 – Extrait du story board mATM.

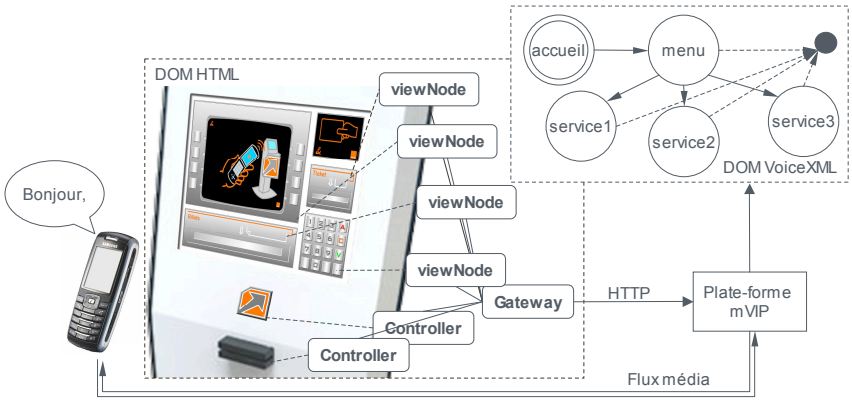


FIG. 4.7 – Architecture mATM.

4.2.3 Scénario du simulateur mATM

Le scénario mATM a été élaboré lors de séances de créativité regroupant le chef de projet, le concepteur d'applications web, un expert de la plate-forme mVIP, un expert du monde bancaire et un expert en cartes à puce. L'articulation du scénario autour de canevas représentés par des objets graphiques assure une compréhension de la problématique commune aux personnes disposant de prérequis propres à leur domaine de compétence. Les différentes planches (figure 4.8) représentent les spécifications fonctionnelles attendue du service mATM.

Choix de la langue La lecture du tag NFC identifie le téléphone de l'utilisateur ce qui provoque l'initialisation d'une session mVIP et le chargement de la logique de dialogue (figure 4.8(a)). La logique de dialogue pilote les interfaces graphiques et vocales (figure 4.8(b)). La première séquence de dialogue concerne le choix de la langue. Si cette information est contenue dans les informations transmises par le tag NFC le dialogue se poursuit. Sinon, les interfaces vocales et graphiques demande à l'utilisateur de choisir la langue à utiliser pour poursuivre. Une fois cette requête exécutée, le système initie un nouvelle requête. La séquence suivante concerne l'insertion de la carte bancaire.

Dépassement de délai d'attente d'un événement utilisateur Le système possède un mécanisme qui lui permet d'exiger d'obtenir une réponse dans un délai imparti. Un dépassement de ce délai provoque une mise en question de l'utilisateur. De la même manière, une réponse non attendue donne lieu à une mise en question différente. Au bout de trois mises en question infructueuses, le système considère le dialogue comme étant divergent. Il en est de même, si l'utilisateur termine brusquement la communication téléphonique. Le système initie alors la procédure de terminaison du dialogue qui consiste à alerter l'utilisateur du retour de sa carte si elle n'a pas été bloquée. Nous avons factorisé cette procédure son la forme d'une séquence unique (DOM VOiceXML, figure 4.7).

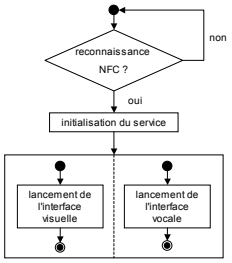
Saisie du code confidentiel Une fois la carte insérée, le système joue la séquence demandant la saisie du code confidentiel. L'expert du monde bancaire nous révèle qu'un tel scénario se doit d'attendre la saisie du code uniquement sur l'automate pour des raisons de sécurité. Ce point soulève le problème de la combinaison des modalités. Dans le cas présent la combinaison est exclusive. Le système se met donc en attente des événements provenant du lecteur de cartes à puce.

Les événements provenant des autres dispositifs ne seront pas traités par le système. Le lecteur de cartes possède sa propre logique de dialogue. Du point de vue du système, les événements utilisateurs reçus pendant la phase d'authentification sont : *chiffre*, *correction*, *annulation*, *code correct* et *carte bloquée*. Le blocage de la carte ou l'annulation par l'utilisateur provoque la sortie du service. La saisie des chiffres ou la correction provoque le retour sémantique sonore (buzzer) et graphique. L'événement *code correct* permet de progresser dans le dialogue et d'accéder au choix des services.

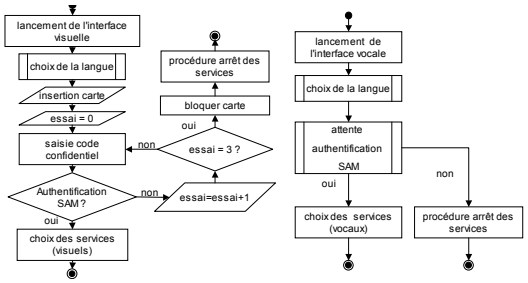
Le choix des services Le système affiche les services disponibles et procède à l'énumération des commandes permettant d'y accéder (figure 4.8(c)). Cette séquence de dialogue est caractéristique de la structure de contrôle VoiceXML *menu* (figure 1.18). La réponse de l'utilisateur provoque le jeu du service souhaité.

Gestion de sous-dialogue Sur notre simulateur, nous proposons un service de consultation des comptes qui retourne à l'utilisateur une synthèse de ses comptes sur un ticket ou sur son mobile par envoi d'un SMS (figure 4.8(d)). Nous proposons également un service de retrait d'espèces et un service permettant de recharger son compte mobile (figures 4.8(e) et 4.8(f)). Ces deux services suivent le même canevas. La séquence débutant le sous-dialogue énonce à l'utilisateur l'ensemble des sommes prédéfinies pouvant être affectées au compte mobile ou en espèces. Avant cela, le système annonce à l'utilisateur qu'en tapant 8 il peut saisir lui-même le montant désiré. L'utilisateur peut alors interrompre le système dans son énumération et accéder au sous-dialogue de saisie du montant (figure 4.8(j)). En VoiceXML, l'affectation à *true* de la propriété *bargein* permet cette navigation rapide dans les services. Une fois l'opération effectuée le système procède à la fermeture du sous-dialogue. Cette fermeture passe par la remise d'un document attestant du bon déroulement de l'opération. Ce document est un ticket papier ou sa dématérialisation sous forme d'un SMS envoyé sur le mobile de l'utilisateur. Comme l'était la sortie des billets, la sortie du ticket est simulée sur l'écran de l'interface.

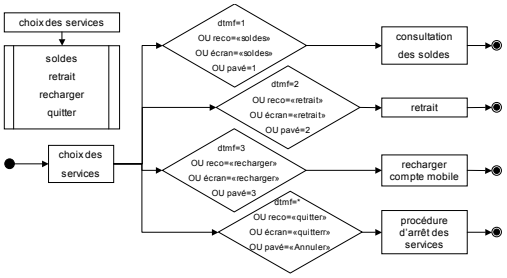
Synthèse sur le simulateur mATM Le simulateur mATM illustre dans un scénario orienté accessibilité ayant trait au domaine bancaire, comment les services mVIP permettent d'un part d'intégrer le mode vocal dans les interfaces web et d'autre part comment ils assurent la convergence du dialogue selon le modèle projectif du dialogue de Vernant (figure 1.17 page 30). mATM illustre l'utilisation que nous avons faite des structures de contrôle du dialogue de VoiceXML permettant la définition de sous-dialogues, de mise en question, de relance et de mise en cause. Le scénario mATM montre également comment les services mVIP permettent de mettre en œuvre les propriétés E et R de CARE. L'interaction multimodale est principalement équivalente, à chaque instant l'utilisateur peut effectuer ses choix sur le mobile par la voix ou les DTMF ou bien sur l'automate avec le pavé numérique ou les interacteurs de l'écran. Dans le cas de la saisie du code secret, l'interaction est assignée au pavé numérique pour des raisons liées à la confidentialité du code. Dans mATM, les éléments des interfaces graphiques et vocales ne sont pas porteurs de sens par eux même : ils sont constitués d'un ensemble d'outils disposant de fonctionnalités unitaires, ce n'est que structurés en séquences qu'ils deviennent cohérents et assurent une sémantique globale.



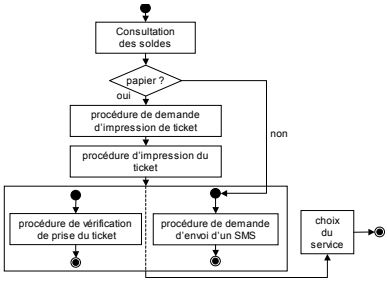
(a) Activation du service mATM.



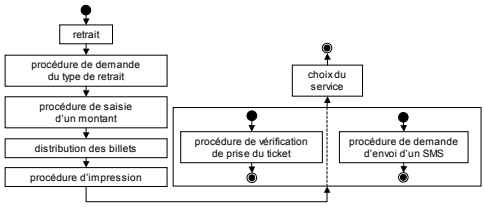
(b) Initialisation des interfaces.



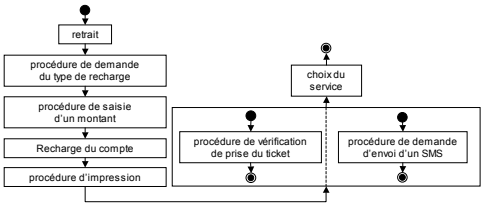
(c) Choix des services.



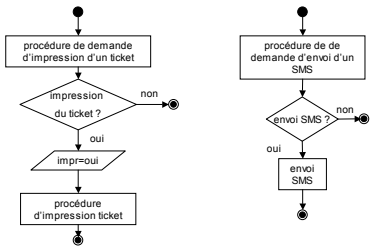
(d) Consultation des soldes.



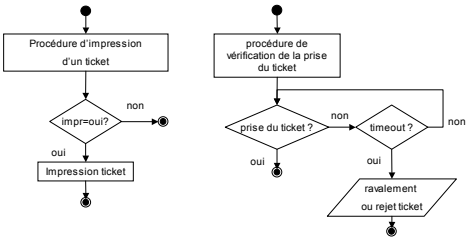
(e) Retrait d'espèce.



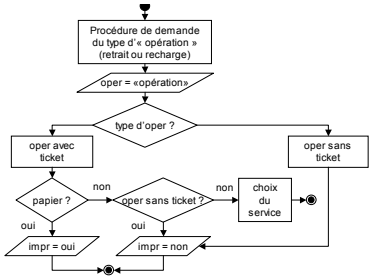
(f) Recharge du compte mobile.



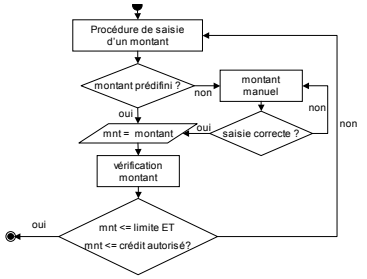
(g) Demande d'impression d'un ticket.



(h) Impression du ticket.



(i) Demande d'opération.



(j) Saisie du montant.

FIG. 4.8 – Spécifications fonctionnelles du simulateur mATM.

4.3 Extension de l'application web Slidy HTML

Dans le cadre de nos travaux de recherche [Chuffart *et al.*, 2005], nous avons présenté un cas d'usage mettant en scène un utilisateur pilotant une présentation depuis son téléphone mobile. Devant l'enthousiasme suscité par ce cas d'usage, nous avons pris en charge la réalisation d'un démonstrateur implémentant ce cas d'usage. L'objectif n'était pas de concevoir de bout en bout un logiciel de présentation mais d'illustrer comment les services mVIP permettent d'étendre les applications existantes.

L'application web Slidy Nous avons identifié Slidy HTML comme une application suffisamment performante pour rendre compte du scénario et suffisamment ouverte pour pouvoir en étendre les capacités d'interaction. Slidy HTML est une application web destinée à la réalisation de présentations. C'est une alternative au logiciel PowerPoint de Microsoft, OpenOffice ou encore LaTeX beamer. Le framework Slidy repose sur les technologies web. Il intègre le HTML afin de structurer le document composite. Chaque diapositive est contenue dans une division HTML structurant des images, des animations et du texte, par des listes, des paragraphes... Les attributs de style du document sont factorisés dans une feuille de style CSS. Les classes CSS définissent les attributs de style des objets qui composent le document. On y retrouve les attributs de style des titres, de la table des matières, des arrières plan, de l'entête et des bas de page des diapositives, le symbole utilisé pour les puces... La navigation de diapositive en diapositive est assurée par le moteur ECMA de l'application. Le mécanisme repose sur la manipulation du DOM. Elle permet un parcours séquentiel du document. Les diapositives peuvent être déroulées de manière incrémentale afin de préserver la tension dramatique du discourt. Ainsi, les documents slidy sont composés d'un seul fichier XHTML. L'utilisation d'une feuille de style adaptée permet l'exportation du document vers une imprimante ou vers d'autres médias.

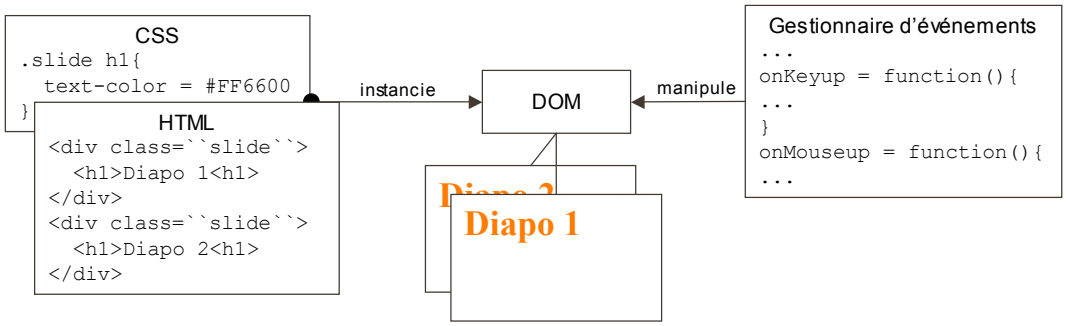


FIG. 4.9 – Fonctionnement général de l'application Slidy HTML. Le DOM du document est instancié à partir de la fusion des informations sur le contenu structuré et le style. Le DOM instancié est manipulé par les objets ECMA qui, de manière événementielle, déroulent la présentation.

Fonctionnement du moteur de Slidy Le moteur de Slidy utilise le gestionnaire d'événements du navigateur. Il réalise une liaison (un « bind ») des événements sur les fonctions de l'application. Ainsi, à l'occurrence d'un événement les paramètres de l'événement sont passés à la fonction de call-back. Par exemple, si l'utilisateur redimensionne la fenêtre du navigateur, la fonction *resized* est appelée avec les paramètres correspondant à la nouvelle taille de la fenêtre. Les événements *document.onkeydown* provoque l'exécution de la fonction *keyDown*. Cette fonc-

tion analyse la valeur *key* de la touche qui a été pressée. En fonction de cette valeur, le moteur de Slidy anime la présentation.

```
function startup() {
  ...
  // bind event handlers
  document.onclick = mouseButtonClick;
  document.onmouseup = mouseButtonUp;
  document.onkeydown = keyDown;
  window.onresize  = resized;
  window.onscroll = scrolled;
  ...
}
```

```
function keyDown(event) {
  var key;
  ...
  key = event.which;
  ...
  if (key == 34) // Page Down {
    nextSlide(false);
    return cancel(event);
  }
  ...
}
```

Prise de contrôle de l'application Afin d'étendre les capacités d'interaction de Slidy, nous avons choisi de réaliser à notre tour une liaison des événements mVIP sur les fonctions de Slidy. Ainsi, nous utilisons le prototype *Gateway* (figure 3.16) afin d'établir une session mVIP entre l'application web et la plate-forme mVIP. Nous étendons le prototype afin de rendre l'application Slidy réactive à un ensemble d'événements téléphoniques. Nous définissons une logique de dialogue en charge de capturer ces événements et nous intégrons le prototype Gateway dans l'application Slidy grâce à la définition d'un snippet. Le snippet ne fait que quelques lignes de code HTML et assure la prise de contrôle de l'application web depuis un terminal téléphonique.

Ergonomie de l'interaction téléphonique Nous avons donc commencé par définir l'ergonomie du service mVIP. Pour ce faire, nous nous sommes inspiré des caractéristiques que nous avions déterminé en 2005. Depuis son téléphone, l'utilisateur pourra naviguer dans la présentation, augmenter ou diminuer la taille de la police, afficher et faire disparaître la table des matières du document et le widget associé à notre service. Nous associons aux touches du téléphone et à un ensemble de mots clefs, des fonctionnalités de l'application Slidy. Le tableau 4.1 présente cette association. Cette disposition des fonctionnalités sur le clavier du téléphone forme une croix. Nous avons choisi cette configuration car nous la retrouvons sur plusieurs dispositifs : les télécommandes des téléviseurs (figure 2.4 page 49) ou encore les consoles de jeu (figure 3 page 3). La figure 4.12 met en évidence cette configuration.

Événement	**DTMF**	**Commande Vocale**
fontSizeUp	2	« *Plus grand.* »
previousSlide	4	« *Précédent.* »
toggleFullScreen	5	« *Plein écran.* »
nextSlide	6	« *Suivant.* »
fontSizeDown	8	« *Plus petit.* »
toggleTOC	*	« *Table des matières.* »
toogleWidget	#	« *Aide.* »

Tab. 4.1 – Ergonomie de la télécommande vocal/DTMF.

Dialogue VoiceXML associé Le dialogue est initié par le client web. Un événement web (clic sur le bouton vert du widget de la figure 4.12) déclenche un appel de service web *subscribe()* qui provoque l'initialisation de la session mVIP. La session mVIP suit la logique de dialogue définie par le script VoiceXML de la figure 4.10. La logique de dialogue utilise le patron VoiceXML *menu*. Ce menu attend les événements téléphoniques précédemment évoqués. À chaque événement, le système notifie l'objet *gateway* instancié dans le DOM de l'application Slidy que l'on souhaite contrôler. L'objet *gateway* accède aux primitives du moteur Slidy et les invoque en conséquence. La figure 4.10 représente ce dialogue sous forme d'un graphe. Une fois l'objet *gateway* notifié, le focus retourne sur le menu et le système se met en attente d'un nouvel événement téléphonique. On remarque que le code VoiceXML associé dépend directement des informations contenues dans le tableau 4.1.

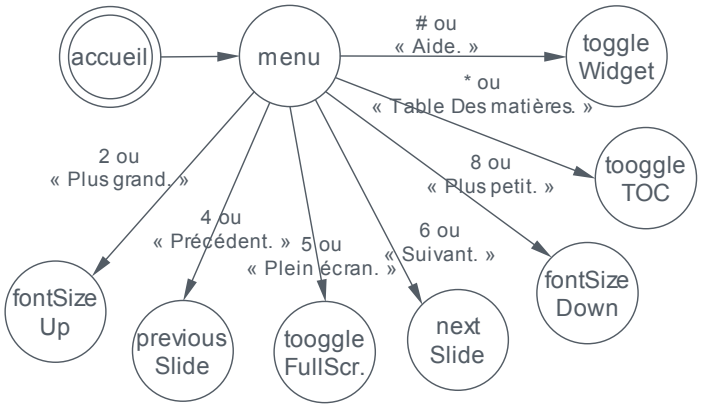


Fig. 4.10 – Logique de dialogue de l'outil télécommande.

```
... <meta name="send" content="[event : isConnected]"/> ...
<menu id="menu">
  <choice dtmf="2" next="#send_fontSizeUp">Plus grand</choice>
  <choice dtmf="4" next="#send_previousSlide">Précédent</choice>
  <choice dtmf="5" next="#send_toggleFullScreen">Plein écran</choice>
  ...
  <choice dtmf="#" next="#send_toogleWidget">Aide</choice>
</menu>
...
<form id="send_fontSizeUp">
  <block>
    <prompt>
      <meta name="send" content="[event : fontSizeUp:wq]"/>
      <goto next='#menu'/>
    </prompt>
  </block>
</form>
...
```

Fig. 4.11 – Script VoiceXML associé.

Objet ECMA associé Sur le navigateur web, dans le DOM de Slidy nous instancions l'objet *Gateway*. Nous avons étendu l'objet Gateway en lui ajoutant des fonctionnalités du moteur Slidy. En effet, la réception d'un événement provenant du téléphone de l'utilisateur provoque un appel aux fonctonnalités du moteur de slidy (diapositive suivante...). L'objet Gateway possède une interface graphique (widget) permettant d'initier la session mVIP et d'offrir un retour sémantique de l'état de l'outil. L'interface graphique est composée d'un champ de saisie. L'utilisateur y inscrit son numéro de téléphone. Une fois le numéro entré, l'utilisateur valide sa saisie en tapant *entrée* ↩ ou en cliquant sur le pictogramme vert représentant un téléphone. Si le numéro de téléphone est valide, le widget fait patienter l'utilisateur en lui présentant un sablier jusqu'à ce que la communication soit établie et que les flux média soient connectés. L'utilisateur accède alors aux fonctionnalités de slidy depuis son téléphone. Dès lors, l'utilisateur utilise son téléphone comme une télécommande. Il peut aussi clore la session en raccrochant son téléphone ou en utilisant le pictogramme rouge du widget.

FIG. 4.12 – Widget mSlidy permet la mise en relation du téléphone de l'utilisateur et du système. Il offre également une interface de contrôle de la présentation. Cette interface de contrôle a surtout vocation à rappeler à l'utilisateur comment utiliser le téléphone pour contrôler la présentation. Sur ce schéma on remarque la croix permettant de naviguer dans la présentation et d'effectuer un zoom sur les diapositives. Une première pression sur la touche # du téléphone permet de faire apparaître le widget, une seconde pression le fait disparaître. La touche * fait apparaître et disparaître la table des matières du document. La touche 5, siglée F11 active et désactive le mode plein écran du navigateur. Le champ de saisie permet d'entrer le numéro de téléphone de l'utilisateur et d'initier la session mVIP.

Logique d'état de l'objet L'objet *Gateway* possède sa propre logique d'état. Dans le cas présent, ces états sont : *déconnecté*, *en connexion*, *connecté*. Initialement le widget est dans l'état *déconnecté* Le changement d'état de l'objet est provoqué par l'événement utilisateur *validation de la saisie*. Dans l'état *en connexion*, l'objet se met en attente d'un événement indiquant l'initiation du dialogue par le système. Typiquement, cet événement est envoyé dans le premier prompt de la logique de dialogue (figure 4.11). Cet événement modifie l'état du widget. Son état est maintenant *connecté*. L'objet Gateway joue alors le rôle de passerelle, assurant la propagation du dialogue dans l'interface graphique. La figure 4.13 représente les états et les transitions de l'objet. À chaque état est associé un comportement. De fait, la communication entre différents organes du service passe par ce composant. Dans la section 2.3.3, nous explicitions un composant unique, racine d'une hiérarchie d'agents interactifs. L'objet *Gateway* est ce composant dans le cas de l'extension de l'application Slidy. En se référant au modèle PAC, le widget est la présentation de ce composant, ses états constituent son abstraction.

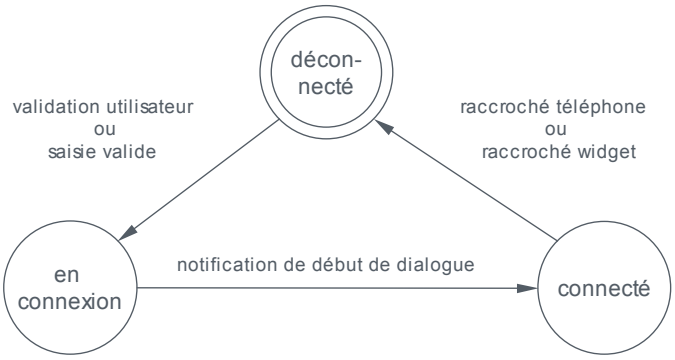


FIG. 4.13 – États internes de l'objet *Gateway*.

Cheminement des messages L'objet *Gateway* intègre le patron *observer/observed* (section 3.5.2). C'est ce patron qui assure la communication entre l'état interne de l'objet, sa représentation graphique (widget), la logique de l'outil mVIP (scénario VoiceXML) et l'application Slidy. À chaque état de l'objet, est associé un comportement. L'objet va tour à tour recevoir des instructions d'un organe et le piloter. Le cheminement des messages passe nécessairement par l'objet *Gateway*, ceci assure une logique globale de l'outil dirigée par l'état de la passerelle. La passerelle relaie les événements web provenant du widget vers la plate-forme mVIP. Les événements mVIP sont relayés au widget et au moteur de slidy.

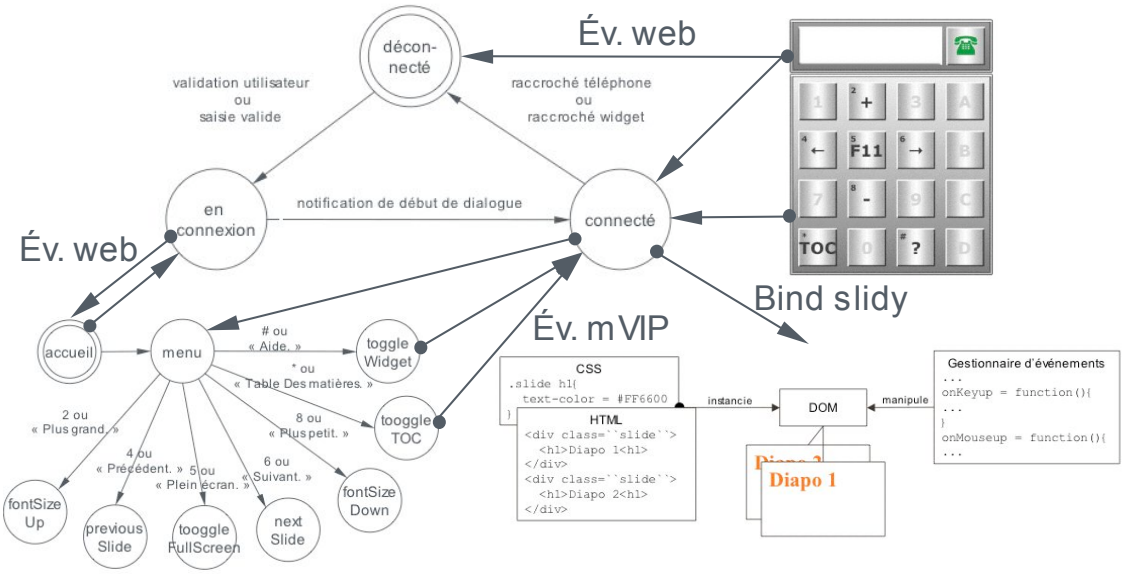


FIG. 4.14 – Organisation des composantes.

Snippet mSlidy Afin d'assurer une intégration rapide de notre savoir faire dans une présentation Slidy, nous avons défini un snippet qui permet d'un simple copier/coller, d'étendre une présentation existante. Son fonctionnement est simple. Cette modification du code source de la présentation provoque le chargement de la feuille de style *widget.css* qui permet de surcharger

les attributs de style du DOM HTML. Les fichiers *prototype.js* et *utils.js* regroupent un ensemble de fonctions ECMA assurant la compatibilité des primitives sur plusieurs navigateurs. Le fichier *mvc.js* contiennent une implémentation ECMA du patron *observer/observed*. Le fichier *gateway.js* contient le prototype de notre Objet *Gateway*. C'est cet objet qui réalise le « bind » des primitives Slidy. Les fichiers *caller.js* et *numpad.js* contient les objets graphiques qui composent le widget mSlidy. Le fichier *mSlidyMain.js* contient le corps du programme (*mSlidyload()*). Le corps du programme instancie les nœuds du DOM HTML du navigateur qui viendront accueillir les composantes graphiques de nos objets. Il instancie les composantes de l'objet *Gateway*, réalise les liaisons entre les observateurs et les observés et déplace le focus du navigateur dans le champ de saisie du widget. Le corps de notre programme est appelé quand le corps DOM HTML est chargé, juste après l'initialisation du moteur slidy *(startup())*.

```
<!-- CSS import -->
<style type="text/css"> @import "../html/widget.css"; </style>
<!-- include objects framework facilities -->
<script type="text/javascript" src="../js/prototype.js"></script>
<script type="text/javascript" src="../js/utils.js"></script>
<script type="text/javascript" src="../js/mvc.js"></script>
<script type="text/javascript" src="../js/gateway.js"></script>
<script type="text/javascript" src="../js/caller.js"></script>
<script type="text/javascript" src="../js/numpad.js"></script>
<!-- run mSlidy widget -->
<script type="text/javascript" src="../js/mSlidyMain.js"></script>
...
<body onload="startup() ; mSlidyload()">
```

FIG. 4.15 – Snippet mSlidy. Le snippet est composé de trois parties : la première partie décrit les élément de style du widget ; la seconde permet le chargement des bibliothèques utilisées par l'outil mVIP ; la dernière initie l'outil et lui donne son comportement. Cette initiation (*mSlidyload()*) a lieu à la fin du chargement du document HTML (*onload()*), juste après le chargement l'initialisation de Slidy (*startup()*).

Apports du démonstrateur mSlidy Le travail réalisé dans le cadre de l'extension de l'application web Slidy révèle la définition d'un outil réutilisable exploitant les fonctionnalités de la plate-forme mVIP. Cette outil fondé sur des technologies normalisées permet l'extension de la plupart des applications web existantes. Il fournit au concepteur de services un ensemble de fonctionnalités permettant de penser les interfaces web de manière distribuée. Il repose sur le patron VoiceXML *menu* et exploite ses capacités d'interaction. Les sources impliquées dans la réalisation de cette extension sont disponibles en annexe du document.

Applications mVIP Les trois mises en œuvre de notre modèle UbiArch exploitent les fonctionnalités de la plate-forme mVIP. Les services déployés sont donnés à titre d'illustrations. Ce qu'il faut retenir de ces réalisations est l'apport majeur que constituent les services mVIP pour les acteurs du domaine. Le service mVIP constitue une réponse pragmatique à un problème existant. mVIP est une solution web permettant :

- le déploiement massif des nouvelles technologies et de la VoIP[112] ;
- l'accès multimodal aux interfaces web ;
- l'intégration des fonctionnalités téléphoniques dans les applications web existantes.

[112] mBioSign - application web permettant l'utilisation de la biométrie vocale dans les pages web.

Conclusion

Bilan

Au cours de ce mémoire, nous avons pu observer les indicateurs du déploiement des services de téléphonie. Nous avons constaté l'émergence de nouveaux usages, liés à l'évolution des terminaux communicants et à la couverture du réseau de communication. Si les utilisateurs accèdent à un panel de services riche sur internet, la mobilité limite le pouvoir d'expression des interfaces web traditionnelles. En revanche le mode vocal semble plus adapté au contexte de mobilité. Nous avons donc cherché à coupler ces modalités.

Pour ce faire, nous avons identifié les modèles qui sous-tendent l'interaction dans deux situations et nous les avons combinés. Nous avons défini une architecture de services permettant le déploiement de cette combinaison. Nous avons implémenté les protocoles permettant à l'infrastructure d'offrir ces nouveaux services et les langages permettant au concepteur d'application web d'intégrer ces fonctionnalités. Nous avons d'ailleurs mis en œuvre des démonstrateurs laissant entrevoir de nouvelles possibilités d'interaction jusqu'alors inexploitées.

La mise en œuvre des modèles de la littérature nous a permis de définir des applications web aux interfaces évoluées. Ces applications permettent d'accéder à des services de messagerie, de bureautique ou d'information. Notre modèle d'architecture étend les capacités d'interaction des interfaces traditionnelles. Il permet un contrôle vocal du service mais aussi un accés distribué au service. L'utilisateur ira chercher dans son environnement les périphériques qu'il désire activer. Mieux, l'environnement s'adapte au cas d'usage. Notre approche ouvre de nouvelles perpectives en terme d'environnement communicant.

Notre plate-forme, mVIP, offre au concepteur de services un espace lui permettant d'exploiter des outils interactifs prêts à être intégrés dans les interfaces web traditionnelles. Ces outils intègrent la voix comme modalité de contrôle, d'authentification, de présentation. Cet espace lui assure un prototypage rapide de services innovants grâce à une bibliothèque de composants interactifs et de dialogues évolués. Il peut lui même étendre ces outils et enrichir la bibliothèque. L'architecture 3-tiers de la plate-forme assure la confidentialité des données cibles et la mutualisation des équipements.

Notre démarche de recherche, fut guidée par des intérêts industriels, nous conduisant à inscrire notre travail dans une optique d'exploitation. Pour cette raison, nous avons utilisé les briques logicielles du centre de recherche de France Telecom. Leurs développements étaient déjà inscrits dans un processus d'industrialisation impliquant une tierce maintenance et donnant lieu à des offres commerciales. Nous avons étendu ces organes afin de les intégrer dans notre architecture. Cette extension fut réalisée dans le respect des standards du domaine afin d'assurer le

passage à l'échelle, anticipant les évolutions futures de la technologie. Les extensions que nous avons proposées intègrent aujourd'hui les feuilles de route de la plate-forme du groupe sur ses aspects multimodaux.

Cependant, mVIP continue d'être une plate-forme expérimentale, une seconde thèse CIFRE a débuté en novembre 2006. Cette thèse poursuivra les travaux de normalisation initiés, elle verra la commercialisation des premiers services mVIP et continuera la démarche pédagogique entreprise auprès des développeurs d'applications web, des concepteurs de services. Les premières réalisations de cette thèse conduisent à implémenter un « snippet » pour le moteur de wiki *WikiMedia*. Ce snippet permet l'ajout d'un composant interactif permettant le dépôt d'un message vocal sur les pages du wiki. Ce service ubiquitaire permet la réalisation et la lecture de documents composites intégrant des séquences audio et du texte. Le but de cette réalisation et de mener à bien une expérimentation sur un panel d'utilisateurs au cours du second semestre 2007 et de proposer une offre commerciale au premier semestre 2008. La version bêta de ce service sera présentée au salon de la recherche Hiver 2007-2008 de France Telecom.

Limites

Nous avons intégré la présence de l'utilisateur sur le réseau dans notre modèle d'architecture. Cette prise en compte de la répartition du service permet un ajustement des coûts, une répartition de la charge ainsi qu'un retour quantitatif et qualitatif d'informations aux exploitants de l'infrastructure, aux fournisseurs et aux concepteurs de services et à l'utilisateur. L'interface d'administration de notre plate-forme nécessite une connaissance avancée de l'outil informatique. Si dans un contexte de recherche et d'expérimentation le traitement des données peut s'effectuer de manière spécifique, un passage à l'échelle impose la définition d'outils génériques adaptés aux exploitants de l'infrastructure. L'industrialisation de notre technologie nécessite la réalisation de cette tâche. De la même manière, si le déploiement de services est simplifié par l'utilisation de snippets, son développement nécessite tout de même des connaissances avancées en programmation web (Serveur web Apache, HTML, JavaScript, CGI). L'existence d'outils graphiques permettant la génération de code VoiceXML [Ould Ahmed Limam, 2003] laisse à penser qu'à moindre coût, les ergonomes et les designers pourront eux même étendre la bibliothèque de services.

Nous évoquions la construction d'un maillage de flux média. Dans ce mémoire, nous avons surtout fait référence aux flux audio. Lors de nos travaux, nous avons pu manipuler des flux multimédia synchronisant une séquence vidéo et une (ou plusieurs) séquences audio. La manipulation de flux multimédia implique la prise en compte des spécificités de ces flux. En effet ces flux multimédia sont synchronisés, volumineux et leur conversion est coûteuse. La synchronisation distribuée des flux nécessite l'intégration d'une composante de synchronisation dans l'architecture ou la définition d'une base de temps distribuée dans le réseau. Le transport de flux multimédia nécessite un dimensionnement adéquat de l'infrastructure et en particulier de la bande passante. Notons que des solutions telles qu'IPv6[113] inscrivent dans leur fondement la multi-diffusion et donc la factorisation du flux média par domaine et sous domaine.

[113]IPv6 - successeur du protocole IPv4, il permet entre autre de fournir un plus grand nombre d'adresses.

Perspectives

Les expérimentations tels que *Timéo* ou l'audio guide *D-Day* mis en place pour les cérémonies commémoratives du débarquement ont permis de valider l'architecture PMX et de susciter, chez les utilisateurs et les fournisseurs de services un vif intérêt pour les technologies vocales. Le travail accompli depuis sur la multimodalité et les premières présentations de nos démonstrateurs ubiquitaires ont reçu un accueil enthousiaste de nos partenaires. Les expérimentations prochaines, à plus large échelle nous permettrons de lancer les premiers services intégrant notre technologie et peut être d'observer l'émergence de nouveaux usages.

Les plates-formes d'interrogation en langage naturelle nécessitent une modélisation fine du domaine que l'on cherche à traiter [Sadek, 1991]. L'intégration du langage naturel dans nos services passe par la définition d'un protocole de communication entre les plate-formes de dialogue naturel et notre composante de dialogue. L'architecture 3-tiers mVIP permet de développer des stratégies de dialogue indépendantes de l'infrastructure de communication. Par conséquent, la plate-forme mVIP permet le déploiement à large échelle des technologies de langage naturel sur des interfaces multimodales. Nous étudions dès à présent l'intégration de ces nouvelles stratégies de dialogue dans nos services.

Si l'implémentation de notre modèle permet l'interaction entre un utilisateur et un service distribué, elle permet également de réunir plusieurs utilisateurs à travers le réseau dans une unique session web. Nous n'avons pas défini de cas d'usages illustrant cette fonctionnalité. Cependant, nous participons à un groupe de réflexion depuis septembre 2007 dont le but est de programmer, pour l'automne 2008, une manifestation sur le thème de la ville communicante. Dans ce cadre, nous pensons proposer des cas d'usage mettant en situation plusieurs utilisateurs. Les utilisateurs utiliseront leur téléphone mobile pour interagir avec des documents composites diffusés par du mobilier urbain.

Bibliographie

[Asakawa *et al.*, 2003] Asakawa, C., Takagi, H., Ino, S. et Ifukube, T. (2003). A proposal for a dial-based interface for voice output based on blind users' cognitive listening abilities. *In* ICS-Forth, C. S. et of Crete, U., éditeurs : *Tenth International Conference on Human-Computer Interaction*, volume 4, pages 1250–1254, London. Lawrence Erlbaum Associates.

[Bachelet, 2004] Bachelet, A. (2004). Etude des languages SALT et X+V. Rapport technique, France Telecom.

[Bai *et al.*, 2005a] Bai, A., Ferrieux, A., Rommel, B., Despax, G. et Markmann, F. (2005a). Transfert de la plate-forme vocale VIP. Rapport technique, France Telecom.

[Bai *et al.*, 2005b] Bai, A., Rommel, B., Despax, G. et Markmann, F. (2005b). La plate-forme vip, ses atouts, son transfert. Rapport technique, France Telecom.

[Basset, 2003] Basset, T. (2003). Monographie d'un logiciel libre : VideoLAN. Mémoire de D.E.A., IEP Paris.

[Bazin *et al.*, 2006] Bazin, C., Chuffart, F. et Madelaine, J. (2006). Construction d'une application vocale pour la sélection d'objets à l'aide d'un modèle basé sur les hypergraphes. *In 9éme Conférence Internationnale sur le Document Numérique*.

[Beaudouin-Lafon, 2004] Beaudouin-Lafon, M. (2004). Designing interaction, not interfaces. *In AVI '04 : Proceedings of the working conference on Advanced visual interfaces*, pages 15–22, New York, NY, USA. ACM Press.

[Bellik et Teil, 1992] Bellik, Y. et Teil, D. (1992). Définitions terminologiques pour la communication multimodale. *In 4ièmes Journées sur l'ingénierie des interfaces Homme-Machine*. IHM'92.

[Berge, 1987] Berge, C. (1987). *Hypergraphes. Combinatoires des ensembles finis*. Gauthier-Villars.

[Berube, 1991] Berube, L. (1991). *Terminologie de neuropsychologie et de neurologie du comportement*. Les Editions de la Chenelière.

[Bolt, 1980] Bolt, R. A. (1980). p̈ut-that-there¨ : Voice and gesture at the graphics interface. *In SIGGRAPH '80 : Proceedings of the 7th annual conference on Computer graphics and interactive techniques*, pages 262–270, New York, NY, USA. ACM Press.

[Bouchet et Nigay, 2004] Bouchet, J. et Nigay, L. (2004). Icare : a component-based approach for the design and development of multimodal interfaces. *In CHI '04 : CHI '04 extended abstracts on Human factors in computing systems*, pages 1325–1328, New York, NY, USA. ACM Press.

[Breton *et al.*, 2006] Breton, G., Pelé, D. et Garcia, C. (2006). Modeling gaze behavior for a 3d eca in a dialogue situation. *In IUI '06 : Proceedings of the 11th international conference on Intelligent user interfaces*, pages 333–335, New York, NY, USA. ACM Press.

[Buisson et Jestin, 2001] Buisson, M. et Jestin, Y. (2001). Design issues in distributed interaction supporting tools : mobile devices in an ATC working position. *In Proceedings of Mobile HCI 2001 : 3rd Int. Workshop on IHM-HCI 2001*, Lille, France.

[Cabal, 2003] Cabal, C. (2003). Office parlementaire d'évaluation des choix scientifiques et technologiques - rapport sur les méthodes scientifiques d'identification des personnes à partir de données biométriques et les techniques mises en œuvres. Enregistré à la présidence de l'Assemblée nationale le 16 juin 2003.

[Caelen et Villaseñor-Pineda, 1997] Caelen, J. et Villaseñor-Pineda, L. (1997). *Dialogue Homme-Machine et Apprentissage*. Europia Productions, Paris, France.

[Catoire *et al.*, 2006] Catoire, C., Lepetit, O. et Dipanda, A. (2006). étude de la caractérisation de la main par son réseau veineux. Rapport technique, France Telecom.

[Chevrin, 2006] Chevrin, V. (2006). *L'Interaction Usagers/Services, multimodale et multicanale : une première proposition appliquée au domaine du e-Commerce*. Thèse de doctorat, Thèse de doctorat en informatique Pr Alain Derycke CUEEP, Université de Lille1 Directeur de thèse, Dr José Rouillard CUEEP, Université de Lille1 Co-directeur de thèse. PDF : `http://noce.univ-lille1.fr/cms/uploaddocs/These_finale_11_05_2006.pdf`.

[Chuffart *et al.*, 2005] Chuffart, F., Van Gool, F. et Courval, L. (2005). Aurora, a framework enabling multimodal interactions. *In International Workshop on Multimodal Multiparty Meeting Processing*.

[Chuffart *et al.*, 2006] Chuffart, F., Van Gool, F. et Courval, L. (2006). Aurora, multimodal messaging framework for ubiquitous web context. *In W3C Ubiquitous Web Workshop*.

[Cotto, 1992] Cotto, D. (1992). *Traitement automatique des textes en vue de la synthèse vocale*. Thèse de doctorat, Université Paul Sabatier de Toulouse III.

[Coutaz, 1987] Coutaz, J. (1987). PAC, an implementation model for dialog design. *In Interact'87*.

[Coutaz, 1997] Coutaz, J. (1997). PAC-ing the architecture of your user interface. *In* Harrison, M. D. et Torres, J. C., éditeurs : *Design, Specification and Verification of Interactive Systems '97*, pages 13–27, Wien. Springer-Verlag.

[Coutaz, 1998] Coutaz, J. (1998). le futur ne manque pas d'avenir. *In ERGO-IA'98*. ESTIA/ILS.

[Coutaz et Caelen, 1991] Coutaz, J. et Caelen, J. (1991). A taxonomy for multimedia and multimodal user interface.

[Coutaz *et al.*, 1995] Coutaz, J., Nigay, L., Salber, D., Blandford, A., May, J. et Young, R. M. (1995). Four easy pieces for assessing the usability of multimodal interaction : The care properties. *In InterAct*, pages 115–120.

[Coutrix et Nigay, 2006] Coutrix, C. et Nigay, L. (2006). Mixed reality : a model of mixed interaction. *In AVI '06 : Proceedings of the working conference on Advanced visual interfaces*, pages 43–50, New York, NY, USA. ACM Press.

[David Thevenin, 2000] David Thevenin, Gaëlle Calvary, J. C. (2000). La multimodalité en plasticité. *In colloque sur les interfaces multimodales*, Grenoble.

[Depaulis *et al.*, 2006] Depaulis, F., Jambon, F., Girard, P. et Guittet, L. (2006). Le modèle d'architecture logicielle h4 : Principes, usages, outils et retours d'expérience dans les applications de conception technique. *Revue d'Interaction Homme-Machine*, 7 (1):93–129.

[Doublet *et al.*, 2006] Doublet, J., Lepetit, O. et Revenu, M. (2006). Contact less hand recognition using shape and texture features. *In 8th International Conference on Signal Processing*.

[Ferrieux *et al.*, 2002] Ferrieux, A., Ferrieux, L. et Monne, J. (2002). Protocole de communication entre un module d'apllication vocale et une plate-forme dans un serveur vocal. brevet FR 02/06475.

[Gamma *et al.*, 1995] Gamma, E., Helm, R., Johnson, R. et Vlissides, J. (1995). *Design patterns : elements of reusable object-oriented software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[Georgescu et Christopher, 2005] Georgescu, S.-M. et Christopher, J. (2005). Multimodal ims services : The adaptive keyword spotting interaction paradigm. *In* [ICAS, 2005], page 21.

[Haton *et al.*, 1991] Haton, J.-P., Pierrel, J.-M., Perennou, G., Caelen, J. et Gauvain, J.-L. (1991). *Reconnaissance automatique de la parole*. Dunod Informatique,Paris.

[Horchani *et al.*, 2007] Horchani, M., Nigay, L. et Panaget, F. (2007). A platform for output dialogic strategies in natural multimodal dialogue systems. *In IUI '07 : Proceedings of the 12th international conference on Intelligent user interfaces*, pages 206–215, New York, NY, USA. ACM.

[ICAS, 2005] ICAS (2005). *Joint International Conference on Autonomic and Autonomous Systems 2005 / International Conference on Networking and Services 2005 (ICAS/ICNS 2005), 23-28 October 2005, Papeete, Tahiti*. IEEE Computer Society.

[Jain *et al.*, 2004] Jain, A. K., Ross, A. et Prabhakar, S. (2004). An introduction to biometric recognition. *Circuits and Systems for Video Technology, IEEE Transactions on*, 14(1):4–20.

[José, 2004] José, R. (2004). *VoiceXML. Le langage d'accès à Internet par téléphone*. aux éditions Vuibert, ISBN : 271174826X, 197 pages, Paris, 2004.

[Kranz *et al.*, 2006] Kranz, M., Holleis, P. et Schmidt, A. (2006). Ubiquitous presence systems. *In SAC '06 : Proceedings of the 2006 ACM symposium on Applied computing*, pages 1902–1909, New York, NY, USA. ACM Press.

[Krasner et Pope, 1988] Krasner, G. E. et Pope, S. T. (1988). A cookbook for using the model-view controller user interface paradigm in smalltalk-80. *J. Object Oriented Program.*, 1(3):26–49.

[Lachenal et Coutaz, 2003] Lachenal, C. et Coutaz, J. (2003). Introspac : a tool to teach and understand pac-amodeus. *In IHM 2003 : Proceedings of the 15th French-speaking conference on human-computer interaction on 15eme Conference Francophone sur l'Interaction Homme-Machine*, pages 212–215, New York, NY, USA. ACM Press.

[Laurillau, 2002] Laurillau, Y. (2002). *Conception et réalisation logicielles pour les collecticiels centrés sur l'activité de groupe : le modèle et la plate-forme Clover*. Thèse de doctorat, IIHM - Université Joseph Fourier.

[Lebert, 2001] Lebert, M. (2001). Le livre 010101. `http://www.etudes-francaises.net/entretiens/00livre.htm`.

[Lécuyer *et al.*, 2004] Lécuyer, A., Burkhardt, J.-M. et Etienne, L. (2004). Feeling bumps and holes without a haptic interface : the perception of pseudo-haptic textures. *In CHI '04 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 239–246, New York, NY, USA. ACM Press.

[Loisel *et al.*, 2005] Loisel, A., Chaignaud, N. et Kotowicz, J.-P. (2005). Un agent conversationnel pour l'amelioration de requtes dans le cadre d'une recherche documentaire. *In Workshop Francophone sur les Agents Conversationnels Animes*, pages 149–150, Grenoble, France.

[Loqué *et al.*, 2004] Loqué, N., Courval, L. et Marie, T. (2004). VOLCAN : un nouveau service de vocalisation des horaires du tramway de Caen. Rapport technique, France Telecom.

[Millán *et al.*, 2007] Millán, J., Ferrez, P. et Buttfield, A. (2007). The idiap brain-computer interface : An asynchronous multi-class approach. *In* Dornhege, G., d. R. Millán, J., Hinterberger, T., McFarland, D. et Müller", K.-R., éditeurs : *Towards Brain-Computer Interfacing*. The MIT Press.

[Nassar *et al.*, 2006] Nassar, M., State, R. et Festor, O. (2006). Intrusion detection mechanisms for voip applications. *In Third annual VoIP security workshop - VSW'06*.

[Nigay, 1994] Nigay, L. (1994). *Conception et modélisation logicielles des systêmes interactifs : application aux interfaces multimodales*. Thèse de doctorat, Université Joseph Fourier - Grenoble I.

[Nigay, 2006] Nigay, L. (2006). Architecture Logicielle des Systèmes Interactifs. *In* Akoka, J. et Comyn-Wattiau, I., éditeurs : *Encyclopédie de l'informatique et des systèmes*, chapitre II/3, pages 294–309. Vuibert, Paris.

[Nigay et Coutaz, 1993] Nigay, L. et Coutaz, J. (1993). A design space for multimodal systems : concurrent processing and data fusion. *In CHI '93 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 172–178, New York, NY, USA. ACM Press.

[Nigay et Coutaz, 1997] Nigay, L. et Coutaz, J. (1997). Multifeature systems : The care properties and their impact on software design. *In* Lee, J., éditeur : *Intelligence and Multimodality in Multimedia Interfaces : Research and Applications*. AAAI Press.

[Ochs *et al.*, 2005] Ochs, M., Sadek, D. et Pelachaud, C. (2005). La représentation des émotions d'un agent rationnel. *In Workshop Francophone sur les Agents Conversationnels Animés*, pages 43–52, Grenoble, France.

[Ould Ahmed Limam, 2003] Ould Ahmed Limam, M. (2003). *Interaction avec Feedback Sémantique dan sun environnement dédié à la recherche d'informations géographiques*. Thèse de doctorat, GREYC, UMR CNRS 6072, Université de Caen, France, Caen.

[Oviatt, 1999] Oviatt, S. (1999). Ten myths of multimodal interaction. *Communications of the ACM*, 42(11):74–81.

[Pearce *et al.*, 2000] Pearce, D., Kopp, D. et Wajda, W. (2000). The impact of distributed speech recognition on multi-modal interfaces to the mobile web. Position Papers. W3C/WAP Workshop : the Multimodal Web.

[Pédauque et Sèdes, 2007] Pédauque, R. et Sèdes, F. (2007). Les temps du document numérique. *In* Pédauque, R., éditeur : *La Redocumentarisation du Monde*, pages 103–134. Cépaduès Editions, http ://www.cepadues.com/.

[Pfaff, 1985] Pfaff, G. E., éditeur (1985). *User Interface Management Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[Projet de loi sur l'égalité des droits, 2005] Projet de loi sur l'égalité des droits (2005). pour l'égalité des droits et des chances, la participation et la citoyenneté des personnes handicapées. adopté par l'Assemblée nationale le 18 janvier 2005 en deuxième lecture.

[Pédauque, 2006] Pédauque, R. (2006). *Le document à la lumière du numérique*. CF éditions, http ://cfeditions.com.

[Quaedvlieg et de Wildt, 2006] Quaedvlieg, C. et de Wildt, T. (2006). Arm gesture analysis for direct television control : simple versus complex movements. Rapport technique, Philips Consumer Electronics,.

[Quaedvlieg *et al.*, 2006] Quaedvlieg, C., de Wildt, T. et Willems, D. (2006). Recognizing sensory-based 3d arm gesture trajectory using a combined classifier. Rapport technique, Philips Consumer Electronics, Radboud University Nijmegen.

[Sabouret, 2002] Sabouret, N. (2002). *Étude de modèles de représentation, de requêtes et de raisonnement sur le fonctionnement des composants actifs pour l'interaction homme-machine*. Thèse de doctorat, Université Paris XI Orsay.

[Sadek, 1991] Sadek, D. (1991). *Attitudes mentales et interaction rationnelle : vers une théorie formelle de la communication*. Thèse de doctorat, Université Rennes I.

[Salaün, 2006] Salaün, J.-M. (2006). Téléphone, bande passante et modèle de médiarendez-vous. `http://blogues.ebsi.umontreal.ca/jms/index.php/2006/12/16/139-telephone-bande-passante-et-modele-de-media`.

[SIGCHI, 1992] SIGCHI (1992). A metamodel for the runtime architecture of an interactive system : the UIMS tool developers workshop. *SIGCHI Bull.*, 24(1):32–37.

[Truillet *et al.*, 2000] Truillet, P., Oriola, B. et Vigouroux, N. (2000). Présentation multimodale de documents électroniques structurés. *In colloque sur les interfaces multimodales*, Grenoble, France.

[Turbout, 2002] Turbout, C. (2002). *Construction d'hypertexte et recherche d'informations hétérogènes : la spécificité de l'information géographique*. Thèse de doctorat, GREYC, UMR CNRS 6072, Université de Caen, France.

[Van Gool, 2004] Van Gool, F. (2004). Position paper intesi on w3c MMI workshop. *In W3C Workshop on Multimodal Interaction*.

[Vernant, 1992] Vernant, D. (1992). Modèle projectif et structure actionnelle du dialogue informatif. *Du Dialogue - Recherches sur la philosophie du langage*, pages 295–314.

[Villaseñor-Pineda, 1999] Villaseñor-Pineda, L. (1999). *Contribution à l'apprentissage dans le dialogue homme-machine*. Thèse de doctorat, Université Joseph-Fourier, Grenoble I.

Webographie

[1] Firefox, 1998-2007. http://www.mozilla-europe.org/fr/.

[2] Microsoft Unveils Road Map for Speech Server 2007, April 2006. http://www.microsoft.com/presspass/press/2006/apr06/04-05MSS07BetaPR.mspx.

[3] Adobe Flash Player, 2007. http://www.adobe.com/fr/products/flashplayer/.

[4] Adode SVG Viewer, 2007. http://www.adobe.com/svg/.

[5] Apple Quicktime Player, 2007. http://www.apple.com/quicktime/.

[6] Biométrie Online, un espace d'information, de rencontre, de dialogue, de réflexion le lien entre tous les acteurs de la biométrie et les utilisateurs, 2007. http://www.biometrie-online.net/.

[7] Microsoft Windows Media Player, 2007. http://www.microsoft.com/windows/windowsmedia/fr/.

[8] 118710, service de renseignements automatisé, 2005. http://www.francetelecom.com/fr/groupe/rd/une/118710/.

[9] ARCEP, Autorité de Régulation des Communications Électroniques et des Postes, 1997-2007. http://www.art-telecom.fr/.

[10] Asterisk. Asterisk, the open source IP PBX, 2007. http://www.asterisk.org/.

[11] Florent Chuffart. Démontrateurs web autour de l'ubiquité, 2007. http://users.info.unicaen.fr/~fchuffar/demos/.

[12] CSS Zen garden, la beauté de la conception css. http://www.csszengarden.com/.

[13] eSpeak, text to speech solution, 2007. http://espeak.sourceforge.net/.

[14] EUREKA a network for market oriented research and development, 2007. http://www.eureka.be/.

[15] OpenInterface Foundation. OpenInterface Platform, 2007. http://www.openinterface.org/.

[16] The IDIAP research institute, 1991 - 2007. http://www.idiap.ch/.

[17] The Internet Engineering Task Force, 1992-2007. http://www.ietf.org/.

[18] IIDRIS, index international et dictionnaire de la réadaptation et de l'intégration sociale, 2004. http://www.med.univ-rennes1.fr/iidris/.

[19] ITEA 2 - home, 2006 - 2007. http://www.itea2.org/.

[20] ITEA-Aurora Project, 2004-2006. http://itea-aurora.org.

[21] International Telecommunication Union, 1992-2007. http://www.itu.int/.

[22] Synthèses vocales, 2002. http://www.compuzik.com/catalogue/syntheses_vocales/kali.php.

[23] Anatole Lécuyer, Laurent Etienne adn Bruno Arnaldi, and Jean-Marie Burkhardt. "images tactiles" : ressentir le relief des images, 2004. http://www.irisa.fr/tactiles/.

[24] Mozilla Developer Center. About JavaScript, 2007. http://developer.mozilla.org/en/docs/About_JavaScript.

[25] Open Mobile Alliance, 2007. http://www.openmobilealliance.net/.

[26] OpenLDAP, open source implementation of Lightweight Directory Access Protocol, 2007. http://www.openldap.org/.

[27] OpenSIP Project, 2000-2007. http://www.opensip.org/.

[28] Jörk Ott. SIP Conferencing, 2001. http://www.cs.columbia.edu/sip/talks/sip-conferencing.pdf.

[29] David Pearce. Distributed speech recognition. W3C presentation, 2005. http://www.w3.org/2005/05/DSR.pdf.

[30] The Model View Controller Framework for php Applications, 2007. http://www.phpmvc.net.com/.

[31] Radvison, Video Conferencing Solutions and Developer Tool kits, 2007. http://www.radvision.com/.

[32] Dave Raggett. HTML Slidy : Slide Shows in XHTML, 2005. http://www.w3.org/Talks/Tools/Slidy/.

[33] Dave Raggett. The Ubiquitous Web. Multimodal Web Application for Embedded Systems, Juin 2005. http://www.w3.org/2005/Talks/0621-dsr-ubiweb.

[34] Dave Raggett. Google Techtalks, Web Applications and the Ubiquitous Web, February 2006. http://video.google.com/videoplay?docid=8950294834635667990.

[35] RSI, Rendez-vous des Systèmes d'Information, Solution Internet, et Stratégies Innovantes, 2000 - 2007. http://www.rsinormandie.com/.

[36] SALT Forum. Speech Application Language Tags (SALT) Forum, 2007. http://www.saltforum.org.

[37] Apache Struts project, 2007. http://struts.apache.org/.

[38] Roland Trique. Le jargon français, 1995-2006. http://www.linux-france.org/prj/jargonf/.

[39] VideoLAN. VLC media player, 1995-2007. http://www.videolan.org/.

[40] Vovida.org, Your Source for Open Source Communication, 2003. http://www.vovida.org/.

[41] The World Wide Web Consortium, 1994-2007. http://www.w3.org/.

[42] W3C. Voice browser activity, 1995-2007. http://www.w3.org/Voice/.

[43] W3C. Multimodal interaction framework, May 2003.

[44] Ubiquitous web domain, 2007. http://www.w3.org/UbiWeb/.

[45] W3C. Web Accessibility Initiative, 2007. http://www.w3.org/WAI/.

[46] Wikipedia, The Free Encyclopedia, 2001-2007. http://www.wikipedia.org/.

RFCs, recommandations et spécifications

[Draft CCXML, 2007] Draft CCXML (2007). Voice Browser Call Control : CCXML Version 1.0. http://www.w3.org/TR/ccxml/.

[Draft SCXML, 2005] Draft SCXML (2005). State Chart XML (SCXML) : State Machine Notation for Control Abstraction 1.0. http://www.w3.org/TR/scxml/.

[Note X+V, 2001] Note X+V (2001). XHTML+Voice Profile 1.0. http://www.w3.org/TR/xhtml+voice/.

[Rec. CSS, 2006] Rec. CSS (2006). Cascading Style Sheets, level 2 revision 1- CSS2.1 Specification. http://www.w3.org/TR/CSS21.

[Rec. DOM, 1998] Rec. DOM (1998). Document Object Model (DOM) Level 3 Core Specification. http://www.w3.org/DOM/.

[Rec. HTML, 1999] Rec. HTML (1999). HTML 4.01 Specification. http://www.w3.org/TR/html401.

[Rec. SRGS, 2004] Rec. SRGS (2004). Speech Recognition Grammar Specification Version 1.0. http://www.w3.org/TR/speech-grammar/.

[Rec. SSML, 2004] Rec. SSML (2004). Speech Synthesis Markup Language (SSML) version 1.0. http://www.w3.org/TR/speech-synthesis/.

[Rec. SVG, 2003] Rec. SVG (2003). Scalable Vector Graphics (SVG) 1.1 Specification. http://www.w3.org/TR/SVG/.

[Rec. URI URL URN, 2001] Rec. URI URL URN (2001). URIs, URLs, and URNs : Clarifications and Recommendations 1.0. http://www.w3.org/TR/uri-clarification/.

[Rec. VXML, 2004] Rec. VXML (2004). Voice Extensible Markup Language (VoiceXML) Version 2.0. http://www.w3.org/TR/voicexml20/.

[Rec. XHTML, 2002] Rec. XHTML (2002). XHTML 1.0 The Extensible Hypertext Markup Language (Second Edition). http://www.w3.org/TR/xhtml1.

[Rec. XML, 2006] Rec. XML (2006). Extensible Markup Language (XML) 1.0 (Fourth Edition). http://www.w3.org/TR/2006/REC-xml-20060816.

[Rec. XML event, 2007] Rec. XML event (2007). XML Events 2, An Events Syntax for XML. http://www.w3.org/TR/xml-events/.

[RFC 1122, 1989] RFC 1122 (1989). Requirements for Internet Hosts – Communication Layers. http://www.ietf.org/rfc/rfc1122.txt.

[RFC 1738, 1994] RFC 1738 (1994). Uniform Resource Locators (URL). http://www.ietf.org/rfc/rfc1738.txt.

[RFC 1889, 1996] RFC 1889 (1996). RTP : A Transport Protocol for Real-Time Applications. http://www.ietf.org/rfc/rfc1889.txt.

[RFC 1945, 1996] RFC 1945 (1996). Hypertext Transfer Protocol - HTTP/1.0. http://www.w3.org/Protocols/rfc1945/rfc1945.

[RFC 2068, 1997] RFC 2068 (1997). Hypertext Transfer Protocol – HTTP/1.1. http://www.ietf.org/rfc/rfc2068.txt.

[RFC 2326, 1998] RFC 2326 (1998). Real Time Streaming Protocol (RTSP). http://www.ietf.org/rfc/rfc2326.txt.

[RFC 2396, 1998] RFC 2396 (1998). Uniform Resource Identifiers (URI) : Generic Syntax. http://www.ietf.org/rfc/rfc2396.txt.

[RFC 2543, 1999] RFC 2543 (1999). SIP : Session Initiation Protoco. http://www.ietf.org/rfc/rfc2543.txt.

[RFC 2616, 1999] RFC 2616 (1999). Hypertext Transfer Protocol – HTTP/1.1. http://www.ietf.org/rfc/rfc2616.txt.

[RFC 3261, 2002] RFC 3261 (2002). SIP : Session Initiation Protocol. http://www.ietf.org/rfc/rfc3261.txt.

[RFC 3265, 2002] RFC 3265 (2002). Session Initiation Protocol (SIP)-Specific Event Notification. http://www.ietf.org/rfc/rfc3265.txt.

[RFC 3489, 2003] RFC 3489 (2003). STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (nats). http://www.ietf.org/rfc/rfc3489.txt.

[RFC 3550, 2003] RFC 3550 (2003). RTP : A Transport Protocol for Real-Time Applications. http://www.rfc-editor.org/rfc/rfc3550.txt.

[RFC 4313, 2005] RFC 4313 (2005). Requirements for Distributed Control of Automatic Speech Recognition (ASR), Speaker Identification/Speaker Verification (SI/SV), and Text-to-Speech (TTS) Resources. http://tools.ietf.org/html/rfc4313.

[RFC 4329, 2006] RFC 4329 (2006). Scripting Media Types. http://www.ietf.org/rfc/rfc4329.txt.

[RFC 4463, 2006] RFC 4463 (2006). A Media Resource Control Protocol (MRCP) Developed by Cisco, Nuance, and Speechworks. http://www.ietf.org/rfc/rfc4463.txt.

[RFC 4510, 2006] RFC 4510 (2006). Lightweight Directory Access Protocol (LDAP) : Technical Specification Road Map. http://tools.ietf.org/html/rfc4510.

[RFC 4566, 2006] RFC 4566 (2006). SDP : Session Description Protocol. http://www.ietf.org/rfc/rfc4566.txt.

[Spec. ECMA, 1999] Spec. ECMA (1999). ECMAScript Language Specification, 3rd Edition. http://www.ecma-international.org/publications/files/ecma-st/ECMA-262.pdf.

[Spec. SALT, 2002] Spec. SALT (2002). Speech Application Language Tags (SALT) 1.0 Specification. http://www.saltforum.org/devforum/spec/SALT.1.0.a.asp.

[Spec. SpeechSC, 2005] Spec. SpeechSC (2005). Speech Services Control Charter. http://www.ietf.org/html.charters/speechsc-charter.html.

[Spec. X+V, 2004] Spec. X+V (2004). XHTML+Voice Profile 1.2. http://www.voicexml.org/specs/multimodal/x+v/12/.

[Spec.X+V mobile, 2005] Spec.X+V mobile (2005). Mobile X+V 1.2. http://www.voicexml.org/specs/multimodal/x+v/mobile/12/.

Glossaire

AJAX : Asynchronous JavaScript And XML.
ARCEP : Autorité de Régulation des Communications Électroniques et des Postes.
Bargein : interruption de l'énoncé vocal du système par l'utilisateur.)
CAO : Conception Assitée par Ordinateur
CARE : Complementarity, Assignment, Redundancy, Equivalence [Coutaz *et al.*, 1995, Nigay et Coutaz, 19
CCXML : Voice Browser Call Control XML.
Cepstrale : anagramme de spectral.
Cepstre : anagramme de spectre.
CGI : Common Gateway Interface.
CIFRE : Convention Industrielle de formation par la Recherche.
CLI : Command Line Interface.
CNIL : Commission Nationale de l'Informatique et des Libertés.
CSS : Cascading Style Sheets.
DOM : Document Object Model.
Domotique : automatisation des activités domestiques.
DSR : Distributed Speech Recognition.
DTD : Document Type Definition, syntaxe du document.
DTMF : Dual Tone Modulaltion Frequency, signalisation associée à chacune des touches du téléphone et véhiculée sur les lignes téléphoniques dans les fréquences audibles.
Démon UNIX : programme résident, UNIX daemon.
ENSI Caen : École Nationale Supérieure d'Ingénieurs de Caen.
EUREKA : organisme européen qui finance et coordonne des projets de recherche et de développement [14].
Framework : espace de travail modulaire, ensemble de bibliothèques, d'outils et de conventions permettant le développement rapide d'applications.
Gabarit : (biométrie) empreinte de référence, donnée à partir de laquelle les comparaisons futures seront effectuées
GPRS : General Packet Radio Service, téléphonie 2G, permet un accès internet bas débit mobile.
GREYC : Groupe de Recherche en Informatique, Image, Automatique et Instrumentation de Caen (UMR 6072).
GSM : Global System for Mobile communications, réseau de téléphonie mobile.
GUI : Graphic User Interface, aussi appelé WIMP pour Window Incon Menu Pointer.
Handle de session : valeur numérique identifiant un objet informatique et permettant donc sa « manipulation » ou sa gestion [38].
HTML : HyperTexte MetaLanguage [Rec. HTML, 1999].
HTML Slidy : *Slide Shows in XHTML*, dévellopé par Dave Raggett, fondé sur S5 de Eric Meyer, il utilise exclusivement des standards du web (XHTML, CSS et JavaScript) [32].

HTTP : HyperText Transfer Protocol.
IDIAP : l'Institut Dalle Molle d'Intelligence Artificielle Perceptive, est un centre de recherche semi-privé basé à Martigny (CH) [16].
IETF : Internet Engineering Task Force, groupe informel, international, ouvert à tout individu [17].
IntrosPAC : outil graphique permettant de voir transiter les messages de proche en proche de la hiérarchie PAC[Lachenal et Coutaz, 2003].
IP PBX : IP Private Branch eXchange, équivalent IP des centrales téléphoniques analogiques PABX.
IPv6 : successeur du protocole IPv4, il permet entre autre de fournir un plus grand nombre d'adresses.
ITEA : Information Technology for European Advancement [19].
ITEA Aurora : projet ITEA 03005 [20].
JSON : JavaScript Object Notation, est un format de structure de données générique. Il utilise la notation des objets JavaScript pour transmettre de l'information structurée. C'est le seul point commun qu'il a avec le langage JavaScript [46].
LATEMS : Laboratoire de Transactions Électroniques, de Monétique et de Sécurité.
LDAP : Lightweight Directory Access Protocol [RFC 4510, 2006].
LSD : Langage de spécifications des dialogues.
mATM : multimodal Automatic Teller Machine, Ditributeur de billets innovant [11].
mBioSign : application web permettant l'utilisation de la biométrie vocale dans les pages web.
mKiosque : Kiosque multimodal, Service de diffusion de contenus multimédia innovant [11].
MRCP : Media Resource Control Protocol [RFC 4463, 2006].
mSlidy : Extensioin multimodale de l'application web Slidy HTML [11].
MVC : Model View Controller, patron de conception permettant de modéliser le système comme un agent autonome interactif.
mVIP : VIP multimodale.
NFC : Near Field Communication, protocole permettant la lecture d'étiquettes interactives s'apparentant à des codes barre mais basé sur un technologie RFID (RFID - Radio-frequency identification), standard ISO depuis le 8 décembre 2003, poussé par le NFC-Forum.
OpenVXI : implémentation opensource d'un interpréte VoiceXML]
P2P : peer-to-peer, échange d'égal à égal à travers le réseau.
PAC : Présentation Abstraction Contrôle.
Pattern : concept destiné à résoudre les problèmes récurrents suivant le paradigme objet
PDA : Personnal Digital Assitant (Newton, Palm Pilot, Pocket PC), terminal mobile utilisant la métaphore du bureau et possédant éventuellement une connectivité réseau (GSM, WiFi) [Buisson et Jestin, 2001].
Plugin : en informatique, le terme anglais plugin (ou plug-in, se prononçant /plœgin/, du verbe to plug in qui signifie brancher), est employé pour désigner un programme qui inter-agit avec un logiciel principal, appelé programme hôte, pour lui apporter de nouvelles fonctionnalités [46].
PME : Petites et Moyennes Entreprises
PMX : Plate-forme Multimédia XML.
QoS : Quality of Service.
RFC : Request For Comments, peu de RFC sont des standards, mais tous les standards d'internet sont enregistrés en tant que RFC.
RFC : Request for Comments.

RFF : Réseau Ferré de France, organisme en charge de la gestion des infrastructures ferroviaires.
RIA : Rich Internet Applications.
RTC : Réseau Téléphonique Commuté.
RTCP : RTP Control Protocol[RFC 3550, 2003].
RTP : Real-time Transport Protocol [RFC 1889, 1996, RFC 3550, 2003].
RTSP : Real Time Streaming Protocol.
SALT : Speech Application Language Tags [Spec. SALT, 2002].
SAX : Simple API for XML.
SCXML : State Chart XML (SCXML) : State Machine Notation for Control Abstraction.
SDP : Session Description Protocol [RFC 4566, 2006].
SIP : Session Initiate Protocol [RFC 2543, 1999, RFC 3261, 2002].
Snippet : quelques lignes de code source réutilisables.
Somesthésie : (n.f.) conscience du corps [Berube, 1991]. source : [18].
SpeechSC : Speech Service Control, Version 2 de MRCP [Spec. SpeechSC, 2005].
SRGS : Speech Recognition Grammar Specification [Rec. SRGS, 2004].
SSML : Speech Synthesis Markup Language [Rec. SSML, 2004].
STUN : Simple Traversal of UDP through NATs.
SVG : Scalable Vector Graphics, SVG est un dialecte de XML permettant la descrition d'objet vectoriel [Rec. SVG, 2003].
T2 : autrement appelée PRI (Primary Rate Interface), une ligne T2 est une interface d'accès à un réseau ISDN.
TCP/IP : Transmission Control Protocol / Internet Protocol, séparation de la couche de transport et de la couche applicative du modèle OSI [RFC 1122, 1989].
TEE : Taux d'Égale Erreur
TFA : Taux de Faux Acceptés, acceptés à tord.
TFR : Taux de Faux Rejetés, rejetés à tord.
TGV : Très Grand Vocabulaire, modèle de reconnaissance vocale
Timéo : service d'accés vocal à l'informations voyageur du réseau de tramways et de bus Twisto (Caen).
Trackpad : dispositif physique permettant d'interagir avec le curseur de la souris en faisant glisser son doigt sur une surface sensible.
UA : User Agent.
UDP : User Datagram Protocol.
UMTS : Universal Mobile Telecommunications System, téléphonie 3G, permet un accès internet haut débit mobile.
URI : Uniform Resource Identifiers [RFC 2396, 1998, Rec. URI URL URN, 2001].
URL : Uniform Resource locators [RFC 1738, 1994].
VIP : Voice Interactive Protocol.
VoIP : Voice over IP, service de téléphonie utilisant le réseau internet.
W3C : World Wide Web Consortium, consortium fondé en octobre 1994 dont la gestion est assurée conjointement par le Massachusetts Institute of Technology (MIT) aux États-Unis, le European Research Consortium for Informatics and Mathematics (ERCIM) en Europe (auparavant l'Institut national de recherche en informatique et en automatique français (INRIA)) et l'Université Keio au Japon [41].
Web Service : le service web est un programme informatique permettant la communication et l'échange de données en utilisant le protocole de transport HTTP est un protocole de description adapté (JSON, XML).
Widget : composant d'interface graphique.

WIMP : Acronyme anglais pour « Windows, Icons, Menus and Pointing device », (fenêtres, icônes, menus et dispositif de pointage), le paradigme WIMP présente des bases fonctionnelles d'une interface graphique informatique.

WIMP : Windows Icon Menu Pointer, interface comprenant une interface graphique, un clavier, une souris sur un ordinateur de bureau, ou portable.

X+V : XTHML + VoiceXML.

XHTML : Extensible HyperText Markup Language.

XML : eXtensible Markup Language [Rec. XML, 2006].

Annexe A

Code source du widget mSlidy

Sommaire

A.1 Fichier widget.css

```
/*************************************************************************
*                                                                        *
*   Copyright (C) FRANCE TELECOM 1998-2007 - TOUS DROITS RESERVES        *
*                                                                        *
*                                                                        *
*   Avertissement : ce logiciel est protégé en France par le Code de la  *
*   Propriété Intellectuelle - Propriété Littéraire et Artistique - et   *
*   par les Conventions Internationales sur le droit d'auteur.           *
*                                                                        *
*   Toute reproduction, adaptation ou distribution partielle ou totale   *
*   du logiciel, autre qu'une seule copie de sauvegarde, par quelque     *
*   moyen que ce soit, est strictement interdite ainsi que pour la       *
*   documentation qui y est associée.                                    *
*                                                                        *
*   Toute personne ne respectant pas ces conditions se rendra coupable   *
*   de délit de contrefaçon et sera passible des sanctions pénales       *
*   prévues par la loi.                                                  *
*                                                                        *
*   L'APP  est mandatée  par l'auteur  pour faire  sanctionner toutes    *
*   copie et/ou utilisation non autorisées.                              *
*                                                                        *
*************************************************************************/

/*************************************************************************
* JB GUI interface stylesheet : objet style property and positionning
*/

/*************************************************************************/
/* Debugging*/
#divTimer{
  position:absolute;
  top:50px; left:215px;
  font-family:Bitstream Vera Sans, Verdana;
  font-size:15px;
  font-weight:bold;
  text-align:left;
  color:#FF6600;
  z-index : 300 ;
}
#divDebug{
  position:absolute;
  top:10px; left:10px;
  color:#FF0000;
  font-weight:bold;
  z-index : 300 ;
}
```

```
/*************************************************************************/
/* widget numpad style property*/
#divNumpad{
  position:absolute;
  top:100px; left:25px;
  z-index : 300 ;
}

#divCaller{
  position:absolute;
  top:35px; left:25px;
  width: 249px;
  height: 67px;
  background-image: url('../img/caller-bg.png');
  background-repeat: no-repeat;
  z-index : 300 ;
}

#divCaller input {
  border: 1px #333 dotted;
  background-color: #EEE;
  margin-top:1px;
  margin-bottom:1px;
  font-size: x-large;
}

#divCaller table, #divNumpad table {
  margin :  0px ;
}

.mask {
  display:none;
  position:absolute;
  top:0; left:0;
  width:100%;
  height:100%;
  background-image: url('../img/fond.gif');
  text-align:center;
}
```

A.2 Fichier index.html

```
<!-------------------------------------------------------------------------
*                                                                        *
*   Copyright (C) FRANCE TELECOM 1998-2007 - TOUS DROITS RESERVES        *
*                                                                        *
*                                                                        *
*   Avertissement : ce logiciel est protégé en France par le Code de la  *
*   Propriété Intellectuelle - Propriété Littéraire et Artistique - et   *
*   par les Conventions Internationales sur le droit d'auteur.           *
*                                                                        *
*   Toute reproduction, adaptation ou distribution partielle ou totale   *
*   du logiciel, autre qu'une seule copie de sauvegarde, par quelque     *
*   moyen que ce soit, est strictement interdite ainsi que pour la       *
*   documentation qui y est associée.                                    *
*                                                                        *
*   Toute personne ne respectant pas ces conditions se rendra coupable   *
*   de délit de contrefaçon et sera passible des sanctions pénales       *
*   prévues par la loi.                                                  *
*                                                                        *
*   L'APP  est mandatée  par l'auteur  pour faire  sanctionner toutes    *
*   copie et/ou utilisation non autorisées.                              *
*                                                                        *
-------------------------------------------------------------------------->

<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
  <head>
  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1"/>
    <!-- CSS import -->
    <style type="text/css"> @import "widget.css"; </style>
    <!-- include objects framework facilites-->
    <script type="text/javascript" src="../js/prototype.js"></script>
    <script type="text/javascript" src="../js/utils.js"></script>
    <script type="text/javascript">
/*
 * Include
 */
include('../js/mvc.js'); include('../js/gateway.js'); include('../js/button.js');
include('../js/box.js'); include('../js/jbNumpad.js'); include('../js/caller.js');

/*
 * Glogals
 */
var BROWSER ; var TIMER_DELAY=1000; var CALL = false ;
var DEBUG='true' ; var DEBUG_LINE=0 ; DEBUG_MAX = function () { return 30; }
```

```
/*
 * Main
 */
function mVIPload(){
 /*
  * DOM extension
  */
  var node = document.createElement('div') ;
  node.setAttribute('id', 'divCaller');
  document.getElementsByTagName('body')[0].appendChild(node);
  var node0 = document.createElement('div') ;
  node0.setAttribute('id', 'divNumpad');
  document.getElementsByTagName('body')[0].appendChild(node0);
  var node1 = document.createElement('div') ;
  node1.setAttribute('id', 'divTimer');
  document.getElementsByTagName('body')[0].appendChild(node1);
  var node2 = document.createElement('div') ;
  node2.setAttribute('id', 'divDebug');
  document.getElementsByTagName('body')[0].appendChild(node2);
  debug("[MAIN]");
  debug("[detectBrowser] " + (BROWSER = detectBrowser() ) );
 /*
  * object creation
  */
  var _gateway = new Gateway();
  var _numpad = new jbNumpad(getById('divNumpad'));
  var _caller = new Caller(getById('divCaller'));
 /*
  * Controller View linking
  */
  _caller.controller.addListener(_gateway);
  _numpad.controller.addListener(_gateway);
  _gateway.controller.addListener(_numpad);
  _gateway.controller.addListener(_caller);
  _gateway.controller.addListener(_gateway);
 /*
  * Start Intercaton
  */
  document.CallForm.PhoneNumber.focus() ;
  debug("[MAIN] done.") ;
}

    </script>
  </head>
  <body onload="mVIPload()">
  </body>
</html>
```

A.3 Fichier utils.js

```
/*************************************************************************
*                                                                        *
*   Copyright (C) FRANCE TELECOM 1998-2007 - TOUS DROITS RESERVES        *
*                                                                        *
*                                                                        *
*   Avertissement : ce logiciel est protégé en France par le Code de la  *
*   Propriété Intellectuelle - Propriété Littéraire et Artistique - et   *
*   par les Conventions Internationales sur le droit d'auteur.           *
*                                                                        *
*   Toute reproduction, adaptation ou distribution partielle ou totale   *
*   du logiciel, autre qu'une seule copie de sauvegarde, par quelque     *
*   moyen que ce soit, est strictement interdite ainsi que pour la       *
*   documentation qui y est associée.                                    *
*                                                                        *
*   Toute personne ne respectant pas ces conditions se rendra coupable   *
*   de délit de contrefaçon et sera passible des sanctions pénales       *
*   prévues par la loi.                                                  *
*                                                                        *
*   L'APP  est mandatée  par l'auteur  pour faire  sanctionner toutes    *
*   copie et/ou utilisation non autorisées.                              *
*                                                                        *
*************************************************************************/

/*************************************************************************/
/* "Prototype-like functions */
$TT = function(s) {
    var tag = document.createElement("pre");
    tag.appendChild(document.createTextNode(s));
    return tag;
};

$FUS = function(tagName, arrayNodes) {
    //var args = $A(arguments).slice(1);
    var tag = document.createElement(tagName);
    $A(arrayNodes).each( function(node){
        tag.appendChild(node);
    });
    return tag;
};

$T = function(s) {
    var tag = document.createElement("span");
    tag.appendChild(document.createTextNode(s));
    return tag;
};
```

```
/*************************************************************************/
/* Debugging functions */

function debug(msg){
  if(DEBUG=='true'){
    var date = new Date();
    var h = date.getHours();
    var m = date.getMinutes();
    var s = date.getSeconds();
    if(m<10) m = "0"+m;
    if(s<10) s = "0"+s;
    DEBUG_LINE++;
    if(DEBUG_LINE > DEBUG_MAX()){
      DEBUG_LINE = 0;
      getById('divDebug').innerHTML = "";
    }
  getById('divDebug').innerHTML+="["+h+":"+m+":"+s+"] "+msg+"<br/>" ;
  }
}

function debugIE(msg){
  if(DEBUG=='true' && BROWSER == "IE5+") debug(msg);
}

/*************************************************************************/
/* Utils functions  */

/*
 * Several Javascript functions alias
 */
function getById(id){ return document.getElementById(id); }
function wln(elt){ document.writeln(elt); }

/*
 * Brower detecting functions
 */
function detectBrowser(){
  if (navigator.userAgent.search(/Firefox/) != -1) return "Firefox";
  else if (navigator.userAgent.search(/Safari/) != -1) return "Safari";
  else if (navigator.userAgent.search(/MSIE/) != -1) return "IE5+";
  else return "Autre" ;
}

/*
 * Create a new HTTPRequest object : browser dependance
 */
httpRequest = function(){
```

```
      if(window.XMLHttpRequest) return new XMLHttpRequest();
      else{
        if(window.ActiveXObject)
          return new ActiveXObject("Microsoft.XMLHTTP");
        else{
          debug("browser don't support HttpResquet Object");
          return null;
        }
      }
    };

  function include(fileName) {
    if (document.getElementsByTagName) {
      var Script = document.createElement("script");
      Script.type = "text/javascript";
      Script.src = fileName;
      var Body = document.getElementsByTagName("head");
      if (Body) {
        Body[0].appendChild(Script);
      }
    }
  }

  function mod(divisee,base) { /* Created 1997 by Brian Risk.
                                      http://members.aol.com/brianrisk */
        return Math.round(divisee - (Math.floor(divisee/base)*base));
  }

  function exist(a){/* 11.11.2006 have to be test */
    var b;
    return (! (a == b));
  }

  /* function sleep(millisecondi)
  {
      var now = new Date();
      var exitTime = now.getTime() + millisecondi;
      while(true){
        now = new Date();
        if (now.getTime() > exitTime) return;
      }
  } */
```

A.4 Fichier mvc.js

```
/*************************************************************************
*                                                                        *
*   Copyright (C) FRANCE TELECOM 1998-2007 - TOUS DROITS RESERVES        *
*                                                                        *
*                                                                        *
*   Avertissement : ce logiciel est protégé en France par le Code de la  *
*   Propriété Intellectuelle - Propriété Littéraire et Artistique - et   *
*   par les Conventions Internationales sur le droit d'auteur.           *
*                                                                        *
*   Toute reproduction, adaptation ou distribution partielle ou totale   *
*   du logiciel, autre qu'une seule copie de sauvegarde, par quelque     *
*   moyen que ce soit, est strictement interdite ainsi que pour la       *
*   documentation qui y est associée.                                    *
*                                                                        *
*   Toute personne ne respectant pas ces conditions se rendra coupable   *
*   de délit de contrefaçon et sera passible des sanctions pénales       *
*   prévues par la loi.                                                  *
*                                                                        *
*   L'APP  est mandatée  par l'auteur  pour faire  sanctionner toutes    *
*   copie et/ou utilisation non autorisées.                              *
*                                                                        *
*************************************************************************/

/*************************************************************************
* Basic MVC framework
*/

/*************************************************************************/
/* Controller class */

function controller(){

    /* class attributes */
    /************************************************************************/

    /*
     * List of listeners
     */
              this.listeners   = new Array();

    /* class methods */
    /************************************************************************/

    /*
     * Add a listener to controller listeners list
     */
```

```
                this.addListener = function(listener){
                  //debug('[controller] addListener' + listener);
                  this.listeners.push(listener);
                  return false ;
                };

        /*
         * Send to all controller listeners the event named by
         * name with optional object params
         */
                this.send        = function(name,params){
                  debug('[controller] send name='+name+' params='+params);
                  this.listeners.each(function(listener) {
                  listener.receive(name,params);
                  });
                  return false ;
                };

    };//end class controller

    /**************************************************************************/
    /* View class, had a controller to prevent (if necessary) other component
     * of his modification */

    function view(controller){

        /* class attributes */
        /**************************************************************************/

        /* his own controller*/
                this.controller = controller;

        /* class methods */
        /**************************************************************************/

        /*
         * View can receive a event named by name with optional object named by params
         * If implemented, it activate the method ("on"+__eventname__)
         * corresponding to received event;
         */
                this.receive = function(name,params){
                  var func = "on"+name;
                  if(this[func]) this[func](params);
                  return false ;
                };

    }; //end class controller
```

A.5 Fichier gateway.js

```
/*************************************************************************
*                                                                        *
*   Copyright (C) FRANCE TELECOM 1998-2007 - TOUS DROITS RESERVES        *
*                                                                        *
*                                                                        *
*   Avertissement : ce logiciel est protégé en France par le Code de la  *
*   Propriété Intellectuelle - Propriété Littéraire et Artistique - et   *
*   par les Conventions Internationales sur le droit d'auteur.           *
*                                                                        *
*   Toute reproduction, adaptation ou distribution partielle ou totale   *
*   du logiciel, autre qu'une seule copie de sauvegarde, par quelque     *
*   moyen que ce soit, est strictement interdite ainsi que pour la       *
*   documentation qui y est associée.                                    *
*                                                                        *
*   Toute personne ne respectant pas ces conditions se rendra coupable   *
*   de délit de contrefaçon et sera passible des sanctions pénales       *
*   prévues par la loi.                                                  *
*                                                                        *
*   L'APP  est mandatée  par l'auteur  pour faire  sanctionner toutes    *
*   copie et/ou utilisation non autorisées.                              *
*                                                                        *
*************************************************************************/


/*************************************************************************
 *  Gateway class : a special class
 *   - receive message from mm_module adn propagate them to his listeners.
 *   - can send dtmf (receive form controller he listened) to mm_module.
 *   - use HTTPRequet (XML or ActiveX) object to communicate with mm_module.
 */

function Gateway(){

  /* class inherits*/
  /************************************************************************/
  this._extends = view;
  this._extends(new controller());

  /* class attributes */
  /************************************************************************/
  this.MM_FETCHER    = "mmm/mm_fetcher.cgi";
    // event flow web service for listening mVIP events
  this.MM_TRANSLATER = "mmm/mm_translater.cgi";
    // one shoot web service for sending web event
  this.MM_CALLER = "mmm/mm_caller.cgi";
    // one shoot web service to subscribe to a mVIP session
```

```
      this.MM_SERVICE = "slidy/html/dialogue.cgi";
        // associated VoiceXML script URL

50    this.last    = 0;     //last modification date (timestamp)
      this.timerRef;        //timer reference
      this.count   = 0;     //debug count
      this.started = false; //debug timer start
      this.response = "";

      /* class State */
      /*********************************************************************/
      var CALL = false ;

60    /* class HTTPRequest methods */
      /*********************************************************************/

      /*
       * Create a new HTTPRequest object : browser dependance
       */
      this.httpRequest = httpRequest ; // in utils.js

      /*
       * Gateway post method
70     * link :  [String]  link to post
       * cmd  :  [String]  server side command to execute, cmd also permit to
       *                   activate the right pasing function
       * vars :  [Array]  list of variable to post
       */
      this.post = function(link,cmd,vars){
        var self = this;
        var req  = self.httpRequest();
        //HTTPRequest event handler functions
        req.onreadystatechange = function(){
80      if(req.readyState == 4){
            if(!req.status) { debug("HTTP REQUEST status error"); return ; }
              if(req.status == 200) self.parse(req,cmd); //when response is complete
            else if(req.status != 0)
              alert("error : " + req.statusText);
            }
        };
        req.open("POST",link,true); //send request
        req.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
        req.send("cmd="+cmd+"&"+vars.join('&')); //send variables
90    };
```

```
/*
 * Gateway parsing HTTPRequest response
 * req : [HTTPRequest Object] an HTTPRequest
 * cmd : [String] comman name
 */
this.parse = function(req,cmd){
  var response = req.responseText;
  switch(cmd){
    case "register" :
      // to do when gui is not registered in mm_box
    break;

    case "fetch" :
      //in call but no modification from service
      if(response == 'NoModification') return;
      if(response == 'NoCall'){
        // when call is interrupted
        if(CALL){
            CALL = false; LANG = "fr";
            this.controller.send("Finish"); // finish process
            return;
        }
        return;
      }

      //response is not a NoModification or NoCall
      // response parsing
      debug("[GATEWAY] fetch response="+response)
      // parse event and params
      var args  = response.split(';');
      this.last = parseInt(args[0]); // args[0] : modification date
      //args[2] : optionnal information use in activate methods
      if(args[2]!=='null'){
        var params = this.parseParams(args[2]); //params paring
        if(params.lang) LANG=params.lang; //init lang choice
      }
      CALL=true;
      this.controller.send(args[1],params); //args[1] is the event name,
                                            //send to listeners
    break;

    default:debug("gateway, parse : no command found");
    };
};
```

```
/*
 * Gateway receive parameters parsing
 */
this.parseParams = function(st){
  debug("[GATEWAY] parse params="+st);
  var p = new Object();
  var group = st.split(',');
  group.each( function(g) {
      var elt = g.split(':');
      var tmp = elt[1].split("'");
      if(tmp) p[elt[0]] = tmp[1];
      else p[elt[0]] = elt[1];
    }
  );
  return p;
};

/*
 * Gateway timer methods
 */
 var TIMER_DELAY = 500;

 this.startTimer = function(){
   debug("[GATEWAY] Timer start delay=["+TIMER_DELAY+"]");
   var self = this;
   self.timerRef = window.setInterval(function(){self.fetch();},
                                                    TIMER_DELAY);
   //self.fetch();
   if(DEBUG=='true')
     getById('divTimer').onmouseup = self.stopTimer(self);
   self.started = true;
 };

 this.stopTimer = function(self){
   return function(){
     if(self.started){
       window.clearInterval(self.timerRef);
       self.started = false;
     }
     else self.startTimer();
   }
 };


/* class event mm_module methods */
/*********************************************************************/

/*
```

```
 * Register a new DEV_ID in mm_box when not exists
 */
  this.register =  function(){
    this.post(this.MM_FETCHER,"register",["dev_id="+DEV_ID]);
  };

/*
 * Fetch new information from mm_box
 */
 this.fetch = function(){
   //debug("[GATEWAY] fetch");
   if(DEBUG=='true')
     getById('divTimer').innerHTML = "["+(this.count++)+"]";
   this.post(this.MM_FETCHER,"fetch",["dev_id="+DEV_ID,"last="+this.last]);
 };

/* class event handlers methods */
/*********************************************************************/

this.onSendDTMF = function(params){
    debug("[GATEWAY] onSendDTMF=["+params.dtmf+"]");
    this.post(this.MM_TRANSLATER,
                  "MakeCall",["dev_id="+DEV_ID,"dtmf="+params.dtmf]);
};

this.onCall = function(params){
    debug("[GATEWAY] Call the number " + params.number);
    DEV_ID = params.number.substr(6,4) ;
    this.controller.send("Calling",{number: params.number});
    if (!this.started) this.startTimer();
    this.post(this.MM_CALLER,"sendDTMF",["dev_id="+DEV_ID,
              "number="+params.number,"service="+this.MM_SERVICE]);
};

this.onFinish = function(params){
    debug("[GATEWAY] onFinish ");
    _this = this ;
    window.clearInterval(_this.timerRef);
    this.started = false;
    document.CallForm.PhoneNumber.focus() ;
};

this.onHangUp = function(params){
    debug("[GATEWAY] onHangUp ");
    //this.controller.send("Finish");
 };

this.onIsConnected = function(params){
```

```
        debug("[GATEWAY] onIsConnected ");
    };

    this.onFontSizeUp = function(params){
      debug("[GATEWAY] onFontSizeUp ");
      bigger()
      return false ;
    };

    this.onPreviousSlide = function(params){
      debug("[GATEWAY] onPreviousSlide ");
      previousSlide(true)
      return false ;
    };

    this.onToggleFullScreen = function(params){
      debug("[GATEWAY] onToggleFullScreen ");

      return false ;
    };

    this.onNextSlide = function(params){
      debug("[GATEWAY] onNextSlide ");
      nextSlide(true)
      return false ;
    };

    this.onFontSizeDown = function(params){
      debug("[GATEWAY] onFontSizeDown ");
      smaller()
      return false ;
    };

    this.onToogleTOC = function(params){
      debug("[GATEWAY] onToogleTOC ");
      if (toc)
        showTableOfContents();
      return false ;
    };

    this.onToogleWidget = function(params){
      debug("[GATEWAY] onToogleWidget ");
      toggleWidget();
      return false ;
    };

  };//end class gateway
```

A.6 Fichier caller.js

```
/*****************************************************************************
*                                                                            *
*   Copyright (C) FRANCE TELECOM 1998-2007 - TOUS DROITS RESERVES            *
*                                                                            *
*                                                                            *
*   Avertissement : ce logiciel est protégé en France par le Code de la      *
*   Propriété Intellectuelle - Propriété Littéraire et Artistique - et       *
*   par les Conventions Internationales sur le droit d'auteur.               *
*                                                                            *
*   Toute reproduction, adaptation ou distribution partielle ou totale       *
*   du logiciel, autre qu'une seule copie de sauvegarde, par quelque         *
*   moyen que ce soit, est strictement interdite ainsi que pour la           *
*   documentation qui y est associée.                                        *
*                                                                            *
*   Toute personne ne respectant pas ces conditions se rendra coupable       *
*   de délit de contrefaçon et sera passible des sanctions pénales           *
*   prévues par la loi.                                                      *
*                                                                            *
*   L'APP  est mandatée  par l'auteur  pour faire  sanctionner toutes        *
*   copie et/ou utilisation non autorisées.                                  *
*                                                                            *
*****************************************************************************/

/*
* ATM Numpad View representation
*/

function Caller(div){

  /* class inherits*/
  /*************************************************************************/
  this._extends = view;
  this._extends(new controller());

  /* class attributes */
  /*************************************************************************/
   //nodes
   this.mask = null ;
   //images
   this.imgBG   = '../img/caller-bg.png';
   this.imgDown = '../img/numpad-down.png';
   this.imgOver = '../img/numpad-over.png';
   this.imgOut  = '../img/numpad.png';
```

```
/* class methods */
/*************************************************************************/
 this.activeMask = function(params){
   debug('[CALLER] activeMask') ;
   debug('Calling params.number') ;
   var mask = $('divCallerMask') ;
   mask.style.display = 'inline';
 }

 this.desactiveMask = function(params){
   debug('[CALLER] desactiveMask') ;
   debug('Calling params.number') ;
   var mask = $('divCallerMask') ;
   mask.style.display = 'none';
 }

 this.activeHangup = function(params){
   var _this = this;
   $("buttonCaller").innerHTML =
                        "<img src='../img/telephone_rouge.gif' border='0'/>" ;
   $("buttonCaller").onmousedown = function(){
         getById("divButtonCall").src = _this.imgDown };
   $("buttonCaller").onmouseover = function(){
         getById("divButtonCall").src = _this.imgOver };
   $("buttonCaller").onmouseout = function(){
         getById("divButtonCall").src = _this.imgOut };
   $("buttonCaller").onmouseup   = function(){
     getById("divButtonCall").src = _this.imgOut;
     _this.controller.send("HangUp",null);
     }
   $("buttonCaller").title = "raccrocher";
 }

 this.desactivateInput = function(){
   var input = getById("divInputCaller") ;
   input.onkeyup = null ;
   var td = getById("buttonCaller") ;
   td.onmousedown = null ;
   td.onmouseover = null ;
   td.onmouseout = null ;
   td.onmouseup = null ;
   var form = getById("divFormCaller") ;
   form.onsubmit = null ;
   return false ;
} ;

 this.activateInput = function(){
   document.CallForm.reset() ;
```

```
var _this = this;
var td = getById("buttonCaller") ;
td.onmousedown = function(){
      getById("divButtonCall").src = _this.imgOut ; }
td.onmouseover = function(){
      getById("divButtonCall").src = _this.imgOut ; }
td.onmouseout = function(){
      getById("divButtonCall").src = _this.imgOut ; }
td.onmouseup   = function(){
      getById("divButtonCall").src = _this.imgOut ; }
td.innerHTML = "<img src='../img/telephone_gris.gif' border='0'/>" ;
td.title = "entrez un numero de telephone valide";
var input = getById("divInputCaller") ;
input.onkeyup = function() {
  var value = $('divInputCaller').value ;
  if ( /^[1-6][0-9]{8}$/.test(value) ) {
    td.onmousedown = function(){
          getById("divButtonCall").src = _this.imgDown};
    td.onmouseover = function(){
          getById("divButtonCall").src = _this.imgOver};
    td.onmouseout = function(){
          getById("divButtonCall").src = _this.imgOut};
    td.onmouseup   = function(){
      getById("divButtonCall").src = _this.imgOut;
      _this.controller.send("Call",{number: value});
      }
    td.innerHTML = "<img src='../img/telephone_vert.gif' border='0'/>" ;
    td.title = "mise en relation";
  } else {
    td.onmousedown = function(){
          getById("divButtonCall").src = _this.imgOut ; }
    td.onmouseover = function(){
          getById("divButtonCall").src = _this.imgOut ; }
    td.onmouseout = function(){
          getById("divButtonCall").src = _this.imgOut ; }
    td.onmouseup   = function(){
          getById("divButtonCall").src = _this.imgOut ; }
    td.innerHTML = "<img src='../img/telephone_gris.gif' border='0'/>" ;
    td.title = "entrez un numero de telephone valide";
  }
}
var form = getById("divFormCaller") ;
form.onsubmit = function() {
  var value = $('divInputCaller').value ;
  if ( /^[1-6][0-9]{8}$/.test(value) ) {
    _this.controller.send("Call",{number: value});
  return false ;
  } else return true ;
```

```
    }
  };

  /* class graphical methods */
  /*************************************************************************/
  this.construct = function(div){
    //buttons positionning
    that = this;
    div.innerHTML+=
      "<img id='divButtonCall' src='" + that.imgOut + "'\
            style='position:absolute;left:190px;top:10px;'/>\
       <table width='242' height='60' border='0'\
              style='position:absolute;left:4px;top:4px;text-align:center;'><tr>\
          <td width=75%''>\
            <form name ='CallForm' action='#' id='divFormCaller'>\
               <input name='PhoneNumber' type='text' id='divInputCaller' \
                      value='023150119'  size='10'/>\
            </form></td>\
          <td width=25%' id='buttonCaller'></td></tr>\
       </table>\
       <div id='divCallerMask' class='mask'>\
         <p><img src='../img/loading.gif' alt='waiting'/></p>\
       </div>";
  };

  this.construct(div); //construct the gui

  this.activateInput();  //activate all buttons, with no dtmf sent









  /* class event Handler methods */
  /*************************************************************************/
  /*
   * Finish event handler :  reinitialize all buttons
   */
   this.onFinish   = function(){ this.activateInput(); };

   this.onIsConnected   = function(params){
     debug('[CALLER] isConnected') ;
```

```
        this.activeHangup(null) ;
        this.desactiveMask(null) ;
        return false ; }

      this.onCalling   = function(params){
        debug('[CALLER] onCalling') ;
        this.activeMask({message:'calling ' + params.number}) ;
        this.desactivateInput() ;
        return false ; }

};  //end class Caller
```

A.7 Fichier numpad.js

```
/*************************************************************************
*                                                                        *
*   Copyright (C) FRANCE TELECOM 1998-2007 - TOUS DROITS RESERVES        *
*                                                                        *
*                                                                        *
*   Avertissement : ce logiciel est protégé en France par le Code de la  *
*   Propriété Intellectuelle - Propriété Littéraire et Artistique - et   *
*   par les Conventions Internationales sur le droit d'auteur.           *
*                                                                        *
*   Toute reproduction, adaptation ou distribution partielle ou totale   *
*   du logiciel, autre qu'une seule copie de sauvegarde, par quelque     *
*   moyen que ce soit, est strictement interdite ainsi que pour la       *
*   documentation qui y est associée.                                    *
*                                                                        *
*   Toute personne ne respectant pas ces conditions se rendra coupable   *
*   de délit de contrefaçon et sera passible des sanctions pénales       *
*   prévues par la loi.                                                  *
*                                                                        *
*   L'APP  est mandatée  par l'auteur  pour faire  sanctionner toutes    *
*   copie et/ou utilisation non autorisées.                              *
*                                                                        *
*************************************************************************/

/*
*  Numpad View representation
*/

function Numpad(div){

  /* class inherits*/
  /************************************************************************/
  this._extends = view;
  this._extends(new controller());

  /* class attributes */
  /************************************************************************/

  /*
    * buttons specification
    * name     : button name
    * dtmf      : default associate dtmf
    * row, col : position on numpad grid
    *  imgId   : image reference name
    */
```

```
this.buttons = [
  {dtmf:"0", row:3,col:1,view:"0",     state:"disable",title:""           },
  {dtmf:"1", row:0,col:0,view:"1",     state:"disable",title:""           },
  {dtmf:"2", row:0,col:1,view:"+",     state:"enable", title:"police (+)" },
  {dtmf:"3", row:0,col:2,view:"3",     state:"disable",title:""           },
  {dtmf:"4", row:1,col:0,view:"&larr;",state:"enable", title:"precedent"  },
  {dtmf:"5", row:1,col:1,view:"F11",   state:"enable", title:"plein ecran"},
  {dtmf:"6", row:1,col:2,view:"&rarr;",state:"enable", title:"suivant"    },
  {dtmf:"7", row:2,col:0,view:"7",     state:"disable",title:""           },
  {dtmf:"8", row:2,col:1,view:"-",     state:"enable", title:"police (-)" },
  {dtmf:"9", row:2,col:2,view:"9",     state:"disable",title:""           },
  {dtmf:"A", row:0,col:3,view:"A",     state:"disable",title:""           },
  {dtmf:"B", row:1,col:3,view:"B",     state:"disable",title:""           },
  {dtmf:"C", row:2,col:3,view:"C",     state:"disable",title:""           },
  {dtmf:"#", row:3,col:2,view:"?",     state:"enable", title:"Aide"       },
  {dtmf:"D", row:3,col:3,view:"D",     state:"disable",title:""           },
  {dtmf:"*", row:3,col:0,view:"TOC",   state:"enable", title:"TOC"        }];

//images
this.imgBG   = '../img/numpad-bg.png';
this.imgDown = '../img/numpad-down.png';
this.imgOver = '../img/numpad-over.png';
this.imgOut  = '../img/numpad.png';

/* class buttons event handler */
/*********************************************************************/
this.onmousedown = function(_self,btn){
  return function(){ getById("divButton" + btn.name).src = _self.imgDown};
};
this.onmouseover = function(_self,btn){
  return function(){ getById("divButton" + btn.name).src = _self.imgOver};
};
this.onmouseup = function(_self,btn){
  return function(){
    getById("divButton" + btn.name).src = _self.imgOut;
    if(btn.state == "enable")
      _self.controller.send("SendDtmf",{dtmf:params.dtmf});
  }
};
this.onmouseout = function(_self,btn,img){
  return function(){ getById("divButton" + btn.name).src = _self.imgOut;}
};
```

```
/*
 * Activate all specific buttons onmouseup event
 */
this.activateAll = function(){
  var _this = this;
  var btn, td;
  this.buttons.each(
    function(btn) { // prototype.js notation using iterartor on an array.
      if(btn.state!="disable"){
        td = getById("button"+btn.row+btn.col)
        td.onmousedown = _this.onmousedown(_this, btn);
        td.onmouseup   = function(){
          getById("divButton" + btn.name).src = _this.imgOut;
          if (exist(btn.event)) _this.controller.send(btn.event,btn.params);
          else if (btn.state == "enable")
                 _this.controller.send("SendDtmf",{dtmf:btn.dtmf});}
        td.onmouseover = _this.onmouseover(_this, btn);
        td.onmouseout = _this.onmouseout(_this, btn);
        td.innerHTML = "<div style='position:absolute;'>"+
                       "<div class=\"button\">" + btn.name + "</div></div>" + btn.view;
        td.title = btn.title;
        td.style.color = "#333333";
      } else {
        td = getById("button"+btn.row+btn.col)
        td.innerHTML = btn.view;
        td.style.color = "#CCCCCC";
      }
    }
  );
};

/*
  * Deactivate all onmousup buttons event
  */
this.desactivateAll = function(){
  _this = this;
  this.buttons.each(
    function(btn) {
        td = getById("button"+btn.row+btn.col)
        td.innerHTML = btn.name;
        td.style.color = "#CCCCCC";
        td.onmousedown = null ;
        td.onmouseup   = null ;
        td.onmouseover = null;
        td.onmouseout = null ;
    }
  );
};
```

```
    /* class graphical methods */
    /*************************************************************************/
    this.construct = function(div){
      //background
      div.innerHTML += "<img id='imgNumpad' src='"+this.imgBG+"' />";
        //could be done in CSS using the div background-image property
        //buttons positionning
      that = this;
      this.buttons.each(function(btn){ // prototype.js notation using iterartor
          if(btn.name){
            var x = 60 * btn.col + 10 ;  var y = 60 * btn.row +10;
            div.innerHTML+="<img id='divButton" + btn.name + "' src='" +
              that.imgOut + "' style='position:absolute;left:"+x+"px;top:"+y+"px;'/>";
          }
        });
      div.innerHTML+="<table>\
                        <tr><td id='button00'></td><td id='button01'></td>
                          <td id='button02'></td><td id='button03'></td></tr>\
                        <tr><td id='button10'></td><td id='button11'></td>
                          <td id='button12'></td><td id='button13'></td></tr>\
                        <tr><td id='button20'></td><td id='button21'></td>
                          <td id='button22'></td><td id='button23'></td></tr>\
                        <tr><td id='button30'></td><td id='button31'></td>
                          <td id='button32'></td><td id='button33'></td></tr>\
                      </table>";  // could be better using div
    };
    this.construct(div); //construct the gui
    this.desactivateAll();  //activate all buttons, with no dtmf sent

    /* class event Handler methods */
    /*************************************************************************/

    this.onFinish   = function(){ this.desactivateAll(); };

    this.onIsConnected   = function(){ this.activateAll(); };

  }; //end class Numpad
```

A.8 Fichier dialogue.cgi

```
#!/usr/bin/php
<?php
/*******************************************************************************
*                                                                              *
*   Copyright (C) FRANCE TELECOM 1998-2007 - TOUS DROITS RESERVES              *
*                                                                              *
*                                                                              *
*   Avertissement : ce logiciel est protégé en France par le Code de la        *
*   Propriété Intellectuelle - Propriété Littéraire et Artistique - et         *
*   par les Conventions Internationales sur le droit d'auteur.                 *
*                                                                              *
*   Toute reproduction, adaptation ou distribution partielle ou totale         *
*   du logiciel, autre qu'une seule copie de sauvegarde, par quelque           *
*   moyen que ce soit, est strictement interdite ainsi que pour la             *
*   documentation qui y est associée.                                          *
*                                                                              *
*   Toute personne ne respectant pas ces conditions se rendra coupable         *
*   de délit de contrefaçon et sera passible des sanctions pénales             *
*   prévues par la loi.                                                        *
*                                                                              *
*   L'APP  est mandatée  par l'auteur  pour faire  sanctionner toutes          *
*   copie et/ou utilisation non autorisées.                                    *
*                                                                              *
*******************************************************************************/
$DEV_ID = substr($_REQUEST["called_id"],6,4) ;
$num_line  = sprintf("%03d",$_REQUEST["num_line"]);
echo '<?xml version="1.0" encoding="ISO-8859-1"?>'
?>
<!DOCTYPE vxml PUBLIC "-//W3C//DTD VOICEXML 2.0//EN"
      "http://www.w3.org/TR/voicexml20/vxml.dtd">
<vxml version="2.0" application="./root.php?num_line=<?php echo $num_line; ?>">

<form id="accueil">
  <block>
    <prompt bargein="true">
      <meta name="send"
            content="array('event'=>'IsConnected',
                           'params'=>'null',
                           'from'=>'<?php echo $num_line ; ?>',
                           'to'=>'<?php echo $DEV_ID ; ?>')"/>
      Bonjour et bienvenue sur aime slailldi.</prompt>
    <goto next="#menu"/>
  </block>
</form>
<menu id="menu">
  <choice dtmf="2" next="#send_fontSizeUp">Plus grand</choice>
```

```
    <choice dtmf="4" next="#send_previousSlide">Precedent</choice>
    <choice dtmf="5" next="#send_toggleFullScreen">Plein ecran</choice>
    <choice dtmf="6" next="#send_nextSlide">Suivant</choice>
    <choice dtmf="8" next="#send_fontSizeDown">Plus petit</choice>
    <choice dtmf="*" next="#send_toogleTOC">Table des matieres</choice>
    <choice dtmf="#" next="#send_toogleWidget">Aide</choice>
    <nomatch>
      <!--prompt>Je ne comprends pas votre saisie.</prompt-->
    </nomatch>
    <noinput> <!-- 5 secondes -->
      <!--prompt>Je n'ai detecte aucune saisie.</prompt-->
    </noinput>
  </menu>

  <form id="send_fontSizeUp">
    <block>
      <prompt>
        <meta name="send"
              content="array('event'=>'FontSizeUp',
                             'params'=>'null',
                             'from'=>'<?php echo $num_line ; ?>',
                             'to'=>'<?php echo $DEV_ID ; ?>')"/>
      </prompt>
        <goto next='#menu'/>
    </block>
  </form>

  <form id="send_previousSlide">
    <block>
      <prompt>
        <meta name="send"
              content="array('event'=>'PreviousSlide',
                             'params'=>'null',
                             'from'=>'<?php echo $num_line ; ?>',
                             'to'=>'<?php echo $DEV_ID ; ?>')"/>
      </prompt>
        <goto next='#menu'/>
    </block>
  </form>

  <form id="send_nextSlide">
    <block>
      <prompt>
        <meta name="send"
              content="array('event'=>'NextSlide',
                             'params'=>'null',
                             'from'=>'<?php echo $num_line ; ?>',
                             'to'=>'<?php echo $DEV_ID ; ?>')"/>
```

```
      </prompt>
        <goto next='#menu'/>
    </block>
  </form>

  <form id="send_fontSizeDown">
    <block>
      <prompt>
        <meta name="send"
              content="array('event'=>'FontSizeDown',
                             'params'=>'null',
                             'from'=>'<?php echo $num_line ; ?>',
                             'to'=>'<?php echo $DEV_ID ; ?>')"/>
      </prompt>
        <goto next='#menu'/>
    </block>
  </form>

  <form id="send_toogleTOC">
    <block>
      <prompt>
        <meta name="send"
              content="array('event'=>'ToogleTOC',
                             'params'=>'null',
                             'from'=>'<?php echo $num_line ; ?>',
                             'to'=>'<?php echo $DEV_ID ; ?>')"/>
      </prompt>
        <goto next='#menu'/>
    </block>
  </form>

  <form id="send_toogleWidget">
    <block>
      <prompt>
        <meta name="send"
              content="array('event'=>'ToogleWidget',
                             'params'=>'null',
                             'from'=>'<?php echo $num_line ; ?>',
                             'to'=>'<?php echo $DEV_ID ; ?>')"/>
      </prompt>
        <goto next='#menu'/>
    </block>
  </form>

  </vxml>
```

A.9 Fichier root.php

```
<!--------------------------------------------------------------------------
*                                                                         *
*   Copyright (C) FRANCE TELECOM 1998-2007 - TOUS DROITS RESERVES         *
*                                                                         *
*                                                                         *
*   Avertissement : ce logiciel est protégé en France par le Code de la   *
*   Propriété Intellectuelle - Propriété Littéraire et Artistique - et    *
*   par les Conventions Internationales sur le droit d'auteur.            *
*                                                                         *
*   Toute reproduction, adaptation ou distribution partielle ou totale    *
*   du logiciel, autre qu'une seule copie de sauvegarde, par quelque      *
*   moyen que ce soit, est strictement interdite ainsi que pour la        *
*   documentation qui y est associée.                                     *
*                                                                         *
*   Toute personne ne respectant pas ces conditions se rendra coupable    *
*   de délit de contrefaçon et sera passible des sanctions pénales        *
*   prévues par la loi.                                                   *
*                                                                         *
*   L'APP  est mandatée  par l'auteur  pour faire  sanctionner toutes     *
*   copie et/ou utilisation non autorisées.                               *
*                                                                         *
--------------------------------------------------------------------------->

<?php
/*
* VoiceXML Application Root document for ATM service
* defined general behavoir
*/
/* multi modal box cleaner */
$CLEANER = "mmm/mm_cleaner.cgi";
/*generate a vxml document */

echo '<?xml version="1.0" encoding="ISO-8859-1"?>' ;
?>
<!DOCTYPE vxml PUBLIC "-//W3C//DTD VOICEXML 2.0//EN"
      "http://www.w3.org/TR/voicexml20/vxml.dtd">
<vxml version="2.0">

<?php
/*POST or GET Vars*/
$num_line = $_REQUEST["num_line"];
/*make a VXML variable session*/
if($num_line) echo "<var name='num_line' expr=\"'$num_line'\"/>\n";
?>
```

```
<!-- hangup_atm is a mm_cleaner command-->
<!-- catch platform  error event-->
<error>
  <if cond='_message != undefined'>
    <log>Error: <value expr='_event'/>, <value expr='_message'/></log>
  <else/>
    <log>Error: <value expr='_event'/></log>
  </if>
    <!-- if error clean the multi modal communication box-->
    <var name="cmd" expr="'hangup_atm'"/>
    <var name="lang" expr="'<?php echo $lang?>'"/>
    <submit next="<?php echo $CLEANER ?>"  method="post"
            namelist="cmd num_line lang"/>
</error>

<!-- catch call disconnect event -->
<catch event="connection.disconnect">
  <!-- if call is disconnected clean the multi modal communication box-->
  <var name="cmd" expr="'hangup_atm'"/>
  <submit next="<?php echo $CLEANER ?>"
              method="post"
              namelist="cmd num_line lang"/>
</catch>

</vxml>
```

Résumé - Les technologies web ont permis de définir des services accessibles par tous, n'importe où et depuis de multiples terminaux. Si les utilisateurs intègrent maintenant l'usage des interfaces graphiques traditionnelles, la miniaturisation des équipements et l'émergence de nouvelles modalités offrent de nouvelles perspectives en matière d'informatique ubiquitaire. Dans ce mémoire, nous modélisons l'interaction ubiquitaire selon les principes fondamentaux des modèles de la littérature. Nous proposons une implémentation de notre modèle UbiArch sous forme d'une plate-forme de services et nous mettons à la disposition du concepteur d'applications web un ensemble d'outils interactifs, prêts à être intégrés dans des interfaces web. Nous illustrons les principes de notre modèle et le fonctionnement de notre plate-forme au travers de trois démonstrateurs. Chacune de ces réalisations intègre les capacités d'interaction des téléphones dans une application web et illustre les apports de notre travail en terme d'interaction multimodale distribuée.

Mots-clefs - interfaces utilisateur (informatique), interaction homme-ordinateur, informatique omniprésente, architecture_logiciels, téléphonie internet, services web, AJAX (informatique), VoiceXML (langage de balisage).

Title - Conception of an ubiquitous services platform integrating distributed multimodal interfaces

Abstract - Web-Based Information Technologies & Distributed Systems represent a powerful shift in computation, where people live, work, and play in a seamlessly interweaving computing environment. It postulates a world where people are surrounded by computing devices and a computing infrastructure that supports us in everything we do. Current means of interactions with applications are almost exclusively the keyboard and the mouse or emulations thereof. This kind of interface is well adapted for classic management of information, but the new usages, mobility, ubiquitous access to information need new interfaces and new interaction modes. This thesis aims to modelize and develop a distributed software platform allowing the user access to ubiquitous services through distributed interfaces in these new usage contexts. We propose the original architecture model UbiArch. This system extends the user interface in order to allow several modes of interactions, offering users the choice of using their voice thanks to headset or phone, or an input device such as a keypad, keyboard or other input device. For output, users will be able to listen to audio devices, and to view information on graphical displays such as phones, PDA or TV screen or by using a projector. This concept is called "distributed modality". The 3-tiers design of the mVIP services based on UbiArch offers a framework in order to implement multimodal services for many application contexts. Here, we have described the use of our technology in a biometric authentication context, in a multimodal cash point simulator and in a slide show application controlled from a vocal and DTMF remote phone. This thesis has shown the feasibility of such an architecture dealing with many components and protocols.

Keywords - user interfaces (computer systems), human-computer interaction, ubiquitous computing, architecture_computer program, internet telephony, web services, AJAX (web site development technology), VoixeXML (document markup language).

France Telecom - Orange Labs
42 rue des Coutures
BP 6243
14066 CAEN CEDEX 4

GREYC - CNRS UMR 6072
boulevard du Maréchal Juin
BP 5186
14032 CAEN CEDEX