HDR 2006 00017 Année 2006

HABILITATION A DIRIGER DES RECHERCHES

Présentée devant

L'Institut National des Sciences Appliquées de Lyon
Et l'Université Claude Bernard Lyon I

INTERACTION HOMME-MACHINE ADAPTATIVE

par
Franck TARPIN-BERNARD

Soutenue le 7 décembre 2006 devant la Commission d'examen

(par ordre alphabétique)

COUTAZ Joëlle, Professeur, Université Joseph Fourier Grenoble (rapporteur)
DAVID Bertrand, Professeur, Ecole Centrale de Lyon
DUSSAUCHOY Alain, Professeur, Université Claude Bernard, Lyon I
MILLOT Patrick, Professeur, Université de Valenciennes (rapporteur)
PREVOT Patrick, Professeur, INSA de Lyon
VANDERDONCKT Jean, Professeur, Université Catholique de Louvain (rapporteur)

Laboratoire ICTT (Interaction Collaborative, Téléformation, Téléactivités)

Partie II

Interaction Homme-Machine Adaptative

Introduction

La conception et la production de systèmes informatiques hésitent entre deux tendances. La première consiste à uniformiser/standardiser son matériel et ses logiciels et en particulier leurs Interfaces Homme-Machine, afin de garantir simultanément la cohésion des pratiques, la cohérence du parc informatique et la réduction des coûts. La seconde consiste à adapter le matériel et les logiciels à l'utilisateur et son environnement selon ses besoins, ses possibilités et le contexte d'utilisation. L'adaptation et la standardisation forment un couple, dont on ne peut séparer les termes toujours en tension : l'histoire informatique se ponctue d'allers et retours vers plus d'uniformité ou plus de singularité. Après une phase marquée par l'uniformisation, la conception s'oriente actuellement vers plus de flexibilité. Dans un contexte industriel qui tend à la normalisation, elle s'efforce de trouver les critères pertinents susceptibles de réguler l'adaptation, sans compliquer outre mesure la production tout en tirant profit de ses avantages.

Le choix final ne peut émaner de la considération d'une seule dimension, mais résulte de la pondération de divers facteurs : techniques, commerciaux, sociaux, industriels, psychologiques. Plus profondément, le terme « adapter » révèle une polysémie irréductible : le sens de l'adaptation varie selon les objets sur lesquels elle porte et selon les points de vue. Adapter l'interface ou le noyau fonctionnel, adapter la machine aux profils de l'utilisateur ou aux tâches, adapter les modèles ne correspondent pas aux mêmes réalités. L'Interaction Homme-Machine (IHM), le génie logiciel, la psychologie, la sociologie offrent encore autant de points de vue sur la dialectique entre uniformiser et adapter, sans qu'un consensus clair ne s'établisse. De même, la standardisation s'applique à la production matérielle, aux différentes couches des logiciels, aux usagers comme tendance à l'uniformisation, sans qu'un concept unique ne se dégage. Bien que le dilemme, et le compromis nécessaire, entre l'adaptation et la standardisation ne soit pas propre à l'informatique, de par la puissance et les potentialités de l'ordinateur, mais aussi de par sa diffusion qui recouvre désormais tous les champs de la vie sociale, l'informatique dépasse l'alternative classique. En effet, la malléabilité, qui caractérise les systèmes interactifs par rapport aux autres machines, induit un développement singulier. Elle suscite en outre une autre conception de l'usager, ainsi qu'une relation originale entre les concepteurs, les usagers et les machines.

Pour tenter d'y voir plus clair, j'ai récemment conduit un atelier de recherche pluridisciplinaire au sein du laboratoire. A partir d'études techniques, psychologiques et philosophiques, nous avons tenté d'évaluer l'opportunité des choix entre uniformiser et adapter. Le point de vue historique nous a semblé utile car il éclaire un problème plus complexe qu'il peut apparaître. L'exemple des environnements de programmation est à ce titre exemplaire : on y a vu apparaître des difficultés de collaboration entre les informaticiens lorsque la personnalisation de leurs outils de production et de leurs méthodes n'était pas suffisamment encadrée, ce qui a pu, au moins temporairement, engendrer un retour de balancier violent avec une uniformisation et un verrouillage des postes de travail de mauvais aloi. La tension entre ces deux tendances se traduit donc par des avantages et inconvénients relatifs à l'une ou l'autre de ces options lorsqu'elles sont exclusives. Le Tableau 1 résume les points détaillés dans (Bobillier-Chaumon *et al.*, 2005).

	Uniformisation	Adaptation
Intérêts	− Référence communautaire voire culture de masse − Transfert d'apprentissage entre applications − Réutilisation et interopérabilité − Simplicité de mise en œuvre et de maintenance (économie) − "Sureté" et contrôle du processus	− Ergonomie (confort et efficacité dans l'interaction) − Interaction personnalisée et reconnaissance de l'usager − Flexibilité et souplesse d'usage − Prise en compte du contexte et de la situation d'usage
Inconvénients	− Non prise en compte des différences inter-individuelles − Décontextualisation − Rigidité	− Risque d'isolement − Complexité de mise en œuvre et d'évaluation − Effets négatifs lors d'une mauvaise adaptation

Tableau 1 *:* Intérêts et inconvénients des approches extrêmes d'uniformisation et d'adaptation de l'IHM

Il apparaît alors clairement, qu'une solution optimale ne pourra être trouvée qu'en arrivant à combiner ces deux approches. Trop d'uniformisation rigidifie les produits et empêche toute contextualisation tandis que trop d'adaptation peut conduire à une perte de repères pour l'utilisateur et peut engendrer une complexité insurmontable pour les concepteurs et des coûts de développement excessifs.

A la recherche de cet équilibre, nos travaux se situent donc à la frontière des champs disciplinaires que sont l'Interaction Homme-Machine d'une part et le génie logiciel d'autre part. Notre objectif général est de fournir des modèles, méthodes et outils pour construire, implémenter à moindre coût des applications interactives adaptatives et/ou facilement adaptables à divers contextes d'utilisation. Les solutions proposées doivent systématiquement faire l'objet d'évaluation dans des applications les plus réalistes possibles tant sur le plan technique que sur la qualité des adaptations proposées.

Ceci nous a amené au fil de nos recherches à mener quatre types de travaux que nous allons présenter dans les quatre chapitres suivants.

Le premier chapitre compile nos réflexions théoriques sur la notion de flexibilité et les concepts qui l'entourent comme l'adaptativité, la plasticité ou la notion de contexte. Les recherches étant particulièrement actives dans ces domaines ces dernières années, un soin particulier est porté sur l'état de l'art et sur les différentes classifications que nous avons été amenées à établir.

Le chapitre 2 aborde le problème sous l'angle des méthodes de conception et de construction de logiciels facilement adaptables. Nos principales contributions ont consisté à élaborer un processus complet inspiré des approches à base de modèles. Ce processus s'appuie sur le modèle d'architecture multiagents multifacettes AMF qui aggrège les informations en provenance des autres modèles du système et permet son exécution. Un effort soutenu a été mené ces dernières années pour instrumenter l'approche et fournir des outils indispensables à sa validation.

Le troisième chapitre recense les travaux que nous avons menés sur des techniques spécifiques d'adaptation. Ceci englobe à la fois la définition de nouveaux modèles et

l'élaboration d'algorithmes spécifiques. Ce travail a principalement porté sur des problématiques d'adaptation au profil cognitif des utilisateurs dans des contextes d'hypermédias pédagogiques et a reposé très largement sur les collaborations avec la société SBT. La plupart des techniques imaginées sont cependant transposables à d'autres types d'adaptation.

Le dernier chapitre présente nos réflexions et expérimentations portant sur la question de l'évaluation des systèmes adaptatifs. Nous l'avons vu, il ne suffit en effet pas d'avoir construit un système qui s'adapte à des changements du contexte d'utilisation, encore faut-il être capable de vérifier que cette adaptation est pertinente et qu'elle apporte quelque chose, qu'il s'agisse d'un accroîssement de la performance, d'une baisse de la surcharge cognitive voire d'un simple sentiment subjectif d'amélioration du confort de l'interaction.

Enfin, nous concluons ce mémoire par un bilan de ces travaux et la définition d'un projet de recherche destiné à approfondir ces travaux et explorer de nouvelles voies.

Contexte de recherche :
- Animation groupe de recherche interne laboratoire ICTT (durée 12 mois)

Publications :
- BOBILLIER-CHAUMON M-E, CARVALLO S., TARPIN-BERNARD F., VACHERAND-REVEL J., Adapter ou uniformiser les interactions personnes-systèmes ?, *Revue d'Interaction Homme Machine*, Europia, vol 6, 2, 2005, pp. 91-129

1 Réflexions théoriques et classification de l'état de l'art

Dans ce premier chapitre, nous allons présenter une synthèse des différentes réflexions théoriques que nous avons menées à propos de la notion de flexibilité des systèmes et donc des notions corrollaires que sont le contexte ou les espaces de conception de l'adaptation. Puis nous donnerons les grandes lignes d'un travail de fond qui a été conduit avec les différents doctorants que j'ai encadrés pour élaborer des classifications des différents travaux recensés dans l'état de l'art.

1.1 La flexibilité

L'objectif de ma thèse de doctorat était de proposer un **modèle d'architecture logicielle adapté aux besoins des collecticiels temps réel**, c'est-à-dire des logiciels supportant un travail collaboratif synchrone. Or, en cherchant à identifier ces fameux besoins, il m'est apparu clairement que le besoin le plus important et le plus difficile à satisfaire était celui de flexibilité, souvent appelé malléabilité ou adaptabilité (Tarpin-Bernard, 2000). L'adaptabilité est, avec la compatibilité, l'un des critères que l'ergonomie des logiciels utilise pour apprécier l'adéquation entre les exigences de l'activité, les ressources de l'usager et les caractéristiques du système (Scapin et Bastien, 1996 ; Bastien *et al.,* 1998).

Selon Dewan (1992), la flexibilité permet aux utilisateurs (programmeurs et/ou utilisateurs finaux) de facilement adapter les collecticiels à leurs besoins. Cependant, la flexibilité peut porter sur de nombreux concepts et couvre des domaines hétérogènes : flexibilité dans l'interaction, dans la distribution des données, dans le partage, dans le contrôle d'accès… Par exemple, en cherchant à caractériser la flexibilité dans la seule interaction homme-machine, Gram & Cockton (1996) définissent dix propriétés regroupées en trois catégories :

- représentation de l'information : multiplicité des périphériques (clavier, souris, tablette…), multiplicité des représentations (représentations alternatives aussi bien en entrée qu'en sortie), réutilisation d'entrées/sorties (possibilité d'utiliser des entrées/sorties précédentes comme nouvelle entrée);
- planification dans l'exécution des tâches : multiplicité des rôles de l'utilisateur, cheminement multiple (les utilisateurs peuvent s'engager dans plusieurs tâches en même temps), non-préemptivité (degré de liberté laissé à l'utilisateur pour décider des actions suivantes à réaliser), atteignabilité (possibilité de naviguer à travers les états du système, par exemple possibilité de faire/défaire);
- adaptation des formes de dialogue : reconfigurabilité (capacité du système à supporter des personnalisations des interactions par l'utilisateur), adaptativité (capacité du système à décider de personnalisations), migrabilité (capacité du système à supporter des transferts de responsabilité entre l'utilisateur et le système).

En adoptant un point de vue fonctionnel, Jameson (2003) a pour sa part dressé un panorama de ce que les interfaces adaptatives peuvent faire. Il distingue les fonctions facilitant l'usage (prise en charge d'opérations routinières, adaptation de l'interface, conseil d'utilisation, contrôle du dialogue) de celles aidant à l'acquisition d'information (aide à la recherche, mise en forme de résultats, recommandation de produits, support à la collaboration, support à l'apprentissage).

Bien entendu, ces propriétés sont aussi valables dans les applications coopératives, mais il est nécessaire de considérer d'autres formes de flexibilité. Parce que les activités collaboratives sont éminemment dynamiques et changeantes, les collecticiels doivent introduire de la flexibilité dans tous les rouages de leur mécanique, depuis les services de haut-niveau proposés aux utilisateurs, comme le support de l'activité (ex : le *workflow* ou la notion de groupe et de rôles), jusqu'aux stratégies de gestion de l'architecture qui peuvent variées considérablement en fonction des critères prédominants qui sont retenus : mobilité, robustesse, performance... En allant plus loin dans l'analyse, Dourish (1995) affirme que les applications coopératives doivent tout d'abord être flexibles statiquement, par exemple en proposant des paramétrages prédéfinis adaptés à des pratiques individuelles ou à des styles de travail (approche explorée en détail par Greenberg *et al.*, 1992). En second, elles doivent proposer une flexibilité dynamique qui répond aux changements de comportement des groupes dans la réalisation de collaborations spécifiques voire au sein de sessions complètes. Troisièmement, elles doivent être flexibles sur le plan de leur implémentation, dans la mesure où les besoins en terme d'infrastructure et d'interopérabilité évoluent. Nous adhérons pleinement à cette analyse.

Malone (1995) qui est un ardent défenseur d'une adaptation maximale par l'utilisateur qu'il appelle « tailorabilité radicale » (capacité à être taillé sur mesure), propose un ensemble de composants élémentaires (objets, vues, agents et liens) permettant aux utilisateurs finaux d'assembler à leur guise un système collaboratif. D'autres, comme Dourish ou Bourguin (2000), ont proposé d'utiliser des architectures réflexives, c'est-à-dire contenant la description de leur propre cœur. Dynamiquement, des utilisateurs avertis peuvent ainsi faire évoluer les algorithmes ou les données constitutives du système. Le pouvoir d'adaptation devient ainsi extrême mais la complexité de l'approche le réserve à une minorité. Dans des travaux plus récents traitant notamment de la migration d'applications, Dourish s'oriente vers des solutions moins lourdes techniquement mais qui conservent des caractéristiques d'inspection et de modification pour l'utilisateur. Une telle activité réflexive peut être intégrée dans l'usage naturel de l'outil tout comme nous utilisons chaque jour certains objets domestiques à la fois comme artefacts et comme outils. Ceci passe notamment par la prise en compte des « pratiques » (formes réelles d'engagement dans une activité) plutôt que par la représentation des contextes (ensemble descriptif de caractéristiques) afin de donner un rôle central aux significations attribuées aux objets et aux actions par les utilisateurs en terme de conséquences et d'interprétation pour eux-mêmes et pour les autres (Dourish, 2004). Si, pour des raisons de modélisation, nous n'allons pas aussi loin que Dourish et conservons la notion de contexte, nous verrons plus loin qu'il nous semble effectivement indispensable de mieux prendre en compte l'activité.

Durant mes premières années de recherche, j'ai donc exploré le moyen de structurer la mise en œuvre de la flexibilité dans les collecticiels à travers un modèle d'architecture souple supportant à la fois le multi-vues, les télépointeurs sémantiques, le paramétrage du WYSIWIS (*What You See Is What I See*) ou la sélection de politiques de contrôle de concurrence. En effet, les rôles différents joués par les utilisateurs qui partagent des données induit de façon encore plus forte la nécessité de proposer des représentations différentes de l'information. Ces vues doivent permettre aux acteurs de discuter et de se coordonner sans que leurs perceptions différentes n'induisent de malentendus. L'utilisation d'outils de désignation comme le télépointeur implique de prendre en compte la sémantique des scènes désignées. A la notion classique de *feed-back* vient s'ajouter la

notion de *feed-through* qui traduit la façon dont les actions distantes sont perçues. Pour le confort de l'utilisateur, il est indispensable que cette rétroaction de groupe soit réglable en fonction de la situation. En effet, s'il est important de pouvoir régler la façon dont sont traitées localement les actions des utilisateurs distants, il est aussi indispensable que chacun puisse maîtriser la façon dont sont notifiées ses propres actions. Ceci se traduit par exemple par la possibilité de réglage de la relaxation du WYSIWIS (*What You See Is What You Get* : rendu fidèle) : depuis un couplage strict, où les vues sont synchronisées, à des couplages beaucoup plus relâchés (délais de synchronisation plus long, présentation dégradée, etc.). L'utilisateur doit aussi pouvoir décider du moment où il rend public des données privées qu'il a constituées.

Enfin, si l'on descend plus profondément dans les couches basses des collecticiels, on est inévitablement confronté au problème de la gestion des accès concurrents aux données partagées. En fonction du degré de cohérence maintenu entre des copies d'un système répliqué et des stratégies de contrôle, un très large panel d'algorithmes peut être utilisé. La granularité des objets contrôlés est aussi un facteur déterminant. Parce que ces choix sont commandés par des critères techniques (fréquence des conflits, taille des données, nature des opérations, etc.) mais aussi par ce que le contrôle de concurrence peut avoir des effets de coordination (on voit qu'un autre utilisateur vient de prendre le contrôle d'un objet), il est indispensable que les collecticiels proposent un ensemble de stratégies de contrôle de concurrence flexibles.

L'adaptation peut donc porter sur de très nombreux aspects des applications interactives. Pour simplifier l'analyse, nous avons suggéré de les organiser en 4 familles :

- l'IHM que nous scindons en 3 sous éléments : la présentation de l'interface (position, taille, images, sons, couleurs, structure…), les techniques et styles d'interaction (menus, choix multiples, etc.) et la navigation ;
- les données ou informations qui sont filtrées et/ou réorganisées ;
- les services qui peuvent être modifiés (ajout, restriction, paramétrage) ;
- les fonctions d'assistance. Ce niveau méta, qui permet à un utilisateur d'apprendre à utiliser un système interactif, est un élément essentiel sur lequel l'adaptation doit porter.

Comme nous le verrons § 4.1, cette structuration sera la base d'une proposition de caractérisation du degré d'adaptabilité d'une application.

Finalement, on aura compris que la flexibilité est nécessaire dès que des différences apparaissent dans le contexte d'utilisation. Le contexte est donc une notion clé qu'il convient de détailler.

Contexte de recherche :
- Exploitation des travaux de ma thèse de doctorat

Publications :
- TARPIN-BERNARD F., La flexibilité dans les collecticiels in *Le temps, l'espace et l'évolutif (Tome 2).* Cépaduès. H. Prade, R. Jeansoulin & C. Garbay (eds). Septembre 2000. p. 449-458.
- DAVID B.T., TARPIN-BERNARD F. Evolution de la conception des logiciels in *Conception : entre science et art.* Presse Polytechniques et Universitaires Romande, Lausanne. Perrin J. (eds). Juillet 2001. p. 61-77.

1.2 La notion de contexte d'utilisation

La définition du contexte d'utilisation a fait l'objet de plusieurs tentatives dans la littérature. L'expression « *context aware* » a été introduite pour reconnaître l'identité des personnes, leur localisation, celle des objets à leur proximité ainsi que les modifications pouvant intervenir sur ces objets (Schilit et Theimer, 1994). D'autres chercheurs se sont basés sur cette définition en ajoutant des éléments comme l'heure, la saison, la température (Brown *et al.,* 1997) ou l'environnement et l'heure (Ryan *et al.,* 1997).

Dey finit par définir le contexte comme : « toutes les informations qui peuvent être utilisées pour caractériser la situation d'une entité. Une entité est une personne, un lieu, ou l'objet que l'on considère pertinent pour l'interaction entre un utilisateur et une application, y compris l'utilisateur et l'application eux-mêmes » (Dey, 2001).

Dans la continuation de Thévenin (Thévenin, 2001), Rey essaye de donner une définition formelle pour le contexte d'interaction en s'intéressant aux changements dans le temps, « *Etant donné un utilisateur U, engagé dans une activité A, le contexte d'interaction à l'instant t est la composition des situations entre les instants t_0 et t pour la réalisation de A par l'utilisateur U* » en expliquant que la notion de **situation** traduit les « circonstances qui entourent l'action » (Rey *et al.,* 2002).

La définition de contexte d'utilisation ne cesse d'évoluer. Aujourd'hui, la définition la plus souvent adoptée par les équipes travaillant dans le domaine de l'adaptation des systèmes interactifs décrit le contexte par le triplet <Utilisateur, Plate-forme, Environnement> (Calvary *et al.,* 2003) (Samaan *et al.,* 2004) (Vanderdonckt *et al.,* 2005) où :

- l'**utilisateur** est typiquement décrit par ses capacités (perceptuelles, motrices et cognitives), mais aussi par ses préférences et ses caractéristiques culturelles (ex : langue naturelle) ;
- la **plate-forme** (*user device*) se décline en deux sous catégories :
  - la plate-forme physique (le matériel) ;
  - la plate-forme logique (le logiciel).
- l'**environnemen**t se réfère à l'environnement physique accueillant l'interaction. Il est décrit par un ensemble d'informations périphériques à la tâche en cours mais susceptibles de l'influencer. Par exemple, la luminosité, le bruit, la localisation géographique, etc.

En prolongation de notre analyse préliminaire, il paraît important d'étendre cette définition à la prise en compte de l'**activité** puisque typiquement, un même utilisateur utilisant une même plate-forme d'interaction dans un environnement identique peut avoir des besoins très différents en fonction de l'activité qu'il mène à l'instant t. Souvent, cette notion d'activité courante voire de tâche en cours est diluée dans l'« utilisateur » sous la forme de ses buts, ce qui nous semble impropre. Cette différence de point de vue se retrouve clairement dans les différentes communautés scientifiques que j'ai eu l'occasion de fréquenter ces dernières années : *User Modelling*, *CSCW* et *HCI.*

Par exemple, si l'on s'en tient aux attributs des contextes d'usage définis dans les normes IEC CDV TR 61997 (2000) et ISO 9241-11 (1998), on retrouve clairement ces caractéristiques supplémentaires ainsi que la notion d'environnement socio organisationnel (Cf. remarques précédentes sur le travail collaboratif).

Dans notre travail nous adoptons donc une vision large du contexte. Nous développons dans ce qui suit cette définition en détaillant les modèles et en proposant des exemples d'approches d'adaptation par rapport à chaque modèle.

Contexte de recherche :
- Thèses de doctorat Kinan Samaan et Halima Habieb-Mammar

1.2.1 L'utilisateur

Le modèle de l'utilisateur est l'un des premiers modèles à avoir été utilisé dans les processus d'adaptation des systèmes interactifs. "Un modèle utilisateur est une connaissance à propos de l'utilisateur explicitement ou implicitement codée, utilisée par le système afin d'améliorer son interaction" (Höök *et al.*, 1996).

Rich (1983) distingue différentes manières de modéliser l'utilisateur :

- modèle explicite / implicite ;
- modèle individuel / stéréotype / générique ;
- modèle à court terme / long terme / statique / dynamique.

Les **modèles implicitement** codés sont les modèles utilisateurs types insérés par le concepteur dans le système tandis que les **modèles explicites** codent les informations acquises à propos de l'utilisateur dans un module séparé. Les approches actuelles utilisent des modèles explicites.

Le **modèle générique** suppose une population homogène d'utilisateurs ; c'est-à-dire que tous les utilisateurs d'une même classe sont traités de la même façon. Le **modèle individuel** essaie de modéliser l'information spécifique à chaque utilisateur. Le **modèle stéréotype** se situe entre ces deux modèles. Selon Rich toujours, un stéréotype est un groupe de caractéristiques liables entre elles. Si le stéréotype est activé par des actions de l'utilisateur, celui-ci sera activé dans son intégralité même si le déclencheur ne concerne qu'un aspect particulier du stéréotype. Si les modèles stéréotypes et génériques sont plus faciles à mettre en œuvre, leur pertinence repose sur la capacité à définir des classes d'utilisateur, ce qui dans la réalité est très souvent difficile ou pour le moins extrêmement réducteur.

Les modèles à long terme gardent uniquement les caractéristiques pérennes de l'utilisateur tandis que les modèles à court terme stockent les informations qui évoluent (non stationnarité intrinsèque de l'utilisateur qui apprend, décide, cherche, etc.). Le caractère dynamique / statique d'un modèle réfère à la capacité du modèle d'être mis à jour à la volée.

Notons que toutes ces familles de modèles sont généralement transposables aux autres aspects du contexte. Par exemple, on pourra trouver des stéréotypes de plateformes (PDA, PC, etc.), des modèles à court ou long terme de l'activité, etc.

Intéressons nous maintenant à ce que contient le modèle de l'utilisateur. Il est possible de distinguer plusieurs catégories de caractéristiques pour décrire les attitudes d'un utilisateur ou d'un groupe d'utilisateurs. Il est donc opportun de diviser le modèle utilisateur en différentes parties (Delestre, 2000) (Habieb-Mammar, Tarpin-Bernard *et al.*, 2003) :

- les préférences ;
- les expériences et connaissances ;
- les plans et buts courants ;
- les capacités perceptives, motrices et cognitives.

Le choix d'une présentation (modalité, métaphore, etc.) ou d'organisation de l'espace de travail fait partie des **préférences** de l'utilisateur. Contrairement aux autres caractéristiques, les préférences sont rarement déduites par le système car difficilement déductibles (hormis les préférences de consommation dans les sites web marchands par exemple). En effet, chaque utilisateur possède ses propres préférences, et c'est généralement lui qui les communique explicitement au système en personnalisant son environnement de travail. Le système utilise ces préférences pour des raisons d'adaptation, de sélection de stéréotypes, ou encore pour inférer des hypothèses sur l'utilisateur (Kobsa, 1993). La combinaison de différents modèles utilisateurs individuels favorise également l'extraction de modèles de groupes. Ces modèles seront employés comme des préférences de base pour les nouveaux utilisateurs dans le groupe (Brusilovsky, 1996a).

Par **expérience** de l'utilisateur, nous entendons toute information liée à l'expérience passée de celui-ci sans lien avec le domaine de l'application (par opposition aux connaissances que nous aborderons ensuite). Ceci comprend l'expérience professionnelle de l'utilisateur, sa formation, mais également, sa familiarité avec la structure et le fonctionnement général de l'outil. L'expérience est souvent modélisée par un stéréotype (débutant, initié et expert). Selon le niveau d'expertise de l'utilisateur pour une commande donnée, le système procédera par étape (guidage du débutant) ou, pour l'expert, présentera en une seule fois tous les paramètres nécessaires à la commande.

Les **connaissances** de l'utilisateur constituent une caractéristique importante pour les systèmes adaptatifs. La majorité des techniques de présentation adaptatives utilisent les connaissances de l'utilisateur comme source d'adaptation. Le modèle de connaissance est largement utilisé dans le domaine pédagogique. Un système adaptatif qui utilise les connaissances de l'utilisateur doit gérer les changements d'état de celles-ci et mettre à jour le modèle utilisateur en permanence. Par exemple, AHXEL (Adaptive Hypermedia using XML for E-Learning) est un système développé au laboratoire ICTT par S. Benadi (2004) qui utilise un modèle de structuration des activités pédagogiques réalisées sous forme hypermédia afin de les rendre adaptatives. Le modèle a pour caractéristique essentielle de faciliter la construction et la distribution d'activités pédagogiques adaptées au profil et au niveau de connaissance de chaque apprenant.

Les **capacités perceptives, motrices et cognitives** des utilisateurs ont, également, une forte influence sur leur aptitude à effectuer efficacement des tâches (Borgman, 1989) (Dufresne, 2001). Si ces informations sont essentielles pour la prise en charge du handicap, elles s'avèrent également très utiles pour les utilisateurs « normaux ». Notons qu'elles peuvent d'ailleurs être corrélées à des facteurs biologiques comme le sexe et l'âge mais également à l'expérience et en particulier au niveau d'étude. Il existe de nombreuses caractéristiques cognitives qui peuvent être pertinentes pour modéliser un utilisateur mais elles sont souvent négligées car difficile à manipuler. Ceci constitue une de nos contributions majeures largement développée dans le chapitre 3.

1.2.2 La plate-forme interactive

Les dispositifs d'interaction se diversifient par leur forme et leur finalité. L'ordinateur tout usage se voit prolongé d'appareils dédiés comme les assistants personnels numériques (PDA) et les téléphones mobiles. Inversement, avec la convergence télévision/numérique, un objet à finalité bornée devient un dispositif tout usage. Les PDA, incluent progressivement les services de téléphonie et inversement, les fabricants de téléphones font du "portable" un véritable PDA. Avec l'informatique ubiquitaire, pervasive ou diffuse (Weiser, 1993), l'espace et les objets de la vie courante deviennent des dispositifs d'interaction. Cette interaction peut être en entrée ou en sortie (Nigay, 1995) (Dragicevic, 2004). Il en résulte un foisonnement de solutions techniques correspondant chacune à des usages émergents.

Face à cette prolifération incontrôlée de plates-formes interactives, il est rapidement devenu nécessaire de construire des modèles pour faire face à une telle complexité.

L'une des premières solutions consistait à regrouper les supports dans des familles (PC, PDA, téléphone portable, etc.) et à proposer des versions pour chaque famille. Ainsi, l'outil CTTE (Mori *et al.,* 2002) par exemple donne la possibilité au concepteur de spécifier pour chaque tâche les familles de plates-formes pour lesquels cette tâche aura du sens. Bien que cette approche apporte un début de solution, le fait qu'elle se limite à la plate-forme physique et qu'elle ne prenne pas en compte la diversité des plates-formes et leurs caractéristiques spécifiques, rend cette classification par famille de supports incomplète. Comme nous le verrons plus loin, une de nos contributions dans ce domaine consiste à considérer les caractéristiques des dispositifs d'interaction de façon indépendante plutôt que de s'intéresser à des familles de dispositif. On s'intéresse ainsi aux caractéristiques du dispositif d'affichage, de celui de pointage ou de celui de saisie, ce qui permet de supporter des dispositifs hybrides comme un smartphone avec clavier.

Afin d'universaliser l'accès au web, le W3C a publié en 2004 une recommandation dite CC/PP (*Composite Capabilities / Preference Profiles*) (Graham *et al.,* 2004), prolongée aujourd'hui dans le cadre du DIWG (*Device Independence Working Group)*[2]. CC/PP représente un cadre de travail, basé sur XML *(eXtensible Markup Language)*[3], pour la description des caractéristiques logicielles et matérielles d'une plate-forme mais aussi les informations relatives aux préférences de l'utilisateur. Le profil CC/PP se base sur RDF (Resource Description Framework), un modèle pour l'utilisation des meta données sur le Web. Cette description peut être utilisée pour guider l'adaptation du système au profil de la plate-forme en question.

La structure générale d'un profil CC/PP prend la forme d'un arbre à deux niveaux : les **composants** et les **attributs**, chaque composant ayant la possibilité de faire appel à un ensemble de valeurs. Un profil CC/PP contient un ou plusieurs composants, qui contiennent à leur tour un ou plusieurs attributs. Chaque composant est représenté par une ressource de type *ccpp:component* (ou par certaines sous-classes RDF de celle-ci) et relié à la ressource du profil client par une propriété *rdf:description*. L'intérêt de l'utilisation d'un tel profil est qu'il se base sur des standards (XML, RDF) extensibles, supportant l'ajout de nouvelles dimensions du contexte et le développement des vocabulaires dédiés à des familles de plates-formes.

[2] http://www.w3.org/2001/di/
[3] http://www.w3.org/XML/

Le Profil des Agents Utilisateurs UAProf (*User Agent PROFile*) est un des exemples des vocabulaires basés sur le profil CC/PP. Développée par L'OMA[4] (*Open Mobile Alliance*), UAProf est un vocabulaire dédié à la description de téléphones mobiles. Comme exemple d'extension du profil CC/PP, citons les travaux de Lemlouma *et al.* (2002) qui proposent un profil nommé UPS. Ce profil étend le cadre de travail CC/PP pour couvrir, en plus des caractéristiques du client, la description détaillée des capacités des serveurs, des proxys, l'état du réseau, ainsi que les caractéristiques du contenu des données échangées. Un autre exemple est le modèle de la plate-forme proposé dans UsiXML (Limbourg *et al.,* 2004) qui repose également sur le profil CC/PP et sur le vocabulaire UAProf avec des attributs supplémentaires.

Finalement, nous pouvons distinguer deux grandes composantes pour le modèle de la plate-forme. D'une part, la **plate-forme physique** regroupe les informations sur le **support** employé pour exploiter le système interactif. Ces informations se déclinent en trois catégories : les dispositifs d'interaction en entrée (souris ou dispositif de pointage, clavier, microphone, etc.), les dispositifs d'interaction en sortie (écran, haut-parleur, etc.) et les capacités physiques du support (processeur, mémoire, capacités de calcul, etc.). D'autre part, la **plate-forme logicielle** regroupe les informations sur le système d'exploitation et l'ensemble des logiciels, machine virtuelle, et navigateur disponibles sur la plate-forme.

1.2.3 L'environnement

Grâce à la miniaturisation des objets communicants, la diversification des modes de travail, et les progrès de la communication sans fil et des capteurs de toute sorte (ex : dispositifs RFID[5]), l'accès au système interactif devient possible en tout lieu et à toute heure. Il en résulte des environnements d'interaction extrêmement diversifiés.

Pour mieux identifier l'environnement d'interaction nous proposons la définition suivante. Considérant un utilisateur qui interagit avec un système interactif à travers une plate-forme, l'**environnement**, se réfère à l'ensemble des informations exogènes au couple < utilisateur, plate-forme >, susceptibles d'influencer le déroulement de l'activité de l'utilisateur. Nous distinguons les caractéristiques ambiantes (ex : la luminosité ou le bruit), la localisation (géographique type GPS et sémantique comme la classe, la bibliothèque…), mais également des informations plus socio organisationnelles comme les responsabilités vis-à-vis de l'organisation, la dimension mono ou multi utilisateurs, l'accès à de l'assistance ou les risques d'interruption…

L'adaptation des systèmes interactifs à leur environnement d'utilisation est aujourd'hui un axe de recherche important. Cependant, l'exploitation des données liées à l'environnement et celles de la plate-forme sont souvent mal distinguées dans la littérature. Ainsi, la présence ou non d'une connexion réseau sans fil peut être considérée comme une information de la plate-forme ou de l'environnement selon les approches. Toutefois, des informations telle que la localisation (Beadle *et al.,* 1997) sont bien identifiées comme des données de l'environnement et commencent à être exploitées pour l'adaptation des systèmes. Par exemple pour indiquer à un touriste localisé à proximité d'un site historique des événements censés l'intéresser (Cheverst *et al.,* 2001) ou pour l'envoi de spots

[4] http://www.openmobilealliance.org/
[5] RFID (Radio Frequency Identification – Identification par fréquence radio)

publicitaires ciblés (Aalto *et al.,* 2004) ou encore la diffusion par SMS d'informations concernant les changements de la qualité de l'air dans une région (Peinel *et al.,* 2003).

Schmidt (Schmidt, 2000) propose un système qui exploite plusieurs données de l'environnement : ContextNotePad, service de bloc notes sur Palm Pilot. Ce système est sensible aux mouvements de faible amplitude (le système se met en marche dès que l'utilisateur le prend en main), au déplacement (il change la taille de l'affichage) et à la luminosité (l'écran passe en mode rétro éclairage si la lumière ambiante baisse).

Si l'adaptation à l'environnement se focalise actuellement sur l'exploitation des données de la localisation, d'autres exemples sont simples à imaginer et sur le point d'être disponibles. Chez soi, le PDA sert de télécommande universelle. Il s'adapte automatiquement à l'objet le plus proche, mais ne fonctionne pas lorsqu'il est actionné par un jeune enfant. En réunion, la sonnerie du téléphone passe automatiquement en mode vibreur ou au contraire utilise une sonnerie discernable dans un milieu bruyant. A la gare, le système de réservation de titre de transports conduit efficacement l'acheteur à son but et sans erreur malgré les conditions de stress (plusieurs personnes attendent et le train part dans cinq minutes!). Toutefois au domicile, le système se permet quelques digressions, offrant la possibilité de consulter les voyages en promotion. Toutes ces informations environnementales peuvent être aujourd'hui modélisées et donc utilisées par les processus d'adaptation.

A ce jour, nous n'avons pas encore exploré de façon opérationnelle des situations d'adaptation à l'environnement (hormis dans le contexte des collecticiels) bien que nos travaux sur les architectures logicielles tentent de prendre en compte cette dimension. Celle-ci constituera assurément une de nos futures directions de recherche.

1.3 Les espaces de conception de l'adaptation

Nous l'avons dit, notre principale préoccupation se résume à » comment rendre flexible un système interactif » c'est-à-dire comment l'adapter aux besoins de(s) l'utilisateur(s) et comment le doter de capacités d'auto adaptation ? Mais pour répondre correctement à cette question, il est indispensable d'élargir la problématique abordée dans la section précédente à savoir le « quoi » et le « pourquoi » par les autres dimensions d'une classique analyse QQOQCP à savoir Qui, Où, Quand et Comment ?

De nombreuses taxonomies pour identifier un espace d'adaptation des interfaces ont vu le jour pour tenter de répondre aux questions suivantes : à quoi s'adapter ? Qui adapte et quand a lieu l'adaptation ? Qu'adapte-t-on et Comment ? Selon quels critères ? Avec quoi ? (Thévenin, 2001) (Calvary, 2003) (Habieb-Mammar, 2004). Dans le même esprit, Vanderdonckt *et al.* (2005) présentent un espace de conception pour la prise en compte du contexte. Nous avons adapté cet espace à notre propre vision de la problématique (Figure 1).

Les axes dans cet espace représentent les questions qu'il faut poser pour l'identification du champ de l'adaptation. Sur chaque axe des éléments de réponse sont positionnés. Ces éléments sont souvent **complémentaires, parfois exclusifs**. La partie supérieure de cet espace décrit quel type de variation de contexte peut inciter l'adaptation (l'action qui cause l'adaptation : le pourquoi), tandis que la partie inférieure représente les éléments impliqués dans l'adaptation (la réaction par rapport au contexte).

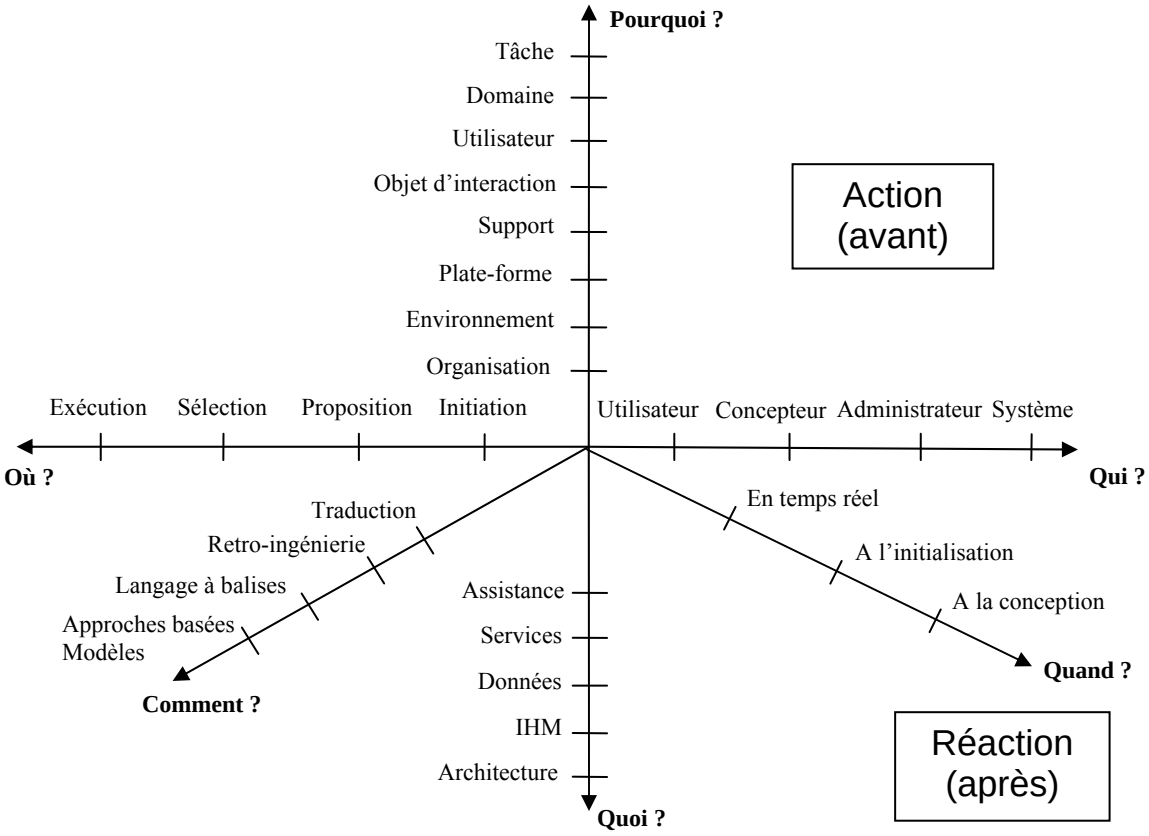

Figure 1 : L'espace de conception pour la prise en compte du contexte.

Notons que les axes Qui et Où sont très largement basés sur les travaux de Dieterich *et al.* (1993) et Kobsa (2001) qui identifient cinq type d'acteurs (intervenants) possibles dans le processus d'adaptation : concepteur, administrateur du système, expert local, utilisateur final et système. Ils introduisent également quatre niveaux dans le processus d'adaptation où il convient d'intervenir :

- initiation : décision d'un acteur de suggérer une adaptation ;
- proposition : élaboration de suggestions ou recommandations ;
- sélection : processus de choix parmi les propositions d'adaptation ;
- exécution : mise en œuvre de l'adaptation choisie.

Le contrôle de l'utilisateur s'exerce sur ces niveaux, et définit sur cette base ce qui distingue l'adaptabilité de l'adaptativité.

Les systèmes où l'utilisateur est responsable de l'initiation, de la proposition, de la sélection et de la production d'adaptation sont appelés **adaptables**. Par exemple, un utilisateur d'un site Web a besoin de présenter un raccourci pour une page du Web qu'il visite fréquemment (initiation d'adaptation). L'utilisateur propose alors de créer un lien sur la barre de navigation (si cela s'avère possible) ou définir un signet dans le navigateur. Ensuite, il choisit le lien de raccourci et exécute les étapes nécessaires pour produire cette adaptation.

En revanche, les systèmes qui sont capables d'exécuter automatiquement toutes ces étapes sans l'intervention de l'utilisateur sont appelés **adaptatifs**. Par exemple, le système

AVANTI (Fink *et al.,* 1997) insère automatiquement des liens de raccourci personnalisés dans des pages qu'un utilisateur visite fréquemment.

Un processus d'adaptation peut également relever de configurations qui combinent les caractéristiques de l'adaptabilité (i.e. contrôle émanant de l'utilisateur) et de l'adaptativité (i.e. contrôle émanant du système).

Notons cependant que pour (Stephanidis *et al.,* 1998), l'adaptabilité fait référence à un processus d'adaptation basé sur des connaissances (l'utilisateur, son environnement, etc.) disponibles ou acquises par le système avant que ne soient engagées les interactions utilisateur/système. Les adaptations sont donc réalisées lors de l'initialisation du système qui se présente dans une version adaptée à l'utilisateur. Les connaissances utilisées par le système sont, de plus, supposées rester inchangées au cours de la session d'utilisation. L'adaptativité traduit une vision plus dynamique du processus d'adaptation. Les connaissances sont ici acquises ou modifiées par le système au cours des interactions, via des techniques de suivi de session (apprentissage, détection d'évènements, etc.). Le système procède à des adaptations pendant que l'utilisateur interagit avec lui. Cette vision est également partagée par (Frasincar *et al.,* 2002), qui appellent l'adaptabilité **adaptation statique** et l'adaptativité **adaptation dynamique**.

Comme nous le verrons plus tard, nous partageons l'idée que l'adaptation doit combiner ces différentes formes d'adaptation statique et dynamique. En fait, l'utilisabilité et plus généralement l'ergonomie d'un système adaptatif requièrent des reconfigurations complexes lorsque le changement de contexte est important. Or, comme nous l'avons vu en introduction en présentant les limites de l'adaptation, il est extrêmement difficile voire hasardeux d'élaborer des processus entièrement automatique de reconfiguration. Ceci souligne l'importance d'adopter des démarches qualités pour évaluer le bien-fondé de l'adaptation effective (Cf. Chapitre 4).

1.4 Les approches existantes pour adapter une IHM

Parmi les différents axes de l'espace de conception de l'adaptation, il nous reste à aborder celui qui est au cœur de nos préoccupations : le comment ? Nous l'avons clairement dit en introduction : notre objectif général est de fournir des modèles, méthodes et outils pour construire et implémenter à moindre coût des applications interactives adaptatives et/ou facilement adaptables à divers contextes d'utilisation. Dans la section qui suit, nous allons dresser un rapide état de l'art des approches existantes pour adapter une IHM dont certaines peuvent également adapter d'autres aspects des applications (Cf. § 1.1).

Pour mieux comprendre les différentes approches employées pour l'adaptation des systèmes interactifs, nous proposons de les classer dans quatre familles :

- La traduction des interfaces : cette solution est employée largement dans la traduction des pages web, mais il existe également des traductions plus complexes comme HTML vers Java et vice-versa ;
- La rétro-ingénierie et la migration des interfaces : il s'agit d'analyser et de segmenter une application de manière à en extraire une représentation plus ou moins abstraite afin de reconstruire une nouvelle interface ;
- Les approches basées sur les langages à balises (*markup languages*) : l'objectif est de créer une description indépendante de la plate-forme à partir d'un langage facilement utilisable dans des environnements hétérogènes ;

- Les approches à base de modèles : elles s'appuient sur un ensemble de modèles (tâches, domaine, présentation, dialogue, etc.) pour la conception et la génération des systèmes interactifs.

Nous justifions cette catégorisation par le fait que ces approches emploient des techniques d'adaptation bien différentes et que leur point de départ et leurs étapes intermédiaires sont distincts. Les deux premières, à savoir la traduction et la rétro-ingénierie (dans sa première phase), s'insèrent dans une logique ascendante. Elles partent des interfaces déjà existantes pour un contexte spécifique pour en retirer de nouvelles interfaces. Les deux dernières approches sont des démarches descendantes. Elles partent d'une spécification du système (à différents niveaux d'abstraction), pour en extraire l'interface de l'application. Aujourd'hui, une certaine porosité existe entre ces 4 grandes familles puisqu'il est fréquent que les approches à base de modèles s'appuient sur des langages à balises, de même que dans le cas de la rétro-ingénierie, la construction de la représentation abstraite de l'IHM peut reprendre certains principes des approches à modèles. Malgré cette tendance naturelle à combiner les avantages des différentes approches, nous allons voir que cette catégorisation permet de mieux comprendre les principes de base de l'adaptation des IHM.

1.4.1 La traduction des IHM

Même si la traduction automatique des interfaces représente une solution rapide qui profite des solutions déjà existantes, les résultats obtenus sont généralement peu satisfaisants car les transformations supportées sont essentiellement d'ordre syntaxique, ne sont pas paramétrables et sont limitées à certaines opérations de base. Elles ne permettent pas de sélection des composants à conserver ou à ignorer en fonction des contextes. Ceci pose des problèmes ergonomiques d'utilisabilité.

La conception par dégradation élégante (Florins *et al.*, 2004) a récemment proposé un début de réponse à ce problème. Elle commence par la spécification d'une interface utilisateur conçue pour la plate-forme possédant les meilleures capacités, en termes notamment de résolution d'écran et de bibliothèques graphiques disponibles. Lors de la phase de conception, des règles d'adaptation appelées règles de dégradation sont proposées par un système d'aide à la conception et sélectionnées par le concepteur afin d'adapter la présentation à la plate-forme cible. Toutefois, la solution de traduction des interfaces reste essentiellement limitée au passage d'un langage à un autre sur la même classe de plate-forme.

1.4.2 La rétro-ingénierie

La rétro-ingénierie (ou ingénierie inverse) consiste à repenser ce qui a été conçu dans une démarche d'ingénierie. En informatique, la rétro-ingénierie consiste à analyser un produit fini (un système d'information, des processus, un logiciel ou des interfaces) pour déterminer la manière dont celui-ci a été conçu et identifier ses composants et leurs dépendances (Müller *et al.*, 2000).

Le processus de rétro-ingénierie d'IHM (Chikofsky et Cross, 1990) d'un contexte d'utilisation vers un autre peut être décomposé en deux étapes successives :

- l'ingénierie régressive (ou abstraction ou réification inverse) qui consiste à analyser une application de manière à en extraire une représentation abstraite (indépendante de la plate-forme et du langage) ;

- l'ingénierie progressive (ou concrétisation ou réification) durant laquelle une nouvelle interface est générée à partir des spécifications actualisées et d'une représentation abstraite.

Les premiers travaux sur la rétro-ingénierie avaient pour objectif de remplacer l'interface textuelle d'applications anciennes par une interface graphique. Aujourd'hui, les travaux s'orientent plutôt vers la rétro-ingénierie des pages web pour les rendre disponibles pour plusieurs plates-formes. Javahery et al. (2004) proposent une démarche qui se base sur les *patterns*[6] et principalement sur les *patterns* d'utilisabilité (*Usability Patterns*) dans l'objectif de générer des interfaces utilisateur multiples (MUI) (Seffah et Javahery, 2004). Paganelli et Paternò (2003) proposent une démarche de rétro-ingénierie des pages web qui permet d'obtenir une description du modèle de tâches exprimé en CTT (Paternò, 1999). VAQUITA (Bouillon *et al.*, 2004) est un autre environnement qui permet d'effectuer une rétro-ingénierie des pages web de manière à en obtenir un modèle de présentation indépendant du contexte d'utilisation. Il offre la possibilité d'obtenir plusieurs modèles de présentation à partir d'une même page par la sélection d'heuristiques de rétro-ingénierie différentes. En revanche, cette approche est utilisée uniquement pour générer une représentation abstraite de la page web. Elle ne s'intéresse pas à la phase de régénération des nouvelles interfaces.

Pour exprimer la spécification abstraite de l'interface, ces approches utilisent aujourd'hui de manière croissante des langages à balises utilisés à l'origine dans des démarches descendantes.

1.4.3 Les approches basées sur les langages à balises

Les langages à balises (*Markup Language*), popularisés par le développement d'Internet, présentent un certain nombre d'atouts comme la simplicité d'utilisation, l'interopérabilité et la multitude des outils qui les supportent. Ils ont rendu possible la **séparation du contenu d'un document de sa forme** à travers la possibilité de recourir à des **feuilles de styles** utilisés pour l'externalisation de la définition de la présentation d'un document (les couleurs, les polices, le rendu et d'autres caractéristiques). Ceci tout d'abord avec les CSS (*Cascading Style Sheets*) associées à HTML puis avec XSL et les langages associés, proposés autour de XML. Dès lors, la création de différentes feuilles de style (qui contiennent les spécifications des adaptations souhaitées) permet de produire différents documents à partir d'un même contenu.

Ces cinq dernières années, le nombre de langages de description des interfaces utilisateur dit UIDLs (*User Interface Description Language*) pour le développement des interfaces utilisateur multi-supports ne cesse d'augmenter. Les efforts dans ce domaine se concentrent essentiellement autour des langages basés sur XML (Luyten *et al.*, 2004). Des moteurs de rendu traitent les fichiers de description pour les adapter à la technologie support (HTML, WAP, VoiceXML, etc.).

Dans leur mise en œuvre, toutes ces propositions suivent deux types d'approches :

- Définir des **balises génériques pour les objets d'interaction abstraite** (AIO). La concrétisation de l'interface passe par le remplacement des AIO par des objets d'interactions concrets (CIO) compatibles à la plate-forme cible. Par

[6] Solution récurrente décrivant et résolvant un problème général dans un contexte particulier. Il existe différents types de *patterns* : d'analyse, de conception, d'implémentation…

exemple une balise <part class="Button " id="MonBouton" /> dans UIML sera traduite par une balise <Button name= "MonBouton"/> par un moteur de rendu HTML ou conduira à l'instanciation d'un objet JButton par le moteur de rendu Java Swing.

- Décrire l'**objectif** de l'utilisateur (choisir commande, sélectionner élément, etc.). Les langages XForms et AUIML permettent par exemple d'exprimer que l'objectif de l'utilisateur est d'effectuer un choix. Cette tâche peut alors être concrétisée par un ensemble de boutons radio ou par un menu déroulant selon les caractéristiques de la plate-forme.

	Langages cibles	**Plate-forme**	**Approche**	**Outils associés**
XUL (Boswell *et al.*, 2004)	XUL	PC standard (Navigateur Mozilla)	Balises génériques	Moteurs de rendus, Mozilla
Xforms (Dubinko *et al.*, 2003)	HTML, XHTML, WML,	PC standard, Terminaux mobiles.	Objectif	Editeur Xforms
UIML (Abrams et Helms, 2004)	HTML, WML, VoiceXML, .Net, Java, c++.	PC standard, Terminaux mobiles, IU Vocale.	Balises génériques	Liquid UI: générateur de code (HTML et Swing), UIML2ALL.
AUIML (Azevedo *et al.*, 2000)	HTML, DHTML, java swing, WML	PC standard, Terminaux mobiles,	Objectif	AUIML Visual Builder, plug-in Eclipse : Assistant et Générateur.
Plastic ML (Rouillard, 2003)	HTML, WML, VoiceXML	PC standard, Terminaux mobiles, IU Vocale.	Balises génériques	ToolKit Plastic ML: assistant et générateur de code.
RIML (Koskimies *et al.*, 2004)	HTML, XHTML, WML, VoiceXml	PC standard, Terminaux mobiles, IU Vocale.	Balises génériques, Objectif	Moteurs de rendus.

Tableau 2 : Différents langages de description des interfaces utilisateurs abstraites.

Les langages que nous présentons dans le Tableau 2 se limitent à la description du modèle de présentation et du modèle de dialogue dans un objectif d'adaptation de cette présentation à la plate-forme. Si cette démarche s'avère intéressante pour des interfaces basées sur les formulaires ou des interfaces orientées Web en général, elle reste peu efficace pour des systèmes interactifs plus complexes.

Ainsi, d'autres langages et approches tentent de prendre en considération plus d'éléments et de modèles dans une démarche plus générique de conception et d'adaptation des systèmes interactifs. Nous regroupons ces approches sous le nom d'approches à base de modèles.

1.4.4 Les approches à base de modèles

Les approches à base de modèles s'appuient sur un ensemble de modèles (tâches, domaine, présentation, dialogue, etc.) pour la conception et la génération des système interactifs. Dans une démarche descendante, ces approches procèdent à une concrétisation progressive des spécifications de modèles pour la génération de l'interface finale du système interactif (Van den Bergh et Coninx, 2004).

Ces approches cherchent à prendre en considération plusieurs modèles sous-jacents à un système interactif et ne se limitent pas à l'adaptation du modèle de la présentation. Dans ces approches, **certains modèles (tels que les modèles de tâches, de dialogue ou de l'interface) subissent l'adaptation tandis que d'autres modèles (contexte, transition, évolution) influencent cette adaptation**. Le projet CAMELEON (Calvary *et al.*, 2003) propose un cadre de référence qui unifie les approches à base de modèles pour la conception et l'exécution d'IHM plastiques. Il structure le processus en quatre niveaux de réification ou concrétisation (Figure 2). Il est important de signaler que le nombre de modèles pris en considération ainsi que le niveau de contextualisation d'un modèle (l'obtention d'un modèle propre au contexte) diffèrent selon les approches.

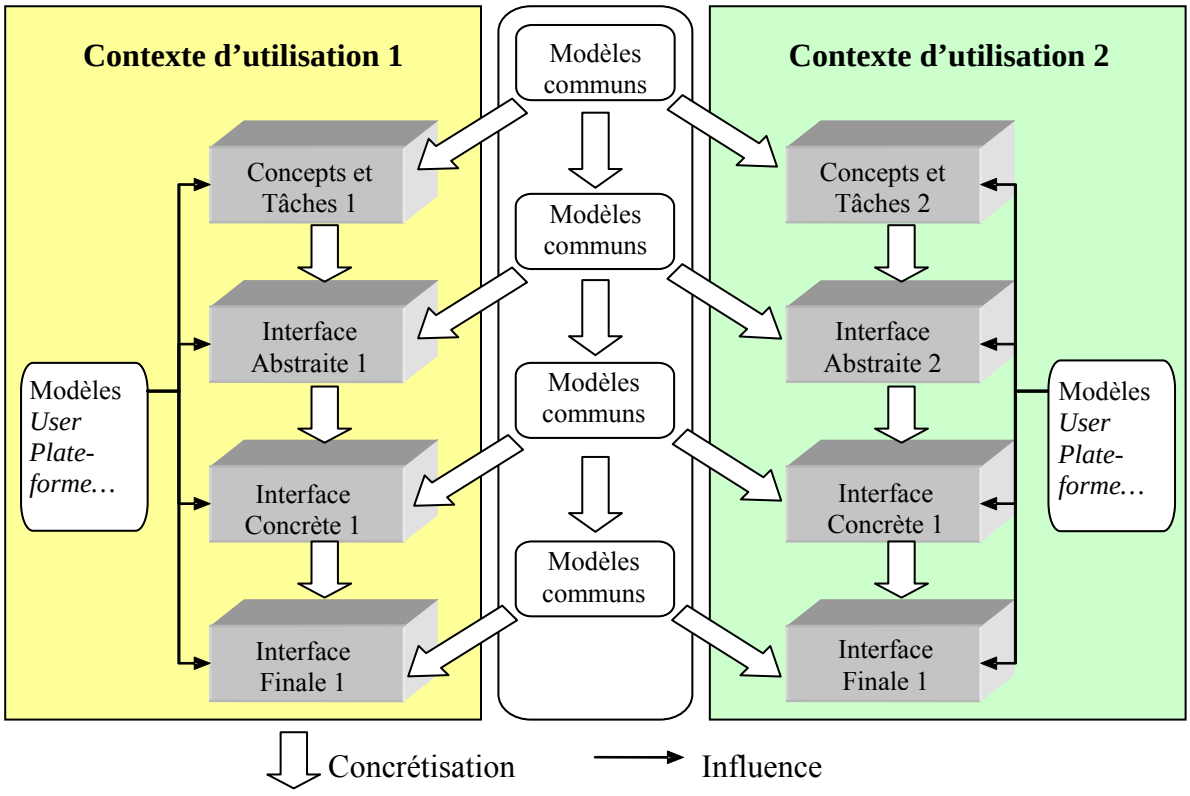

Figure 2 : Approche à base de modèles pour l'adaptation des interfaces (adaptée de Calvary *et al.*, 2003)

Les approches à base de modèles définissent comment construire progressivement les différents modèles, jusqu'à obtenir l'ensemble des caractéristiques nécessaires à la génération d'interfaces finales. La distinction entre niveaux d'abstraction d'interface permet notamment d'intégrer les contraintes de la conception multi-contextes aux modèles qui définissent le système interactif. Cette prise en considération favorise l'adaptation de l'ensemble du système interactif et se reflète en dernière étape sur le modèle de la présentation. En prenant en compte les modèles de tâches notamment, les approches à base

de modèles sont sans doute celles conduisent aux meilleurs résultats sur le plan de l'utilisabilité des versions adaptées lorsque les contextes d'utilisation varient fortement.

Le Tableau 3 récapitule différentes approches à base de modèles en spécifiant les modèles pris en compte, le type de contexte cible, les langages de spécification, la méthodologie employée et les outils associés.

	Modèles	**Contexte**	**Langage de spécification**	**Méthodologie**	**Outils associés**
XIML (Puerta et Eisenstein, 2004)	Tâches, Domaine, Dialogue, Présentation.	Utilisateur,	XIML	Description générique.	Des outils qui utilisent XIML : DiaTask, VAQUITA.
UsiXML (Limbourg *et al*., 2004)	Domaine, tâches, IUA, IUC, transformation, correspondance.	Plate-forme, Utilisateur, Environnement	UsiXML	Description générique, collaboration entre modèles. Génération assistée	SketchiXML, GrafiXML, FlashiXML, QtkXML, IdealXML, TransformiXML
ART-Studio (Thévenin, 2001)	Domaine, tâches, IUA, IUC, IUF	Plate-forme	XML	Génération assistée, Interface plastique.	ART-Studio
TERESA (Mori *et al*., 2004)	Domaine, tâches, IUA, IUC, IUF	Plate-forme	TERESA XML, CTT XML	Centré tâche, Génération assistée.	TERESA, CTTE
Dygimes (Luyten, 2004)	Domaine, tâches, Dialogues, IUA, IUC	Plate-forme	CTT XML, SEESCOA XML	Collaboration tâche-dialogue, Génération automatique	Moteur de rendu (UiBuilder), Layout Specification, TaskLib,
Comets (Calvary *et al*., 2005)	Domaine, tâches, IUA, IUC, IUF, Transformation, Evolution.	Plate-forme, Utilisateur, Environnement	Non	Couplage approche modèles et interacteur	Non

Tableau 3 *:* Les principales approches à base de modèles

Chaque approche, met en avant un certain nombre de modèles et cible l'adaptation au contexte ou à des parties du contexte en employant différentes méthodologies d'adaptation. Cependant, quand certaines approches se concentrent sur le modèle de tâches comme modèle central pour l'adaptation (ex : TERESA), d'autres font collaborer le modèle de tâches avec le modèle de dialogue (ex : Dygimes). En outre, certaines approches proposent un support particulier à chaque modèle (ex : UsiXML) accompagné d'une palette d'outils pour relier ces modèles. D'autres approches, préfèrent encapsuler la description des modèles dans un composant (ex : Comets) en le dotant ainsi de capacités d'adaptation et d'auto-adaptation.

Aujourd'hui, et dans un objectif de convergence (autour de l'OMG), des efforts considérables sont menés pour calquer les approches à base de modèles sur l'approche MDA (*Model Driven Architecture*) (David *et al.*, 2005) (Sottet *et al.*, 2005).

1.4.5 Synthèse des différentes approches

A partir de l'état de l'art précédent, nous avons dressé le tableau comparatif suivant :

Approche	Avantages	Inconvénients
Traduction de l'interface	☺ rapide ☺ automatique ☺ réutilisation de solutions existantes	☹ orientée Web ☹ transformations d'ordre syntaxique ☹ peu paramétrables
Rétro-ingénierie	☺ + ou – rapide ☺ semi-automatique	☹ dirigée par la solution de départ ☹ niveau d'abstraction et de paramétrage différents selon les approches
Langage à balises	☺ très adaptée aux hypermédias ☺ automatique ☺ extensible	☹ limitée généralement au modèle de la présentation ☹ moteur de rendu ou CCS spécifique pour chaque plate-forme
Approche à base de modèles	☺ adaptation progressive ☺ qualité ergonomique des résultats de l'adaptation ☺ extensible	☹ liens difficiles entre les modèles ☹ complexe à mettre en œuvre

Tableau 4 : Avantages et inconvénients des grandes familles d'approches pour l'adaptation de l'IHM

L'usage de langages à balises est une approche très efficace. Nous avons d'ailleurs eu l'occasion de la tester lorsque nous cherchions à construire des hypermédias adaptatifs (Cf. section 3.5 à venir).

Elle est très adaptée à ce type de problématique mais s'avère insuffisante si l'on cherche à modéliser des applications interactives plus riches. Seules les approches à base de modèles (qui d'ailleurs reposent souvent sur des langages à balises) nous semblent capables d'être utilisable dans ce genre de situation, où l'on désire à la fois maîtriser les coûts et fournir des applications adaptatives de bonne qualité ergonomique profitant pleinement des dispositifs d'interaction offerts. L'évaluation de ces critères (coûts et qualité) reste un sujet difficile sur lequel nous comptons travailler dans les années à venir.

Dans le chapitre qui suit, nous allons présenter l'approche que nous avons élaborée au fil des années avec K. Samaan, les étudiants de Master que j'ai encadrés (J. Poquet, C. D. Tran) ainsi qu'avec le Pr. B. David et son doctorant G. Masserey.

2 Méthodes de conception et de construction d'applications adaptables

2.1 Introduction

Nous considérons comme une évolution normale que le logiciel devienne un produit (industriel) comme un autre, qu'il puisse faire l'objet des mêmes méthodes de conception – étude – développement – fabrication – contrôle qualité – homologation…
Malheureusement, hormis un nombre restreint de domaines très spécifiques, la grande majorité des logiciels n'a pas encore atteint ce stade de maturité.

S'il existe un certain nombre d'explications à cette situation, il est indéniable que les progrès effectués en trente ans en matière de conception et de réalisation de logiciels sont considérables. On peut même dire qu'aucune industrie n'a fait de tels progrès en si peu de temps. Aujourd'hui, les langages de développement proposés aux programmeurs contiennent tous les ingrédients nécessaires à l'amélioration de la qualité des logiciels produits. Pour atteindre cet objectif, un effort conséquent est actuellement réalisé sur l'amélioration des modèles d'architectures sous-jacents qui permettent de travailler sur un **modèle unifié** depuis les premières phases de spécification et de conception jusqu'à l'implémentation. Ce modèle doit favoriser la génération d'applications fiables et évolutives par **composition d'éléments réutilisables**. En outre, les ateliers de génie logiciel associés évoluent également très rapidement afin que les informaticiens disposent de tous les moyens indispensables à une réelle industrialisation de l'informatique.

Aujourd'hui, les plus gros obstacles sont sans doute d'ordres **méthodologique** et **organisationnel** car le développement tend à devenir de plus en plus concourant et implique des équipes de travail de plus en plus nombreuses. De plus, depuis quelques années on constate une prise de conscience du fait que la conception des logiciels n'est pas seulement l'apanage des informaticiens mais qu'elle doit résulter de coopérations interdisciplinaires impliquant notamment les utilisateurs finaux de ces logiciels.

En effet, en retraçant l'évolution des méthodes de conception des logiciels interactifs nous avons montré qu'il n'était plus pertinent d'appliquer une approche technocentrée mais que l'activité des utilisateurs devait être pleinement prise en compte à partir d'approches centrées sur l'usager. Dans cette perspective, nous avons mis en lumière les limites des modèles théoriques de l'interaction issus du paradigme cognitiviste et l'intérêt des approches issues des théories de l'action qui permettent, notamment, de contextualiser l'activité humaine, d'articuler les dimensions sociales et cognitives, et un cadre interprétatif pour appréhender la fonction médiatrice des artefacts pour l'activité.

De fait, pour prendre en compte les multiples facettes de cette problématique une des solutions les plus adaptées consiste à impliquer les futurs utilisateurs dès la conception des produits, reprenant par la même le concept d'ingénierie concourante du génie industriel. Ainsi, de nombreuses techniques de conception participative peuvent être mises en œuvre pour obtenir une conception consensuelle dans laquelle se retrouvent concepteurs et usagers.

Sur le plan technique, ces nouvelles pratiques doivent s'accompagner d'une évolution de l'outillage informatique support de la conception à savoir : les modèles d'architectures, les bibliothèques de composants réutilisables et, plus largement, les méthodologies de conception associées.

Pour construire des logiciels utiles, utilisables et donc acceptables pour l'utilisateur, c'est sur ce triptyque, modèle théorique – pratique de conception « participative » – environnement informatique support, que doivent s'appuyer les méthodologies modernes de conception des logiciels interactifs.

Par ailleurs, il nous semble important de prendre en compte le concept de co-évolution introduit par Bourguin *et al.* (2001) et qui consiste à partager la (re)conception d'un système entre les acteurs (utilisateurs, experts, informaticiens…) dans un environnement supportant un domaine d'activité, mais supportant aussi sa propre activité coopérative de (re)conception. Dans ses derniers travaux, Bourguin souligne l'importance de considérer la méta-activité qui consiste à faire évoluer une activité donnée, comme une activité à part entière et pas seulement une extension de la dite activité (Bourguin et Derycke, 2005). Dans notre contexte, cela signifie que si l'on souhaite laisser un certain nombre de degrés de liberté à l'utilisateur pour l'adaptation d'un système, l'activité d'adaptation doit être considérée comme une activité à part entière, éventuellement collaborative, et qu'elle ne peut se résumer au paramétrage d'un outil.

La co-évolution conduit également à considérer les systèmes socio-techniques comme des systèmes dont les composants (humains et techniques) interagissent et influencent leurs évolutions mutuelles. De fait, il est nécessaire que les concepteurs des systèmes comprennent les structures sous-jacentes et les natures des changements potentiels, les processus d'adaptation étant généralement non prédictibles (Kaplan & Seebeck, 2001).

Parce que l'adaptation par l'utilisateur d'un système nécessite obligatoirement un certain niveau de compréhension de la structure des logiciels et de leurs modes de fonctionnement, Mørch (2000) milite quant à lui pour une collaboration indirecte entre les concepteurs et les utilisateurs finaux passant par l'utilisation de représentations multiples (modèles plus ou moins orientées système) permettant à ces derniers de mieux comprendre la structure des logiciels.

De ces différentes analyses, nous avons retiré deux éléments clés. Il est tout d'abord indispensable de travailler avec des **formalismes utilisables durant la totalité du cycle de vie des logiciels**, depuis les phases amont de conception jusqu'à l'implémentation. En effet, si la conception générale est un problème crucial qui est aujourd'hui assez bien cerné, il est fréquent, en particulier lorsqu'il s'agit d'élaborer des applications interactives, que le passage entre la conception générale et la conception détaillée et surtout l'implémentation se fasse difficilement. Faute d'outils et méthodes adaptés, le développeur est souvent livré à lui-même pour tenter de produire une application en respectant des modèles plus ou moins abstraits. Les outils industriels et en particulier de production multimédia s'avèrent malheureusement souvent inadaptés. Le second point porte sur l'importance de trouver des **formalismes susceptibles d'être compris, voire manipulés, par les différents acteurs d'un projet informatique et en particulier les utilisateurs**, notamment durant l'exécution du logiciel. Dans la suite de ce chapitre, nous allons montrer qu'il nous semble possible de satisfaire ces deux enjeux en s'appuyant sur un modèle d'architecture pivot qui cristallise les différents modèles du système et assure le fonctionnement du système.

Publications :
- DAVID B.T., TARPIN-BERNARD F. Evolution de la conception des logiciels in *Conception : entre science et art.* Presse Polytechniques et Universitaires Romande, Lausanne. Perrin J. (eds). Juillet 2001. pp. 61-77.

- VACHERAND-REVEL J., TARPIN-BERNARD F., DAVID B.T., Des modèles de l'interaction à la conception "participative" des logiciels interactifs in *Conception : entre science et art.* Presse Polytechnique Romande. Lausanne. Perrin J. (eds). Juillet 2001. pp. 239-25

2.2 Notre approche : un modèle d'architecture pivot

Pour des raisons de maîtrise de la complexité et des coûts, les multiples dimensions de la notion de contexte imposent de mettre en place des solutions modulaires, standardisées mais suffisamment flexibles non seulement pour supporter la combinatoire qui en résulte, mais aussi pour anticiper le support de dispositifs d'interaction innovants (gants numériques, objets communicants à base de technologie RFID[7], etc.).

Avec un tel objectif, les approches à base de modèles prennent une importance capitale car elles proposent des solutions modulaires dans des démarches descendantes de spécification et de construction de systèmes interactifs. Le problème dans ces approches est que les liens entre ces modèles les uns avec les autres et surtout entre le modèle du domaine et les autres modèles restent plus ou moins explicites selon les approches. Un besoin émerge alors clairement. **Comment exprimer d'une façon claire et explicite les relations entre les modèles de la présentation ou le modèle des tâches d'un côté et le modèle du domaine de l'autre ?**

Pour répondre à cette question, nous nous sommes appuyés tout d'abord sur l'expertise que nous avions développée à l'occasion de la thèse de doctorat dans le domaine des modèles d'architecture logicielle (Tarpin-Bernard, 1997). En effet, le modèle d'architecture logicielle constitue un des éléments indispensables sur lequel s'appuie le génie logiciel moderne. Il systématise à la fois l'élaboration et l'utilisation des logiciels et joue un rôle primordial dans leur **qualité** et leur **efficacité**. Garlan et Perry soulignent le fait que l'utilisation de modèles d'architecture peut avoir des effets bénéfiques sur au moins cinq aspects du développement des logiciels (Garlan *et al.,* 1995) :

- la **compréhension** est simplifiée en présentant la structure de grands systèmes à un niveau d'abstraction adapté. Les contraintes et choix de conception peuvent ainsi être mieux explicités et expliqués ;
- la **réutilisation** est favorisée. Bien sûr, la réutilisation des composants est largement encouragée par la définition d'architectures, mais il est également possible de réutiliser des structures plus complexes définies sous forme de motifs de conception ou *design patterns* ;
- l'**évolution** et la **maintenance** profitent largement de la définition d'architectures logicielles. En comprenant les ramifications et les connexions de chaque composant, il est beaucoup plus facile de déterminer a priori les coûts potentiels de toute modification ;
- l'**analyse** des systèmes est simplifiée. Suivant les modèles utilisés, on peut envisager des formes de vérification de cohérence, de respect de styles, de satisfaction de critères de qualité voire d'adaptation à des domaines spécifiques d'application ;

[7] RFID (Radio Frequency Identification – Identification par fréquence radio)

- la **gestion** des projets industriels doit s'appuyer sur cet élément essentiel pour déterminer au plus tôt la viabilité des options choisies, les prérequis indispensables et les perspectives d'évolution.

Nous adhérons à l'analyse de Garlan et Perry et nous considérons que, par **le rapprochement du modèle de fonctionnement et du modèle d'utilisation**, le découpage en entités autonomes peut servir de support d'explication et être utile non seulement aux concepteurs mais également aux utilisateurs. En intégrant les mécanismes et contraintes du système, ils peuvent mieux se l'approprier. Le modèle d'architecture constitue le pivot des activités d'élaboration (spécification, conception et réalisation). On peut ainsi spécifier quatre objectifs pour le modèle d'architecture :

- servir de support lors de la spécification (en tant que formalisme) ;
- constituer l'ossature de la réalisation (en tant que *framework*) ;
- assurer la cohérence de fonctionnement à l'exécution (au *runtime*)
- servir de guide d'utilisation et/ou de support à la reconfiguration par l'utilisateur.

Les modèles d'architecture partent du principe qu'un système interactif comporte une partie **interface** et une partie **noyau fonctionnel** qui se réfère au domaine de l'application. Le noyau fonctionnel est considéré comme préexistant, et les modèles de systèmes interactifs décrivent essentiellement la partie interface et ses relations avec les objets du domaine.

La plupart des modèles identifient *au moins* trois types d'éléments :

- les éléments en contact direct avec l'utilisateur (présentations) ;
- les éléments en contact direct avec le noyau fonctionnel ou qui en font partie (abstractions, interfaces du noyau fonctionnel, etc.) ;
- les éléments de communication entre les deux premiers éléments (contrôleurs, adaptateurs).

Malgré ces points communs, certains modèles procèdent d'approches différentes, et ne se limitent pas à ce découpage sommaire. Nous ne détaillerons pas ici les différents modèles d'architecture mais notons simplement qu'ils peuvent être classés en trois grandes catégories. Les **modèles à couches** décrivent la structure globale d'une application interactive sous forme de couches logiques (ex : Seeheim (Pfaff, 1985) et Arch (UIMS, 1992)). Ces modèles sont également appelés modèles logiques, ou sont qualifiés d'abstraits. Les **modèles à agents**, ou encore modèles orientés objet (ex : MVC (Goldberg, 1984), PAC (Coutaz, 1987), AMF (Ouadou, 1994)). Ces modèles proposent une décomposition de l'application en un ensemble d'agents communicants séparant en leur sein l'interface du noyau fonctionnel. Enfin, des **modèles mixtes** tel que PAC-Amodeus (Nigay, 1993) ou H4 (Guittet, 1995) tentent de tirer partie des avantages respectifs de ces deux approches.

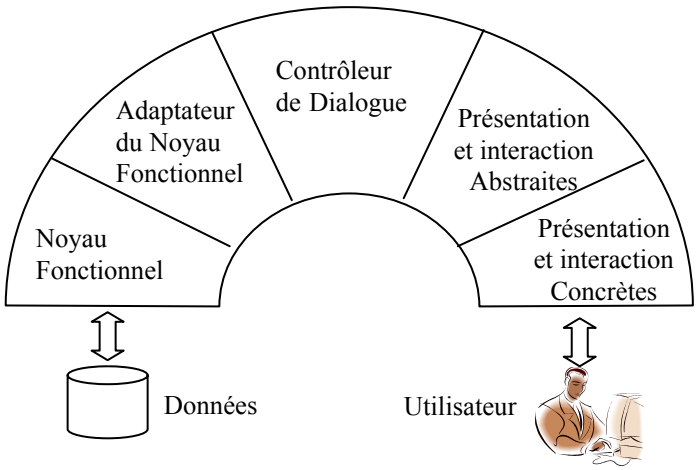

Figure 3 : Les composants du modèle Arch

Rappelons rapidement les bases du modèle Arch (Figure 3). Du côté du pilier applicatif, il distingue 2 niveaux :

- le **noyau fonctionnel** de l'application qui gère les données internes et leur rémanence ;
- l'**adaptateur du noyau fonctionnel** qui est un composant médiateur entre le contrôleur de dialogue et le noyau fonctionnel. Il est responsable de la réorganisation des données du domaine dans des buts de manipulation interactive, ou de la détection des erreurs sémantiques.

Du côté du « pilier » présentation, il décompose l'interface utilisateur en 2 niveaux :

- un **niveau abstrait**, comportant les objets de présentation et d'interaction indépendants de l'interface concrète et donc des dispositifs d'interaction avec l'utilisateur ;
- un **niveau concret**, lié à un « *toolkit* graphique » précis, gérant le dialogue avec l'utilisateur (et donc lié à des dispositifs d'entrée/sortie précis).

Dans le modèle original (UIMS, 1992) ces deux couches sont appelées (*Presentation Component*) et (*Interaction Toolkit Component*) ce qui, en référence à MVC, peut porter à confusion en laissant sous-entendre qu'une couche est dédiée aux sorties (affichage à l'écran) alors que l'autre se consacre aux entrées (souris, clavier) ce qui est évidement faux. C'est pourquoi, nous rencontrons dans la littérature d'autres termes comme « Présentation logique » et « Présentation physique ». Mais le mot « physique » ne nous semble pas approprié car, l'usage de ce mot peut prêter à confusion avec les objets physiques ou l'instrument d'interaction. Nous trouvons que « Présentation et Interaction abstraites » et « Présentation et Interaction concrètes » proposés par (Chalon, 2004) sont plus appropriés.

On le voit, les modèles d'architecture ont naturellement vocation à créer un lien entre le modèle du domaine (traduit par le noyau fonctionnel) et les modèles de l'IHM (traduit par la présentation). De fait, nous pensons qu'il est possible de définir un modèle d'architecture comme modèle central dans le processus de conception, de développement, d'adaptation et d'exécution, ce que nous appelons le **modèle d'interaction**. Les objectifs d'un tel modèle sont donc :

- représenter les éléments clés des différents modèles du système et leurs interactions ;

- jouer le rôle de chef d'orchestre du système interactif en assurant le contrôle du système ;
- être composable par éléments pour se doter de capacités de réutilisation ;
- s'intégrer dans un processus de spécification descendante (conceptuel, abstrait, concret, final).

L'architecture AMF que nous avions largement enrichie à l'occasion de nos travaux sur les systèmes collaboratifs (AMF-C) apporte plusieurs éléments de réponse à nos besoins et peut être étendue pour satisfaire l'ensemble de nos prérequis. Dans les sections qui suivent, nous présentons plus en détail le modèle AMF puis son utilisation comme modèle d'interaction.

2.3 Le modèle d'architecture AMF hybride

Comme nous l'avons déjà indiqué, le but d'AMF est de servir à la fois à la conception, la réalisation et l'utilisation. La dualité de vues, entre la segmentation thématique en couches (présentation, abstraction, etc.) et le rapprochement de tous les aspects concernant un même agent nous conduit à privilégier la vue « agent » dans la conception et la vue « en couches » dans la réalisation. Pour marier ces deux approches, une extension du modèle AMF a été proposée (Poquet, 1998). Nous appelons ce modèle : **AMF hybride**.

Ainsi, les agents d'une application peuvent être considérés comme des entités duales : une partie AMF pour les connexions et les communications gérant la dynamique d'interaction, et une partie application, pour la gestion des objets interactifs de la présentation ainsi que les données et les traitements applicatifs. On retrouve ainsi les 5 niveaux du modèle Arch, comme représenté sur la Figure 4.

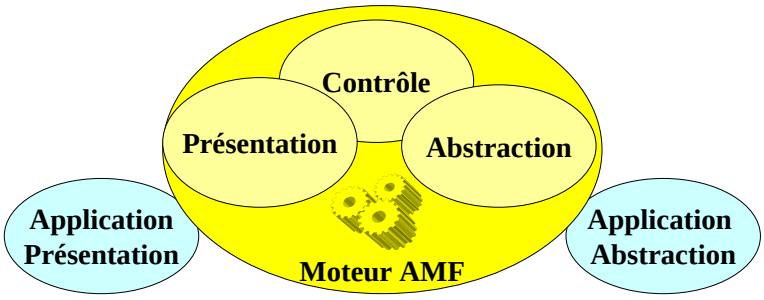

Figure 4 : Dualité des objets de l'application

Toute la modélisation AMF et les échanges dynamiques de messages (communications inter et intra agents) est gérée par ce que nous appelons le **Moteur AMF**. Ce rapprochement d'AMF avec Arch permet d'aller au-delà d'un simple modèle d'architecture. En effet, le modèle AMF est directement instancié en un contrôleur de dialogue et le moteur AMF pilote le dialogue de l'application en gérant les échanges entre facettes selon le contrôle exprimé dans le modèle AMF.

Pour qu'une application puisse être exécutée, les objets de l'application (issus soit de classes de présentation gérant les objets graphiques (*widgets*) soit du noyau fonctionnel) doivent être liés à leurs correspondants AMF par une relation d'association. C'est la raison pour laquelle chaque agent du modèle AMF, ainsi que chacune des facettes a une entité duale : une partie du côté application, une autre complémentaire du côté moteur AMF.

Le modèle AMF est un modèle d'architecture multiagents et multifacettes (Tarpin-Bernard et David, 1999). Le nombre de facettes n'est pas figé et chaque facette encapsulent des comportements récurrents. En plus des facettes régulièrement proposées dans les architecture multiagents (Présentation, Abstraction, Contrôle) AMF propose de nouvelles facettes (Aide, Erreur, Distribution, Droits, etc.). AMF possède un formalisme graphique qui exprime le contrôle d'exécution et spécifie le séquencement des opérations (Figure 5). A l'instar des méthodologies objets, AMF favorise l'analyse descendante grâce à une décomposition fine de l'application. Il favorise aussi la réutilisation de composants logiciels.

L'**agent** est le composant de base du modèle AMF. Chaque agent peut être constitué de **sous-agents**, de **facettes** et d'**administrateurs de contrôle**. Chaque facette est associée à une **classe applicative** qui la concrétise et dispose de ports de communication qui sont chacun associés à ce que nous appelons un **démon**, c'est-à-dire à une méthode de la classe en question. Un port joue donc en quelque sorte le rôle d'interface avec le service réel. La facette regroupe des fonctions (services) d'une nature semblable au sein d'un même composant. Tout agent qui agrège une facette dispose ainsi des services qu'elle fournit. Par défaut, un port n'est accessible qu'au sein de l'agent qui contient sa facette, mais il peut être rendu visible afin d'être accessible à partir d'autres agents.

Chaque service est associé à un **port de communication**. Il existe trois sortes de ports : les **ports d'entrée, les ports de sortie et les ports d'entrée/sortie**. Lorsqu'un port d'entrée est activé, le contrôle d'exécution est donné au démon qui lui est associé. Il exprime donc un service offert par la facette. A contrario, un port de sortie représente un service dont a besoin la facette. Sauf prétraitement particulier, aucun traitement n'est directement associé à un port de sortie. Un port entrée/sortie se comporte d'abord comme un port d'entrée, puis comme un port de sortie. Il n'est cependant associé qu'à un seul démon. La Figure 6 représente un meta-modèle de l'architecture AMF.

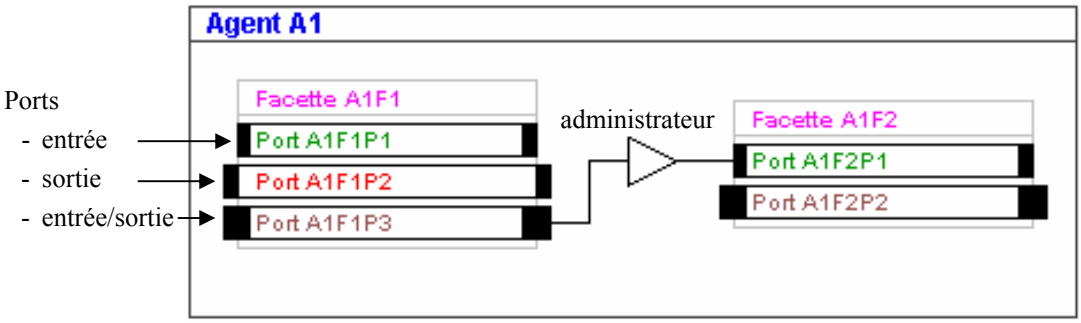

Figure 5 : Représentation graphique des éléments de base de l'architecture AMF

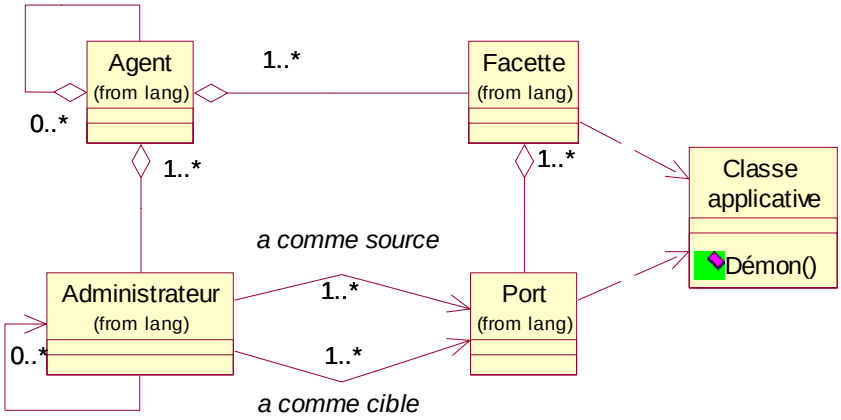

Figure 6 : Meta-modèle d'AMF

Les administrateurs de contrôle (Figure 7) jouent trois rôles : la gestion des connexions entre ports, l'exécution d'un traitement particulier sur les données échangées entre ports et la gestion de l'ordre d'activation des ports sources avant propagation aux ports cibles. Les administrateurs peuvent être connectés à plusieurs ports sources et plusieurs ports cibles. Ils peuvent jouer un rôle de traducteur et convertir les données reçues du port source dans un format compréhensible par le port cible. Ils peuvent aussi gérer des règles d'activation des ports sources pour ne propager le contrôle d'exécution au(x) port(s) cible(s) que si l'ordre des activations des ports sources est conforme à celui attendu. Plusieurs administrateurs peuvent aussi être mis en séquence afin de cumuler la fonctionnalité de chacun d'eux. Les administrateurs peuvent propager l'activation à plusieurs ports cibles de façon successive ou simultanée. Ils peuvent aussi, par filtrage, propager le contrôle d'exécution au port d'un ensemble de sous-agents à instanciation multiple en prenant en compte une condition qui permette de sélectionner le(s) sous-agent(s) dont le port doit être activé.

Administrateur simple	Administrateur avec filtrage	Administrateur avec retour	Administrateur AND	Administrateur de séquence	Administrateur utilisateur (personnalisable)

Figure 7 : Les principaux administrateurs AMF.

En résumé, une application est un ensemble d'agents, chaque agent possédant une ou plusieurs facettes, chaque facette possédant un ou plusieurs ports. Grâce à des administrateurs, chaque port peut être mis en relation avec des ports d'autres facettes de l'agent ou d'un agent fils direct de cet agent. On peut également passé par l'agent parent pour faire une connexion plus lointaine. Dans ce cas, il faut cascader des administrateurs.

Les caractéristiques de l'architecture AMF nous ont permis de supporter le modèle d'interaction à deux niveaux :

- Au niveau interne, il joue le rôle d'articulation entre les modèles du système par intégration des éléments clefs de ces modèles dans des facettes dédiées,
- Au niveau externe, il assure l'interaction entre l'utilisateur et le système en faisant le lien avec l'implémentation du système, c'est-à-dire avec les classes applicatives (classes du noyau fonctionnel, classes de présentation, etc.).

2.3.1 Articulation entre les modèles du système

Au niveau interne, le modèle d'interaction supporte les relations entre les modèles du système et exprime en partie le contrôle du dialogue.

L'état de l'art que nous avons dressé sur les approches à base de modèles, nous conduit à nous intéresser tout d'abord aux modèles de tâches et de domaine. Ces modèles seront mis en relation avec les différents éléments du formalisme AMF.

Le modèle des **tâches d'interaction** contient l'ensemble des actions et séquences d'interaction de l'utilisateur. Il représente donc notamment les services de l'interface utilisateur manipulables par l'utilisateur. Dans le modèle AMF, les services de l'interface utilisateur sont interfacés par les ports qui sont regroupés dans une ou plusieurs facettes de présentation d'un agent ou de plusieurs agents.

Le **modèle du domaine** décrit les concepts du domaine et les services proposés par le système qui sont du côté du noyau fonctionnel de l'application (ex : effectuer un calcul, consulter une base de donnée, tester la faisabilité d'une action). Ainsi, ce modèle peut être rapproché des ports des facettes abstraction du modèle AMF.

L'originalité du modèle AMF réside dans la formalisation et la représentation graphique de la communication entre les ports des facettes du système interactif à travers les administrateurs de contrôle. L'ensemble de ces administrateurs constitue une partie du **modèle de dialogue** qui décrit les liens entre les services de l'interface (regroupés dans les facettes de présentation) et les concepts du domaine (regroupés dans les facettes d'abstraction). La limite du modèle AMF en ce qui concerne la modélisation du dialogue réside dans le fait que la représentation graphique n'est pas dynamique, il est donc difficile de modéliser des dialogues temporels (au-delà de l'usage d'administrateurs de séquence ou de ports de communication d'entrée/sortie qui matérialise un *workflow*). L'usage de noms de ports explicites est alors très important pour faciliter la compréhension comme nous le verrons plus tard avec des exemples. Nous travaillons actuellement sur des moyens d'enrichir le modèle sur ce plan.

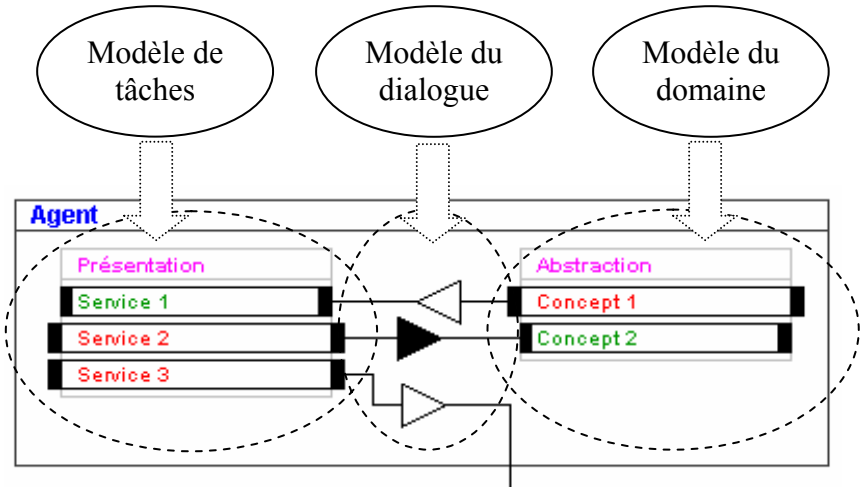

Figure 8 : Elaboration du modèle d'interaction à partir des modèles de tâches d'interaction, du domaine et du dialogue

A ce niveau, le modèle d'interaction exprime les liens entre les tâches d'interaction et les concepts d'une façon claire et expressive.

2.3.2 Communication avec les éléments externes au moteur

Le niveau externe d'interaction joue le rôle de médiateur entre l'utilisateur, les classes applicatives et les différents modèles du système.

A ce niveau, nous nous basons sur l'approche hybride et le moteur de l'architecture AMF qui assure alors le lien entre :

- les objets de l'application, dont les rôles consistent d'une part à gérer les données et en effectuer les traitements (noyau fonctionnel), d'autre part à interfacer le *toolkit* graphique (présentation concrète) ;
- les objets AMF eux-mêmes afin de gérer les interactions selon les activations des ports de sorties.

En pratique, lorsque l'utilisateur agit sur l'interface de l'application (clic sur un bouton, choix dans un menu…), la classe de présentation concrète (ex : JPanel en Java) qui gère cet évènement active un port de sortie du moteur AMF, ce qui lance tout le processus de communication interne au moteur. En bout de chaîne, à l'activation d'un port d'entrée, une méthode d'une classe du noyau fonctionnel est déclenchée.

Dans ce cas, la facette abstraction des agents AMF a valeur d'adaptateur du noyau fonctionnel du modèle Arch et la facette présentation représente le niveau abstrait de l'interface, le niveau concret se trouvant dans la présentation de l'application.

Ce rapprochement entre AMF et Arch permet d'utiliser AMF non seulement pendant la phase de conception mais également comme support à l'exécution. En effet, le modèle AMF est directement instancié en contrôleur de dialogue et pilote l'interaction entre l'utilisateur et l'application (Figure 9).

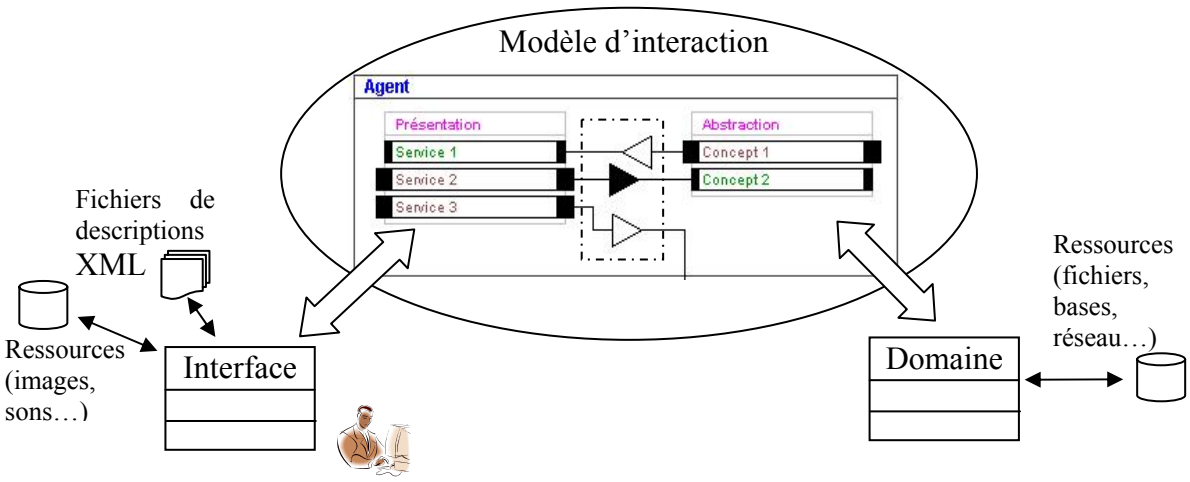

Figure 9 : Interactions entre le moteur AMF les éléments externes pendant l'exécution

2.3.3 Extension de l'approche Arch

Un des intérêts de AMF est naturellement son approche multifacettes. Chaque facette a vocation à encapsuler des données et des comportements spécifiques et donc par extension à supporter des sous-modèles particuliers ce qui est très intéressant dans notre problématique de gestion de l'adaptation par une approche basée sur les modèles.

Historiquement, et dans un contexte de travail collaboratif, des facettes spécifiques de gestion des droits des utilisateurs, de rétroaction de groupe ou d'aide ont été identifiées (thèse de doctorat sur AMF-C). Plus tard, des facettes gestion du workflow ont été proposées (David *et al.*, 2000). Aujourd'hui, nous pouvons y ajouter d'autres facettes spécifiques :

- Une facette « **Contexte d'utilisation »** : cette facette regroupera l'ensemble des informations et des services d'identification et d'adaptation au contexte d'utilisation. Elle peut être éclatée en plusieurs facettes représentant chacune un aspect spécifique du contexte (plate-forme, utilisateur, environnement).
- Une ou plusieurs facettes « **Dispositifs »** : ces facettes prennent en charge l'interfaçage des dispositifs d'interaction avec le système interactif afin de découpler les dispositifs physiques d'interaction des techniques d'interaction associées (Cf. section 2.7.2 plus loin).

Nous pouvons alors parler d'un modèle du contexte et d'un modèle de dispositif. Bien que nous n'ayons pas encore exploré cette voie, on peut sans doute imaginer définir de nouveaux types de facettes spécifiques en charge par exemple du modèle d'évolution des agents.

Avec cette vision du système interactif possédant un nombre non prédéfini de modèles, la représentation en couches (architecture Arch) du système peut être étendue à une vue multidimensionnelle et représentée par trois anneaux concentriques sectorisés (Figure 10) :

- L'anneau extérieur représente l'ensemble des classes applicatives du système, c'est-à-dire des composants concrets de l'application. Ces classes manipulent des ressources (fichiers, médias, bases, etc.) ;
- L'anneau intermédiaire exprime le niveau adaptateur qui est responsable de l'interfaçage des classes applicatives (dans notre cas, il s'agit des agents AMF avec leurs facettes et ports),
- Le niveau central représente le contrôleur du dialogue responsable de l'acheminement des messages entre les différents composants du système interactif (dans notre cas, les administrateurs de contrôle et le moteur AMF à l'exécution).

Dans cette représentation en forme de cible, celle-ci est fractionnée en secteurs. Chaque secteur représente un des modèles du système interactif. La communication entre les composants d'un modèle et les composants d'un autre modèle passe obligatoirement par le niveau central de la cible (le contrôleur du dialogue). Il n'y a donc pas de communications directes entre secteurs.

Sur la Figure 10, nous avons également représenté en pointillé le modèle Arch traditionnel qui est un sous ensemble de ce modèle étendu. AMF est un support pour les deux couches intérieures (contrôleur et adaptateurs).

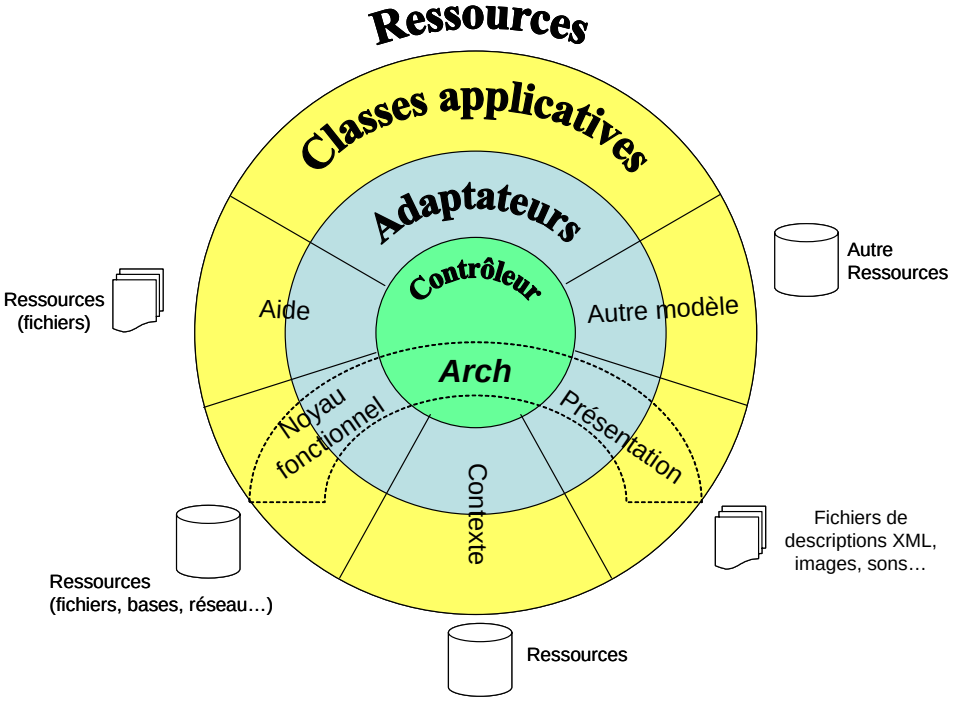

Figure 10 : Le modèle Arch étendu à l'approche multifacettes multimodèles.

Notons que malgré la présence d'un nombre non prédéfini de modèles dans notre approche, la logique de rattachement adoptée pour la relation entre le modèle AMF et le modèle du domaine reste identique pour tous les autres modèles. Ainsi, chaque classe applicative sera rattachée à une facette d'un agent du modèle d'interaction AMF. L'ensemble des classes applicatives qui sont rattachées aux mêmes types de facettes (facette dispositif, facette contexte, etc.) dans les différents agents, forme l'implémentation d'un modèle du système interactif représenté par ailleurs avec tel ou tel formalisme (graphique ou textuel).

Dans la mesure où le modèle AMF hybride était un bon candidat pour servir de pivot à l'élaboration d'applications interactives adaptatives, il nous a fallu sélectionner les modèles à prendre en considération dans notre approche. Dans les sections suivantes nous identifions l'ensemble des modèles, leurs représentations et les choix techniques que nous avons été amenés à faire pour leur intégration dans notre approche de construction et d'adaptation des systèmes interactifs.

Contexte de recherche :
- Thèse de doctorat puis DEA Johnny Poquet et thèse Kinan Samaan
- Projet Régional << RESTER PROPRE >>

Publications :
- **TARPIN-BERNARD F., DAVID B.T.,** AMF : un modèle d'architecture multi-agents multi-facettes. *Techniques et Sciences Informatiques.* Hermès. Paris. Vol. 18. No. 5. Mai 1999. pp. 555-586.
- **SAMAAN K., DELOTTE O., TARPIN-BERNARD F.** Processus de génération d'IHM multicibles pour applications interactives. In *Proceedings of the 14th French-Speaking Conference on Human-Computer interaction (Conférence Francophone Sur L'interaction Homme-Machine)* (Poitiers, France, November 26 - 29, 2002). M. Beaudouin-Lafon, Ed. IHM '02, vol. 32. ACM Press, New York, NY, ISBN:1-58113-615-3, 251-254.
- **DAVID B.T., TARPIN-BERNARD F., POQUET J., SAIKALI K., BOUTROS N.** From theory to practice: cooperation models in a sustainable product life-cycle in *Designing Cooperative Systems, The Use of Theories and Models*, Proceedings 4th International Conference on the Design of Cooperative

Systems (COOP 2000, Sophia Antipolis, 23-26 Mai 2000), Eds G. De Michelis , A. Giboin , L. Karsenty and R. Dieng, Vol 58, IOS Press, Amsterdam, The Netherlands, 2000.
- **DAVID B.T., POQUET J., TARPIN-BERNARD F., GERNER S., VIAL I., BINDER Z.,** Design for Sustainability: Information Technology Support. *2000 International CIRP Design Seminar.* Haifa, Israel, May 16-18, 2000.
- **DAVID B.T., POQUET J., TARPIN-BERNARD F.** Interfaces Opérateurs pour des systèmes contrôle/commande coopératifs. *3° congrès international de génie industriel.* Presse internationales Polytechnique. Montréal, Canada. p. 1027-1036. Mai 1999.
- **TARPIN-BERNARD F., DAVID B.T.,** AMF : un modèle d'architecture multi-agents multi-facettes. *Techniques et Sciences Informatiques.* Hermès. Paris. Vol. 18. No. 5. Mai 1999. pp. 555-586.
- **TARPIN-BERNARD F., DAVID B.T.,** AMF a new design pattern for complex interactive software? *International HCI'97 - Advances in Human Factors/ Ergonomics - Design of Computing Systems.* Elsevier, ISBN: 0 444 82183 X. San Francisco. Vol. 21B. pp. 351-354. Aout 1997.
- **TARPIN-BERNARD F., DAVID B.T., PRIMET P.,** 1998 Frameworks and Patterns for Synchronous Groupware: AMF-C Approach. In *Proceedings of the IFIP Tc2/Tc13 Wg2.7/Wg13.4 Seventh Working Conference on Engineering For Human-Computer interaction* (September, 1998). S. Chatty and P. Dewan, Eds. IFIP Conference Proceedings, vol. 150. Kluwer B.V., Deventer, The Netherlands, 225-241.

2.4 Liens entre modèle de tâches et modèle d'interaction

Pour lier le modèle de tâches et le modèle d'interaction, nous avons élaboré une méthode de mise en correspondance qui prend en compte les deux problèmes suivants :

- Le modèle de tâches contient des tâches d'interaction et des tâches machine. Ainsi, le modèle de tâches CTT (Paternò, 1999) que nous avons choisi d'utiliser identifie plusieurs types de tâches. Nous trouvons les tâches entièrement effectuées par l'utilisateur, les tâches entièrement effectuées par la machine et les tâches d'interaction entre l'utilisateur et la machine. La tâche purement utilisateur (réflexion mentale, tâche physique indépendante de la machine, etc.) n'est pas gérée pas notre système car elle n'implique pas de traitement spécifique de la part de la machine. Les deux autres types de tâches sont mis en relation avec les services proposés par les différentes facettes du modèle d'interaction.
- Le niveau de détail ou de précision des tâches est variable. Le modèle de tâches est un modèle conceptuel dont le niveau de détail diffère selon les approches. Nous distinguons dans notre travail deux niveaux pour ce modèle : un niveau abstrait, indépendant de la plate-forme et du contexte, et un niveau plus détaillé dépendant de la plate-forme. Ces deux niveaux du modèle de tâches doivent être mis en relation avec deux niveaux du modèle d'interaction.

Voyons maintenant en pratique comment nous avons mis en relation les deux modèles.

2.4.1 Liens entre tâches et ports de communication

Les tâches d'interaction sont celles que l'utilisateur doit effectuer pour manipuler le système. Ceci suppose la présence d'un objet d'interaction manipulable sur l'interface de l'application. Par exemple, pour augmenter le volume sonore d'un lecteur de musiques. L'utilisateur doit avoir à sa disposition une touche de contrôle, que cette touche soit physique ou un objet de contrôle graphique selon la nature de la plate-forme.

Dans le modèle d'interaction, l'ensemble des services, proposés par le système et manipulables par l'utilisateur est regroupé dans la ou les facettes de présentation des différents agents. L'idée est alors de faire la correspondance entre l'ensemble des tâches

d'interaction extraite du modèle de tâches CTT et les ports de la facette présentation du modèle AMF.

A titre d'illustration, nous proposons dans les pages qui suivent de nous intéresser à la modélisation d'un lecteur de musiques dont les tâches d'interaction basiques seraient par exemple :

- Activer_lecteur : pour démarrer le système ;
- Sélectionner_titre : pour choisir un titre de musique ;
- Play_titre : pour lancer la lecture ;
- Stop_titre : pour arrêter la lecture ;
- Régler_volume : pour régler le volume sonore du système ;
- Quitter_application : pour quitter le système.

A partir de ces tâches qui sont organisées en arbre (Figure 11), nous déduisons les ports de la facette présentation de cette fraction du système. Ainsi, la facette présentation doit posséder au minimum un ensemble de ports et de services qui correspondent au nombre et à la nature des tâches dans le modèle de tâches.

Les tâches machine sont les tâches que la machine doit effectuer en arrière-plan et qui sont liées au domaine de l'application. Dans notre exemple, imaginons une tâche *Proposer_les_titres* qui s'exerce au démarrage du système et consulte la base des titres disponibles avant de les proposer à l'utilisateur. Ce type de tâches n'est pas lié à une intervention directe de l'utilisateur et ne nécessite pas une manipulation spécifique de sa part. Notons qu'il faut bien faire la différence entre les tâches machine et les *feedbacks* des tâches d'interaction. Les premières se lancent automatiquement une fois que les conditions d'exécution sont rassemblées (consulter une base de données, effectuer un calcul, etc.). Les autres sont la réaction directe du système aux actions de l'utilisateur (déplacement d'un élément sur l'écran, modification du volume sonore du système, etc.).

Dans le modèle d'interaction, l'ensemble des services du domaine est rassemblé dans les facettes d'abstraction des différents agents. La correspondance cette fois sera établie entre les tâches machine du modèle de tâches CTT et les ports de la facette abstraction du modèle AMF.

La Figure 11 représente les liens de correspondance entre les tâches d'interaction et machine avec les ports des facettes présentation et abstraction du modèle d'interaction pour le lecteur de musiques.

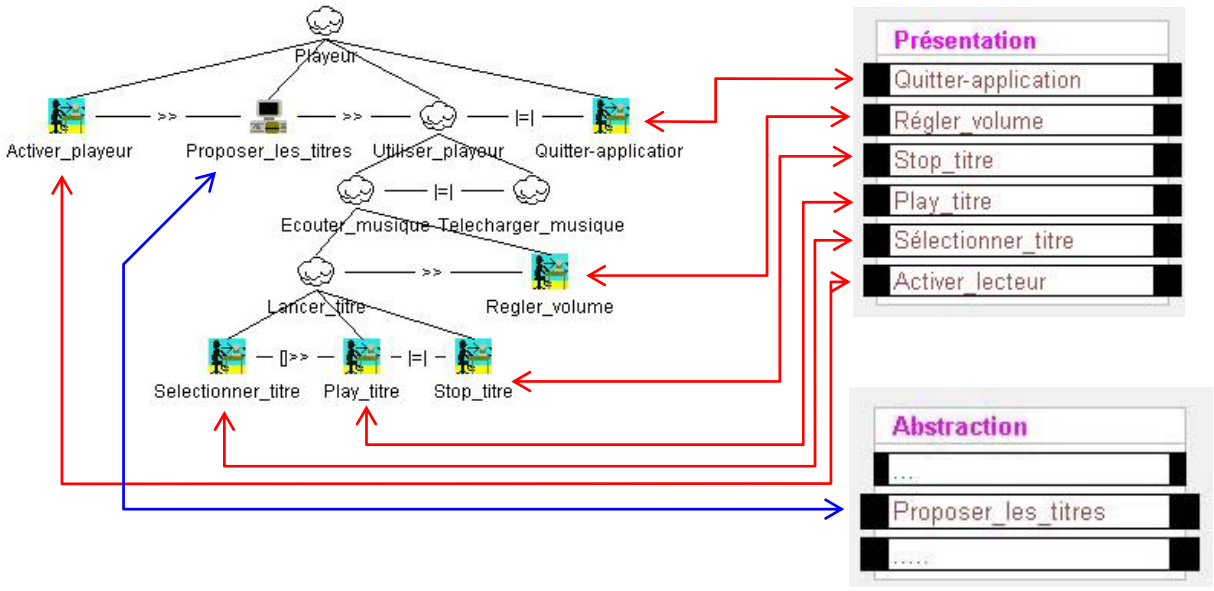

Figure 11 : Exemple de correspondance entre les tâches et les ports du modèle d'interaction.

Souvent les tâches machines sont occultées dans le modèle de tâches car les travaux dans ce domaine se focalisent essentiellement sur les tâches d'interaction. La présence des tâches machine se limite généralement à une représentation graphique dans le modèle de tâches. Pourtant ces tâches sont extrêmement intéressantes car elles établissent également un lien explicite avec le modèle du domaine.

2.4.2 Modèle abstrait et modèle concret de tâches

Dans les approches à base de modèles, le modèle de tâches est considéré comme le modèle de départ pour la conception d'un système interactif. Ce modèle est assez flexible quant au niveau de détail des tâches. Généralement, la spécification d'une tâche doit rester à un niveau d'abstraction assez élevé (ex : modification du volume sonore du système). Cependant, certaines approches utilisent un modèle de tâches plus détaillé (clic droit, touche clavier, etc.). Dans notre approche, nous prenons en considération ces deux niveaux de détail pour le modèle de tâches, mais à deux phases différentes du processus de conception.

Le modèle abstrait de tâches représente une vue générique du système modélisé, et cela indépendamment du contexte d'utilisation en général et plus spécifiquement de la plate-forme. Il ne tient par exemple pas compte des moyens d'interaction disponibles sur cette plate-forme ni de ses capacités d'affichage. Il représente le système en décrivant des tâches avec un niveau d'abstraction élevé. En revenant sur l'exemple du système de lecteur de musiques, les tâches (*Activer_lecteur, Sélectionner_titre, Régler_volume, etc.*) sont des tâches abstraites car elles déclarent l'objectif de l'utilisateur (ou du système dans le cas des tâches machine) mais sans spécifier la manière dont chaque tâche doit être réalisée. Nous pouvons donc reprendre le modèle de tâches originelles en considérant que les tâches d'interaction citées sont en fait des tâches abstraites dans la mesure où elles ne spécifient pas précisément les interactions élémentaires que doit réaliser l'utilisateur. Pour les représenter, nous utilisons le symbole de tâche abstraite ( ) proposé dans le modèle CTT, bien que dans le modèle original de CTT ce formalisme soit utilisée pour exprimer la hiérarchisation du modèle et pour pouvoir masquer ou afficher les détails des sous-tâches.

A partir de ce modèle, nous pouvons construire un modèle abstrait d'interaction en ajoutant au modèle d'interaction AMF la notion de port abstrait. Les **ports abstraits** sont des ports qui expriment les services que le système doit satisfaire, mais qui ne détaillent pas la façon dont ces services seront accomplis. Comme nous le verrons plus loin, lorsque sera décrite la façon concrète dont ces services seront accomplis, chaque port abstrait sera remplacé par un ou plusieurs ports concrets connectés au reste du système par un ou plusieurs administrateurs de contrôle.

Pour représenter graphiquement ces ports abstraits dans le modèle d'interaction et pour faire la différence entre ce modèle et le modèle concret d'interaction, nous avons choisi de les représenter en utilisant une **ligne pointillée**.

Dans le modèle concret, toutes les feuilles de l'arbre des tâches représentent des interactions concrètes avec des objets de la présentation manipulés. Le niveau de détail du modèle concret de tâches peut être suffisamment précis pour identifier les interactions physiques de l'utilisateur (clic souris, saisie clavier, etc.). Ce modèle est alors dépendant du contexte d'utilisation et des moyens d'interaction disponibles sur la plate-forme d'interaction. A ce niveau, et pour un contexte bien identifié, un choix de solutions doit être effectué pour remplacer les tâches abstraites par des tâches ou par des sous-arbres de tâches concrètes.

En revenant sur l'exemple du lecteur de musiques, nous supposons ici que la plate-forme cible ne dispose que d'un clavier comme moyen d'interaction (c'est un cas simplifié du contexte mais suffisant pour illustrer l'approche). Dans ce cas, la tâche abstraite *Sélectionner_titre,* par exemple, peut être remplacée (Figure 12) par un sous-arbre de tâches qui contient les tâches suivantes :

- *Up/down*: pour faire défiler la liste des titres ;
- *Valider_titre* : pour valider le choix effectué.

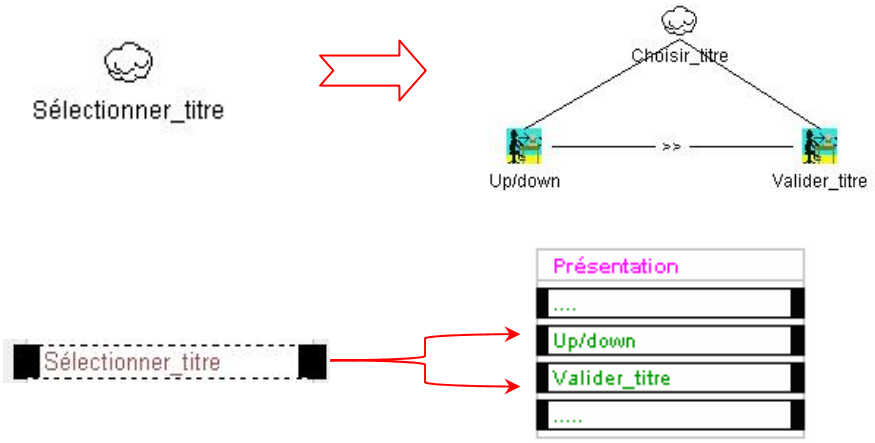

Figure 12 : Exemple de concrétisation d'une tâche abstraite et d'un port abstrait

Parallèlement, et en fonction des solutions adoptées pour le modèle de tâches, les ports abstraits du modèle d'interaction sont remplacés par des ports concrets pour obtenir le modèle concret d'interaction.

A la suite de cette opération de remplacement et en effectuant la même opération pour les autres tâches et ports abstraits, nous obtenons le modèle concret de tâches et d'interaction.

Cette opération de concrétisation est une tâche répétitive qui peut être assistée. En effet, nous avons constaté que de nombreuses tâches (et suite de tâches) se répètent identiquement dans les systèmes interactifs. Le paragraphe suivant introduit à ce propos notre vision pour l'utilisation de patterns de tâches et d'interactions dans le processus de conception et de construction des systèmes interactifs.

2.4.3 Les patterns de tâches et d'interactions

Le modèle d'interaction AMF se situe résolument dans l'approche « design patterns » (Gamma *et al.*, 1995). En effet, l'architecture globale est orchestrée par la structuration en agents AMF de l'application interactive où chaque agent peut contenir différentes facettes. Des fragments de modélisation sont par construction des patterns potentiels dès lors qu'ils constituent une solution éprouvée à un problème clairement spécifié. Les patterns de ce type sont désignés dans notre démarche sous le nom de **pattern d'interaction**. Ainsi, plusieurs motifs élémentaires récurrents ont déjà été identifiés (Tarpin-Bernard *et al.*, 1999). De nombreux autres restent bien sûr à découvrir.

Sur le plan de l'adaptation des systèmes, nous avons défini plusieurs patterns liés aux moyens utilisés pour interagir avec l'application dans différents contextes. Nous nous sommes par exemple intéressés à la tâche de déplacement d'un élément au sein d'un objet conteneur (ex : un icône sur une surface, un item dans une liste, etc.). Le principe de mise en œuvre retenu est le suivant : 1) l'objet conteneur reçoit la notification de demande sélection d'un élément, 2) l'élément concerné est identifié et autorise le déplacement, 3) la destination du déplacement est définie, 4) le déplacement est validé, 5) l'affichage est rafraîchi pour tenir compte du déplacement.

La Figure 13 présente un pattern qui réalise la tâche abstraite de sélection et de déplacement d'un élément dans un ensemble d'éléments. Ce pattern d'interaction propose un agent *Container* contenant les agents composants (agent à instanciations multiples) susceptibles d'être sélectionnés et déplacés (*Element*). C'est le port abstrait (*Select&Move)* de la facette *Presentation* de l'agent composé qui reçoit l'ensemble des actions de l'utilisateur avant de les transmettre aux agents composants concernés. Ce pattern peut se décliner en plusieurs versions selon les dispositifs d'interactions des plates-formes. Les ports et les administrateurs abstraits du pattern sont remplacés par des ports et des administrateurs concrets.

Pour l'instant, seuls quelques patterns ont été sérieusement étudiés et un travail approfondi devra être mené pour constituer un catalogue de patterns utiles et utilisables par les concepteurs / développeurs.

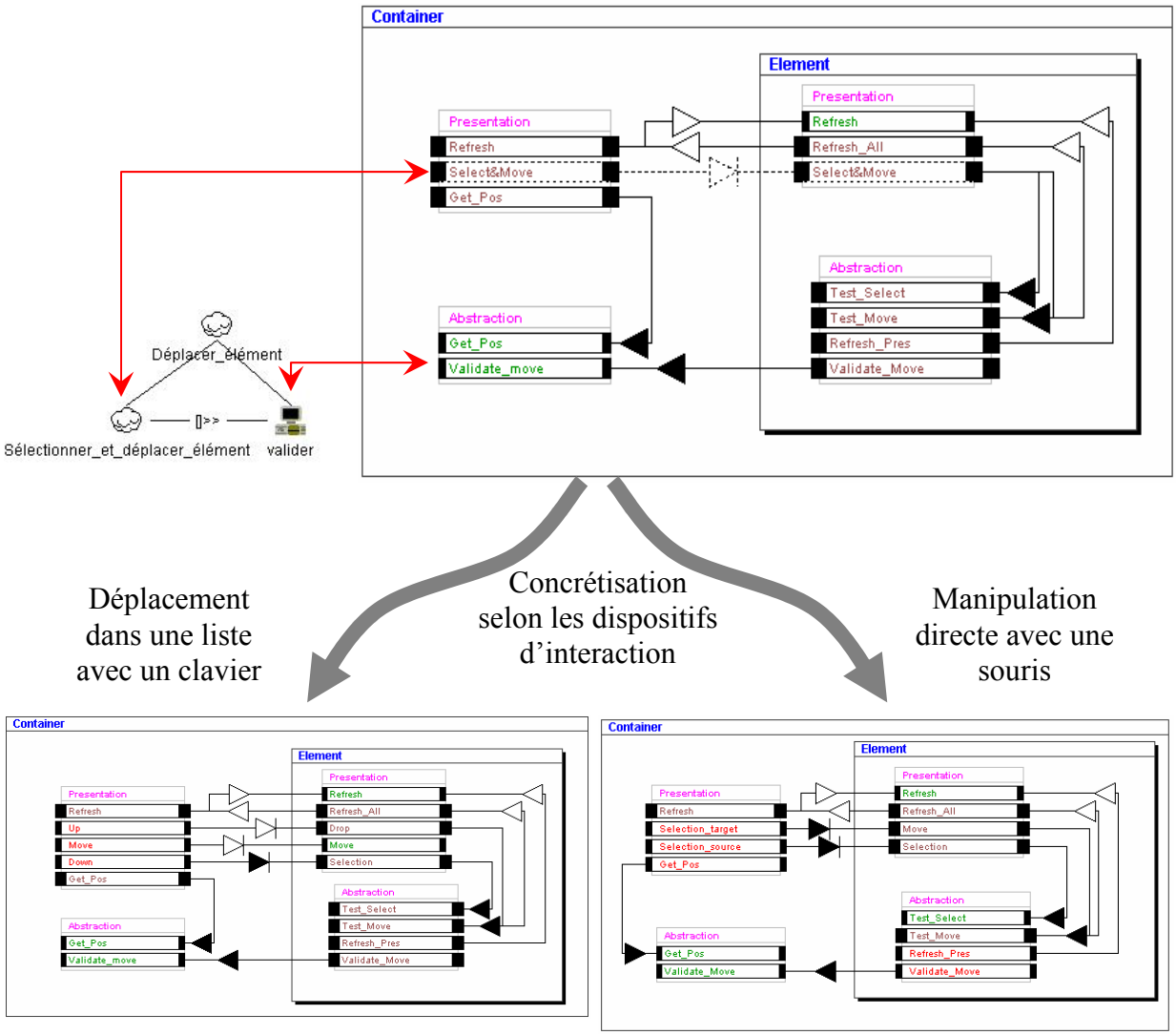

Figure 13 : Un pattern pour la tâche abstraite « sélectionner et déplacer un élément »

Contexte de recherche :
- Thèse Kinan Samaan

Publications :

- **SAMAAN, K.; TARPIN-BERNARD, F.** 2004. Task models and interaction models in a multiple user interfaces generation process. In *Proceedings of the 3rd Annual Conference on Task Models and Diagrams* (Prague, Czech Republic, November 15 - 16, 2004). TAMODIA '04, vol. 86. ACM Press, New York, NY, pp. 137-144, ISBN:1-59593-000-0, 137-144.
- **SAMAAN K.**, **TARPIN-BERNARD F.**, The AMF Architecture in a Multiple User Interface Generation Process, *Proceedings of the ACM AVI'2004 Workshop on Developing User Interfaces with XML: Advances on User Interface Description Languages*, K. Luyten, M. Abrams, J. Vanderdonckt and Q. Limbourg (eds), Gallipoli, Italy, 2004, pp. 71-78.
- **SAMAAN K., TARPIN-BERNARD F.** Using interaction patterns for the adaptation of multi target user interfaces. In *Proceedings of the 15th French-Speaking Conference on Human-Computer interaction on 15eme Conference Francophone Sur L'interaction Homme-Machine* (Caen, France, November 25 - 28, 2003). T. Baudel, Ed. IHM 2003, vol. 51. ACM Press, New York, NY, ISBN:1-58113-803-2, pp. 272-275.

2.5 Modèle d'interaction et modèle du domaine

Le modèle du domaine représente l'ensemble des concepts du noyau fonctionnel manipulables par l'application, leurs relations et les actions qui peuvent être réalisées sur ces objets. La modélisation des concepts du domaine repose généralement sur l'élaboration de modèles UML, en particulier les **diagrammes de classes**. Le diagramme de classes modélise la structure statique en termes de classes et de relations. Une classe définit les propriétés (les attributs et les opérations) de ses objets. La notation UML dispose de quatre types de relations entre ces classes : l'association, l'héritage, l'agrégation et la composition.

Nous rappelons ici que chaque facette du modèle AMF est associée à une classe applicative. Cette classe regroupe l'ensemble des méthodes rattachées aux ports de la facette et toutes les autres méthodes dont elle peut avoir besoin pour son fonctionnement. Idéalement, le modèle du domaine représenté par le diagramme de classes UML reprend donc la structuration des agents déterminée par le modèle AMF. En réalité, cette règle d'association n'est pas obligatoirement une relation bijective. Des classes applicatives peuvent ne pas avoir une représentation dans le modèle AMF. Pour clarifier ce point, nous identifions trois cas de correspondance possibles entre ces deux modèles :

- Les agents sont autosuffisants et chaque classe applicative est rattachée à une facette AMF. Dans ce cas, on a une relation bijective et une symétrie totale entre les deux modèles. Dans les applications interactives que nous présentons en exemple, c'est la situation la plus fréquente.
- Une classe d'agents peut utiliser des services spécifiques réalisés par un ensemble de classes applicatives n'ayant pas vocation à être toutes visibles (application du pattern façade de Gamma *et al.*, 1995). Dans ce cas, ces classes ne sont pas nécessairement représentées dans le modèle AMF. L'agent en question forme l'interlocuteur unique avec ces classes et la symétrie est partielle.
- Plusieurs agents utilisent un ensemble complexe de classes applicatives externes (ex ; gestion de bases de données, bibliothèque de calcul, etc.). Dans ce cas, les classes applicatives externes sont regroupées dans un package et un agent de communication vient s'ajouter au modèle AMF pour assurer les liens entre les agents et ces services externes.

Dans toutes les situations de correspondance complète entre les classes applicatives et les facettes AMF (premier cas décrit précédemment), la modélisation UML des relations entre les classes applicatives (association, agrégation, etc.) ne peut être conservée. En effet, la souplesse et la richesse d'AMF passe par un découplage des classes applicatives afin que les échanges d'information et le contrôle soient gérés par le moteur AMF. Pour ce faire, il ne doit pas exister de relation directe entre les classes applicatives associés aux facettes AMF.

Ainsi, nous avons proposé une traduction vers le modèle AMF pour les relations interclasses du modèle UML les plus couramment utilisées. La Figure 14 illustre la façon dont les relations d'agrégation et d'association du modèle du domaine se traduisent dans le modèle AMF

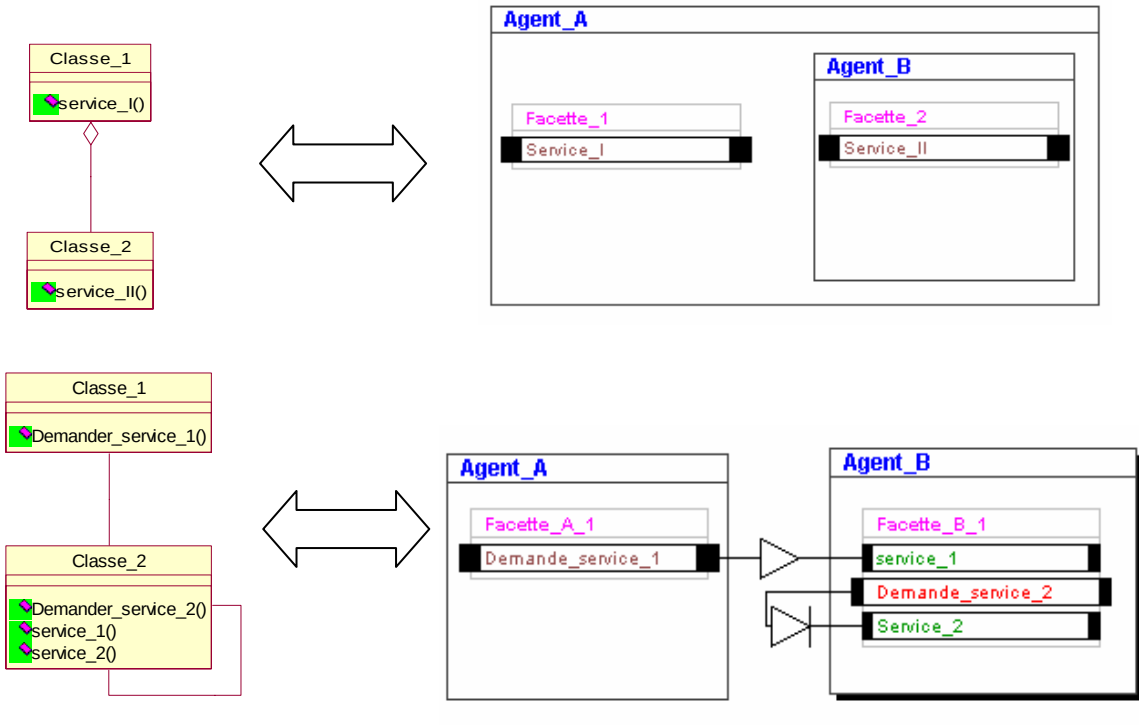

Figure 14 : Traduction des relations d'agrégation et d'association du modèle UML en modèle AMF

Dans le modèle AMF les relations d'héritage sont maintenues au niveau de l'implémentation des classes applicatives, mais cette relation n'est pas visible dans la représentation graphique du modèle d'interaction AMF classique. En revanche, un diagramme complémentaire spécifique peut être utilisé pour montrer des relations d'héritage entre facettes et/ou agents. Dans ce cas, on n'utilise plus une représentation imbriquée (agent / sous-agents) mais une représentation classique de type arbre.

2.6 Processus de construction

A ce stade, nous disposons de tous les éléments de base pour élaborer un processus de conception et de construction d'applications interactives facilement adaptables au contexte d'utilisation. La conception d'applications interactives se situe au confluent des approches du génie logiciel et de l'IHM. Depuis de nombreuses années, les recherches dans ce domaine ont conduit à mettre en avant des démarches de conception participatives ainsi que des cycles de vie itératifs conduisant à faire du maquetage et prototypage avant la production finale. Notons cependant que si les approches de génie logiciel proposent plutôt d'élaborer les différents diagrammes UML dans une démarche progressive centrée sur le modèle du domaine, les méthodes issues de l'IHM proposent principalement de partir d'une analyse des tâches puis de passer à la conception en adoptant ou respectant un des modèles d'architecture (*framework*) comme MVC, ARCH, PAC ou AMF.

De fait, nos recherches s'inscrivent dans ces différentes démarches et se focalisent sur les phases aval de la conception. En effet, si nous apportons notre contribution dans l'élaboration du modèle du domaine et du modèle de tâches, nos apports principaux se situent dans l'explicitation du processus permettant de passer de modèles abstraits à des modèles concrets implémentés et exécutables.

En accord avec la plupart des approches à base de modèles, notre méthodologie consiste à découper le processus de conception et de construction des systèmes interactifs adaptables en quatre phases.

La première phase, consiste à élaborer le modèle du domaine et le modèle des tâches qui se trouvent au niveau d'abstraction le plus élevé. La deuxième phase définit l'interaction abstraite qui représente la structure générale de l'interaction de l'application à l'aide du modèle abstrait d'interaction AMF. La troisième est une phase de concrétisation : à partir du modèle abstrait d'interaction et en fonction du contexte d'utilisation nous obtenons le modèle concret d'interaction. Enfin, l'application finale est construite et exécutée à partir des éléments produits précédemment.

Dans la suite, nous détaillons étape par étape les points clés de ce processus organisé autour de l'élaboration du modèle d'interaction.

2.6.1 Phase d'élaboration du modèle du domaine et du modèle de tâches

En appliquant l'approche par découverte des scénarios, il est fréquemment suggéré de bâtir le plus grand nombre possible de cas d'utilisation. Ces cas d'utilisation font découvrir les acteurs humains, les tâches (décrites assez sommairement mais qui structurent les activités) et les acteurs logiciels. L'assemblage de cas d'utilisation donne une vision globale des interactions de l'application.

En même temps, il est possible de construire des diagrammes de séquences UML montrant pour chaque scénario (cas d'utilisation) les échanges entre l'utilisateur et l'application. De cette manière on collecte à un niveau assez abstrait les relations entre l'utilisateur et l'application. A partir de ces informations, nous sommes en mesure de commencer à bâtir le diagramme de classes qui structure les différentes entités qui apparaissent dans les cas d'utilisation et de façon plus détaillée dans les diagrammes de séquences (traitements ou services activés côté application). On obtient ainsi un modèle du monde applicatif, dans lequel figurent les classes d'objets supportant les activités identifiées dans les cas d'utilisation et les diagrammes de séquences.

De façon plus ou moins concomitante, selon la sensibilité du concepteur, un diagramme de tâches est construit en utilisant par exemple le formalisme CTT de Paternò. L'arbre de tâches organise les tâches en hiérarchie. L'intérêt de CTT est de décrire plus précisément les activités que dans les cas d'utilisation et les diagrammes de séquences et surtout d'organiser, grâce aux opérateurs temporels l'ensemble des tâches. Cette organisation temporelle globale constitue un « plus » indispensable par rapport à la vision parcellaire des cas d'utilisation et des diagrammes de séquences. La modélisation est construite jusqu'au niveau d'interaction abstraite (indépendante des dispositifs d'interaction).

La fin de cette phase fournit un modèle de tâches abstraites et un modèle du domaine. D'autres modèles, selon la nature du système, peuvent également être bâtis à ce stade du processus. Nous pouvons, par exemple, définir les éléments d'un modèle de droits d'utilisation dans un système de travail collaboratif.

La Figure 15 montre une partie des modèles obtenus à la fin de cette phase pour l'exemple du lecteur de musiques. Les modèles ne sont pas encore connectés les uns aux autres. D'un côté nous avons les classes du modèle de domaine et de l'autre, nous avons le modèle abstrait de tâches.

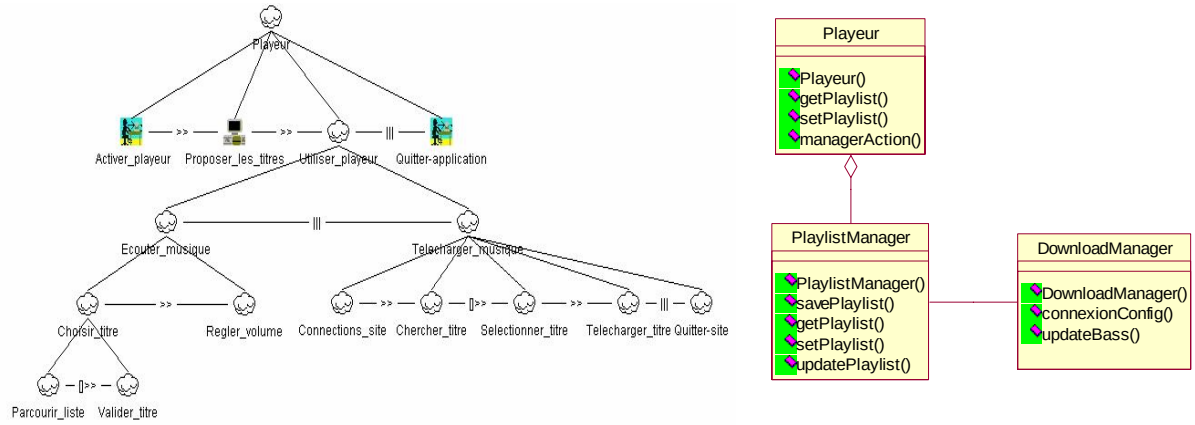

Figure 15 : Les modèles à la fin de la première phase de conception d'un lecteur de musiques

La souplesse de l'approche respecte le mode de travail et la culture (développement ou IHM) de chaque concepteur en le laissant commencer par le modèle du domaine ou par le modèle de tâches. Dans nos propres démarches, le modèle de tâches a toujours précédé celui du domaine.

2.6.2 Élaboration de l'interaction abstraite

Cette phase d'élaboration consiste à lier les différents modèles obtenus à la fin de la première phase, grâce au modèle d'interaction AMF.

La logique temporelle globale du modèle de tâches organise les interactions dans un style modal en introduisant la notion d'**espace d'interaction** (ou environnement d'interactions), ce qui permet de définir les grands blocs de navigation dans l'application. Dans chaque espace, le style peut alors être non modal, c'est-à-dire qu'il n'y a pas d'opérateurs de précédence dans l'arbre de tâches.

Pour identifier ces espaces, nous proposons d'utiliser la fonctionnalité de regroupement des tâches offerte par CTTE. Nous obtenons ainsi des ensembles de tâches de présentation (PTS – *Presentation Tasks Set*) pertinents en fonction du contexte. L'application, par le concepteur, d'heuristiques de regroupement des PTS définit des grands blocs de tâches accessibles simultanément. Ces blocs de tâches conduisent à un premier niveau de structuration d'agents AMF qui est généralement difficile à identifier depuis le modèle du domaine.

Ce niveau correspond aux différents espaces d'interaction de l'application qui ont une influence directe sur la navigation de l'utilisateur. Les tâches abstraites définissent les ports abstraits des facettes de présentation, et les tâches machine définissent des ports dans les facettes Abstraction des agents.

Le modèle du domaine, évoqué précédemment, vient compléter l'organisation des agents. La traduction des relations de composition et d'agrégation de ce modèle vers le modèle AMF engendre la composition hiérarchique des agents à l'intérieur des espaces d'interactions (Cf §2.5). Les relations d'association sont traduites par des connexions entre ports de différentes facettes.

A ce stade du processus, nous obtenons un premier niveau de structuration du modèle d'interaction AMF.

Les tâches abstraites du modèle de tâches d'interaction conduisent à la définition des ports des facettes présentation. Les méthodes des classes du modèle du domaine définissent des ports des facettes Abstraction des agents. Une fois les ports des facettes identifiés, le concepteur établit leurs connexions. De cette manière, il met en correspondance le modèle de présentation et le modèle d'abstraction à travers le modèle d'interaction.

Il reste à affiner ces relations en concrétisant les ports des facettes et en implémentant les classes applicatives du système. C'est l'objectif de la phase suivante. Cette concrétisation se base sur les informations issues du contexte d'utilisation du système.

2.6.3 Concrétisation des modèles

Nous avons vu qu'à ce stade, à partir d'une tâche d'interaction abstraite et en fonction du contexte, nous faisons apparaître progressivement des tâches d'interaction concrètes.

Pour cette opération, le concepteur peut s'appuyer sur les patterns d'interaction pour la concrétisation des tâches abstraites en tâches concrètes en accord avec le contexte d'utilisation. Les ports abstraits des facettes présentation seront remplacés par des ports concrets. A ces facettes qui sont devenues des facettes concrètes, le concepteur doit alors associer les classes applicatives de la présentation. Comme nous le verrons dans la section suivante consacrée aux différentes stratégies d'adaptation possibles, il existe à ce niveau plusieurs options dont certaines reprennent les approches à base de balises développées dans la section 1.4.3.

Du côté abstraction, la modélisation UML des relations entre classes du domaine est traduite par la structuration et les connexions entre ports du modèle AMF. Le concepteur s'occupe uniquement de l'implémentation du contenu des classes applicatives laissant la communication entre les classes à la charge du modèle d'interaction AMF à travers le moteur.

Bien que nous ayons jusqu'à présent focalisé nos travaux sur le modèle de tâches et le modèle du domaine, la concrétisation des autres facettes suit la même logique.

2.6.4 Exécution de l'application

Cette phase est responsable de l'instanciation des éléments de l'application (agents AMF et instances des classes applicatives). Dans l'implémentation que nous avons élaborée, ce rôle a été confié à ce que nous appelons le moteur AMF.

Le **moteur AMF** est un composant logiciel qui forme le cœur de l'architecture AMF. Il assure deux fonctions principales :

- le chargement de la description du modèle d'interaction concret défini dans les phases précédentes en vue d'instancier les différents objets de l'application (les objets AMF et les objets applicatifs).
- le contrôle l'application qui consiste à traiter et acheminer des messages entre des méthodes des classes applicatives à travers les objets AMF (ports, facettes, agents et administrateurs).

Les facettes « concrètes », c'est-à-dire les classes applicatives de présentation, d'abstraction et les autres, sont associées au Moteur AMF pour fournir les comportements finalisés. Il s'agit donc d'assembler les différents éléments constitutifs de l'application. Nous n'abordons pas, à ce stade, les questions de mise en forme de l'interface de

l'application. Cet aspect du problème est traité par les classes applicatives de présentation et nous aurons l'occasion de revenir sur ce point plus loin.

La description graphique du modèle d'interaction AMF est transformée en fichiers de description XML. Ceci a l'avantage de favoriser l'automatisation de la construction de l'application. A partir de la description XML, le moteur AMF est capable d'instancier les objets AMF (agents, facettes, ports et administrateurs) et de charger les facettes concrètes programmées spécifiquement. C'est ainsi que la facette AMF de Présentation indique les services proposés à l'utilisateur et la facette AMF d'Abstraction représente les services du noyau fonctionnel. Ces services abstraits sont implémentés dans des classes applicatives qui jouent finalement le rôle de facettes concrètes.

2.6.5 Vision globale du processus de construction

Dans cette section, nous avons présenté, phase par phase, le processus générique de conception et de construction des systèmes interactifs que nous avons progressivement mis en place. Il s'inspire des démarches du génie logiciel et des modèles d'architecture en les incorporant dans une approche à base de modèles. L'objectif de cette approche est de combiner les avantages issus de ces trois démarches dans un processus structuré, formalisé et adaptable.

Dans cette section nous avons mis en avant les liens qui doivent exister entre le modèle de tâches et le modèle du domaine à travers le modèle d'interaction, ainsi, que les liens entre ces modèles et leurs classes applicatives qui passent aussi par le modèle d'interaction.

La Figure 16 donne un aperçu général sur ce processus appliqué à l'exemple du lecteur de musiques. Les flèches expriment les différentes phases du processus. Ainsi, la flèche en pointillé (en haut) exprime la phase d'élaboration du modèle d'interaction abstraite à partir du modèle de tâches abstraites et du modèle du domaine système. Ensuite, les flèches latérales expriment la phase de concrétisation des modèles. Les flèches pleines traduisent l'implémentation des classes applicatives associées aux facettes du modèle d'interaction à partir des éléments extraits des modèles concrets. Finalement, le moteur AMF récupère les classes applicatives et la description XML du modèle d'interaction pour l'exécution de l'application.

Le processus que nous venons de présenter correspond à un cas donné et un contexte prédéfini. Il se base sur le modèle d'interaction pour assurer les liens internes et externes du système. Dans l'exemple que nous avons utilisé pour expliquer les phases, nous nous sommes limités à un minimum de modèles (tâches et domaine). Cependant, nous avons indiqué à plusieurs reprises que le rôle du modèle d'interaction ne se limite pas à assurer les liens entre ces deux modèles, mais qu'il s'étend à tous les autres modèles du système pour peu que ceux-ci soient explicités à travers des facettes dédiées. Cette caractéristique ainsi que le rôle de pilote du contrôle joué par AMF nous a amené à identifier un certain nombre de stratégies d'adaptation en exploitant la flexibilité du modèle d'interaction.

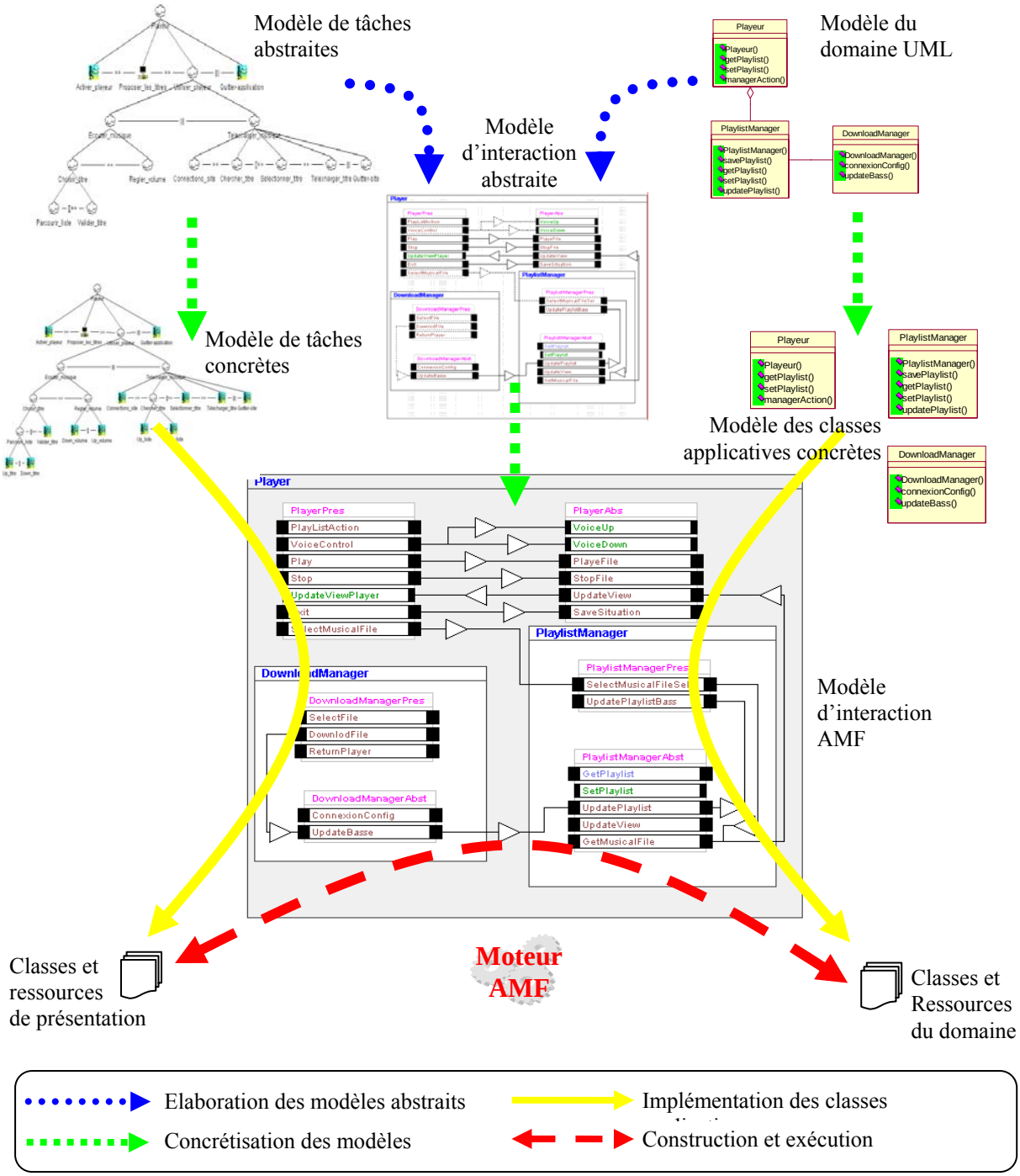

Figure 16 : Processus générique de conception et de construction des systèmes interactifs

2.7 Stratégies d'adaptation

Une fois le processus de conception et de construction défini, nous devons nous intéresser aux options offertes au concepteur pour mettre en œuvre l'adaptabilité du système en fonction des objectifs d'adaptation et des différences entre les contextes cibles.

Rappelons que nous avons adopté une définition du contexte d'utilisation qui se décline en quatre parties : la plate-forme (cible principale de notre intérêt), l'utilisateur, l'environnement et l'activité. La distance entre deux contextes d'utilisation caractérise l'ensemble des différences entre les caractéristiques des contextes.

A ce jour, nous avons identifié quatre stratégies d'adaptation qui peuvent être combinées dans une application réelle selon les niveaux d'adaptation à mettre en oeuvre.

2.7.1 L'adaptation est superficielle

Lorsque la différence entre deux contextes ne remet en question ni la structuration ni les connexions du modèle d'interaction AMF, ce sont les classes applicatives qui sont rattachées aux facettes *Présentation* des agents qui doivent être modifiées. C'est par exemple le cas lorsque deux plates-formes possèdent les mêmes capacités interactionnelles (même dispositif d'interaction) mais se distinguent par leur capacité d'affichage (résolution, couleurs, son…).

Trois possibilités s'offrent alors :

- Le développeur génère plusieurs versions des classes applicatives selon les contextes d'exécution. Au moment de la construction, le moteur charge la même description AMF du modèle d'interaction puis il rattache la bonne classe applicative à chaque facette *Présentation* du modèle.
- La deuxième solution consiste à bâtir des classes de présentation paramétrables. Selon la valeur d'un paramètre, la classe concrète de présentation utilise tel ou tel *widget* ou change les attributs d'un même *widget*. Cette approche est utilisable lorsqu'on connaît de façon précise les degrés de liberté possibles et qu'ils s'inscrivent dans une même boîte à outils.
- La dernière solution consiste à définir une classe de présentation générique susceptible de traiter une description formelle ou semi-formelle de l'interface. Les classes concrètes de présentation héritent ou sont associées à cette classe générique. Cette approche est clairement dans la droite ligne des approches UIDL. En effet, un mécanisme de rendu générique (les *renderers* classiques) interprète un fichier XML de description d'interface abstraite qu'il concrétise en fonction du contexte.

Cette forme d'adaptation peut être traitée dans la troisième phase du processus de conception.

L'utilisation d'un agent multi-présentations (multiples facettes de présentation) rejoint la stratégie adoptée pour la description du « polymorphisme » des Comets (Calvary *et al.,* 2004). Cependant, et dans l'état actuel de notre système, les capacités d'auto-description et d'auto-adaptation requises par les Comets ne sont pas encapsulée dans les agents AMF. Cette adaptation est effectuée manuellement par le concepteur qui, une fois que les spécifications du contexte cible sont connues, sélectionne la bonne facette *Présentation* à fournir au moteur pour la construction du système.

La question qui reste posée est celle de la granularité de l'agent. Est-ce un agent composant de base (bouton multiforme) ou un agent de service (ex : agent d'identification) ? Cette question est à rapprocher de la gestion de l'ergonomie et de l'homogénéité de l'interface dans une approche d'auto-adaptation (cohérence globale au sein d'un agent). A notre avis, l'utilisation des agents multi-présentations favorise l'aspect réutilisation des composants. Un agent d'identification, par exemple, peut être utilisé pour différents systèmes interactifs.

2.7.2 L'adaptation porte sur les dispositifs d'interaction

Pour utiliser un système interactif, l'utilisateur doit avoir à sa disposition un ou plusieurs dispositifs d'interaction. Ces dispositifs l'aident à interagir avec le système en employant différentes techniques d'interaction. Une technique d'interaction est une méthode pour accomplir une tâche générique à l'aide d'une IHM (matérielle et/ou logicielle) (Bowman, 1999, 2002). Il n'y a pas de bijection entre techniques d'interaction et dispositifs d'interaction puisqu'un même dispositif permet de réaliser différentes techniques d'interaction de même qu'une même technique peut être mise en œuvre par différents dispositifs. Par exemple, lorsque le style d'interaction retenu est la manipulation directe, la tâche « sélectionner et déplacer un objet » peut être effectuée par glisser-déposer à l'aide une souris, un stylet ou un autre pointeur d'écran. La même technique d'interaction (glisser-déposer) avec les mêmes dispositifs d'interaction en entrée est employée pour accomplir d'autres tâches telles que la destruction ou la création d'un objet. Cependant, il est évident que certaines techniques d'interaction sont mieux adaptées à certains dispositifs qu'à d'autres, ce qui se traduit généralement par des règles ergonomiques.

Si les plates-formes cibles possèdent des dispositifs d'interaction en sortie qui sont similaires et des dispositifs d'interaction en entrée qui supportent les mêmes techniques d'interaction, l'architecture du système peut être conservée pour les différentes plates-formes. Toutefois, des facettes dédiées aux dispositifs d'interaction en entrée seront ajoutées dans le modèle d'interaction. Le rôle de ces facettes est d'interfacer d'un côté l'interaction entre ces dispositifs et les composants de l'interface, et de l'autre, tout le reste du système.

Dans l'absolu, ces facettes sont peu intéressantes pour interfacer les dispositifs standards (souris, clavier) dont la gestion est embarquée nativement dans les langages de programmation. En revanche, l'utilisation de facettes *Dispositif* prend un réel intérêt pour interfacer les dispositifs non standards (gant numérique, *eye traking*, RFID, etc.).

Pour illustrer ces facettes *Dispositif* et leur mode d'utilisation, prenons l'exemple d'une calculatrice (Masserey *et al.*, 2005). L'agent principal de l'application est doté de facettes *Dispositif* associés aux différentes API permettant de piloter les dispositifs concrets (clavier, gant numérique, etc.). Les ports de ces facettes sont connectés au port de la facette présentation de l'agent *Calculator*. Ainsi, G. Masserey a développé une facette dispositif pour interfacer les gestes réalisés à l'aide d'un gant numérique (*DataGlove*). L'emploi de facettes *Dispositif* peut également être utile pour modéliser la collaboration de plusieurs dispositifs dans l'accomplissement d'une ou de plusieurs tâches (mouvement du gant et touche clavier) en utilisant des administrateurs « *AND* » (Figure 17).

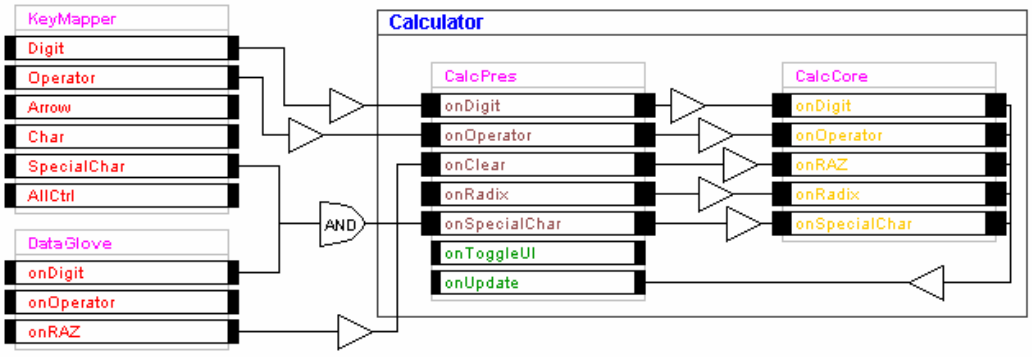

Figure 17 : Utilisation conjointe de deux facette dispositifs

De fait, nous pouvons remplacer complètement une facette *Dispositif* par une autre en la chargeant de la totalité du contrôle de l'interaction externe du système. Ce changement n'affecte à aucun moment l'architecture des agents qui compose le système. Ainsi, ce sont ces facettes spécialisées (et non directement le *widget* de l'interface) qui récupèrent les actions de l'utilisateur et les acheminent vers l'objet concerné de l'interface.

L'idée portée par cette stratégie d'adaptation des interactions en entrée, rejoint les travaux effectués par Dragicevic (2004) qui propose un modèle basé sur des configurations d'entrée, où des dispositifs d'entrée sont librement connectés aux applications à travers des adaptateurs. Ce modèle a donné lieu à une très intéressante boîte à outils (ICon : *Input Configurator*[8]), pour la construction d'applications interactives configurables en entrée. De son côté, notre approche essaye de traiter la totalité du processus de conception et d'adaptation des systèmes. L'adaptation en entrée ne forme qu'une des stratégies d'adaptation que nous proposons. Toutefois, la formalisation des facettes *Dispositif* et aspect réutilisation de ces facettes (pattern de facettes) forment des points de rapprochement avec les propositions de Dragicevic.

Parmi les axes de recherche futurs, notons qu'il est sans doute possible d'imaginer des administrateurs de contrôle chargés de piloter des flux d'évènements pour supporter la multimodalité.

2.7.3 L'adaptation porte sur le style d'interaction

Les stratégies d'adaptation proposées précédemment sont des solutions qui s'appliquent dans la troisième phase de notre processus. Jusqu'alors, les distances entre contextes d'utilisation évoquées se limitent à l'aspect visualisation de l'interface et aux dispositifs d'interaction tout en gardant les mêmes techniques d'interaction.

Le changement de technique d'interaction associé à un changement de style d'interaction (glisser-déposer avec souris vs. interaction clavier) passe généralement par une spécification différente des ports et par un changement plus profond des connexions pour les administrateurs de contrôle. Dans ce cas, nous privilégions une stratégie d'adaptation basée sur l'utilisation de patterns d'interaction. Comme nous l'avons évoqué (cf. § 2.4.3), les patterns d'interaction sont des patterns de conception associant un fragment de modèle AMF spécifique à une technique d'interaction (Samaan *et al.,* 2003). Ces patterns peuvent être des patterns intra-agent (applicables au sein même d'un agent) ou inter-agents (agissant ainsi sur plusieurs niveaux dans la hiérarchie des agents).

En optant pour une technique spécifique d'interaction, le concepteur peut employer les bons patterns de tâches en vue de concrétiser les tâches abstraites du modèle de tâches. Il remplace chaque tâche abstraite du modèle de tâches par le pattern de tâches (un sous-arbre de tâches concrètes). Ensuite, il incorpore dans son modèle d'interaction AMF le pattern d'interaction associé à ce pattern de tâches dans le modèle d'interaction.

Revenons sur l'exemple du lecteur de musiques. Pour concrétiser le modèle de tâches abstraites, nous pouvons concrétiser la tâche de sélection simple dans une liste « Sélectionner_titre », soit par deux tâches d'interaction élémentaires (défilement avec des flèches et sélection si on dispose d'un clavier) soit par une simple tâche dans le cas d'une désignation directe (clic souris par exemple).

[8] http://inputconf.sourceforge.net/

Parallèlement, l'utilisation d'un pattern d'interaction associé au pattern de tâches, comme ceux présentés précédemment, implique généralement un enrichissement des facettes et des connexions du modèle d'interaction. Cette restructuration fait émerger de nouveaux ports dans les facettes Présentation et des nouveaux administrateurs dans le système. Les ports des facettes Abstraction restent globalement inchangés. Cependant, et pour des besoins spécifiques, nous pouvons être amenés à enrichir également les facettes Abstraction et en conséquence le modèle même du domaine.

L'utilisation des solutions d'adaptation basées sur les patterns est une approche en plein essor (Van Weilie *et al.,* 2003) (Gaffar *et al.,* 2004). Sinnig (2004) propose d'associer directement des composants de la présentation aux patterns de tâches. Luyten (2004) propose des patterns de tâches classés par famille de supports et associés à des composants de l'interface utilisateur. Dans notre travail, nous avons adopté une approche qui associe les patterns de tâches à des fragments d'architecture AMF. Cette solution vise en premier une adaptation de la structuration du système qui se répercute en deuxième temps sur la structuration de la présentation.

2.7.4 L'adaptation porte sur la structuration du système

Pour des contextes d'utilisation éloignés où les plates-formes cibles ne possèdent ni les mêmes capacités d'affichage ni les mêmes capacités de calcul, les techniques d'interaction supportées par les plates-formes sont également différentes. Une restructuration totale du modèle d'interaction s'avère, dans ce cas, nécessaire. Néanmoins, cette restructuration peut très bien être supportée au cours de la deuxième phase du processus en ayant recours à deux actions complémentaires :

- le filtrage des tâches non significatives dans un contexte,
- la restructuration des espaces d'interaction.

Pour le filtrage du modèle de tâches, le concepteur élague de l'arbre des tâches abstraites les tâches non réalisables sur la plate-forme cible. Le concepteur élague également les ports abstraits du modèle d'interaction correspondants aux tâches élaguées.

Reprenons le lecteur de musiques : pour une plate-forme de disposant pas de connexion Internet, le concepteur peut élaguer de l'arbre de tâches abstraites la tâche *Télécharger_musique* pour obtenir un arbre de tâches abstraites filtré.

Pour restructurer les espaces d'interaction, le concepteur doit utiliser les différentes heuristiques de regroupement de tâches proposées par CTTE. En effet, si cet outil génère automatiquement un premier niveau de regroupement en PTS (*Presentation Task Set*), ce regroupement est minimaliste, le contenu d'un PTS se limitant fréquemment à une seule tâche. Le nombre de PTS est alors relativement grand. Ensuite, l'outil propose quatre heuristiques de regroupement de tâches :

- si deux (ou plus) PTS diffèrent seulement d'un élément et que ces éléments sont au même niveau et sont connectés par un opérateur de séquence, les deux PTS peuvent être regroupés ensemble.
- si un PTS est composé d'un seul élément, il peut être inclus dans un autre PTS contenant cet élément.
- si un certain nombre de PTS partagent la plupart des éléments, ils peuvent être unifiés.

- s'il y a un échange d'information entre deux tâches, elles peuvent être regroupées dans les mêmes PTS pour mettre en évidence les transferts de données.

Ces heuristiques sont celles implémentées dans l'outil CTTE. D'autres heuristiques peuvent très bien être identifiées. Dans nos travaux nous avons ajouté l'heuristique suivante :

- Si une tâche machine se trouve isolé dans un PTS, celle-ci peut être regroupée avec le PTS précédent ou suivant.

A ce jour, ces heuristiques de regroupement doivent être appliquées manuellement par le concepteur.

L'objectif final est d'obtenir un regroupement pertinent de tâches selon le contexte d'utilisation. Par exemple, un PC possédant un large écran est capable d'afficher dans le même espace d'interaction un nombre plus grand de tâches qu'un téléphone portable avec un écran restreint. Ainsi, le concepteur peut appliquer successivement les heuristiques de regroupement pour arriver à une structuration satisfaisante pour une plate-forme avec un grand espace d'affichage. Toutefois, d'autres caractéristiques que l'espace d'affichage peuvent amener à une restructuration en espaces d'interaction. L'homogénéité des informations portées par les tâches et le profil de l'utilisateur (préférences, compétences, etc.) peuvent également conduire à une restructuration en espaces d'interaction.

Cette stratégie d'adaptation est comparable à celle utilisée par Mori *et al.* (2004). Cependant, le filtrage de tâches appliqué par Mori s'effectue par famille de plate-formes (PC, PDA, Téléphone portable). Dans notre approche, nous avons choisi de laisser ce filtrage suivre la volonté du concepteur, car le filtrage des tâches et leur regroupement en PTS doivent respecter des caractéristiques plus détaillés que celui apporté par le découpage en famille de supports. L'exemple que nous venons de présenter plus haut est un exemple type des limites de la solution basée sur les familles de plates-formes. Une tâche de téléchargement des données sur Internet n'est plus restreinte à un PC et peut s'appliquer à un grand nombre de plates-formes alors qu'il existe encore des PCs sans connexion Internet.

Dans les années à venir, nous allons nous intéresser à d'autres techniques que celles basées sur les PTS pour restructurer les espaces d'interaction. En particulier, nous souhaitons étudier l'utilité des tâches intermédiaires de l'arbre des tâches (celles qui ne sont pas des feuilles) dans la définition des espaces d'interaction.

2.8 Instrumentation de l'approche

Après avoir présenté notre approche pour la construction d'applications interactives adaptables, nous avons identifié l'ensemble des modèles nécessaires à son bon déroulement ainsi que les liens qui les rattachent, puis nous avons exposé les différentes stratégies d'adaptation supportées par notre approche.

Le génie logiciel montre qu'un modèle d'architecture ne prend tout son intérêt que s'il est associé à une **méthode de conception** et à des **outils de développement**. La nécessité de fournir de tels outils de développement est d'autant plus importante avec AMF que nous avons montré que son utilisation implique la définition de nombreuses opérations. En contrepartie, ces outils masquent cette complexité, alors qu'ils sont à même de générer automatiquement la structure logicielle du système interactif. Ils ont l'avantage de guider l'implémentation.

Malgré l'importance du travail à mettre en oeuvre, nous avons pu au fil des années instrumenter notre approche selon trois axes :

- la définition d'une description XML du modèle d'interaction pour agréger les informations en provenance des autres modèles et assurer l'interopérabilité ;
- le développement de l'entité moteur AMF en java pour assurer l'exécution ;
- la construction d'un éditeur graphique du modèle d'interaction qui préfigure ce que pourrait être un atelier de génie logiciel basé sur AMF. Cet outil contient en effet un générateur de squelettes des classes applicatives et est destiné à supporter l'usage des patterns d'interaction. Il permet aussi de lancer l'exécution du modèle en cours.

Ces trois composants forment le cœur de l'instrumentation de notre approche. Ils ont atteint aujourd'hui une maturité suffisante pour faire des développements de plus grande envergure. Nous allons donc prochainement les proposer au téléchargement pour qu'ils soient évalués par d'autres chercheurs.

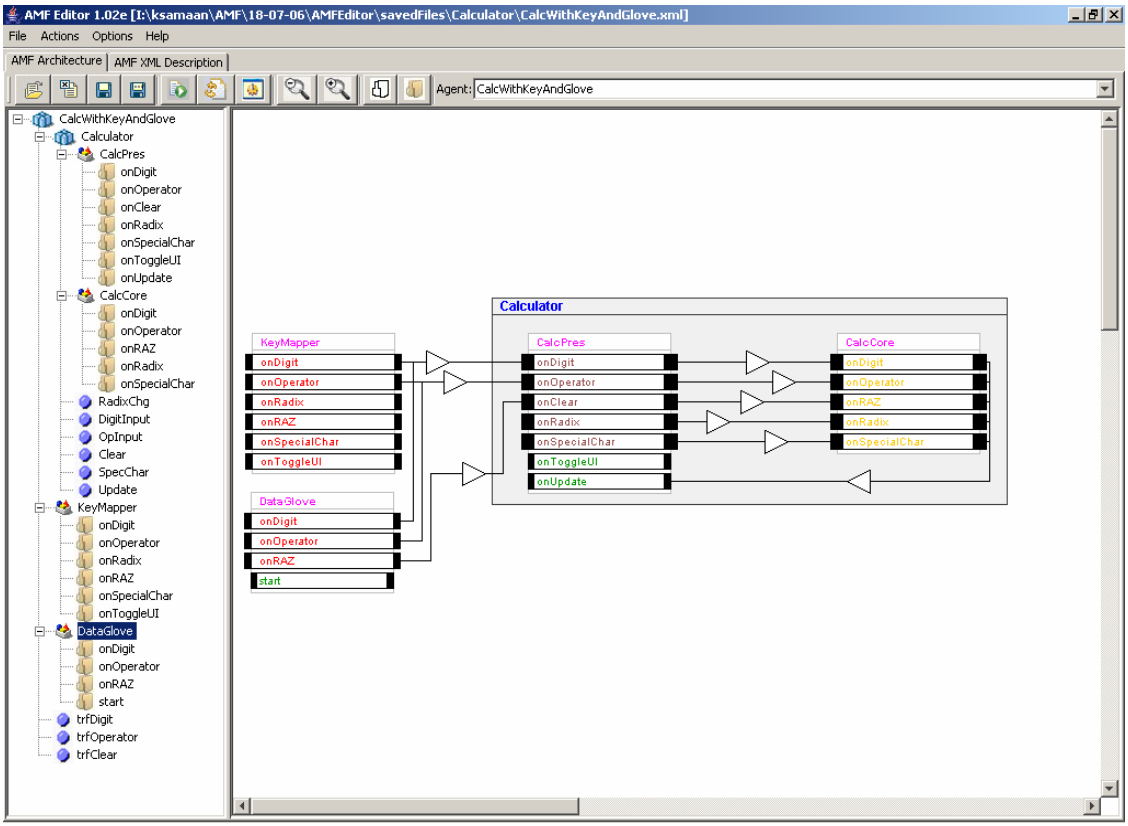

Figure 18 : Un écran tiré de l'éditeur AMF v1.02

Contexte de recherche :
- Thèse Kinan Samaan

Publications :
- **TARPIN-BERNARD F., SAMAAN K.**, A Method for building adaptable software that respect usability, *Computer Human Interaction and Software Engineering*, vol 2, Springer HCI Series, A. Seffah, J. Vanderdonckt, M. Desmarais (eds.), soumis 10/2006

- **MASSEREY G., CHI DUNG T., SAMAAN K., TARPIN-BERNARD F., DAVID B.T.**, Environnement de conception et développement d'applications interactives basées sur l'architecture AMF, *IHM'2005*, Démonstration, Toulouse, 2 pages

2.9 Conclusion et perspectives autour d'AMF

Notre travail constitue une étape importante vers la définition de modèles, méthodes et outils pour construire et implémenter des applications interactives adaptables aux contextes d'utilisation. En fournissant une première proposition opérationnelle validée à travers des exemples présentant un premier niveau de complexité, nous ouvrons la porte à de très nombreuses pistes de recherche qui vont pouvoir enfin être expérimentées et sortir du cadre seulement théorique.

A l'occasion d'une réunion du Groupe de Travail CESAME[9] (Conception et Evaluation de Systèmes interactifs Adaptables et/ou MixtEs) en mai 2006, nous avons pu réaliser à quel point notre travail se trouvait au centre de tous les travaux menés en France aujourd'hui dans la mesure où il apparaissait clairement qu'il était possible d'enrichir notre modèle des différentes approches proposées. Ainsi, les Comets (Calvary *et al.,* 2005) peuvent être modélisées en AMF. Leur capacité d'auto-description avancée (notamment en terme d'utilisabilité) et leurs liens avec les modèles (contexte, évolution...) pourraient faire l'objet de facettes spécifiques (Utilisateur, Plate-forme, Adaptation, etc.). La multimodalité d'ICARE (Bouchet *et al.,* 2005) pourrait être prise en compte par de nouveaux administrateurs de contrôle (redondance, équivalence...) et probablement des agents spécifiques. Comme nous l'avons montré avec le support d'interactivité à base de tags RFID, AMF peut servir de plateforme pour la réalité mixte. Les approches comme IRVO (Chalon, 2004) se prêtent en effet bien à la définition de facettes spécialisées.

Dans le même esprit, si l'on veut être capable de gérer des contextes techniques complexes (détection d'environnement, migration de services, etc.) il faudra faire évoluer le moteur AMF pour le rendre **moins monolithique**. En proposant par exemple une implémentation WComp (Cheung Foo Wo *et al.,* 2006) du moteur, on serait alors capable de distribuer les applications AMF. Finalement, on obtiendrait également ici, une nouvelle façon de proposer un support de travail collaboratif, plus robuste que l'implémentation AMF-C (Tarpin-Bernard, 1997) réalisée il y a quelques années. On a également vu qu'une des forces de AMF résidait dans **la formalisation et la standardisation du contrôle**. Si nous avons eu peu d'occasions d'élaborer de nouveaux administrateurs de contrôle (hormis un administrateur piloté, c'est-à-dire offrant une entrée spéciale de contrôle), il est sans doute possible d'aller plus loin dans la programmation de ces administrateurs en s'inspirant notamment des travaux menés sur ISL (Blay-Fornarino *et al.,* 2004).

Enfin, depuis nos premiers travaux sur AMF, nous avons toujours eu la conviction que, moyennant certaines adaptations dans la présentation du modèle, en particulier dans le masquage de certains détails, la représentation graphique d'AMF pouvait constituer un moyen pour l'utilisateur final d'accéder à la structure interne de l'application et ainsi la reconfigurer (débrancher des administrateurs de notification dans une activité collaborative, échanger une facette de présentation, etc.). De ce point de vue, enrichie d'une gestion de la réalité mixte type IRVO, une telle représentation AMF pourrait servir de méta-IHM externe, au sens de (Roudaut et Coutaz, 2006) puisqu'elle permettrait aux utilisateurs de contrôler et évaluer l'état de leur espace interactif ambiant (en voyant les

[9] http://www.irit.fr/CESAME/

agents en présence et leurs relations). Ceci rejoint également les questions relevées dans le § 2.1 traitant de la collaboration entre concepteurs et utilisateurs, et en particulier celles de Mørch (2000) qui soulignait que, parce que l'adaptation par l'utilisateur d'un système nécessite obligatoirement un certain niveau de compréhension de la structure des logiciels et de leurs modes de fonctionnement, il faut une collaboration indirecte entre les concepteurs et les utilisateurs finaux. Cette « collaboration » passe par l'utilisation de représentations multiples (modèles plus ou moins orientées système) permettant aux utilisateurs de mieux comprendre la structure des logiciels.

D'autres axes de recherche et développement devront également être menés autour de l'éditeur AMF qui a vocation à devenir un véritable atelier de génie logiciel capable de fournir des services de haut niveau qu'il ne remplit actuellement que partiellement :

- lien direct avec les **autres modèles** et en particulier le modèle de présentation (la structure de l'interface) qui n'apparaît pas aujourd'hui explicitement dans notre approche car traitée directement par les classes applicatives de présentation ;
- meilleur support de **l'intégration de composants** AMF déjà développés ;
- fourniture d'une **bibliothèque extensible de patterns** de conception AMF ;
- communication avec des **éditeurs externes** type Eclipse et CTTE ;
- amélioration des fonctions de mise en forme des schémas AMF (interface zoomable pour plonger dans l'architecture, facilités pour afficher/cacher certains éléments comme par exemple les administrateurs de création et de destruction, gestion de vues multiples, etc.)

Notre travail s'est focalisé sur la mise à disposition du concepteur de modèles et technologies susceptibles de l'aider à concevoir et mettre en œuvre différentes déclinaisons cohérentes d'une application en fonction des contraintes du contexte. Il s'agit donc bien de concevoir des applications facilement adaptables. La poursuite de nos travaux conduira naturellement à ce poser la question de **l'adaptativité,** c'est à dire la possibilité pour le système de passer automatiquement d'une configuration à une autre au gré des changements de contexte. Ceci supposera notamment de conserver dans l'application une trace des premières étapes du processus à savoir les modèles abstraits. Des facettes spécifiques et/ou des agents spécialisés devront être définis pour modéliser à la fois le contexte et les opérations à réaliser lors d'un changement de contexte.

L'approche AMF nous semble parfaitement préparée pour supporter ces approches dans la mesure où elle a été conçue pour gérer des **changements architecturaux dynamiques** d'une part et gérer à travers des facettes spécifiques des liens vers d'autres modèles (l'environnement, l'évolution, l'utilisateur, etc.).

Sur ce dernier point (le modèle de l'utilisateur) mes collaborations avec les neurologues et les spécialistes en psychologie de SBT m'ont permis d'envisager de nouveaux développements originaux pour caractériser et exploiter le profil cognitif des utilisateurs. Pour simplifier les expérimentations indispensables à la validation de telles recherches, nous avons choisi de nous intéresser à des applications stéréotypées de type hypermédia. Dans le prochain chapitre nous résumons nos principales contributions.

Notons que Kinan Samaan vient de soutenir sa thèse le 3 octobre 2006, et qu'un certain nombre de travaux parmi les plus récents présentés dans ce chapitre n'ont pas encore été publiés.

3 Techniques spécifiques d'adaptation

A l'occasion de nos différentes recherches, nous avons été amené à mettre au point différentes techniques spécifiques d'adaptation en particulier dans le champ des hypermédias adaptatifs. Ce chapitre présente les innovations les plus significatives.

3.1 Hypermédias adaptatifs

Sur le Web, les documents électroniques sont généralement présentés aux lecteurs sous la forme d'un hypermédia. La caractéristique de ces hypermédias traditionnels réside dans le fait que les mêmes pages et les mêmes liens sont présentés à tous les utilisateurs. Or ces utilisateurs se différencient les uns des autres selon leurs besoins, leurs connaissances sur le sujet, etc. Ainsi, ils ne seront pas forcément intéressés par les mêmes informations et n'emprunteront pas les mêmes chemins ou liens durant leur navigation. Les informations et liens inutiles entraînent une surcharge cognitive chez l'utilisateur. L'utilisateur risque facilement de se perdre dans l'hypermédia. Cela entraîne une mauvaise représentation mentale du document qu'il parcourt ce qui influence sa compréhension. En effet, en sciences cognitives, la compréhension d'un document est souvent caractérisée par la construction mentale d'une représentation de ce document. La lisibilité du document peut être définie comme l'effort mental nécessaire au processus de construction d'un modèle (Kaheneman, 1973). Assister l'utilisateur dans la bonne construction de ce modèle s'avère nécessaire pour qu'il y ait une meilleure compréhension du document.

L'objectif des hypermédias adaptatifs est d'adapter la présentation de la connaissance et d'aider l'apprenant à se diriger dans les hyperdocuments. De ce fait, dans un hypermédia nous devons pouvoir modifier aussi bien le contenu des pages que les liens entre les différentes pages (Brusilovsky, 1996 a). Mais c'est surtout sur l'adaptation des liens que la plupart des techniques ont été développées (Brusilovsky, 1996 b).

Les hypermédias adaptatifs visent, entre autre, à réduire la charge cognitive de l'utilisateur et les risques d'incompréhension du document qui en résultent.

Selon Brusilovsky, l'hypermédia adaptatif est utile quand le système est appelé à être utilisé par des personnes ayant différentes connaissances et différents buts et quand il est vaste. Ces personnes peuvent être intéressées par différents types d'informations et peuvent utiliser différents liens de navigation. Un système hypermédia adaptatif peut aider l'utilisateur dans sa recherche en éliminant l'information la moins pertinente ce qui réduit utilement les dimensions de l'espace de navigation.

L'architecture des hypermédias adaptatifs, comme pour beaucoup de systèmes de formation assisté par ordinateur, s'appuie principalement sur deux modèles : le modèle du domaine et le modèle de l'apprenant. La combinaison de ces informations est exploitée pour déterminer les médias à utiliser pour présenter les concepts à l'apprenant. Aujourd'hui, la technique la plus répandue consiste à relier chaque concept à une ou plusieurs pages physiques. Ces relations sont représentées par des liens hypertextes.

3.2 Principes de base de l'adaptation au profil utilisateur

Avant d'aborder nos propres contributions, nous allons présenter les différentes familles de techniques de représentation, d'initialisation et de mise à jour des modèles utilisateurs.

3.2.1 Les différents modèles types de l'utilisateur

Différentes techniques sont utilisées pour représenter les utilisateurs. Dans cette section, nous allons décrire très brièvement les approches les plus utilisées.

Dans les modèles de recouvrement (overlay models), les connaissances de l'utilisateur sont situées par rapport à un sous-ensemble des connaissances « idéales ». Ces connaissances idéales sont généralement exprimées par un modèle de référence : celui-ci est préalablement établi par un expert du domaine, ou correspond plus simplement à un gabarit de concepts manipulés, ou encore définit ce que l'on veut que l'utilisateur apprenne (liste de connaissances valuées). Le modèle de l'utilisateur prend la forme d'un ensemble de paires concept-valeur, où le concept représente une connaissance élémentaire et la valeur associée indique le niveau d'assimilation ou de maîtrise que l'utilisateur a de cette connaissance. Cette valeur peut être exprimée à l'aide de probabilités ou d'intervalles d'appartenance qui expriment une croyance du système à l'égard de la maîtrise par l'utilisateur de cette connaissance.

Les profils utilisateurs sont des modèles également basés sur des couples (items, valeurs) mais ne se réfèrent pas uniquement à un modèle de connaissances préétabli. Les items auxquels sont associées les valeurs (booléennes, discrètes, nominales, probabilités) représentent aussi bien des connaissances du domaine, que des préférences, des capacités, un vécu de l'utilisateur, etc. L'objectif est de disposer de tous les ingrédients pour proposer à chaque apprenant du sur-mesure.

Plutôt que d'avoir des modèles individualisés, les modèles de stéréotypes utilisent des combinaisons de couples items-valeurs pour définir des classes d'utilisateur telles que novice, initié, expert. L'utilisateur est donc associé ou parvient à une des catégories élaborées et hérite de ses propriétés. Il dispose, par suite, des adaptations réalisées pour le stéréotype correspondant. Ce manque d'individualisation de l'adaptation peut être contré par le recours à des approches mixtes qui consistent à initialiser un modèle utilisateur à l'aide d'un stéréotype (souvent faute de mieux), puis à l'affiner au fur et à mesure de la session interactive en utilisant un modèle de recouvrement ou un profil. Un exemple est donné par les travaux de (Cannataro *et al.*, 2001) dans lesquels l'association d'un utilisateur à un stéréotype est réévaluée, en fonction de ses actions, par le biais d'une méthode probabiliste. Pour être efficaces, les approches à base de stéréotypes supposent d'établissement d'une taxinomie de cas-types, des batteries de tests/questionnements qui identifient le type et des procédures d'instanciation adaptées à chaque type.

L'exemple cité précédemment illustre un double enjeu de la représentation des connaissances à propos de l'utilisateur : gérer l'incertitude de cette représentation et supporter son évolution. Dans ce but, des travaux exploitant les réseaux de neurones tels que (Chen *et al.*, 2000) ou Bayésiens dans (Henze & Nejdl, 1999) sont proposés.

La construction et la gestion des modèles utilisateur reposent dans la plupart des travaux sur un module (ou composant) spécifique. Deux courants pour la mise en œuvre de ce module s'opposent (cf. (Fink et Kobsa, 2000) pour une revue). D'un côté, une approche monolithique considère le composant dédié au modèle utilisateur comme un élément totalement intégré à l'application. De l'autre, des serveurs de modélisation de l'utilisateur, indépendants de toute application cible, sont proposés. Ces outils sont principalement rencontrés dans le domaine du commerce électronique (e.g. *Personalization Server* et produits complémentaires de l'*Art Technology Group*).

Après avoir décrit les différentes représentations de modèles utilisateur, nous présentons dans la section suivante les techniques les plus couramment utilisées pour l'acquisition de tels modèles.

3.2.2 Les modes d'instanciation du modèle utilisateur

Il existe deux manières d'obtenir de l'information sur les utilisateurs : l'acquisition explicite, selon laquelle nous utilisons une source externe à l'outil pour créer et/ou compléter le modèle de l'utilisateur et l'acquisition implicite, selon laquelle le système infère l'information à partir de connaissances disponibles ou déduites sur l'utilisateur (i.e. acquisition incrémentale). Les méthodes explicites reposent souvent sur un questionnaire et une classification des utilisateurs prédéfinie dans le modèle utilisateur. Si elles tentent de poser des questions aux utilisateurs sur leurs préférences ou connaissances, comme le souligne Rich (1989), elles constituent rarement des sources fiables d'information. Dans tous les cas, les méthodes explicites peuvent être jugées lourdes par les utilisateurs.

D'une façon plus générale, voici quelques méthodes couramment utilisées (les trois premières sont explicites, la dernière est implicite) :

- L'observation directe :

  Il s'agit de la méthode la plus précise mais la plus lourde et la plus coûteuse car elle nécessite des personnes qualifiées derrière chacun des individus observés. On peut ainsi identifier des classes d'utilisateurs, leurs tâches types ainsi qu'un certain nombre de facteurs critiques, comme la pression sociale, qui ont des effets néfastes sur l'utilisation du système.

- Les interviews :

  Cette technique permet d'obtenir d'autres types d'information : l'expérience, les opinions, les motivations comportementales voire dans certains cas les avis sur les outils existants. Ils sont plus courts et moins coûteux que la technique d'observation, néanmoins, ils nécessitent aussi du personnel qualifié.

- Les questionnaires :

  Ou "comment obtenir à moindre coût un maximum de données" ? (Petrelli *et al.*, 1999) Les résultats obtenus permettent des études statistiques et des généralisations plus fortes que les interviews. Les questionnaires peuvent être collectés par des personnes non expérimentées. Ils permettent d'avoir à la fois un aperçu de la situation et des points d'information plus précis. Néanmoins, les questionnaires ne sont pas toujours des sources d'information fiables (parfois, les utilisateurs ne les complètent pas avec tout le sérieux nécessaire).

  Embarqués dans les systèmes eux-mêmes, les questionnaires sont souvent utilisés pou récolter des informations factuelles (âge, sexe, profession, etc.).

- L'analyse de traces d'usage :

  Il est possible d'obtenir de l'information en capturant des traces d'usage du système. Ces données sont analysées par un module d'apprentissage, embarqué dans le système ou externe. Des algorithmes d'apprentissage se chargent de collecter des informations essentiellement comportementales sur l'utilisateur (Kobsa & Pohl, 1995).

La construction du modèle utilisateur soulève un certain nombre de questions :

- de pertinence : quels types de modèles choisir et pourquoi faire ;
- méthodologiques : comment assurer le contrôle qualité des modèles choisis ? Quelles sont leurs durées de vie ?
- techniques : comment instancier le modèle à partir de l'interaction système-utilisateur et en assurer une mise à jour régulière ?

La difficulté dans la conception d'un système adaptatif réside en grande part dans la recherche d'un équilibre entre les adaptations utiles et des techniques de modélisation fiables.

3.2.3 Initialisation et mise à jour du modèle utilisateur

Une fois choisis, en réponse aux objectifs de personnalisation du concepteur, le ou les modèles de l'utilisateur doivent être instanciés par des informations, données, valeurs pas forcément instantanément ou facilement disponibles.

Pour illustrer ce problème, Fischer (2001) présente deux cas de figures extrêmes :

- M. X qui possède des connaissances limitées sur l'ordinateur, ne sait pas ce qu'est un programme de messagerie. Comment peut-il être informé que l'aide existe ?
- M. Y est un programmeur très expérimenté. Il connaît la plupart des logiciels et préfère que tout soit direct et rapide. Il ne supporte pas les programmes qui interrompent ses tâches.

Dans ces conditions, comment initialiser le modèle de l'utilisateur pour que le logiciel adaptatif se comporte de façon adéquate avec ces deux utilisateurs très différents. La solution la plus souvent retenue consiste à confier à l'administrateur du système (s'il existe) la responsabilité d'initialiser l'identité du nouvel utilisateur et donc l'état initial du modèle. S'il n'y a pas d'administrateur (qui peut être l'enseignant dans un système de e-formation) ou si celui-ci n'a pas la connaissance suffisante sur l'utilisateur, le modèle sera initialisé avec une valeur neutre.

Une fois initialisé, le modèle utilisateur doit pouvoir être mis à jour. On retrouve alors des techniques explicites et implicites qu'il convient de combiner :

- La première solution est la plus courante. Elle consiste à appliquer des changements en réponse à des commandes de l'utilisateur. Ceci conduit typiquement à modifier les composantes du modèle utilisateur relatives par exemple aux préférences. Ce paramétrage explicite de l'interface est indispensable pour donner un contrôle à l'utilisateur. Cependant, cette possibilité est souvent très largement sous-utilisée. En effet, Larsoe (1999) souligne que les utilisateurs des systèmes sont généralement obstinés dans leurs usages et préfèrent rester dans leur niveau d'expertise le plus longtemps possible. Ainsi, les utilisateurs continuent à se servir de leurs anciens modèles "it is not what I want but since I use it so seldom, I will not bother to change it". Cette constatation est également valable pour les usagers non expérimentés.
- La deuxième solution repose sur une évolution du modèle de l'utilisateur après ou pendant l'usage d'un programme, basée sur des mesures sensées traduire la capacité de l'utilisateur à utiliser correctement le programme. Le problème dans ce cas se pose sur la façon d'évaluer les actions de l'utilisateur.

La mise à jour du modèle utilisateur peut s'exercer à deux niveaux, bien que les systèmes adaptatifs ne fonctionne généralement qu'au second niveau :

- Un « observateur » (souvent caché) vérifie que le modèle est **structurellement** (ses composants, son type, etc.) toujours pertinent et alerte lorsque cela n'est plus le cas. En général, le système ne sait pas, sauf situation répétitive, changer de modèle ou en élaborer un autre. S'il savait le faire, ce serait un système dit « auto-organisé ».
- Une procédure d'actualisation des **données du modèle** traduisant l'évolution normale du système (progression dans la maîtrise des savoirs, acquisition de comportements, meilleures performances…).

Une dernière question nous semble importante à soulever. Elle concerne la temporalité et la visibilité de l'adaptation. En d'autres termes, quand et comment notifier à l'utilisateur que le système a mis à jour le modèle le concernant. Par exemple, comment doivent apparaître les changements quand l'utilisateur devient de plus en plus expert ? La solution la plus simple, consiste généralement à annoncer les changements explicitement sur l'écran, mais le problème qui se pose alors est à quel moment l'interface annonce un changement au risque de perturber l'utilisateur dans sa tâche ? Enfin, ces questions sont à rapprocher des problématiques de co-évolution que nous avons abordées précédemment puisque nous pouvons être confrontés à des situations où le système change au moment où l'utilisateur commence à apprendre, ce qui le perturbe, voire où le système peut ne jamais atteindre une situation stable. Dès que le système s'adapte l'utilisateur change et inversement.

3.3 Notre modèle cognitif de l'utilisateur

La plupart des travaux sur l'adaptation des systèmes au profil cognitif de l'utilisateur se basent sur les styles cognitifs tels que la dépendance du champ en utilisant des tests tels que celui de Felder (Felder, 1996). Or la notion de style cognitif est un concept souvent décrié. Citons simplement Coffield *et al.* (2004) qui ont récemment tenté de valider les principaux styles d'apprentissage, notion très proche de la notion de style cognitif, et ont montré que la plupart des tests utilisés pour déterminer ces styles souffrait d'un net manque de fiabilité. De plus, les classifications grossières qui en découlent conduisent très souvent à des résultats peu exploitables (par ex : 60% style A – 40% style B). En outre, ces styles demandent aux auteurs des supports de cours de très gros efforts de restructuration.

Après avoir tenté nous-même d'explorer les styles cognitifs, nous avons cherché à identifier des éléments plus fiables et plus facilement utilisables dans des processus d'adaptation. Dans les sections suivantes, nous allons montrer comment nous avons su exploiter dans un contexte d'hypermédia pédagogique adaptatif un modèle cognitif basé sur l'évaluation des habiletés cognitives élémentaires élaboré à SBT.

Contexte de recherche :
- Thèse Halima Habieb-Mammar

Publications :
- **HABIEB-MAMMAR H., TARPIN-BERNARD F.** Cognitive Styles and Adaptive Multimodal Interfaces. *CogSci'2002, the 24th annual meeting of the Cognitive Science Society.* Gray, W. D., Schunn, C. D. (Ed.). Hillsdale, NJ: Erlbaum. Août 2002. pp. 626-630.
- **HABIEB-MAMMAR H., TARPIN-BERNARD F.** How to Incorporate Cognitive Styles into Adaptive Multimodal Interfaces. *World Conference on Educational Multimedia, Hypermedia and*

Telecommunications (EDMEDIA 2002), editeur(s): Association for the Advancement of Computing in Education (AACE), Denver, Colorado, Etats Unis d'Amérique, 24-25 juin 2002.

3.3.1 Le modèle HAPPYneuron™

Le modèle utilisateur qui a été développé à SBT en collaboration avec une équipe de psychologues et neurologues adopte une posture radicalement différente de celle des styles cognitifs. Il s'agit en effet de définir la performance de l'individu selon un certain nombre d'habiletés. Le modèle Happyneuron se compose ainsi :

- d'une **facette cognitive** dans laquelle sont décrites les habiletés cognitives de l'utilisateur. Ces capacités correspondent aux mécanismes grâce auxquels un sujet reconnaît et acquiert des informations, les traduit en représentations puis en connaissances, et enfin les utilise dans la génération de comportements plus ou moins élaborés (Lemaire, 1999). Nous donnons plus loin la liste des habiletés utilisées.
- d'une **facette démographique** dans laquelle se trouvent les informations relatives à l'âge, au sexe et au niveau socio-éducatif de l'utilisateur. Ces données sont renseignées par un questionnaire que doit compléter l'utilisateur lors de sa première connexion. Ces informations permettent de comparer ses performances avec d'autres utilisateurs qui appartiennent à la même classe.

SBT propose un site Web d'entraînement cognitif supervisé, www.happyneuron.fr, dont les innovations principales seront décrites dans la section 3.4. Ce service est constitué d'exercices de stimulation cognitive et d'un processus de supervision de l'entraînement. Toutes les performances réalisées sont archivées et analysées. Au fil du temps, un profil cognitif est élaboré (Tarpin-Bernard *et al.*, 2001). Ce profil décrit 5 grands domaines de compétence regroupant différentes fonctions cognitives élémentaires :

- La **mémoire** est l'ensemble des mécanismes utilisés pour acquérir, stocker, puis réutiliser des informations. La mémoire n'est pas une fonction unitaire, elle est divisée en différents sous-systèmes selon la nature consciente ou inconsciente de l'apprentissage ou du rappel, le type de stratégie employée (verbale ou visuelle) la qualité du souvenir mémorisé (expérience personnelle, faits culturels, procédures motrices spécialisées).

- L'**attention** est fondamentale pour l'efficacité intellectuelle car elle est sollicitée dans la plupart des tâches cognitives. La mobilisation des ressources attentionnelles permet dans certains cas de privilégier un stimulus parmi d'autres (attention focalisée) ou au contraire de distribuer simultanément sa concentration sur plusieurs stimuli de l'environnement (attention partagée). On regroupe également sous la catégorie attentionnelle les formes de mémoire à court terme, indispensables à la réalisation de tâches. La mémoire de travail est une forme spécialisée de mémoire à court terme qui maintient temporairement une information pendant l'exécution d'activités cognitives complexes.

- Le **langage** regroupe l'ensemble des processus d'expression et de compréhension selon deux codes : oral et écrit. Il constitue le plus important moyen de communication d'un individu qui dispose de modules de gestion des sons, de l'orthographe, du sens des mots, des enchaînements grammaticaux, des procédures de compréhension, d'organisation du discours. Bien que le stock de mots constitue finalement une forme de mémoire sémantique, celui-ci est géré par une zone distincte du cerveau.

- Le domaine **visuo-spatial** est celui des systèmes de reconnaissance et d'identification des formes ainsi que des procédures d'analyse des positions relatives de plusieurs formes. Il

permet aussi la réalisation de manipulations mentales de formes telles que les rotations spatiales, on parle alors d'imagerie mentale.

- Les **fonctions exécutives** correspondent à des fonctions plus élaborées de logique et de stratégie, de flexibilité mentale et de planification, de résolution de problèmes, de raisonnement hypothético-déductif, et de flexibilité cognitive (changement de comportement). Ces fonctions sont amodales, c'est-à-dire non liées à une modalité sensorielle.

Toute activité cognitive résulte du fonctionnement parallèle ou séquentiel de plusieurs fonctions cognitives élémentaires appartenant aux cinq grands domaines de compétences précédents. Dans le cadre du programme d'entraînement HAPPYneuron™, SBT a défini 25 indicateurs cognitifs à l'intérieur des cinq secteurs (Tarpin-Bernard *et al.*, 2001).

Ce profil cognitif se rapproche de travaux sur les architectures cognitives, domaine de l'intelligence artificielle qui cherchent à encapsuler des aspects de la cognition humaine relativement constants et indépendants des tâches réalisées (Gray *et al.*, 1997; Ritter *et al.*, 2001). Principalement destinés à simuler des comportements humains, ces architectures exploitent des systèmes reposant sur des règles de production de type « SI…ALORS » qui manipulent des paramètres factuels cognitifs, perceptifs et moteurs.

Les systèmes les plus connus sont Soar (Newell, 1990; Altmann & John, 1999), EPIC (Kieras & Meyer, 1997), et ACT-R/PM (Byrne & Anderson, 1998; Byrne, 2001). EPIC par exemple utilise des paramètres types comme la durée moyenne de traitement du processeur cognitif (50 ms) ou le temps moyen de reconnaissance d'une forme (250 ms). ACT-R/PM est une autre architecture cognitive à deux couches combinant un modèle de la cognition centrée sur la mémoire avec des capacités perceptives et motrices (Figure 19).

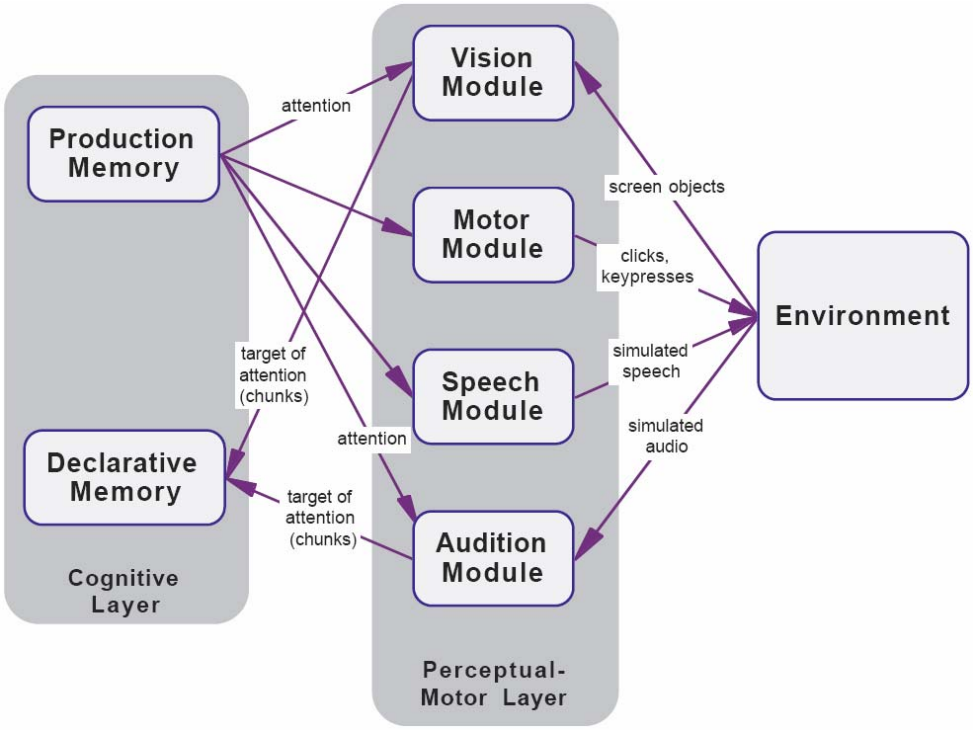

Figure 19 *:* Structure de l'architecture ACT-R/PM

Malgré l'intérêt potentiel de ces architectures elles se heurtent jusqu'à présent à de nombreux problèmes dont l'un des plus manifestes est la complexité du système qui rend l'approche extrêmement difficile à utiliser pour des novices (Byrne, 2003).

Bien que nous n'envisagions pas à court terme d'explorer le champ des intelligences artificielles, nous pensons que nos travaux sur les habiletés cognitives pourraient apporter de nouveaux développements aux architectures cognitives.

3.3.2 Un modèle cognitif pour l'adaptation des hypermédias

Pour l'adaptation des hypermédias nous n'avons retenu pour l'instant que 14 indicateurs cognitifs parmi les 25 (Habieb-Mammar & Tarpin-Bernard, 2003). Nous n'avons pas pris en compte, par exemple, les indicateurs liés aux fonctions exécutives car nous n'avons pas encore identifié de conséquences pertinentes sur la présentation des informations bien que nous pensions que des débouchés pertinents pourraient sans doute être trouvés. Le modèle appliqué est ainsi composé de 14 **indicateurs cognitifs** représentant les habiletés cognitives suivantes : mémoire de travail verbale, visuelle et musicale, mémoire court terme verbale, visuelle et musicale, mémoire long terme verbale, visuelle et musicale, catégorisation, compréhension, richesse lexicale, reconnaissance de formes et exploration visuo-spatiale.

Pour établir rapidement le profil cognitif de l'utilisateur, nous avons élaboré une batterie de 6 exercices alimentant les indicateurs cognitifs sélectionnés. Les exercices étant étalonnés (plusieurs milliers de personnes pour chaque exercice), il nous a été possible de bâtir un module de test dont la durée d'exécution varie entre 25 et 30 minutes. Chaque exercice a été spécialement conçu pour solliciter 2 ou trois indicateurs. Les performances, exactitude et vitesse, de l'individu sont comparées aux étalons.

Nous appelons performance brute réalisée lors d'un exercice, l'ensemble des valeurs capturées lors de la réalisation de l'exercice correspondant aux variables jugées pertinentes par le concepteur de l'exercice. Dans les cas les plus simples (empan verbal ou visuel) cette performance est un score entier (nombre de séquences reproduites avec succès). Dans la majorité des cas, on trouve un couple taux d'exactitude (pourcentage) et temps de réponse (en secondes). Il arrive que la performance brute soit détaillée (score dans phase 1, score dans phase 2, etc.). Les exercices les plus complexes contiennent 8 à 10 valeurs.

La méthode classique qui a été utilisée lors d'une première version du système de scoring est appeler Z score. Elle consiste à construire une performance centrée réduite en considérant que les performances des individus suivent une loi statistique normale.

Pour produire des scores compris entre 0 et 100, la formule suivante a été adoptée pour chaque élément de la performance brute :

$$S = 50 + 20 \times \frac{P - M}{\sigma}$$

Avec P la performance de l'individu, M la performance moyenne et σ l'écart type

Une moyenne pondérée combine ensuite les scores d'exactitude (Se) et de vitesse (Sv). Par exemple S = 0,6 x Se + 0,4 x Sv

Un score inférieur à 10 soit une performance inférieure à M-2σ est très faible alors qu'un score supérieur à 90 est excellent.

Le principal inconvénient de cette méthode réside dans le fait que lorsque les performances ne suivent pas des lois normales centrées, l'algorithme est mis en échec et donne des résultats contestables. Or, il existe un certain nombre de situations où une performance ne suit pas une loi normale centrée. Citons deux cas fréquemment rencontrés dans les exercices HAPPYneuron :

- La performance est naturellement bornée : taux d'exactitude (0 à 100), nombre de mots ou de figures (0 à n)…
- Le temps de résolution est illimité

Dans les deux cas, la distribution des performances ne suit pas des lois normales (gaussiennes) puisque dans le premier cas la courbe de répartition est tronquée d'un côté et que dans le second cas la courbe est étirée d'un côté à cause de personnes mettant un temps beaucoup trop long. De fait, dans ces situations, la moyenne et l'écart-type ne sont plus pertinents.

La nouvelle méthode de scoring que j'ai mise au point repose sur l'outil statistique des percentiles[10]. Il s'agit de construire un histogramme des performances discrétisées puis de construire la courbe inverse de **variation de la performance en fonction du pourcentage de population cumulée** (Figure 20). Dans cet exemple, quelqu'un avec une performance de 10 se situe sur le percentile 41 (41% de la population fait un moins bon score que lui), tandis que le percentile 50 qui sépare la population en 2 groupes égaux se situe à une performance de 11.

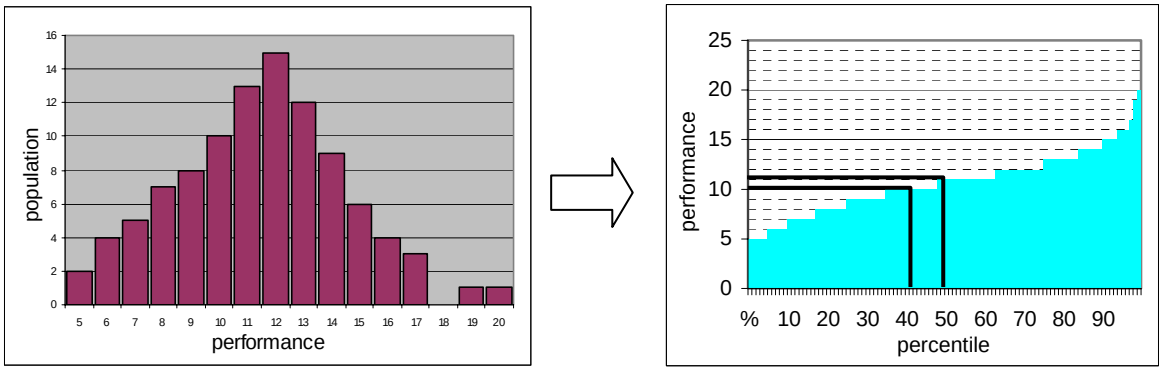

Figure 20 : Passage d'un histogramme de performance à une courbe de percentiles

On voit que cette technique permet d'établir un score sur 100 pour chaque performance si l'on dispose du graphe des percentiles. Ce score est significatif même si la répartition des performances n'est pas centrée.

Ce mode de calcul de score est beaucoup plus robuste que celui mettant en œuvre moyenne et écart-type car il n'impose aucune contrainte sur la répartition des performances. Les seuils jusqu'alors utilisés pour les comparaisons M-2σ et M+2σ peuvent être remplacés par les performances des percentiles 3 et 97 car lorsque la performance suit une loi normale parfaite, la population dont la performance est extérieure à cet intervalle représente 2 fois 2,5%. Pour des raisons de commodité, nous arrondissons à 3%.

[10] En statistique descriptive, un percentile ou un centile est chacune des 99 valeurs qui divisent les données triées en 100 parts égales, de sorte que chaque partie représente 1/100 de l'échantillon de population.

L'inconvénient de cette méthode est qu'elle impose d'avoir une courbe pour chaque performance. Pour obtenir une précision garantie au percentile près, il faut donc stocker 101 points (de 0 à 100).

Pour simplifier le codage des étalons, nous avons choisi d'utiliser 9 scores particulièrement importants dans une courbe de percentile pour le scoring HAPPYneuron : 0, 3, 13, 25, 50, 75, 87, 97, 100. Si l'on linéarise localement la courbe des percentiles autour de chacune de ces 9 valeurs, on obtient généralement de très bonnes approximations (Figure 21).

Notons enfin que les outils d'évaluation du profil cognitif ont été améliorés au fil des années jusqu'à devenir aujourd'hui un programme d'évaluation du potentiel cognitif des individus (PEPCo™[11]). Cet outil est utilisé par des grandes entreprises et des cabinets spécialisés pour des recrutements ou des programmes dits d'*assessment* pour déterminer des trajectoires professionnelles dans des groupes ou effectuer des reclassements lors de fusions ou de restructurations.

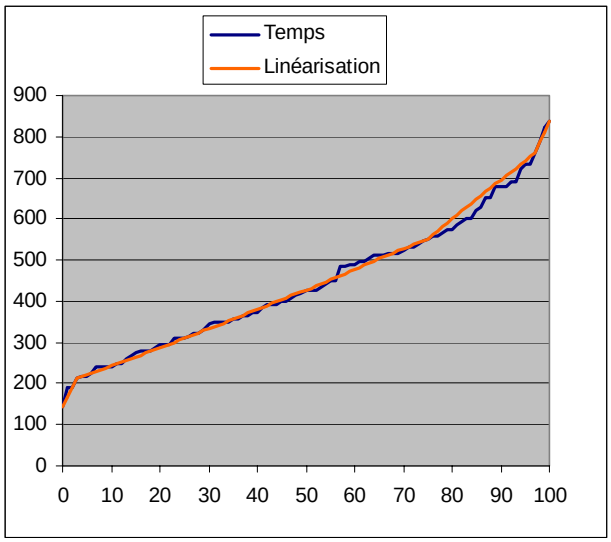

Figure 21 : Exemple de linéarisation des percentiles de performance pour définir des étalons compacts

Contexte de recherche :
- R&D SBT, Projet Européen Eurêka!

Publications (pour des raisons de propriété industrielle, ces travaux n'ont pas été publiés dans le détail) :
- **TARPIN-BERNARD F., HABIEB-MAMMAR H., CROISILE B., NOIR M.** User cognitive model for adaptive interfaces. *NÎMES-TIC'2001. 2nd International Conference, Maîtrise des Systèmes Complexes et la Relation Homme-Système*, Nîmes, France, Décembre 2001.

3.4 Supervision d'un entraînement cognitif à partir du profil cognitif

www.happyneuron.com est le site d'entraînement cognitif supervisé commercialisé par SBT. Ce site destiné principalement aux seniors a pour vocation d'aider les utilisateurs à entretenir voire développer leurs capacités cognitives pour se préserver des effets du vieillissement (déclin cognitif naturel) et diminuer le risque d'apparition de pathologies neurodégénératives (type Alzheimer). Il propose une cinquantaine d'exercices interactifs

[11] http://www.pepco.fr

paramétrables destinés à la stimulation de toutes les facettes cognitives couplés à des mécanismes de supervision et de *coaching*. La Figure 22 présente le principe général de fonctionnement du site.

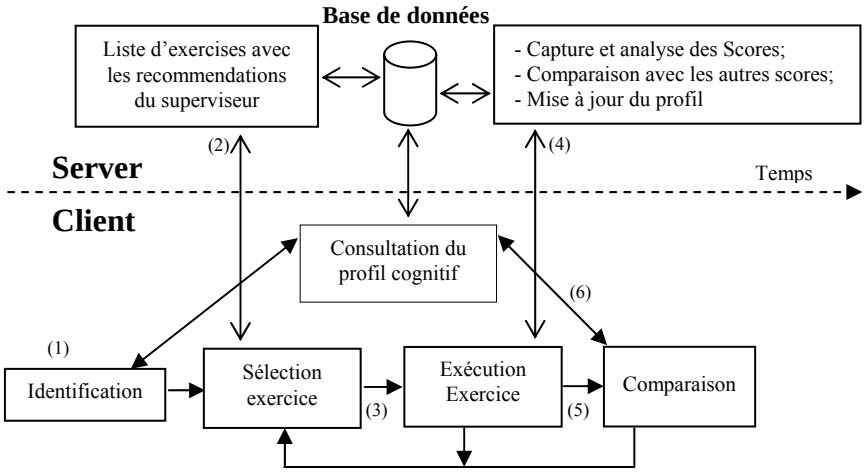

Figure 22 : Principe de fonctionnement d'une séance d'e-training cognitif supervisé

L'objectif de la supervision est triple :

- élaborer un profil cognitif pour que l'utilisateur puisse suivre ses progrès ;
- confectionner des séances d'entraînement adapté au profil de l'utilisateur (choix des exercices et de leur paramétrage) ;
- faire des commentaires et délivrer des conseils pour progresser.

Le premier point consiste à **agréger les résultats** obtenus à l'ensemble des exercices pour construire le profil. Ceci s'effectue facilement en appliquant la formule suivante :

$$ind_i \leftarrow \frac{p_i \times S + Cp_i \times ind_i}{p_i + Cp_i} \qquad Cp_i \leftarrow Min\,(p_i + Cp_i, 100)$$

Avec S : performance normalisée sur 100

p_i : poids de l'exercice sur l'indicateur i sachant que pour chaque exercice $\sum_{i=1}^{25} p_i = 1$

Cp_i le poids cumulé de l'indicateur i depuis que l'utilisateur s'entraîne (limité à 100 pour limiter le poids du passé.

ind_i : valeur de l'indicateur cognitif i de l'utilisateur (le profil contient 25 indicateurs)

Figure 23 : Evolution du profil cognitif

Pour **sélectionner les exercices** suggérés à l'utilisateur, le superviseur calcule des scores pour chaque exercice selon différents critères (ceux qui feront le plus travailler les points faibles, les plus anciens, etc.). Ensuite, il effectue un tirage aléatoire en affectant une probabilité de tirage à chaque exercice lié au score cumulé. Pour favoriser davantage les exercices arrivant en tête, la probabilité est calculée à partir d'une puissance quatrième du score cumulé. Grâce à cette technique probabiliste, on évite tout phénomène de tirage trop répétitif en cas de phénomène d'hystérésis.

Pour chaque exercice, il reste à **définir le paramétrage des variantes**, celles-ci ayant été qualifiées préalablement en trois niveaux (facile, moyen et difficile). Pour ce faire, nous

calculons le score théorique de l'utilisateur en combinant son profil et le profil de l'exercice : $S_{th} = \sum_{i=1}^{25} p_i \times ind_i$. Ce score théorique est le score que devrait obtenir l'utilisateur si sa performance était exactement conforme à son profil cognitif. Si S_{th} est inférieur à 40, le superviseur choisira une variante facile, s'il est supérieur à 70, ce sera une variante difficile. Entre les deux la variante sera moyenne.

Lorsque l'utilisateur a terminé un exercice, en fonction des scores obtenus, l'objectif est alors de **proposer un commentaire adapté**. La combinatoire des situations possibles est à nouveau extrêmement importante (configurations x phase de jeu x exactitude x vitesse). Pour simplifier la tâche de rédaction des commentaires, il n'est pas obligatoire de créer un commentaire distinct pour chaque situation possible. Un tableau stocke pour chaque phase du jeu des commentaires identifiés par une clé structurée. Il est possible d'associer à une clé un lien vers une autre clé. Cela signifie que pour ces deux clés, le commentaire proposé est le même.

Lorsqu'une personne réalise un exercice, il est nécessaire de construire la clé de commentaire correspondant à ses résultats. La première étape consiste à vérifier si la performance ne correspond pas à un cas spécial, par exemple : aucune erreur (exactitude=100%), pas fait sérieusement (exactitude < 10 % et temps de réponse < seuil)... Ensuite, si ce n'est pas le cas, la clé est générée en utilisant une table de la base de données qui indique comment construire la clé et calculer les scores en tenant compte des variantes et des éléments de score (exactitude, vitesse..).

Cette technique permet de produire des commentaires d'une grande richesse. Par exemple, une personne ayant échoué dans une configuration difficile parce qu'elle a voulu aller trop vite se verra proposer le commentaire suivant : « Votre exactitude n'a pas été bonne cette fois ci. En revanche vous avez été très rapide. Prenez plus de temps pour répondre pour obtenir de meilleurs résultats. Vous pouvez également vous entraîner sur le niveau facile avant de revenir sur ce niveau difficile. » Cette technique garantit également des commentaires de qualité en cas de traduction dans une autre langue.

Contexte de recherche :
- R&D SBT, Projet Européen Eurêka!

Publications (pour des raisons de propriété industrielle, ces travaux n'ont pas été publiés dans le détail) :
- **CROISILE B., TARPIN-BERNARD F., NOIR M.** Expérience d'un site internet d'entraînement cognitif destiné aux seniors. *Neurologie.* Paris. Vol. 3 supp 1. No. 2S126. Juin 2002.
- **TARPIN-BERNARD F., HABIEB-MAMMAR H., CROISILE B., NOIR M.** A supervised Program for Cognitive e-Training. *WebNet'2001, World Conference on Web technologies.* Orlando. Octobre 2001, pp. 1208-1213.

Licence : une licence exclusive sur les technogies Happyneuron a été vendue en 2005 à la société américaine Quixit Inc.

Dans la section suivante, nous allons montrer comment nous avons pu exploiter certains éléments du profil cognitif HAPPYneuron pour réaliser une adaptation de la présentation d'hypermédias pédagogiques au profil des apprenants.

3.5 Adaptation de la présentation d'un hypermédia selon le profil cognitif

Dans cette section nous allons présenter les travaux que nous avons réalisés avec Halima Habieb-Mammar dans le cadre de sa thèse de doctorat qui consistaient à étudier un Environnement de Développement et de Présentation d'Hyperdocuments Adaptatifs (EDPHA).

Bien que nous ayons cherché à élaborer des propositions utilisables pour d'autres composants du contexte d'utilisation, nous nous sommes focalisés sur l'adaptation de la présentation d'un hyperdocument au profil cognitif de l'utilisateur. En effet, comme nous l'avons déjà évoqué, les utilisateurs assimilent plus ou moins facilement les concepts qui leurs sont présentés suivant leurs capacités perceptives (audition, vision, etc.) et cognitives (attention, langage, etc.) (Weil-Barais, 2001). Une **adaptation à ces capacités** a pour objectif de **réduire la charge cognitive** des utilisateurs lors de l'exploration des hyperdocuments.

L'environnement EDPHA a cette ambition. Il est basé sur un modèle utilisateur, un modèle de document, ainsi qu'un moteur d'adaptativité qui, à partir des deux modèles précédents et d'une feuille de style, produit un document hypermédia adaptatif (Figure 24). Il peut être étendu à d'autres critères d'adaptation que les critères cognitifs. Nous décrivons dans un premier temps chacun des composants de cet environnement, puis nous présentons un exemple d'hyperdocument adaptatif.

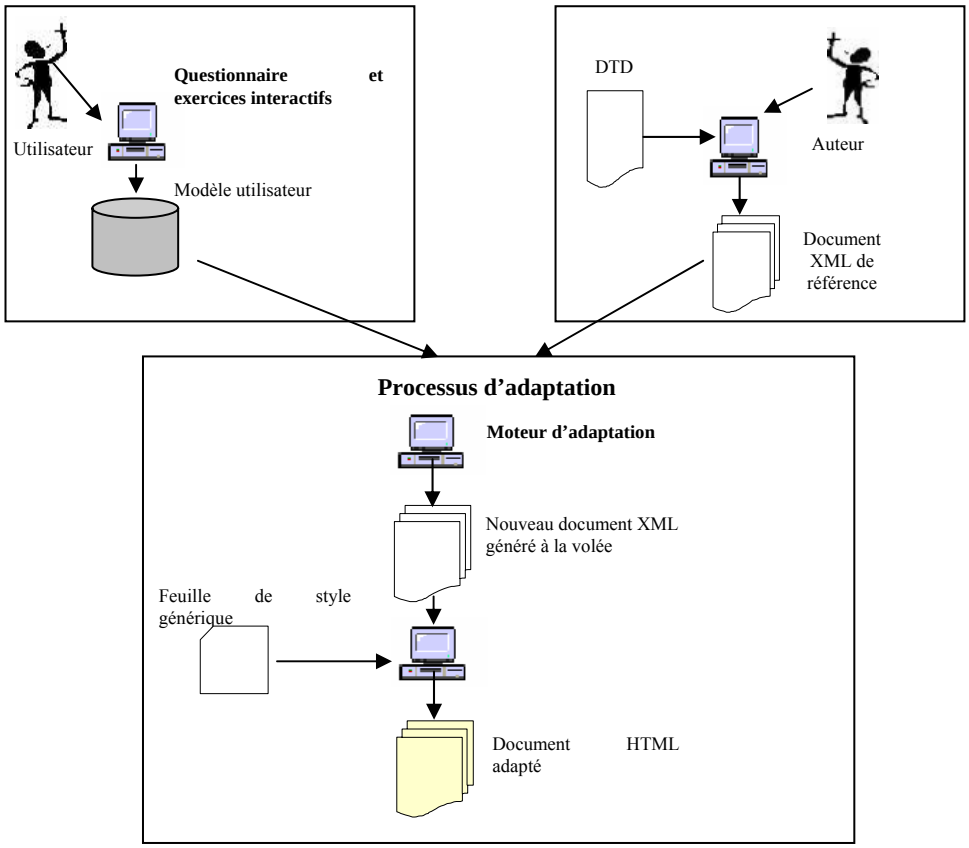

Figure 24 : Architecture de notre Environnement de Développement et de Présentation d'Hyperdocuments Adaptatifs (EDPHA)

3.5.1 Le modèle de documents

Dans EDPHA, le document est vu comme un ensemble de pages (au sens des pages Web). Chaque **page** est découpée en blocs de données (Figure 25). Un **bloc** est une collection d'**éléments** dont chacun est une description du même concept (définition, exemple, description, exercice, etc.) à partir de divers médias ("texte + image + son", "texte + son", "animation + son", etc.). Selon les situations, ces éléments sont des présentations alternatives ou complémentaires (sur le plan de la forme bien sûr). Les grains d'information, les éléments dans notre approche, sont représentés par une collection de médias (par exemple une image avec sa légende textuelle). Nous regroupons les données que nous jugeons indissociables sémantiquement, c'est pour cela que la granularité n'est pas la plus fine, (i.e., au niveau des médias). Ceci nous différencie des découpages dans lesquels les auteurs s'intéressent aux briques d'un seul média dans le but d'adapter le contenu aux utilisateurs (Delestre, 2000) (Crampes *et al.*, 2000).

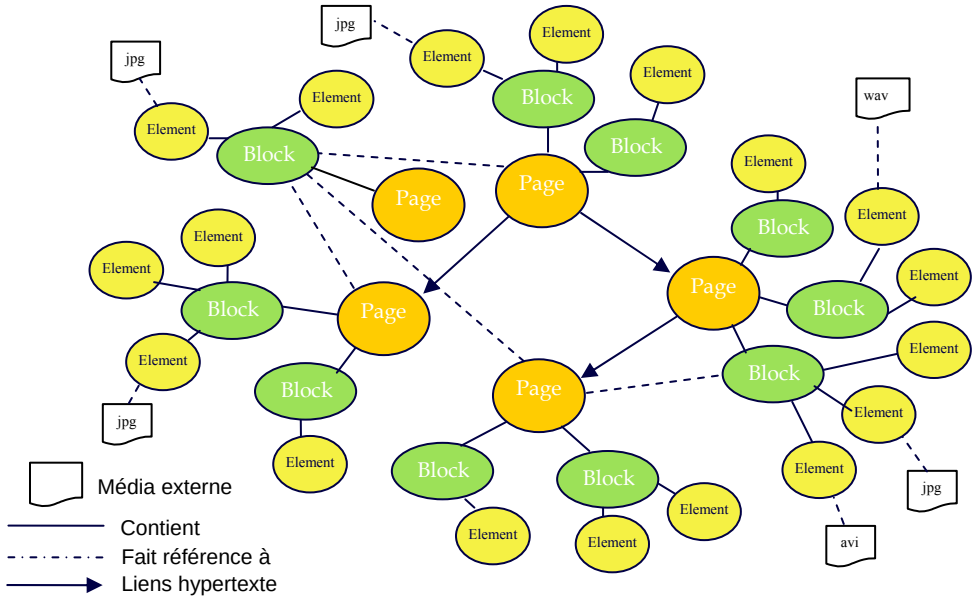

Figure 25 : Graphe des composants d'un hypermédia modélisé avec EDPHA

Pour modéliser un document, nous avons naturellement opté pour XML (*eXtensible Markup Language*) et les techniques qui lui sont associées (DTD, XSLT, XPATH, etc.). En effet, la DTD (Data Type Description) permet de décrire la structure de la page telle que nous l'avons présentée dans le paragraphe précédent. La Figure 26 présente une version simplifiée de la structure que doivent respecter toutes les pages.

Page [title]
- **block** [name, group, position, include]
  - **element** [name, signature, state]
  - **content** [type="text" : "sound" [duration, autostart] : "image" [width, high, link]
- **rules**
  - **combi**
- **exclude**
  - **combi**
- **links**
  - **link** [key_word, url, position]

Figure 26 : Structure simplifiée d'une page

Après avoir présenté le découpage de la page en blocs et éléments, nous allons détailler la structure du bloc et celle de l'élément.

Un bloc tel qu'il a été défini précédemment est une partie du document qui traite un concept donné. Il contient un attribut *name* utile pour l'auteur.

Chaque bloc contient un attribut *group* servant à le rattacher à un groupe de blocs. Ceci est très utile pour contraindre le moteur d'adaptation. En effet, tous les blocs d'un même groupe seront traités de façon identique. Ainsi, lorsque l'on a trois blocs de navigation « précédent », « sommaire » et « suivant » et que chacun d'eux possède plusieurs mises en forme, le fait de les déclarer appartenant à un même groupe garantira que les trois blocs seront présentés de la même façon.

Le bloc peut appartenir à une page comme il peut être défini en dehors de celle-ci. On parlera alors d'un bloc externe ; l'externalisation des blocs permet leur réutilisation dans d'autres pages. L'hyper document peut ainsi être défini comme un graphe et non plus seulement comme un arbre. Le pointeur vers la page des blocs externes est défini par l'attribut *include*.

Afin que les liens hypermédias soient utilisables indépendamment de la forme des objets (image, texte…) et pour simplifier le travail de l'auteur, les liens entre les pages sont externalisés des contenus dans une section *links*.

3.5.2 Qualification de la signature cognitive d'un élément

Nous avons vu qu'un bloc contient une liste d'éléments correspondant à des formes différentes de présentation de l'information. Pour déterminer les éléments proposés à l'utilisateur, nous avons besoin de qualifier chacun d'eux.

Nous avons opté pour une modélisation hybride à 4 dimensions situés entre le modèle perceptif (visuel, auditif, kinesthésique, …) et le modèle de la cognition (langagier, mémoriel, visuo-spatial., …). Un attribut appelé "***signature cognitive***" décrit chaque élément par un quadruplet constitué des composantes suivantes : **Visuelle** (***V***), **Sonore et musicale** (**M**) et **Kinesthésique** (***K***) auxquelles s'ajoute la composante **Langagière** (***L***).

La ***composante visuelle*** est liée à la complexité de l'information visuelle véhiculée par l'élément (principalement par l'image ou la vidéo),

La ***composante sonore*** est relative à la complexité de l'information sonore contenue dans l'élément (i.e. dans un bruit, un jingle, une musique, une voix-off),

La ***composante kinesthésique*** est liée au degré d'interactivité de l'élément (dans l'image, l'animation, les schémas, etc.) ainsi qu'à sa dimension « concrète ».

La ***composante langagière*** correspond à la complexité de l'information verbale contenue dans l'élément (que ce soit dans le texte, l'image, l'animation, le son, etc.).

Le niveau de chaque composante est défini par l'auteur du document. La valeur de chacune de ces composantes est comprise entre 0 et 5 (0 correspond à l'absence de la modalité alors que 5 signifie que la modalité en question est très prégnante). Ainsi, un schéma comporte à

la fois des composantes visuelles et verbales. Une vidéo interactive peut avoir une signature non nulle dans chacune des 4 modalités. Pour aider l'auteur, nous avons défini une grille de référence (Tableau 5).

Composante Signature	V	M	K	L
1	Icône, bouton…	Beep, fond sonore sans sens	Elément cliquable	Légende, titre, mots clefs…
2	Schéma simple	Gingle ou bruit simple à reconnaître (2-3 secondes)	Animation simple	Phrase 2 lignes
3	Schéma, dessin simple	Voix off simple (3 à 10 secondes)	Animation avec menus	Paragraphe 5 lignes
4	Schéma complexe, Photographie taille moyenne	Voix off longue (10 à 30 secondes)	Animation avec manipulation directe	Paragraphe 10-15 ligne
5	Photographie riche	Voix off très longue (plus de 30 secondes)	Animation interactive complexe	Page entière
remarque	Valeur incrémentée si nombreux détails	Valeur incrémentée si la prise de son est directe et avec ambiance		Valeur incrémentée si vocabulaire ou structure complexe

Tableau 5 : Exemples d'assignation de signatures à des éléments

La Figure 27 montre un exemple de page EDPHA dans lequel des signatures cognitives ont été associées à chaque élément. Nous reprendrons plus loin cet exemple lorsque nous construirons dynamiquement une page adaptée à différents profils.

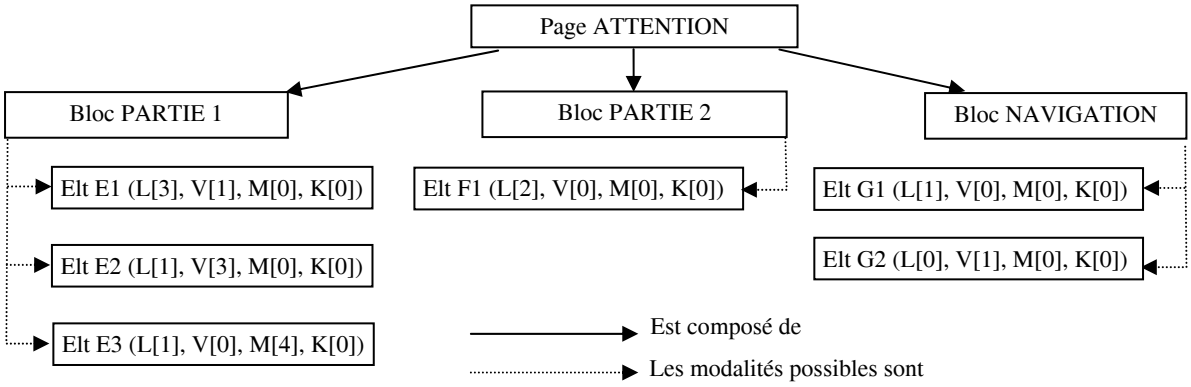

Figure 27 : Exemple de document avec les signatures cognitives de chaque élément

Comme nous le verrons dans la section 3.7, les valeurs des composantes de ces signatures vont être confrontées au profil de l'utilisateur pour trouver les médias qui lui conviennent le mieux.

Notons cependant que, s'il le désire, l'auteur du document peut spécifier la liste exhaustive des combinaisons d'éléments qui doivent être pris en compte (les éléments, qui, selon l'auteur se complètent) ; pour cela le bloc est complété par une partie *rules*. A l'inverse, l'auteur peut également préciser les combinaisons d'éléments qui ne doivent pas être proposées. Les éléments qui s'excluent mutuellement sont référencés par la partie *exclude*.

3.6 Générateur d'hyperdocuments

Plusieurs outils auteur existent actuellement, comme SEPHYR et OPHELIA utilisés dans le projet ARIADNE (ARIADNE, 2002). SEPHYR permet une segmentation conceptuelle à la volée de documents HTML en éléments sémantiquement cohérents. Ainsi, un réseau conceptuel est créé et simultanément affiché. Un hypertexte pédagogique est alors généré, il pourra être utilisé via un navigateur Web. OPHELIA permet, en plus de la segmentation à la volée, de modifier ou de créer le contenu et la structure du document source.

Notre démarche diffère de celle de ces deux outils dans le sens où l'auteur ne fournit que les contenus élémentaires qui composent son cours. En effet, ces contenus sont assemblés en respectant la DTD décrite précédemment. L'auteur insère au fur et à mesure les contenus et établit les règles de leurs compositions. Dans le cas où aucune précision n'est fournie par l'auteur, des combinaisons sont attribuées aux contenus et seront filtrées dans une étape ultérieure. Un document XML est alors généré, il sera utilisé par le processus d'adaptativité

La philosophie principale de notre outil consiste à proposer un grand nombre de degrés de liberté à l'auteur tout en adoptant des comportements par défaut facilitant un apprentissage progressif et un développement incrémental de l'hyperdocument. L'auteur peut basculer de la vue arborescente du document XML (avec visualisation des contenus) au code XML. Il peut également demander une visualisation du document en sélectionnant un profil cognitif spécifique ou type (Figure 28).

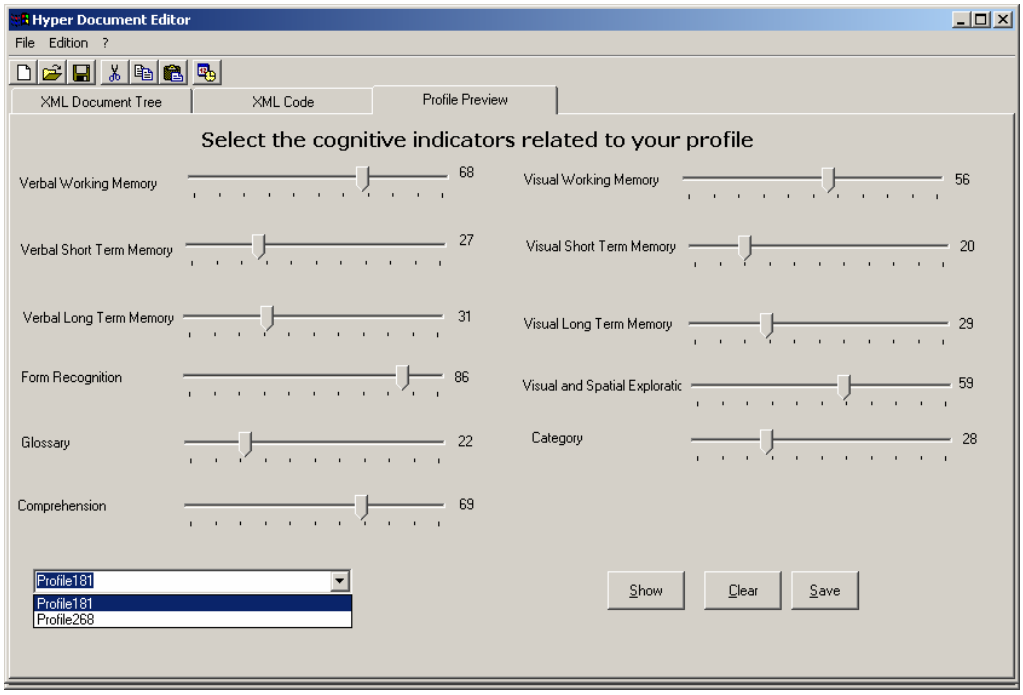

Figure 28 : Sélection d'un profil cognitif pour une prévisualisation d'un document

Il nous reste maintenant à décrire le principe de fonctionnement du moteur d'adaptation que nous avons mis au point pour construire dynamiquement des versions adaptées au profil cognitif de l'utilisateur des pages vues.

3.7 Moteur de l'adaptativité

Dans tout système adaptatif les composants (modèle utilisateur, modèle de document, modèle d'interaction, etc.) sont traités par un processus intelligent qui génère une adaptation. Le processus de base consiste à manipuler ces données en leur appliquant des règles d'inférence (Kobsa, 1994), des règles de sélection (De Bra *et al.*, 1998, 2003) ou des probabilités d'apparition (Pazzani & Billsus, 2002). Dans EDPHA, le processus d'adaptation est basé sur la sélection d'une combinaison d'**éléments** la plus "compatible" avec le modèle utilisateur. La Figure 29 présente les étapes suivies par le moteur d'adaptation.

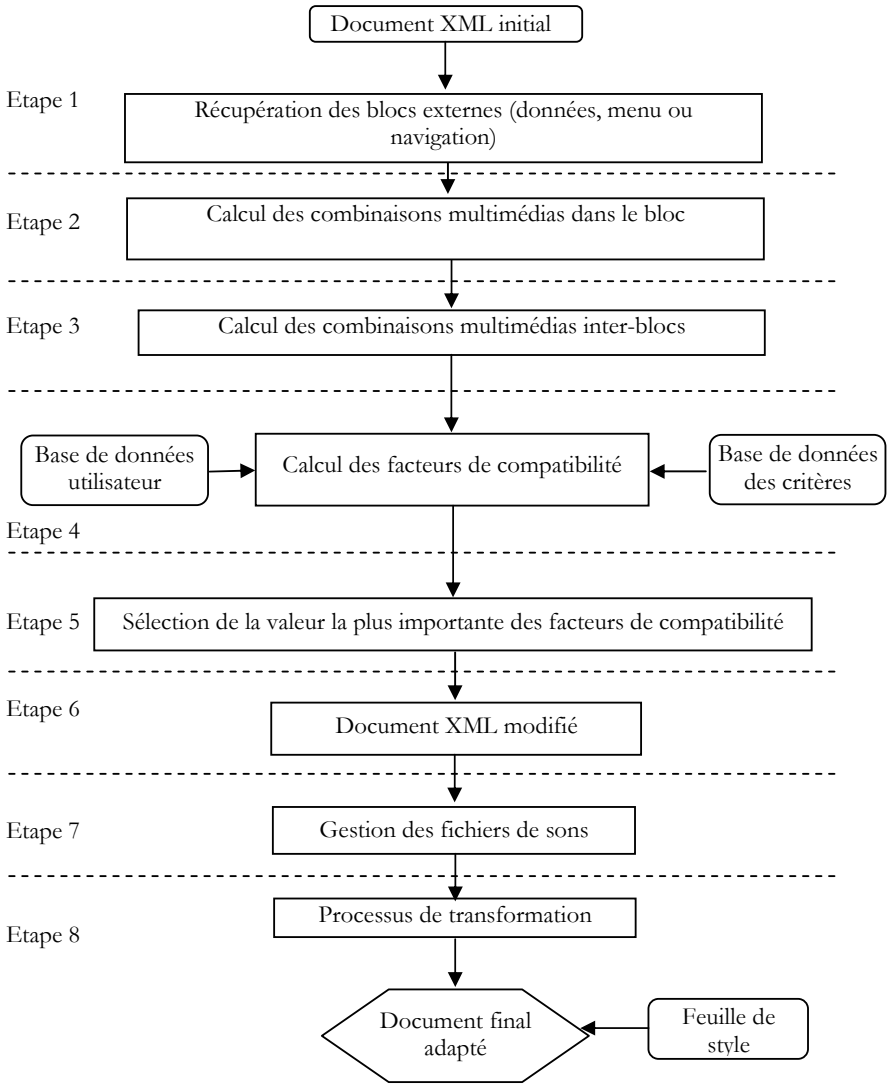

Figure 29 : Etapes du moteur d'adaptation

Si les premières étapes consistent simplement à identifier l'ensemble des combinaisons possibles, l'étape 4 est plus complexe. Pour chaque combinaison un **facteur de compatibilité** est calculé à partir d'un ensemble de critères numériques (paramètres) tels que la maximisation d'une variable. Ces paramètres sont appliqués, d'une part, sur le modèle utilisateur (visuel ou verbal, sa possibilité de gérer deux types de médias, etc.) et, d'autre part, sur le document lui-même (nombre de médias impliqués, degré de complexité de chaque média, etc.). Jusqu'à présent, nous avons utilisé 2 paramètres : P1 et P2. Le paramètre P1 est la moyenne des scores cognitifs de l'utilisateur pondérée par les

caractéristiques des éléments constituant la combinaison évaluée. Il est utilisé pour calculer la combinaison d'éléments dont la signature est a priori la plus compatible avec le profil cognitif de l'utilisateur. Le paramètre P2 rapporte le nombre de signatures pour chaque composante (V, L, M, K) au nombre maximum possible. Il est utilisé pour favoriser la sélection d'une combinaison multimédia. Chaque paramètre peut être pondéré d'un poids. Cette technique, similaire à celle utilisée pour la supervision des séances d'entraînement HAPPYneuron, est ainsi souple et extensible. Si l'on dispose d'un modèle de l'utilisateur plus riche (informations comportementales, modèle des connaissances…) il est possible d'insérer de nouveaux paramètres.

La combinaison pour laquelle la somme des paramètres est maximisée est sélectionnée. Comme l'évaluation se fait au vu de l'ensemble de la page la compatibilité est optimale d'un point de vue global. Ceci est très important si l'on souhaite avoir une cohérence globale du résultat, c'est-à-dire du document adapté.

Les dernières étapes sont relativement « classiques » : le document XML correspondant à la combinaison choisie est élaboré en mémoire en « taggant » les éléments devant être vus et ceux devant être réduits. Finalement une feuille de style CSS assure la mise en forme du résultat final. D'autres feuilles de style plus sophistiquées sont naturellement envisageables. Ce traitement est développé dans une feuille de style XSLT (*eXtensible Stylesheet Transformation*).

3.8 Exemple d'hyperdocument adapté

Illustrons rapidement la technique utilisée en reprenant l'exemple de la Figure 27. La Figure 30 présente les différentes combinaisons intrablocs possibles en tenant compte des contraintes imposées par l'auteur. Supposons maintenant que nous ayons 3 utilisateurs aux profils assez stéréotypés (Pr1, Pr2 et Pr3) présentant une dominante verbale, visuelle ou auditive (Tableau 6).

Name of the intra-block combination	Block	Element	Ve	Vi	M	K
A1	1	E1	3	1	0	0
A2	1	E2	1	3	0	0
A3	1	E3	1	0	4	0
A12	1	E1+E2	4	4	0	0
A13	1	E1+E3	4	1	4	0
A23	1	E2+E3	2	3	4	0
A123	1	E1+E2+E3	5	4	4	0
B1	2	F1	2	0	0	0
C1	3	G1	1	0	0	0
C2	3	G2	0	1	0	0

Figure 30 : Combinaisons intra-blocs

Indicators / Profiles	Sc_{Ve}	Sc_{Vi}	Sc_{M}	WM_{Ve}	WM_{Vi}	Wm_{M}	STM_{Ve}	STM_{Vi}	LTM_{Ve}	LTM_{Vi}	LTM_{M}	Glos	Cat	Und	Rec	Exp
Pr 1 "verbal"	75.0	25.0	50.0	75	25	50	75	25	75	25	50	75	75	75	25	25
Pr 2 "visual"	25.0	75.0	25.0	25	75	25	25	75	25	75	25	25	25	25	75	75
Pr 3 "musical"	25.0	25.0	75.0	25	25	75	25	25	25	25	75	25	25	25	25	25

Tableau 6 : Profils Cognitifs de 3 utilisateurs types (les 13 indicateurs élémentaires + 3 synthétiques)

L'application des algorithmes décrits précédemment conduit à sélectionner pour les profils Pr1, Pr2 et Pr3 les combinaisons respectives : A13 B1 C1, A2 B1 C2 et A3 B1 C2 (Figure 31 a, b et c).

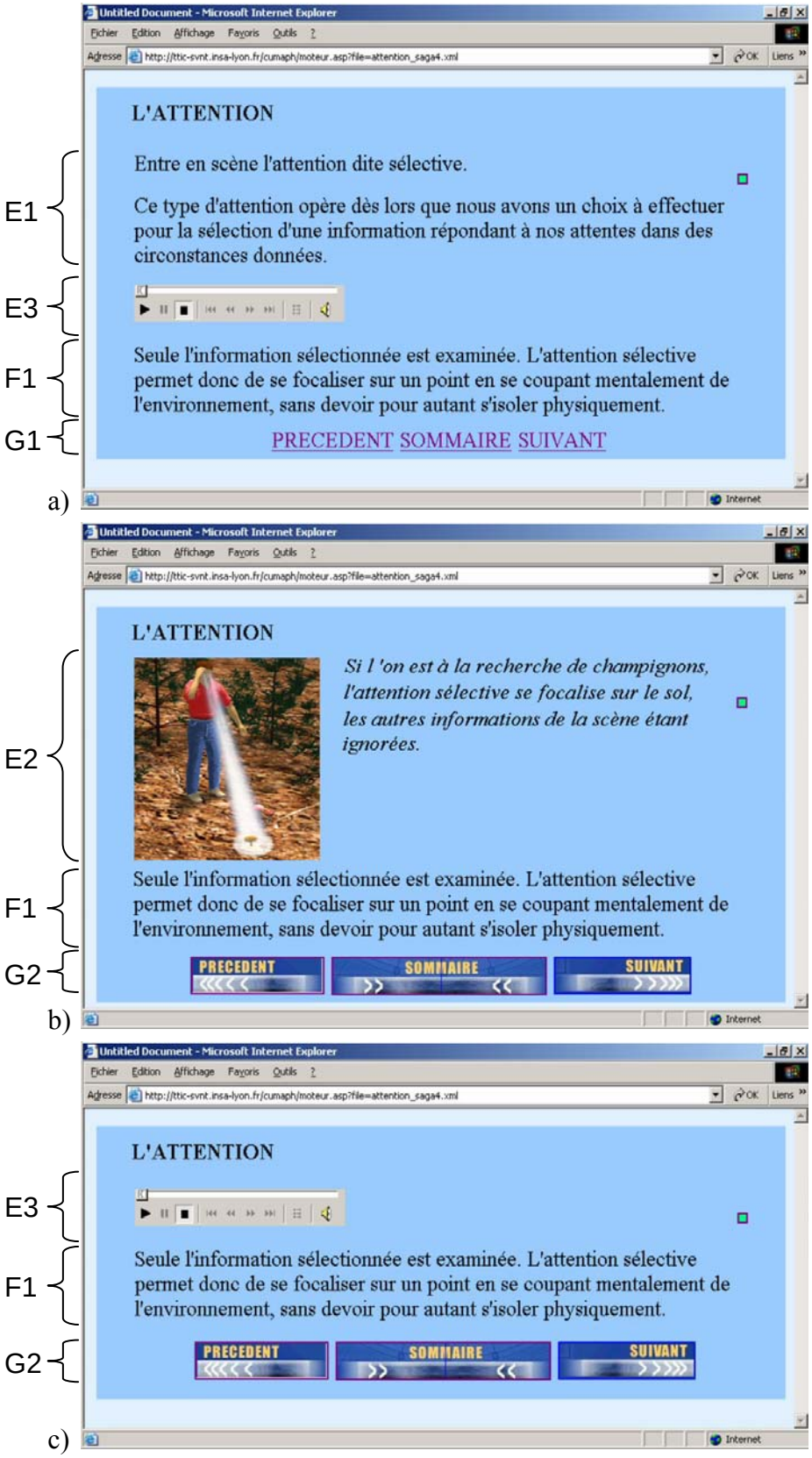

Figure 31 : Une même page adaptée à trois profils différents

Une fois la mécanique d'adaptation en place, il est nécessaire d'étudier dans quelle mesure cette adaptation est utile et efficace. Se pose alors la question de **comment évaluer l'adaptativité d'un système** d'une part et **l'efficacité de cette adaptation** d'autre part, ce qui ramène parfois à la question de **l'évaluation des usages**.

Ce point est fondamental puisqu'il ne sert à rien d'imaginer de nouvelles techniques d'adaptation ou de nouveaux modèles d'architecture supportant l'adaptation si nous ne sommes pas en mesure d'évaluer et certifier leur intérêt (démarche qualité). Celui-ci peut prendre différentes formes : amélioration de l'utilisabilité du système, augmentation de la capacité d'apprentissage de l'utilisateur, gain de productivité, etc.

L'enjeu est donc tel que nous avons choisi d'y consacrer le dernier chapitre de ce mémoire. Notons simplement que les résultats très positifs de ce travail sont exposés dans la section 4.2.3 et ont fait l'objet de nombreuses publications dans des revues et conférences internationales de très haut niveau.

Contexte de recherche :
- Thèse Halima Habieb-Mammar

Publications :
- **TARPIN-BERNARD F., HABIEB-MAMMAR H.,** Modeling Elementary Cognitive Abilities for Adaptive Presentation of Hypermedia, *User Modeling and User-Adapted Interaction (UMUAI)* The Journal of Personalization Research, Kluwer Academic Publishers, Volume 15, Issue 5, Nov 2005, pp. 459 - 495.
- **HABIEB-MAMMAR H., TARPIN-BERNARD F., PREVOT P.,** EDPHA : un Environnement de Développement et de Présentation d'Hyperdocuments Adaptatifs, *Revue d'Interaction Homme Machine*, Europia, Paris, vol. 6, 1, 2005, pp. 55-79.
- **HABIEB-MAMMAR H., TARPIN-BERNARD F.** CUMAPH: Cognitive User Modeling for Adaptive Presentation of Hyper-documents, *Adaptive Hypermedia and Adaptive Web-Based Systems*, Third International Conference, AH 2004, Eindhoven, The Netherlands, August 23-26, 2004, Proceedings, Springer Verlag in the Lecture Notes in Computer Science (LNCS 3137) series, P. de Bra, W. Nejdl (eds), ISBN: 3-540-22895-0, pp. 136-155.
- **HABIEB-MAMMAR H., TARPIN-BERNARD F., PREVOT P.** Adaptative presentation of multimedia interface, case study : "Brain Story" Course, *9th International Conference on User Modelling, UM 2003,* Johnstown, PA, USA, June 22-26, 2003, Proceedings, Springer Verlag in the Lecture Notes in Computer Science (LNCS 2702) series, Brusilovsky, P., Corbett, A., de Rosis, F. (Eds.), 2003, XIV, ISBN: 3-540-40381-7, pp. 15-24
- **HABIEB-MAMMAR H., TARPIN-BERNARD F., PREVOT P.** Modélisation XML des interfaces adaptatives intégrant le profil cognitif de l'utilisateur, *DVP'2002 Documents Virtuels Personnalisables,* Brest. p. 8. Juillet 2002.

Cependant avant d'aborder ce dernier chapitre, prenons le temps de tirer quelques conclusions de nos recherches portant sur des techniques spécifiques d'adaptation.

3.9 Conclusion sur les techniques spécifiques d'adaptation

Dans ce chapitre, nous venons de montrer qu'il était important, non seulement d'apporter des **réponses architecturales et méthodologiques** à la **problématique d'adaptativité** mais également qu'il convenait de continuer à mener des recherches plus spécifiques sur les techniques susceptibles d'être utilisées en bout de chaîne par le processus d'adaptation. Jusqu'à présent nos propositions tant académiques qu'industrielles ont essentiellement consisté à explorer le schéma suivant :

1) être capable de **qualifier le mieux possible les éléments adaptables** (indicateurs, signature, etc.) ;
2) **appliquer des processus de sélection multicritères** capables d'adopter un point de vue global pour éviter les problèmes ergonomiques pouvant naître de maximisations locales (pour l'instant nous avons privilégié des techniques de maximisation de grandeur numérique mais il serait sans doute intéressant de coupler notre approche aux approches plus classiques à bases de règles) ;
3) mettre au point des **techniques de mise en forme des éléments retenus**.

Dans les années à venir, nous allons chercher à intégrer le mieux possible nos travaux portant sur la modélisation cognitive de l'utilisateur aux approches AMF développées dans le chapitre précédent. Ceci passe sans doute par la définition de facettes *Utilisateur* porteuses du modèle de l'utilisateur mais également par la construction de **mécanismes génériques** de rendu de la présentation pour faciliter l'implémentation des classes applicatives.

Dans nos travaux sur un Environnement de Développement et de Présentation d'Hyperdocuments Adaptatifs (EDPHA), nous avons également cherché à prendre en compte la question de l'environnement Auteur. Ce point est un élément crucial sur lequel il conviendra de poursuivre nos réflexions. En effet, quand on connaît la difficulté (et donc le coût) de création d'un support de cours multimédia exploitant pleinement les possibilités des nouvelles technologies, on comprend qu'il est encore plus difficile pour un auteur de concevoir un cours adaptatif, non seulement sur les contenus mais également sur leur forme. Sur ce dernier point, notre approche sera étendue à la modélisation des connaissances pour intégrer les travaux d'un autre jeune docteur du laboratoire S. Benadi (2004). Ceci nous amènera sans doute à mieux étudier les travaux réalisés à ce jour dans le domaine du *knowledge management*.

Enfin, se pose la question de l'évaluation du résultat obtenu, à savoir le système adapté en temps réel. Le problème est si complexe que nous avons choisi d'y consacrer le dernier chapitre de ce mémoire.

4 Outils et méthodologie pour l'évaluation de systèmes adaptatifs

Après plusieurs années de recherche consistant à étudier les propositions existantes et à imaginer de nouvelles propositions, nous pouvons affirmer aujourd'hui qu'un des principaux verrous de la recherche sur les systèmes adaptatifs réside dans le manque d'outils et méthodes utilisables pour **évaluer la qualité des systèmes produits**.

A ce titre, trois questionnements nous semblent importants :

- Pour étudier et surtout comparer des propositions alternatives et identifier les lacunes de tel ou tel système, comment peut-on caractériser voire quantifier l'adaptabilité ou l'adaptativité d'un système ?
- Comment déterminer que le système adaptatif apporte un plus par rapport à un système uniforme ou un autre système adaptatif ? En effet, puisqu'un système adapté à l'utilisateur présente par essence un visage différent à chaque utilisateur, les techniques consistant à comparer des groupes d'utilisateur ne sont généralement pas pertinentes car le nombre de degrés de liberté rend l'analyse quasi impossible.
- L'impact d'un système adaptatif ne se ressent pas forcément sur des indicateurs de performances classiques (temps de réponse, quantité d'information mémorisée, etc.) mais peut avoir des effets psychosociaux plus subtils. Comment mettre en place des protocoles dédiés mesurant de tels effets.

Dans ce chapitre, nous allons présenter les travaux que nous avons réalisés jusqu'alors dans ces trois directions.

4.1 Caractérisation et quantification de l'adaptabilité et de l'adaptativité

Pour mesurer les progrès réalisés par telle ou telle proposition, il nous est apparu clairement que nous ne disposions d'aucun moyen de caractériser et encore moins quantifier l'adaptabilité ou l'adaptativité d'un système donné. Etant donné le grand nombre de degrés de liberté de l'adaptation, il semblait impossible de comparer deux systèmes.

Pour avancer dans ce sens, nous avons élaboré une première grille d'analyse considérant le niveau d'adaptabilité ou d'adaptativité des aspects évoqués dans la section 1.2 en fonction du contexte d'utilisation (Tarpin-Bernard *et al.*, 2007). Rappelons à ce stade que nous avons identifié 4 aspects sur lequel peut porter l'adaptation (la présentation, les données ou informations, les services et l'assistance à l'utilisateur) ainsi que 4 facteurs eux-mêmes subdivisés en sous facteurs : l'utilisateur (ses connaissances, ses préférences, son expérience, ses habiletés cognitives et ses habiletés perceptives et motrices), la plate-forme d'interaction (la puissance de traitement, les périphériques d'entrée, les périphériques de sortie, la connectivité, le système d'exploitation et les logiciels disponibles), l'environnement (les informations ambiantes, la localisation, l'aspect mono/multi utilisateurs, la structure organisationnelle, les politiques de sécurité et de travail) et les caractéristiques de l'activité (les caractéristiques des tâches, leur structure, les relations avec le workflow d'entreprise et les buts courants de l'utilisateur).

Cette grille (Cf. Tableau 7) peut bien sûr être enrichie même si son utilisabilité repose sans doute sur sa simplicité.

		Aspects d'adaptation			
Facteur	Facteur Elémentaire	Présentation P	Données D	Services S	Assistance A
Utilisateur U	Connaissances du domaine				
	Préférences				
	Expérience				
	Habiletés Cognitives				
	Habiletés Perceptives / Motrices				
Plate-forme d'interaction Pl	Puissance de traitement				
	Périphériques d'entrée				
	Périphériques de sortie				
	Connectivité				
	Systèmes d'exploitation				
	Logiciels installés				
Environnement E	Informations ambiantes				
	Localisation				
	Mono/multi utilisateurs				
	Structure organisationnelle				
	Politiques de sécurité et de travail				
Activité Ac	Caractéristiques des tâches				
	Structure des tâches				
	Relation avec le workflow Ets				
	Buts et intentions				

Tableau 7 : Grille de caractérisation de l'adaptabilité d'un système – v 1.0

A partir de cette grille, il nous a semblé possible de définir le degré d'adaptabilité ou d'adaptativité d'un système grâce à une métrique utilisable pour comparer plusieurs systèmes.

Nous avons ainsi proposé les formules équivalentes suivantes :

$GA(x) = F\ (LA_{a/f}(x))$ avec $a \in \{P, D, S, A\}$ et $f \in \{U, Pl, E, Ac\}$

$GA(x) = F_f\ (FA_U(x), FA_{Pl}(x), FA_E(x), FA_{Ac}(x))$

$GA(x) = F_a\ (AA_P(x), AA_D(x), AA_S(x), AA_A(x))$

où

$GA(x)$ est le degré Global d'Adaptabilité d'un système interactif x

$LA_{a/f}(x)$ est le degré Local d'Adaptabilité de l'aspect a selon le facteur f.

$FA_f(x)$ sont des degrés semi-globaux caractérisant le degré d'adaptabilité selon le facteur (U, Pl, E et Ac représentent les 4 facteurs discutés précédemment).

$AA_a(x)$ sont des degrés semi-globaux caractérisant le degré d'adaptabilité de chaque aspect (P, D, S et A représentent les 4 aspects discutés précédemment).

$F(\ )$, $F_a(\ )$ et $F_f(\ )$ sont des fonctions pour calculer le degré Global d'Adaptabilité à partir respectivement des degrés locaux, semi-globaux par aspect ou par facteur.

L'ensemble de ces degrés d'adaptabilité (globaux, semi-globaux et locaux) permet de caractériser le niveau d'adaptation proposé par le système. Ainsi, si un système est capable d'adapter sa présentation au profil d'un utilisateur, le degré local $LA_{P/U}$ sera non nul. Les degrés semi-globaux comme AA_P et FA_U décriront respectivement à quel point la Présentation est adaptable à tous les facteurs et à quel point le facteur Utilisateur est pris en considération par les différents aspects de l'application. Le degré global d'adaptabilité GA synthétise tous les degrés intermédiaires possibles.

Parce que l'adoption et la validation d'une telle métrique, et surtout l'affinement des caractéristiques considérées et des fonctions retenues, requièrent du temps et de la collaboration, nous avons proposé dans cette version 1.0 d'utiliser de simples fonctions moyennes donnant ainsi un poids équivalent à chaque élément. Pour évaluer chaque case de la grille, nous avons choisi d'utiliser une échelle de 0 à 3 (Cf. Tableau 8). Le degré Local d'Adaptabilité de l'aspect a selon le facteur f est calculé en divisant la moyenne des scores des cases correspondantes par 3 (la valeur maximale de l'échelle). Un degré semi-global (AA ou AF) est la moyenne des degrés locaux de l'aspect ou du facteur correspondant. Le degré Global d'Adaptabilité est la moyenne des degrés semi-globaux (AA ou AF). Selon que l'on porte un intérêt à l'adaptation par le système ou par l'utilisateur, la même technique de calcul permet de déterminer des degrés d'adaptabilité ou d'adaptativité.

Adaptation de l'aspect A selon le Facteur élémentaire X
0: A ne tient pas compte de X
1: A est légèrement affecté par X
2: A est affecté par X
3: A est grandement affecté par X

Tableau 8 : Echelle de calcul des degrés élémentaires d'adaptabilité

Nous avons ainsi été en mesure de comparer 3 systèmes connus dédiés à l'adaptation à l'utilisateur : AHA ! (De Bra *et al.*, 2003), NetCoach (Weber *et al.*, 2001) et notre propre système EDPHA (Tarpin-Bernard *et al.*, 2005). Le Tableau 9 présente les différents degrés d'adaptation obtenus.

Degrés locaux d'adaptativité en %	Présentation	Données	Services	Assistance	Degrés Semi-globaux (moyenne)
Utilisateur	40/33/33	13/27/0	0/7/0	7/13/0	**15/20/8**
Plate-forme	0/0/0	0/0/0	0/0/0	0/0/0	0/0/0
Environnement	0/0/0	0/0/0	0/0/0	0/0/0	0/0/0
Activité	27/13/0	19/27/0	0/0/0	7/13/0	13/13/0
Degrés semi-globaux (moyenne)	**17/11/8**	**8/14/0**	**0/2/0**	**3/6/0**	**Degré Global 7/8/2**

Tableau 9 : Les différents degrés d'adaptativité de 3 systèmes (AHA!/NetCoach/EDPHA)

Nous ne discuterons pas ici les résultats obtenus par les trois systèmes si ce n'est pour souligner qu'il reste encore beaucoup de travail à réaliser pour obtenir des degrés globaux d'adaptativité significatifs. Notons simplement que le faible score d'EDPHA s'explique naturellement par la spécificité de l'adaptativité que nous avons mise en œuvre (adaptation de la présentation au profil cognitif). En revanche, les techniques introduites par EDPHA étant extrêmement complémentaires de celles proposées par les deux autres systèmes, leur intégration augmenterait le degré d'adaptativité à l'utilisateur.

Nous estimons notre outil de caractérisation particulièrement intéressant, car il présente les avantages suivants :

- Il est simple d'utilisation et peut même être utilisé partiellement si l'on ne s'intéresse qu'à certains degrés locaux d'adaptativité.
- Il permet de comparer des systèmes adaptatifs. La subjectivité de l'évaluateur est maîtrisée même s'il est clair que des recommandations détaillées et des exemples seront utiles.
- Il est extensible. A partir de cette version 1.0, d'autres chercheurs peuvent ajouter des éléments ou en raffiner certains dans la mesure où ils prennent garde de ne pas déséquilibrer la matrice.
- La qualification d'un degré élémentaire pourrait être raffinée en étendant l'échelle (passer à 0-5 voire 0-10), ceci sans remettre en cause le principe général. Une expertise pourrait ainsi être développée.

Il est cependant important de noter que cet outil sert à évaluer l'adaptabilité ou l'adaptativité d'un système du point de vue de ses potentialités. En aucun cas, il n'aide à caractériser l'efficacité ou la pertinence des adaptations proposées. Celles-ci ne peuvent être évaluées sans mener des expérimentations de terrain qui requièrent de nombreux utilisateurs placés dans des contextes variés, d'où une explosion combinatoire difficile à gérer (Chin, 2001). C'est ce que nous allons voir dans la section suivante.

Contexte de recherche :
- Collaboration avec Concordia University, Canada

Publications :
- HABIEB-MAMMAR H., TARPIN-BERNARD F., SEFFAH A., Towards UI Characterizing and Adaptability Quantifying, *Interacting with Computers*, 24p, soumis 07/2006. 2° révision en cours.

4.2 Méthodologie d'évaluation de la pertinence d'une adaptation

David Chin (2001) a très bien montré l'impérieuse nécessité d'évaluer empiriquement les systèmes adaptatifs tout en mettant en évidence la difficulté de la tâche. Il a ainsi souligné le fait, d'une part que les analyses statistiques requièrent de grands groupes de sujets, d'autre part qu'il était souvent difficile de construire une situation de contrôle (c'est-à-dire une situation de référence pour mesurer les effets de l'adaptation). En effet, l'approche la plus communément utilisée consiste à comparer le système étudié à une version sans adaptativité, approche utilisée pour les évaluations de systèmes comme MetaDoc (Boyle et Encarnacion, 1994) et PUSH (Höök, 1997). Bien que ces évaluations prétendent que les versions adaptatives de ces systèmes ont permis d'augmenter les performances des

utilisateurs dans la réalisation de plusieurs tâches, la qualité des versions de contrôle non adaptées est assez discutable.

D'autres méthodes consistent à classer la population en deux groupes opposés A & B (par exemple des profils visuels vs. des profils verbaux) et à expérimenter le système adapté avec des profils forcés (S_A, S_B) dans des situations contrebalancées (A-S_A, A-S_B, B-S_A, B-S_B). Cette méthode peut être très pertinente s'il est réellement possible de découper la population de sujets dans deux groupes opposés. Dans le cas qui nous a intéressé avec EDPHA, cette technique s'avère impraticable puisque les profils cognitifs sont a priori très variés et hétérogènes.

Pour démontrer les effets positifs de l'adaptation au profil cognitif, il nous fallait donc imaginer un protocole innovant. L'hypothèse que nous avons formulée peut se décrire ainsi : si nous sommes capables de montré qu'une adaptation basée sur un profil cognitif virtuel proche du profil réel de l'utilisateur apporte de meilleurs résultats en terme de performance qu'une adaptation basée sur un profil cognitif virtuel éloigné du profil réel de l'utilisateur, alors cela signifie que les bénéfices ainsi mis en évidence tenait bien au processus d'adaptation lui-même et non par exemple à la qualité des médias utilisés pour tel ou tel profil.

4.2.1 Description du plan d'expérimentation

En expérimentant le modèle EDPHA, nous souhaitions **évaluer l'impact des interfaces adaptatives sur les acquis et les ressentis de l'utilisateur**. Cette expérimentation rentre dans le cadre d'une approche de développement incrémental, où les modèles soumis à la rude confrontation de l'usage sont régulièrement validés et améliorés. Ici, nous nous sommes placés dans une approche empirique fondée sur un diagnostic de l'usage à partir de recueils de données (Senach, 1990).

L'expérimentation s'est déroulée auprès d'une population de 42 élèves-ingénieurs en 3ème année du département Génie Industriel de l'INSA de Lyon, dont trois sujets ont été éliminés a posteriori pour une maîtrise insuffisante du français. Dans de futures expérimentations, nous impliquerons des sujets provenant d'horizons plus diversifiés, ce qui conduira sans doute à l'obtention de profils cognitifs plus variés.

Le protocole suivi s'articule ainsi :

- chaque sujet passe le module d'évaluation cognitive (6 exercices interactifs, Cf. section 3.3.2) ce qui construit son profil cognitif ;

- pour chaque sujet un profil cognitif virtuel aléatoire est généré (en affectant des valeurs aléatoires comprises entre 10 et 90 pour chaque indicateur) – nous conseillons de vérifier l'homogénéité de répartition des profils virtuels / réels ;

- les sujets disposent alors de 20 minutes pour naviguer dans un cours web adapté selon le profil aléatoire (et non le profil réel de l'utilisateur). Ce cours contient une vingtaine de pages et expose des informations sur le fonctionnement et les processus cognitifs sujet dans lequel les étudiants ingénieurs n'ont pas de connaissances préalables (un mini questionnaire a permis de valider cette hypothèse). ;

- les étudiants sont ensuite évalués à l'aide d'un questionnaire final comportant des questions ouvertes. L'évaluation donnée par le correcteur humain est une note sur 100 points (notée plus loin *Mark*). Enfin, une question plus subjective demande aux sujets d'attribuer une note (notée plus loin *Feeling*) au document pédagogique qu'ils viennent de manipuler.

4.2.2 Caractérisation de la population étudiée

Etant donné les critères d'adaptation retenus pour l'expérimentation, nous avons synthétisé les 14 indicateurs cognitifs SBT en 3 scores : visuel, verbal et auditif. À l'issue de cette première étape d'expérimentation, nous avons constaté que les scores visuels et verbaux du test HAPPYneuron™ sont fortement corrélés (coefficient de corrélation=0.70, p=0.0005). Ce résultat nous montre que les deux composantes visuelle et verbale ne sont pas indépendantes. En effet, si on regarde une image, on trouve quasiment toujours un moyen de la verbaliser (la décrire). Réciproquement, lorsqu'on lit du texte, on utilise la modalité visuelle. La Figure 32 présente la répartition des sujets testés en fonction des scores visuel et auditif (ici aucune corrélation n'est visible). Comme nous l'avons indiqué précédemment, les scores varient de 0 à 100 et correspondent à des percentiles de population.

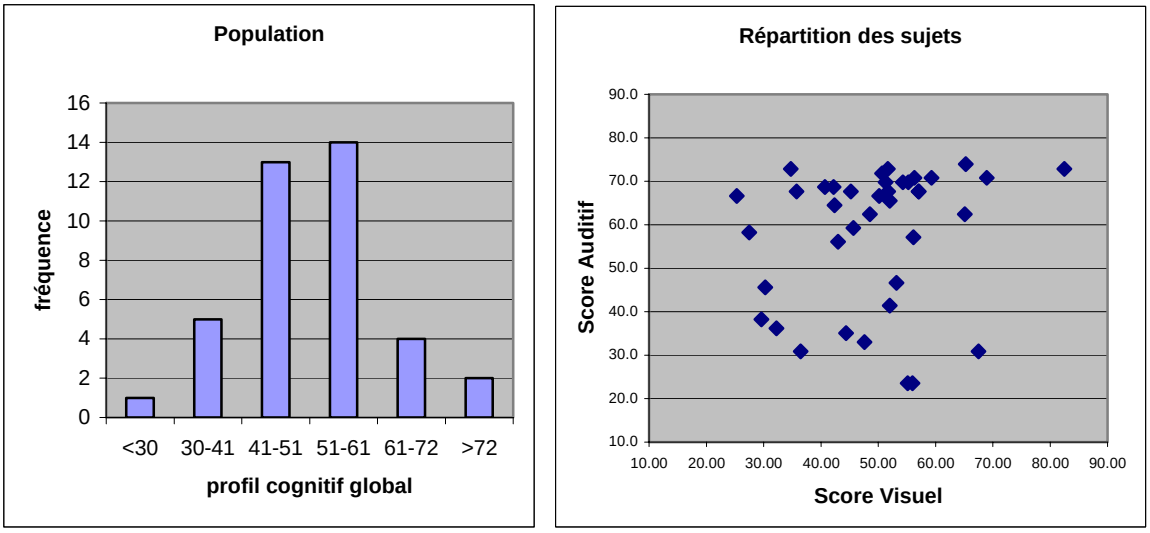

Figure 32: Répartition des sujets de l'expérimentation en fonction de leur profil

Nous avons également calculé, pour chaque individu, un nouveau score dit "Profil" qui représente la moyenne des trois scores cognitifs (visuel, verbal et auditif).

La moyenne des scores de "Profil" (M=51,2) et l'écart type (S=10.5) amènent à subdiviser la population en 6 nouvelles classes : [<M-2S] [M-2S ; M-S] [M-S ; M] [M; M+S] [M+S ; M+2S] [>M+2S]. Pour la suite de l'analyse, nous avons attribué à chaque individu un code "cProfil" dont la valeur appartient à l'intervalle [1..6] et représente la position de l'individu dans les 6 classes précédemment décrites. Cette valeur représente le potentiel cognitif de chaque individu.

À la suite de ces tests, les étudiants passent à la deuxième étape de l'expérimentation pour l'exploration de l'hyperdocument. Après leur avoir laissé 20 minutes pour explorer un hyperdocument adapté à des profils aléatoires, nous les avons interrogés sur le contenu du

cours. Le questionnaire comportait 10 questions ouvertes. Une note finale sur 20 a été attribuée en fonction de la pertinence des réponses. Les résultats de l'expérimentation sont décrits dans la section suivante.

4.2.3 Analyse des résultats

La première étape consiste à subdiviser la population en tenant compte de la "distance d'adaptation". Cette distance est calculée en utilisant la distance euclidienne entre le profil réel (Vis, Ver et Aud) et le profil aléatoire (VisA, VerA et AudA) :

$$distance = \sqrt{(Vis - VisA)^2 + (Ver - VerA)^2 + (Aud - AudA)^2}$$

Les individus sont alors répartis dans 2 groupes de 19 et 20 membres. Chacun de ces 2 groupes se caractérise respectivement par des distances inférieures ou supérieures à la distance moyenne (44,2). On note cDistance=1 et 2 ces deux groupes.

La note obtenue à l'évaluation finale est très corrélée à cDistance classes (cf. Figure 33) :

Correlation Mark / cDistance : -0.45 avec P-value=0.0033

Analyse ANOVA Mark / cDistance: F-Ratio=9.89, P-Value=0.0033

En outre, la corrélation entre Mark et Distance est statistiquement d'autant plus significative que l'on s'intéresse aux personnes présentant les plus faibles profils cognitifs (Tableau 10). L'hypothèse que nous formulons pour expliquer ce phénomène est que plus les personnes sont brillantes et plus elles sont capables de surmonter une inadéquation du cours à leur profil. En revanche, les personnes présentant des profils plus faibles sont plus sensibles à une mauvaise adaptation.

Population	complète	Sans les 3 plus hauts profils	Les 23 profils les plus faibles	Les 15 profils les plus faibles
Nombre de sujets	39	35	23	15
Corrélation Mark / Distance	-0.18	-0.31	-0.40	-0.54
P-Value	0.27	0.06	0.05	0.03

Tableau 10 : Corrélation entre la note obtenue au test et la distance d'adaptation

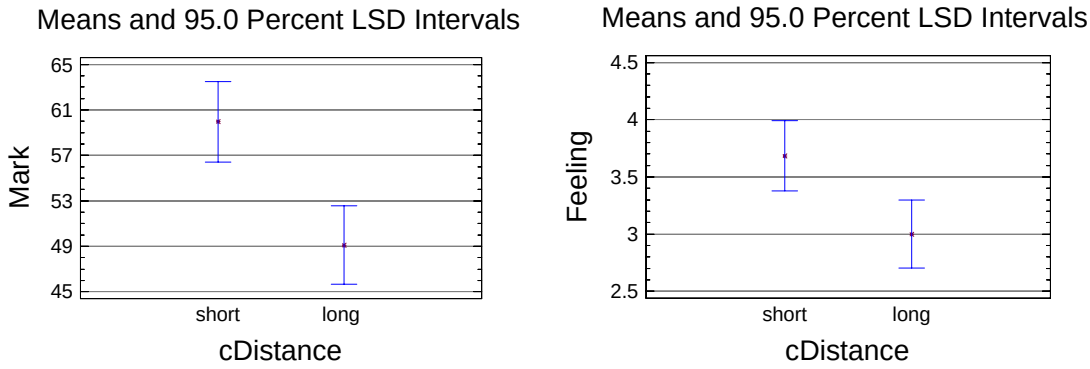

Figure 33 : Différence statistiquement significative des notes et de l'impression subjective selon la distance d'adaptation

La Figure 33 illustre également bien le fait que la qualité de l'adaptation est naturellement perçue par l'utilisateur puisque l'évaluation du cours (Feeling) diffère significativement selon la Distance. Le lecteur intéressé par plus de détail sur l'analyse se reportera à (Tarpin-Bernard et Habieb-Mammar, 2005).

Les résultats issus de cette expérimentation ont montré que les utilisateurs appartenant à la classe des distances faibles ont obtenu de meilleurs résultats au questionnaire final que ceux qui appartiennent à la classe des grandes distances. **Ce n'est donc pas la qualité intrinsèque des médias qui est à l'origine des performances mais bien le fait qu'ils soient adaptés aux profils cognitifs des utilisateurs**.

Les résultats de cette expérimentation nous paraissent très intéressants et motivants pour envisager de nouveaux développements. Ainsi, nous envisageons d'étendre notre modèle utilisateur en y ajoutant d'autres indicateurs représentant d'autres facettes non prises en compte comme les aspects comportementaux, culturels ou d'ordre émotionnel. La structure du moteur que nous avons développé permet de les intégrer facilement. Nous pensons qu'il nous est également possible d'aller plus loin dans l'adaptation de la navigation. Par ailleurs, nous souhaitons mettre en œuvre certains aspects de l'adaptation à la plate-forme notamment pour la prise en compte des modalités auditives.

Le protocole d'évaluation nous semble très intéressant pour valider « en aveugle » l'intérêt d'une adaptation, c'est-à-dire sans chercher à définir a priori des groupes de situations (ex : A, B, contrôle…) dont on sait qu'elles sont extrêmement difficiles à construire et souvent génératrices de biais pour l'analyse. Nous sommes convaincus qu'il peut être transposé à de nombreuses autres situations. Il serait également intéressant d'étudier dans quelle mesure d'autres distances que la distance Euclienne pourraient rendre l'approche encore plus pertinente (la question de la distance entre un profil réél et un profil virtuel rejoint ici la question de distance entre deux contextes d'utilisation)...

Contexte de recherche :
- Thèse Halima Habieb-Mammar

Publications :
- TARPIN-BERNARD F., HABIEB-MAMMAR H., Modeling Elementary Cognitive Abilities for Adaptive Presentation of Hypermedia, *User Modeling and User-Adapted Interaction (UMUAI)* The Journal of Personalization Research, Kluwer Academic Publishers, Volume 15, Issue 5, Nov 2005, pp. 459 - 495.

4.3 Evaluation des effets psychosociaux de systèmes interactifs adaptatifs

Les systèmes de e-training cognitif présentés précédemment peuvent être utilisés dans des contextes de **handicap**, qu'il s'agisse d'un handicap mental innée ou d'un handicap inhérent au très grand âge ou à des maladies neuro-dégénératives comme Alzheimer.

Ceci nous a conduit à bâtir un projet dans le cadre de l'appel "Usages des nouvelles technologies pour la société" lancé par le Ministère délégué à la Recherche et aux Nouvelles Technologies en 2003, intitulé MNESIS et destiné à étudier les usages de l'outil Internet comme moyen d'intégration sociale et de stimulation cognitive dans une résidence de personnes âgées

Le but du projet était d'analyser les effets d'un programme de stimulation cognitive sur Internet sur le déclin cognitif naturel lié au vieillissement de personnes placées en résidences médicalisées et, dans le même temps, d'observer les effets de l'usage du support technologique en termes d'intégration dans l'environnement social, familial et médical du résident usager. Après avoir conçu un logiciel adapté sur le plan ergonomique, nous avons mené une expérimentation de 6 mois dans 9 résidences auprès de 45 personnes âgées en moyenne de 84 ans.

4.3.1 Contexte

En vieillissant, les Personnes Agées (PA) accumulent les "handicaps" : sociaux, physiques, psychologiques, cognitifs et numériques (Gorgeon et Léridon, 2001, Plonton, 2003). Il s'agit d'abord d'un déclin cognitif (avec une réduction des possibilités d'adaptation, des désapprentissages, de la démotivation, des difficultés de mémorisation…) et de dégradations psychologiques importantes (marquées par une plus grande vulnérabilité psychologique, l'absence de nouveaux investissements, une atteinte de l'estime de soi, la dépression ….). Les pertes physiques sont significatives, symbolisées notamment par une plus grande préoccupation sur la santé, des pathologies fonctionnelles importantes et une perte de la dextérité physique et de la coordination sensori-motrice. L'effritement de l'identité et du lien social est également spécifique de cette génération (Meire, 1992, David & Starzec, 1996). Avec le grand âge, on observe ainsi un repli de la personne sur le domicile et un affaiblissement significatif de ses rôles sociaux et familiaux, une vie par procuration, des conduites régressives (alimentation, hygiène, usages sociaux), une perte de but et d'identité conduisant à un état d'anomie (Atchley, 1980). Ce désengagement social s'exprime notamment par la diminution du niveau d'interaction sociale tant par la fréquentation que par le degré d'implication. Ainsi 11 % des octogénaires vivent totalement isolés, c'est-à-dire qu'ils n'ont ni sorties, ni relations, ni contacts téléphoniques avec des tiers (famille, amis…) (David & Starzec, 1996).

L'arrivée dans une maison de retraite est souvent particulièrement mal vécue sur le plan personnel et social et provoque, du fait de cette immersion brutale, une forme de désorientation plus ou moins longue. Cette désorientation peut se caractériser selon trois plans : kinesthésique, personnel et social et conduit à plusieurs formes de déclins. Le déclin **kinesthésique** est caractérisé par une absence de déplacement de la personne en dehors d'une zone géographique socialement significative ; cette zone pouvant être une chambre ou une maison. Les raisons de ce déclin sont principalement liées à des défaillances motrices ou de perception, dues à l'âge ou à un handicap. Le déclin **psychologique et personnel** est caractérisé par un repli sur soi. Dans le cas d'une PA, la personne est refermée sur elle-même et le cas échéant souvent centrée sur sa maladie. Le déclin **social** est caractérisé par une absence de relations sociales qui sont supposées être a priori facilitées par la proximité physique (personnes extérieures ou intérieures à la résidence) ou personnelle (personnel encadrant/famille). Enfin, un dernier déclin, **cognitif**, affecte les facultés intellectuelles des PA en ne leur permettant pas d'appréhender les situations et donc de développer les conduites adaptées.

Les technologies sont alors éligibles comme moyen pour compenser ou pallier ces déclins. Notre propos est de voir, dans le cas de PA immergées dans ce milieu communautaire qu'est la maison de retraite, si les technologies ont effectivement une incidence réelle et si elles ont un effet sur l'adaptation à ce lieu. Nous entendons par « technologie », un logiciel

spécifique (Activital™) d'information et de communication et de stimulation cognitive par le jeu, conçu pour l'occasion. Ce logiciel repose sur des principes simples :

- Un environnement ergonomique adapté (lisibilité, interactivité avec écran tactile).
- 3 activités variées (Figure 34) :
  - un ensemble de jeux cognitifs de difficulté variable ;
  - un outil de création de journal de résidence pour développer la créativité ;
  - un outil de messagerie électronique simplifié pour favoriser les liens sociaux et la communication.

Figure 34 : Extraits des 3 activités d'Activital : les jeux cognitifs, l'éditeur de journal, et la messagerie

4.3.2 Expérimentation

Le projet s'est déroulé sur 22 mois (septembre 2003 – juin 2006) en trois phases :

- Préparation du logiciel et sélection des résidents sur dossier médical (pas d'antécédents neurologiques, psychiatriques ou organiques majeurs, ni pathologies neurologiques, psychiatriques ou organiques évolutives). A l'issue, un pré-test neuropsychologique a été fait pour chaque résident. Il donne un état avant stimulation. Une série d'entretiens et de questionnaires socio comportementaux ont également été menés auprès des sujets et de leur entourage (médical et familial) ainsi que des observations effectuées sur les pratiques et habitudes de vie des résidents. Le matériel informatique a également été installé et configuré dans les établissements. Le personnel d'animation a été formé aux outils.
- L'expérimentation s'est faite en répartissant les résidents en 3 groupes :
  - groupe 1 : entraînement Internet supervisé au moyen d'exercices spécifiquement développés par SBT, et usage d'un outil de messagerie électronique simplifié
  - groupe 2 : utilisation d'un outil informatisé de rédaction du journal de la résidence et usage d'un outil de messagerie électronique simplifié,
  - groupe 3 : entraînement au moyen de fiches papiers destinées à l'animation d'ateliers mémoire et activité traditionnelle de rédaction d'une correspondance papier.

  Il était ainsi possible :
  - d'évaluer l'importance de l'effet Internet en comparant le groupe 3 aux groupes 1 et 2,

- d'évaluer l'importance et la spécificité de l'entraînement supervisé en comparant le groupe 1 au groupe 2.

La stimulation a alors pu commencer. Elle a consisté pour chaque résident en un entraînement de plusieurs (2 à 3) séances hebdomadaires de trente minutes.

Pendant toute la durée de la stimulation sur ordinateur, un système de capture de traces enregistrait pour chaque résident un grand nombre d'actions élémentaires (clic bouton, frappe clavier, messages envoyés et reçus, etc.).

- A l'issue des 6 mois de stimulation, un post-test équivalent au pré-test a été réalisé (tests neuropsychologiques, entretien et suivi de l'activité).

4.3.3 Conclusion du projet MNESIS

Le bilan général du projet MNESIS s'est avéré globalement très positif. Malgré les nombreux problèmes pratiques rencontrés, principalement dus à la sollicitation importante des personnels des résidences et la difficulté à identifier des sujets remplissant toutes les exigences scientifiques que nous nous étions données, nous avons pu obtenir des résultats très intéressants.

Si nous ne devions retenir que quelques points clés, nous mettrions en avant que :

- l'âge avancé (moyenne 84 ans) ne constitue pas un obstacle infranchissable pour l'accès aux nouvelles technologies dans la mesure où celles-ci ont été adaptées sur le plan ergonomique (dans notre cas écran tactile, gros boutons explicites, etc.)
- contrairement aux préjugés souvent ancrés dans les esprits, les personnes âgées sont intéressées par l'ordinateur qui est un objet étrange qui suscite leur curiosité. Passé le stade de la crainte de ne pas y arriver, l'usage de l'ordinateur entraîne une revalorisation de la personne âgée et de l'estime qu'elle a d'elle-même.
- la technologie permet une meilleure intégration de la personne âgée sur le plan physique en la stimulant à participer à des activités manuelles par exemple et sur le plan personnel en lui procurant des objectifs positif et constructifs.
- le logiciel élaboré dans le projet induit une stimulation cognitive directe (les jeux) mais également une stimulation indirecte (surtout la rédaction du journal ou la préparation des mails).
- c'est sur le plan social, avec un rôle prégnant du dispositif comme artéfact symbolique, que l'effet est le plus marquant. On observe en effet un développement net du lien social et même l'apparition de nouvelles formes de construits sociaux basés sur la collaboration.
- sur le plan strictement cognitif, le bilan est plus difficile à tirer car comme nous l'avons indiqué en préambule, les conditions d'expérimentation ont rendu difficile l'interprétation des résultats. Rappelons que les participants à cette étude avaient un âge moyen de 84 ans et que l'intervalle de temps séparant le pré-test et le post-test était de 6 mois. Dans ces conditions, il n'aurait pas été surprenant d'observer une chute des performances dans les tests cognitifs les plus sensibles au vieillissement. Or la plupart des participants présente des performances stables dans ces tests, voire même en augmentation, à l'issue de l'entraînement cognitif. On peut ainsi supposer que les participants à cette étude ont bénéficié de la stimulation cognitive proposée (que ce soit la stimulation cognitive avec des tâches informatisées, la simple utilisation de l'ordinateur, ou encore l'entraînement de type « papier-crayon » spécialement conçu à cet effet). Nos résultats révèlent tout de même un léger avantage, en termes de maintien

du niveau de performances cognitives, en faveur des participants des groupes stimulés via l'informatique (groupes 1 et 2).

Enfin, ce projet nous a permis de mieux appréhender les contraintes inhérentes à ce public très particulier des personnes âgées dépendantes en milieu médicalisé. Cette expérience acquise nous sera très utile pour élaborer de nouveaux projets afin d'étudier notamment de manière plus fine les effets cognitifs de telle ou telle activité.

Dernier point et non des moindres, les outils informatiques de recueil et d'analyse de données d'usage (traces) mis au point par mes collègues à l'occasion de ce projet vont pouvoir être réutilisés dans des contextes très différents (Michel *et al.*, 2005). Ces outils consistent à embarquer dans les logiciels étudiés des capteurs qui tracent les actions brutes de l'utilisateur (mouvements souris, frappes clavier, actions sur boutons, etc.) puis à élaborer des traces primitives d'utilisation à partir d'une ontologie d'usage défini au préalable. Des outils de requêtage permettent à l'analyste d'identifier des motifs d'usage récurrents et de définir des profils types. Une des principales originalités de l'approche est de croiser ces traces d'usage avec les informations psychosociales issues des entretiens et questionnaires. Je suis personnellement convaincu que ces travaux s'avèreront déterminant dans l'amélioration des techniques d'analyse des usages de logiciels applicatifs. Comme nous l'avons montré précédemment, ceci s'avèrera d'autant plus important que cette analyse est encore plus difficile pour l'observateur humain lorsque le logiciel est adaptatif et que les expériences des utilisateurs sont donc très différentes.

Contexte de recherche :
- projet MNESIS

Publications :
- MICHEL C., BOBILLIER-CHAUMON M.-E., COHEN-MONTANDREAU V., TARPIN-BERNARD F., (2006) Les personnes âgées en EHPAD. Les TIC sont-elles un mode de reliance sociale ? In Conférence EUTIC 2006 "ENJEUX ET USAGES DES TIC : Reliance sociale et insertion professionnelle", Bruxelles, 13-15 septembre 2006
- MICHEL C., BOBILLIER-CHAUMON M-E., MONTANDREAU V., TARPIN-BERNARD F., Démarche d'évaluation de l'usage et des répercussions psychosociales d'un environnement STIC sur une population de personnes âgées en résidence médicalisée, *17ème conférence francophone sur l'Interaction Homme-Machine (IHM'05)*, ACM International Conference Proceedings Series, ACM Press, New York, 2005, ISBN 1-59593-192-9, pp. 195-198
- MICHEL, CH., BOBILLIER-CHAUMON, M.-E., COHEN-MONTANDEAU V., TARPIN-BERNARD F., Immersion de la personne âgée en maison de retraite : étude des incidences possibles des TIC dans sa (re)construction psychosociale, *LUDOVIA 2006*, Saint-Lizier, France, 5-7 juillet 2006, 15 pages
- TARPIN-BERNARD F., NOIR M., CROISILE B., MICHEL CH., BOBILLIER-CHAUMON M-E., MONTANDREAU-COHEN V., KOENIG O., COLLIOT P., OJEDA N., FILLON V., OUDART S., FAVRE B., Rapport final du projet MNESIS, 30/09/2006, SBT, 56 pages

5 Programme de recherche

Mon objectif à cinq ans est de fédérer une équipe de recherche autour de la thématique de l'**adaptation des interfaces homme-machine**. La puissance de calcul des dispositifs d'interaction accessibles aux utilisateurs et leur connectivité rend aujourd'hui possible d'un point de vue purement technique la mise en œuvre de nombreuses adaptations qu'elles soient fonctionnelles ou interactionnelles. Nous avons montré qu'une simple analyse QQOQCP (Qui, Quoi, Où, Quand, Comment, Pourquoi) aide à cerner l'étendue des problèmes à résoudre. Mes travaux actuels n'ont apporté que quelques pierres à l'édifice et de très nombreux verrous scientifiques restent encore à lever.

Les premiers verrous portent sur la modélisation et les méthodologies de conception et développement de logiciels adaptatifs. Pour limiter les coûts de développement de versions *ad hoc* d'un logiciel, il est indispensable de modéliser, à des niveaux d'abstraction suffisants, les programmes et en particulier leurs interfaces homme-machine. S'il est relativement simple de définir un objet générique de type « bouton » qui prend telle ou telle forme selon le support d'exécution, on comprend que cela devient plus complexe lorsqu'on souhaite pouvoir aussi utiliser un bouton physique du support, voire un autre mode d'interaction comme une commande vocale. Si la communauté scientifique est en pleine effervescence sur ce sujet, les réponses apportées restent encore trop théoriques et s'appliquent généralement à des sous-problèmes, par exemple les applications basées sur des formulaires. Au-delà de notre approche actuelle centrée sur AMF, je souhaite pouvoir étudier le rapprochement de nos modèles avec les modèles d'UML notamment, et au delà, nos méthodologies avec les approches MDA. En outre, toute méthode de génie logiciel doit s'accompagner d'outils de mise en œuvre. Dans le cas particulier de l'adaptation, ceci pose de très nombreuses questions. **Comment aider un concepteur ou un développeur à bâtir un logiciel polymorphe susceptible d'évoluer au gré des contextes d'utilisation tout en respectant les contraintes économiques ?** Si des outils d'émulation et de simulation peuvent apporter des réponses, il reste que la qualité du logiciel construit, notamment en terme d'utilisabilité est beaucoup plus difficile à assurer. L'intégration de technologies d'adaptation au profil des utilisateurs, et en particulier à leur profil cognitif est également problématique, tant on sait qu'une mauvaise adaptation peut rapidement conduire les utilisateurs à abandonner un logiciel.

Ceci amène à une deuxième famille de verrous à lever liée aux **usages** de ces approches. Pour faciliter le travail du concepteur, les chercheurs tentent d'automatiser le plus possible les mécaniques d'adaptation alors qu'il a été démontré depuis longtemps que dans tout système interactif, certaines fonctionnalités étaient traitées beaucoup plus efficacement par l'utilisateur que par le système. Parce que la résolution automatique d'un tel système de contraintes peut produire des interfaces de faible qualité sur le plan ergonomique se pose la question du **contrôle que peut avoir le concepteur et l'utilisateur final sur l'adaptation**. Pour résoudre ce type de questions, il est indispensable de jeter un regard pluridisciplinaire. C'est grâce à cette approche que nous avons par exemple pu prendre en compte des indicateurs cognitifs dans l'adaptation d'hypermédias pédagogiques. *A contrario*, en travaillant avec des collègues psychologues et sociologues, nous avons également montré qu'il n'était pas toujours pertinent de chercher l'adaptation au dépend de l'uniformisation (par exemple dans des contextes de collaboration où les acteurs interagissent avec des versions adaptées différemment d'un même logiciel).

Dans les années qui viennent, je souhaite pouvoir étendre nos travaux actuels à la prise en compte de la capacité des applications modernes à **migrer dynamiquement d'un support technologique à l'autre** (PC⇔PDA par exemple) ou à évoluer dynamiquement lors d'un changement de contexte (environnemental ou lié à l'activité du/des utilisateurs). L'intégration de la problématique de la **collaboration entre usagers de technologies adaptatives** est un autre axe de recherche que je souhaite pouvoir développer. Enfin, dans la continuité du projet ADELA auquel je participe (étude de l'accessibilité de la *e-administration*), il me semble important que des travaux soient menés pour une meilleure prise en compte du handicap (visuel, moteur, etc.) et donc pour une meilleure accessibilité des logiciels (thématique internationale *Universal Access*). Notons au passage que le handicap mental est très souvent absent des recherches STIC, or nos approches cognitives peuvent apporter beaucoup dans ce domaine. Sur le plan plus technique de la normalisation, notons également, le lancement d'un groupe de travail (*Device Independence Working Group*) du W3C. Je trouve regrettable la faible représentativité de la recherche académique française dans ces structures et pense qu'il est nécessaire de ne pas être absent de ces travaux.

L'équipe de chercheurs que je souhaite fédérer devra donc posséder des compétences en Interaction Homme-Machine et en Génie logiciel, mais également être capable de collaborer avec des chercheurs en ergonomie, psychologie ou sociologie. Cette aptitude au travail pluridisciplinaire est encore plus importante si l'on s'intéresse aux environnements d'apprentissage car viennent se greffer des questions d'ordre pédagogique. Notons que les thématiques liées aux EIAH et développées au sein de l'équipe INSA du laboratoire ICTT, à savoir l'étude des apprentissages collectifs, des télé-TPs, des jeux d'entreprise et de l'analyse des usages, trouvent pleinement leur place au sein de mon projet scientifique et l'enrichissent dans la mesure où la question de l'adaptation des technologies d'apprentissage au contexte d'utilisation est prégnante. L'émergence de concepts comme le *m-learning* (mobile) est une autre manifestation de la nécessité de prendre en compte la question de la plasticité des EIAH.

Enfin, si je suis amené à poursuivre mes travaux au sein de mon équipe de recherche actuelle, je souhaite jouer un rôle majeur dans la réussite de la fusion du laboratoire ICTT et du laboratoire PRISMa. Début 2007, le laboratoire LIESP (Laboratoire d'Informatique pour l'Entreprise et les Systèmes de Production) va en effet voir le jour et il est important que de véritables synergies s'opèrent notamment à travers des projets communs. Co-responsable de l'axe Interaction Collaborative Médiatisée, j'oeuvrerai pour que les futurs domaines d'application de nos recherches, tout en s'enracinant dans notre culture d'origine, s'orientent davantage vers le génie industriel, qu'il s'agisse de traiter des questions d'interface opérateur-machine posant des problèmes de mobilité et de sécurité par exemple que de questions de cycle de vie des environnements techniques informatisés avec les questions de qualité et de maintenance.

6 Bibliographie

Aalto L., Göthlin, N., Korhonen J., Ojala T., (2004) Bluetooth and WAP Push Based Location-Aware Mobile Advertising System. In *Second International Conference on Mobile Systems, Applications and Services*, Boston, MA, pp. 49 - 58.

Abrams M., Helms, J., (2004) User Interface Markup Language (UIML) Specification version 3.1. Technical report, Harmonia.

Altmann, E. M., & John, B. E. (1999), Episodic indexing: A model of memory for attention events. *Cognitive Science*, 23 (2), 117-156.

Ariadne. (2002). Une association européenne ouverte au monde, pour partager et réutiliser la connaissance [en ligne]. Disponible sur <http://www.ariadne-eu.org>.

Atchley R. (1980) *The social forces in later Life, an Introduction to social Gerontology*, Belmont, CA, Wadswoeth Publishing Co

Azevedo P., Merrick R., Roberts D., (2000), OVID to AUIML: User-Oriented Interface Modelling. In *Electronic proceedings of the TUPIS2000 Workshop*, York, United Kingdom.

Bastien, J.M.C., Leulier, C. & Scapin, D.L. (1998). L'ergonomie des sites web. In *Créer et maintenir un service Web*, J.-C. Le Moal & B. Hidoine (eds.), ADBS Edition, pp. 111-173.

Beadle P., Harper B., Maguire G.Q., Judge, J., Location Aware Mobile Computing. *Proc. of IEEE Intl. Conference on Telecommunications*, Melbourne, Australia, April 1997.

Benadi S., (2004) *Structuration des données et des services pour le télé-enseignement*, thèse de doctorat, INSA de Lyon.

Blay-Fornarino M., Riveill M. Un service d'interactions. *RSTI (Revue des sciences et technologies de l'information) série TSI, 23(2)*, 2004. pp.175-204.

Bobillier-Chaumon M-E, Carvallo S., Tarpin-Bernard F., Vacherand-Revel J., (2005) Adapter ou uniformiser les logiciels ?, *Revue d'Interaction Homme Machine*, Europia,

Borgman C.L., All Users of Information Retrieval Systems are not Created Equal: An Exploration Into Individual Differences. *Information Processing & Management*, vol. 25, n°3, 1989, pp. 237-251.

Boswell D., King B., Oeschger I., Collins P., Murphy E., (2002) *Creating Applications with Mozilla*, First Edition September, O'Reilly

Bouchet J., Nigay L., Ganille T., The ICARE Component-Based Approach for Multimodal Input Interaction: Application to Real-Time Military Aircraft Cockpits. *HCI International, 3rd International Conference on Universal Access in Human-Computer Interaction*, Las Vegas, Nevada, USA, Juillet 2005.

Bouillon L., Vanderdonckt J., Souchon N., (2004) Rétro-ingénierie du Modèle de Présentation Pour les Pages Web. *Revue d'Interaction Homme-Machine*, Europia, Vol. 3.

Bourguin G., Derycke A., (2000). A Reflective CSCL Environment with Foundations Based on the Activity Theory. In *Proceedings of ITS'2000, Fifth International Conference on Intelligent Tutoring Systems*, vol. 1839, Montreal, CANADA, Verlag, Springer. pp. 272-281.

Bourguin, G., Derycke, A., & Tarby, J.C., (2001). Beyond the Interface : Co-evolution inside Interactive Systems - A proposal founded on Activity Theory, *Proceedings of IHM-HCI 2001 conference, People and computer XV-Interactions without Frontiers*, Blandford, Vanderdonckt, Gray (eds.), Lille, France, Springer Verlag, pp. 297-310.

Bourguin G., Derycke A., (2005). Systèmes Interactifs en Co-évolution, Réflexions sur les apports de la Théorie de l'Activité au support des Pratiques Collectives Distribuées. *Revue d'Interaction Homme-Machine*, vol. 6, n.1, pp. 1-31.

Bowman, D. (2002) Principles for the Design of Performance-oriented Interaction Techniques. In Stanney, K. (ed.), *Handbook of Virtual Environments*, Lawrence Erlbaum Associates, Mahwah, New Jersey, pp. 277-300.

Bowman, D. A. (1999) *Interaction Techniques for Common Tasks in Immersive Virtual Environments: Design, Evaluation, and Application*. Doctoral Thesis. UMI Order Number: AAI9953819.

Boyle, C., Encarnacion, A. O., (1994), MetaDoc: An Adaptive Hypertext Reading System. *User Modeling and User-Adapted Interaction*, 4 (1), pp. 1-19.

Brusilovsky, P. (1996a). Methods and Techniques of Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*, 6, pp. 87-129. 1996 (Reprinted in Adaptive Hypertext and Hypermedia, 1998, pp. 1-43, Kluwer Academic Publishers,).

Brusilovsky, P. (1996b) Adaptative Hypermedia : An Attempt and Generalize, P. Brusilovsky, P. Kommers, N. Streitz (Eds.), *Multimedia, Hypermedia, and Virtual Reality*, Berlin: Springer-Verlag 1077 pp.288-304.

Byrne M., D. (2003), Cognitive architecture, *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications*, Lawrence Erlbaum Associates, Inc., Mahwah, NJ, USA, 97-117.

Byrne, M. D., and Anderson, J. R. (1998), Perception and action. In: J. R. Anderson & C. Lebiere (eds.). *The atomic components of thought*, Hillsdale, NJ: Erlbaum, pp. 167-200.

Byrne, M. D. (2001), ACT-R/PM and menu selection: Applying a cognitive architecture to HCI. *International Journal of Human-Computer Studies*, 55, 41-84.

Calvary G., Coutaz J., Dâassi O., Balme L., Demeure A., (2004) Towards a New Generation of Widgets for Supporting Software Plasticity: the « Comet ». In *Proceedings of EHCI-DSVIS'04, The 9th IFIP Working Conference on Engineering for Human-Computer Interaction*, Bastide, R., Palanque, P., Roth, J. (Eds), Lecture Notes in Computer Science 3425, Springer, Hamburg, Germany, pp 306-323

Calvary G., Coutaz J., Thevenin D., Limbourg Q., Bouillon L., Vanderdonckt J. (2003) A unifying reference framework for multi-target user interfaces, *Journal of Interacting With Computer*, Elsevier Science B.V, June, Vol 15/3, pp 289-308.

Calvary G., Dâassi O., Coutaz J., Demeure A., (2005) Des Widgets aux Comets pour la Plasticité des Systèmes Interactifs. Revue d'Interaction Homme Machine, Europia, Vol 6 N°1.

Cannataro M., Cuzzocrea A., Pugliese A. (2001) A Probabilistic Adaptive Hypermedia System. In *International Symposium on Information Technology* (ITCC 2001), 2-4 April, Las Vegas, NV, USA. IEEE Computer Society, pp. 411-415

Chalon R. *Réalité Mixte et Travail Collaboratif : IRVO, un modèle de l'Interaction Homme-Machine*. Thèse de doctorat d'informatique. Ecole Centrale de Lyon, décembre 2004. p. 212.

Chen Q., Norcio A.F., Wang J. (2000) Neural Network Based Stereotyping for User Profiles, *Neural Computing and Applications*, 9, Springer Verlag, London, pp. 259-265.

Cheung Foo Wo D., Tigli, J.Y., Lavirotte S, Riveill M. (2006) Wcomp: a Multi-Design Approach for Prototyping Applications using Heterogeneous Resources. *In Proceedings of the 17th IEEE International Workshop on Rapid System Prototyping (RSP),* Chania, Crete, June, pp. 119-125.

Cheverst, K. Davies, N. Mitchell, K. Efstratiou, C. (2001) Using Context as a Crystal Ball : Rewards and Pitfalls. *Personal and Ubiquitous Computing*, 5(1), Springer Verlag Publ., pp.8-11.

Chikofsky E.J., Cross J.H., (1990) Reverse Engineering and Design Recovery: A Taxonomy. *IEEE Software*, Vol. 1, No. 7, Janvier 1990, pp. 13–17.

Chin, D. N. (2001), Empirical Evaluation of User Models and User-Adapted Systems, *User Modeling and User-Adapted Interaction*, 11 (1-2), 181–194.

Coffield, F., Moseley D., Hall, E., Ecclestone, K. (2004), *Should we be using learning styles? What research has to say to practice*, Learning and Skills Research Centre (LSRC), available at http://www.lsda.org.uk/files/PDF/1540.pdf

Coutaz J. (1987) PAC, an Object Oriented Model for Dialog Design, in Proceedings Interact'87, North Holland, pp.431-436.

Crampes M., Ranwez S., Plantier M. (2000). Ontology-supported and ontology-driven conceptual navigation on the world wide web. In *Proceedings of HyperText 2000*, pp. 191-199, San Antonio, Texas USA. HT2000, ACM Press.

David B.T., Chalon R., Delotte O., (2005) Model-Driven Engineering of Cooperative Systems, *HCI International 2005*, Las Vegas, 22-27 Juillet.

David B.T., Tarpin-Bernard F., Poquet J., Saikali K., Boutros N. (2000) From theory to practice: cooperation models in a sustainable product life-cycle in Designing Cooperative Systems, The Use of Theories and Models, *Proceedings 4th International Conference on the Design of Cooperative Systems* (COOP 2000, Sophia Antipolis, 23-26 Mai 2000), Eds G. De Michelis, A. Giboin , L. Karsenty and R. Dieng, Vol 58, IOS Press, Amsterdam, The Netherlands.

David M.G, Starzec C. (1996) *Aisance à 60 ans, Dépendance et isolement à 80 ans,* Division études sociales, Insee.

De Bra, P., Aerts, A., Berden, B., De Lange, B., Rousseau, B., Santic, T., Smits, D., Stash, N. (2003). AHA! The Adaptive Hypermedia Architecture. In *proceedings of the ACM Hypertext Conference*. Nottingham, UK, pp. 81-84.

De Bra, P., Brusilovsky, P., Houben, G.J.: 1999, Adaptive Hypermedia: From Systems to Framework. *ACM Computing Surveys*, 31 (4).

De Bra, P., Calvi, L. (1998), AHA! An open Adaptive Hypermedia Architecture. *The New Review of Hypermedia and Multimedia*, pp. 115-139.

De Bra, P., Stash, N., De Lange, B., (2003) AHA! Adding Adaptive Behavior to Websites. *Proceedings of the NLUUG Conference*, p. 10, Ede, The Netherlands, May.

Delestre, N. (2000). L'architecture clients-serveurs d'un hypermédia adaptatif pour la production automatique de cours. In colloque international TICE'2000 Technologies de l'Information et de la Communication dans les Enseignements d'ingénieurs et dans l'industrie, pp. 127-127.

Dewan P. (1992), Principles of Designing Multi-User Interface Development Environments, CSCW '92 Proceedings, November 1992, pp. 51-58.

Dey, A.K. (2001) Understanding and Using Context, *Personal and Ubiquitous Computing Journal*, Vol. 5 (1), pp. 4-7

Dieterich, H. Malinowski, U. Kühme, T. Schneider-Hufschmidt, M. (1993) State of the Art in Adaptive User Interfaces. In M. Schneider-Hufschmidt, T. Kühme and U. Malinowski (eds.): *Adaptive user interfaces: Principles and practice*. Amsterdam, North-Holland, pp. 13-48.

Dourish, P. (2004) What We Talk About When We Talk About Context, *Personal and Ubiquitous Computing*, 8(1), pp. 19-30.

Dourish P. (1995) Developing a Reflective Model of Collaborative Systems, *ACM Transactions on Computer-Human Interaction*, 2(1), March, 40-63.

Dragicevic P. (2004), *Un modèle d'interaction en entrée pour des systèmes interactifs multi-dispositifs hautement configurables*. Thèse de doctorat, École Nationale Supérieure des Techniques Industrielles et des Mines de Nantes.

Dufresne A. (2001), Conception d'une Interface Adaptée aux Activités de l'Education à Distance : ExploraGraph, *Sciences et Techniques Éducatives*, vol. 8, n°3-4, pp. 301-320.

Dubinko M., Leigh L., Klotz Jr., Merrick R., Raman T. V., (2003) *XForms 1.0.*, W3C Recommendation, World Wide Web Consortium, October.

Felder, R.M., (1996), Matters of styles, *PRISM*, ASEE, 6 (4), 18-23.

Felder, R.M., Brent, R. (2001) Effective Strategies for Cooperative Learning. *Journal of Cooperation & Collaboration in College Teaching* 10, no.2, pp. 63–69.

Fink J., Kobsa A., (2000) A Review and Analysis of Commercial User Modeling Servers for Personalization on the World Wide Web, *International Journal of User Modeling and User-adapted Interaction*, 10, Kluwer Academic Publishers, pp. 209-249.

Fink, F., Kobsa, A., Nill, A. (1997) Adaptable and Adaptive Information Access for All Users, Including the Disabled and the Elderly. In A. Jameson, C. Paris, & C. Tasso (eds.), *6th International Conference on User Modelling* (UM '97, Sardinia, Italy), New York: Springer-Verlag, pp. 171-173.

Fischer, G. (2001) User Modeling in Human-Computer Interaction, *User Modeling and User-Adapted Interaction*, Volume 11, Issue 1-2, pp. 65-86.

Florins M., Trevisan D., Vanderdonckt J., (2004) The Continuity Property in Mixed Reality and Multiplatform Systems: a Comparative Study. In *Proceedings of CADUI'04*, Madeira Island, pp. 13-16.

Frasincar F., Houben G-J. (2002) Hypermedia Presentation Adaptation on the Semantic Web. In de Bra P.,Brusilovsky P. & Conejo R., (Eds.), *Proceedings of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, Malaga, Spain, LNCS 2347, pp. 133-142.

Gaffar A., Sinnig D., Seffah A., Forbrig P., (2004) Modeling patterns for task models. In *Proceedings of 3rd International Workshop on TAsk MOdels and DIAgrams for user interface design TAMODIA'2004*. Prague, Czeck Republic, November, ACM Press, pp. 99-104.

Gamma, E., Helm, R., Johnson R. & Vlissides J., (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, Reading, Mass.

Garlan D., Perry D. Introduction to the Special Issue on Software Architecture, IEEE Transactions on Software Engineering, April 1995.

Goldberg A., (1984) *Smalltalk-80: The interactive programming environment*, Addison-Wesley Publ.

Gorgeon C., Léridon H. (2001) Rencontres sur le vieillissement. Rapport du comité scientifique d'organisation. http://www.inserm.fr/serveur/vieil.nsf/Titre/Rencontres+Vieillissement+Sommaire?OpenDocument

Graham K., Reynolds F., Woodrow C., Ohto H., Hjelm J., Butler M. H., Tran L., (2004) *Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0.* W3C Recommendation January 2004.

Gram C. & Cockton G., Eds (1996), *Design Principles for Interactive Software*, Chapman & Hall, pp. 27-36

Gray, W. D., Young, R. M., & Kirschenbaum, S. S. (1997), Introduction to a special issue on cognitive architectures and human-computer interaction. *Human-Computer Interaction*, 12, 301-309.

Greenberg S., Roseman M., Webster D, Bohnet R., (1992), Human and Technical Factors of Distributed Group Drawing Tools, *Interacting with Computers*, 4(3), 364-392.

Guittet L. (1995), *Contribution à l'Ingénierie des Interfaces Homme-Machine, Théorie des Interacteurs et Architecture H4 dans le système NODAOO*. Thèse de doctorat, Université de Poitiers.

Habieb-Mammar H. (2004), *EDPHA : un Environnement de Développement et de Présentation d'Hyperdocuments Adaptatifs*, Thèse de doctorat, INSA de Lyon

Habieb-Mammar H., Tarpin-Bernard F., Prévot P. (2003) Adaptive Presentation of Multimedia Interface Case study: "Brain Story" Course. In *User Modeling 03*, Springer Verlag. Carnegie Mellon University, Pittsburgh, Peter Brusilovsky and Albert T. Corbett and Fiorella de Rosis Eds. pp.15-24.

Habieb-Mammar H., Tarpin-Bernard F., Prévot P. (2003) Modélisation et mise en œuvre d'hyperdocuments pédagogiques adaptatifs, *IHM 2003*. Caen. Novembre.

Henze N., Nejdl X., (1999) *Bayesian Modeling for Adaptive Hypermedia Systems*. Technical Report, University of Hannover, July.

Höök K., Karlgren J., Wærn A. (1996) A glass box approach to adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 6 : pp. 157-184.

Höök, K. (1997), Evaluating the utility and usability of an adaptive hypermedia system. In: J. Moore, E. Edmonds and A. Puerta (eds.): *International Conference on Intelligent User Interfaces*, Orlando, Florida, ACM, pp. 179-186.

IEC CDV TR 61997 (2000) *Guidelines for the user interfaces in multimedia equipment for general purpose use*.

ISO 9241 (1998) *Ergonomic requirements for office work with visual display terminals.*

Jacob, R.J.K., (1994) New Human-Computer Interaction Techniques. In *Human-Machine Communication for Educational Systems Design*, Brouwer-Janse M.D. and Harrington T.L. (eds), Springer-Verlag, Berlin, 1994, pp. 131-138.

Jameson, A. (2003). Adaptive interfaces and agents. In J. A. Jacko & A. Sears (Eds.), *The Human Computer Interaction Handbook*, Mahwah, NJ: Erlbaum, pp.305-330.

Javahery H., Seffah A., Engelberg D., Sinnig D., (2004), Migrating User Interfaces between Platforms Using HCI Patterns. In *Multiple User Interfaces : Cross Platform Applications and Context-Aware Interfaces*, Seffah A., Javahery H. (Eds), John Wiley & Sons, Chichester, England.

Kaheneman, D., (1973) *Attention and Effort*, Englewood Cliffs: Prentice Hall.

Kaplan, S, Seebeck, L. (2001). Harnessing complexity in CSCW, *European Conference on Computer Supported Collaborative Systems* (ECSCW-01), pp. 359-378.

Kieras, D. & Meyer, D.E. (1997). An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-Computer Interaction*., 12, 391-438.

Kobsa A. (1993) User modeling : Recent work, prospects and hazards. Dans Schneider-Hufschmidt M., Kühme T. et Malinowski U., (eds), *Adaptive User Interfaces : Principles and Practice*, North-Holland, Amsterdam.

Kobsa, A. (1994), Conceptual Hierarchies: Approaches from Artificial Intelligence and Connectionism. In: H. Best, B. Endres-Niggemeyer, M. Herfurth und P. P. Ohly, Hrsg. (eds): *Informations- und Wissensverarbeitung in den Sozialwissenschaften: Beiträge zur Umsetzung neuer Informationstechnologien*. Opladen, Westdt. Verlag.

Kobsa, A. (2001) Generic user modeling systems. User Modeling and User-Adapted Interaction 11, pp. 49-63. HTTP:// http://www.ics.uci.edu/~kobsa/papers/2001-UMUAI-kobsa.pdf.

Kobsa A., Pohl W. (1995) The user modeling shell system bgp-ms. *User Modeling and User-Adapted Interaction*, 4 : pp. 59-106.

Koskimies O., Wasmund M., Wolkerstorfer P., Ziegert T., (2004) Practical Experiences with Device Independent Authoring Concepts. In *Advances on User Interface Description Languages, workshop of AVI 2004*, Expertise Center for Digital Media.

Lemaire, P., (1999), *Psychologie cognitive*, De Boeck Université, Bruxelles, 1999.

Lemlouma T., Layaïda N., (2002) Universal Profiling for Content Negotiation and Adaptation in Heterogeneous Environments, *W3C Workshop on Delivery Context*, W3C/INRIA Sophia-Antipolis, France, March.

Limbourg Q., Vanderdonckt J., Michotte B., Bouillon L., Florins M., (2004) UsiXML: A User Interface Description Language Supporting Multiple Levels of Independence. In *Proceedings of DIWE'04 Workshop*, Munich, July.

Luyten K., Abrams M., Limbourg Q., Vanderdonckt J. (Eds.). (2004) Developing User Interfaces with XML: Advances on User Interface Description Languages. *Workshop of Advanced Visual Interfaces (AVI) 2004*, Expertise Center for Digital Media.

Luyten K., (2004) *Dynamic User Interface Generation for Mobile and Embedded Systems with Model-Based User Interface Development*. PhD, Limburgs Universitair Centrum, School of IT, Expertise Centre for Digital Media, Diepenbeek, Belgium, Oct..

Malone, T. W., Lai, K. Y., Fry, C. (1995) Experiments with Oval: A radically tailorable tool for cooperative work. *ACM Transactions on Information Systems*, 13, 2 (April), pp. 177-205.

Masserey G., Tran C. D., Samaan K., Tarpin-Bernard F., David B., (2005) Environnement de conception et développement d'applications interactives basées sur l'architecture AMF, In *proceedings of IHM'2005*, International Conference Proceedings Series, ACM, Toulouse, France. September 2005.

Meire Ph. (1992) Vie affective, psychodynamique et vieillissement In A. Gommers, Philippe Van Den Bosch De Aguilar (Eds) *Pour une vieillesse autonome : vieillissement dynamismes et potentialités*, Mardaga, Paris

Michel, C., Prié, Y., and Le Graët, L. (2005). Construction d'une base de connaissance pour l'évaluation de l'usage d'un environnement STIC. In *Proceedings of the 17th Conference on 17èMe Conférence Francophone Sur L'interaction Homme-Machine* (Toulouse, France, September 27 - 30, 2005). IHM 2005. ACM Press, New York, NY, 199-202.

Mori G., Paternò F., Santoro C., (2004) Design and Development of Multidevice User Interfaces Through Multiple Logical Descriptions. *IEEE Transactions on Software Engineering*. Aug., pp. 507-520.

Mørch, A.I. & Mehandjiev, N.D. (2000) Tailoring as Collaboration: The Mediating Role of Multiple Representations and Application Units. *Computer Supported Cooperative Work* 9(1), pp. 75-100.

Müller H., Jahnke J., Smith D., Storey M-A., Tilley S., Wong K., (2000) Reverse Engineering: A Roadmap. *22nd International Conference on Software Engineering*, Limerick, Ireland, pp. 47 – 60.

Newell, A. (1990), *Unified theories of cognition*, Cambridge, MA: Harvard University Press.

Nigay L., Coutaz J. (1993) A Design Space for Multimodal Systems: Concurrent Processing and Data Fusion. *Proceedings of InterCHI'93*, Amsterdam, pp.172-178.

Nigay L., (1995) MATIS : un Système Multimodal d'Information sur les Transports Aériens. *7èmes journées de l'ingénierie de l'interaction Homme Machine (IHM'95)*, CEPAD Publication, pp. 131-132.

Ouadou K., (1994) *AMF : Un modèle d'architecture multi-agents multi-facettes pour Interfaces Homme-Machine et les outils associés*. Thèse de doctorat, Ecole Centrale de Lyon.

Paganelli L., Paternò F., (2003) A Tool for Creating Design Models from Web Site Code. International Journal of Software Engineering and Knowledge Engineering, World Scientific Publishing Vol.13, N° 2, pp. 169-189.

Paternò, F., (1999), *Model-based Design and Evaluation of Interactive Applications*, Springer-Verlag.

Pazzani, M. and Billsus, D. (2002) Adaptive Web Site Agents. Journal of Autonomous Agents and Multiagent systems, Kluwer Academic Publishers, Amsterdam, 5(2). pp. 205-218.

Peinel G., Rose T., Sedlmayr M., (2003) APNEE: Air Pollution Network for Early Warning and Information Exchange in *Europe 3rd International Symposium on Digital Earth* 2003, Brno, Czech Republic.

Petrelli D., De Angeli A., Convertino G. (1999) A user-centered approach to user modeling. In Kay J. Ed., Proceedings of the Seventh International Conference. UM99 - User Modeling, SpringerWienNewYork, pp. 255-264.

Pfaff, G. E., editor (1985). *User Interface Management Systems: Proceedings of the Seeheim Workshop*, Berlin. Springer-Verlag. Proceedings of the Workshop on User Interface Management Systems, held in Seeheim, FRG, November 1-3, 1983

Ploton L. (2003) *La personne âgée : son accompagnement médical et psychologique et la question de la démence*. Broché, Paris.

Poquet J. (1998), *Architecture AMF des systèmes interactifs : conception d'un moteur de gestion d'interactions en Java*. DEA I.S.C.E. Université Claude Bernard de Lyon.

Puerta A., Eisenstein J., (2004) XIML: A Multiple User Interface Representation Framework for Industry, In *Advances on User Interface Description Languages, workshop of AVI 2004*, Expertise Center for Digital Media.. pp.119–148.

Rey G., Coutaz J. (2002) Le Contexteur : une Abstraction Logicielle pour la Réalisation de Systèmes Interactifs Sensibles au Contexte, In Proc. *IHM2002*, pp 105-112.

Rich E., (1989) Stereotypes and user modeling, in *User models in dialog systems*, A. Kobsa and W. Wahlster, Ed. Springer verlag: Berlin, 1989, p. 35-51.

Rich E., (1983) Users are Individuals: Individualizing User Models. *Int. J. Man-Machine Studies*, Vol. 3, N°18, pp. 23-46.

Ritter, F. E., Baxter, G. D., Jones, G., & Young, R. M. (2000), Cognitive models as users. *ACM Transactions on Computer-Human Interaction*, 7 (2), pp. 1-33.

Roudaut A., Coutaz J. (2006) Méta-IHM ou comment contrôler l'espace interactif ambiant, Ubimob 2006, Actes des Troisièmes Journées Francophones : Mobilité et Ubiquité 2006 (Paris, France). ACM Press, pp. 73-80.

Rouillard J., (2003) Plastic ML and its toolkit. HCI International 2003, Heraklion, Crete, Greece, Volume 4, pp. 612-616.

Ryan N., Pascoe J., Morse D., (1997) Enhanced Reality Fieldwork : the Context-Aware Archeological Assistant. Gaffney V., van Leusen M., Exxon S. (Eds.), Computer Applications in Archeology. British Archaeological Reports, Oxford.

Samaan K., Tarpin-Bernard F., (2004) Task models and interaction models in a multiple user interfaces generation process. In *Proceedings of TAMODIA 2004*, Czech Republic, vol. 86, ACM Press, New York, NY, pp. 137-144.

Samaan K., Tarpin-Bernard F., (2003) L'Utilisation de Patterns d'Interaction pour l'Adaptation d'IHM Multicibles. In *proceedings of IHM'03*, Caen, France, vol. 51, ACM Press, New York, NY, pp. 272-275.

Samaan K., Delotte O., Tarpin-Bernard F., (2002) Processus de génération d'IHM multicibles pour applications interactives. In *Proceedings of the 14th French-Speaking Conference on Human-Computer interaction* (Poitiers, France, November 26 - 29, 2002). M. Beaudouin-Lafon, Ed. IHM '02, vol. 32. ACM Press, New York, NY, pp. 251-254.

Scapin, D-L. Bastien, C. (1996). *Inspection d'interfaces et critères ergonomiques*. INRIA Rapport de Recherche.

Schilit B.N., Adams N.I., Want R., (1994) Context-Aware Computing Applications. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications* (WMCSA'94), IEEE Press, pp 85-90.

Schmidt A., Implicit Human Computer Interaction Through Context. *Personal Technologies*, Vol 4, N° 2&3, June 2000, pp. 191-199.

Seffah A., Javahery H. eds. (2004), *Multiple User Interfaces : Cross Platform Applications and Context-Aware Interfaces*, John Wiley & Sons, Chichester, England.

Senach, B. (1990). *Evaluation ergonomique des interface homme-machine: une revue de littérature*, INRIA Rocquencourt: rapport technique n°383.

Sinnig D., Gaffar A., Reichart D., Seffah A., Forbrig P., Patterns in Model-Based Engineering. In *Proceedings of the 4th Int. Workshop on Computer-Aided Design of User Interfaces (CADUI'04)*, Funchal, Madeira Island, Port., January 13-16, 2004, pp.195-208.

Sottet J.S., Calvary G., Favre J.M., (2005) Ingénierie de l'Interaction Homme-Machine Dirigée par les Modèles. *IDM'05*, Paris, France, pp. 67-82.

Stephanidis C., Paramythis A., Sfyrakis M., Stergiou A., Maou N., Leventis A., Paparoulis G. & Karagiandidis C., (1998) Adaptable and adaptive user interfaces for disabled users in AVANTI Project. In Triglia S., Mullery A., Campolargo M., Vanderstraeten H. & Mampaey M. Eds. *Proceedings of the 5th International Conference on Intelligence in Services and Networks (IS&N'98)*, Technology for Ubiquitous Telecom Services, Antwerp, Belgium, 1998, LNCS 1430, Springer-Verlag, Germany, pp. 153–166.

Tarpin-Bernard F., Habieb-Mammar H., Seffah A., Towards UI Characterizing and Adaptability Quantifying, *Interacting with Computers*, 24 p., soumis 07/2006. 2° révision en cours.

Tarpin-Bernard F., Habieb-Mammar H., Croisile B., Noir, M. (2001) A supervised Program for Cognitive e-Training. In WebNet'2001, World Conference on Web technologies. Orlando. pp 1208-1213. ISBN 1-880094-46-0.

Tarpin-Bernard F., Habieb-Mammar H., (2005) Modeling Elementary Cognitive Abilities for Adaptive Presentation of Hypermedia, *User Modeling and User-Adapted Interaction (UMUAI)* The Journal of Personalization Research, Kluwer Academic Publishers, Volume 15, Issue 5, Nov, pp. 459 - 495.

Tarpin-Bernard, F., David B.T., (1999) AMF : un modèle d'architecture multi-agents multi-facettes. *Techniques et Sciences Informatiques*. Hermès. Paris. Vol. 18, No. 5. Mai, pp. 555-586.

Tarpin-Bernard, F., (1997) *Travail coopératif synchrone assisté par ordinateur : Approche AMF-C*. Thèse de doctorat, Ecole centrale de Lyon, p.154.

Tarpin-Bernard, F., (2000) La flexibilité dans les collecticiels, in *Le temps, l'espace et l'évolutif* (Tome 2). H. Prade, R. Jeansoulin & C. Garbay (eds). Septembre 2000. Cépaduès. pp. 449-458.

Tarpin-Bernard, F., David B.T., Primet P., (1998) Frameworks and Patterns for Synchronous Groupware: AMF-C Approach. In Proceedings of the IFIP Tc2/Tc13 Wg2.7/Wg13.4 Seventh Working Conference on Engineering For Human-Computer interaction (September 14 - 18, 1998). S. Chatty and P. Dewan, Eds. IFIP Conference Proceedings, vol. 150. Kluwer B.V., Deventer, The Netherlands, pp. 225-241.

Thevenin D., (2001) Adaptation en Interaction Homme-Machine: Le cas de la Plasticité. Thèse de doctorat, Université Joseph Fourier, Grenoble I, 2001. 234 p.

Thevenin, D., Coutaz, J. (1999) Plasticity of User Interfaces: Framework and Research Agenda. In Proceedings of INTERACT'99, Edimbourg, UK, pp. 110-117.

UIMS (1992), A metamodel for the runtime architecture of an interactive system. ACM SIGCHI Bulletin, Vol. 24, No 1, pp. 32-37.

Vacherand-Revel J., Tarpin-Bernard, F., David B.T., (2001) Des modèles de l'interaction à la conception "participative" des logiciels interactifs, in *Conception : entre science et art*. Presse Polytechnique Romande. Lausanne. Perrin J. (eds). Juillet 2001. pp. 239-255.

Van den Bergh J., Coninx K., (2004) Model-Based Design of Context-Sensitive Interactive Applications: a Discussion of Notations. In Proceedings of TAMODIA'2004. Prague, Czech Republic, 43-50.

Vanderdonckt J., Grolaux D., Van Roy P., Limbourg Q., Macq B., Michel B. (2005) A Design Space For Context-Sensitive User Interfaces, In Proceedings of IASSE 2005.

Van Welie M., Trætteberg H., Interaction Patterns in User Interfaces. In *7th. Pattern Languages of Programs Conference*, Allerton Park

Weber, G., Kuhl, H-C., and Weibelzahl, S. (2001). Developing adaptive internet based courses with the authoring system NetCoach. In Reich, S., Tzagarakis, M.M., and de Bra, P. (Eds.), *Hypermedia: Openness, Structural Awareness, and Adaptivity*, Berlin:Springer, pp. 226-238.

Weil-Barais, A. (2001) *L'homme cognitif*. Presses Universitaires de France. 2001. 616 p.

Weiser, M. (1993) Some Computer Science Problems in Ubiquitous Computing, *Communications of the ACM*, July 1993. (reprinted as "Ubiquitous Computing". Electronics; pp. 137-143, December 6.