Modélisation et conception d'une plateforme pour l'interaction multimodale distribuée en intelligence ambiante

Gaëtan Pruvost

UNIVERSITÉ PARIS SUD

Comprendre le monde, construire l'avenir®

UNIVERSITE PARIS-SUD

ÉCOLE DOCTORALE D'INFORMATIQUE DE PARIS-SUD

Laboratoire d'Informatique et de Mécanique pour les Sciences de l'Ingénieur

Informatique

THÈSE DE DOCTORAT

soutenue le 11/02/2012

par

Gaëtan PRUVOST

Modélisation et conception d'une plateforme pour l'interaction multimodale distribuée en intelligence ambiante

Directeur de thèse :	Yacine Bellik	Maître de conférences, HDR, Université de Paris-Sud 11

Composition du jury :

Président du jury :	Anne VILNAT	Professeur, Université de Paris-Sud 11
Rapporteurs :	Christophe KOLSKI	Professeur, Université de Valenciennes
	José ROUILLARD	Maître de conférences, HDR, Université de Lille 1
Examinateurs :	Wolfgang MINKER	Professeur, Université d'Ulm, Allemagne

RESUME

Cette thèse s'inscrit dans le domaine de l'intelligence ambiante et de l'interaction homme-machine. Elle a pour thème la génération d'interfaces homme-machine adaptées au contexte d'interaction dans les environnements ambiants. Les travaux de recherche présentés traitent des problèmes rencontrés lors de la conception d'IHM dans l'ambiant et notamment de la réutilisation de techniques d'interaction multimodales et multi-périphériques. Ce travail se divise en trois phases. La première est une étude des problématiques de l'IHM spécifiques à l'Ambiant et des architectures logicielles adaptées à ce cadre théorique. Cette étude permet d'établir les limites des approches actuelles et de proposer, dans la seconde phase, une nouvelle approche pour la conception d'IHM ambiante appelée DAME. Cette approche repose sur l'association automatique de composants logiciels qui construisent dynamiquement une IHM. Nous proposons deux modèles complémentaires qui permettent de décrire les caractéristiques ergonomiques et architecturales des composants. La conception de ces derniers est guidée par une architecture logicielle composée de plusieurs couches qui permet d'identifier les différents niveaux d'abstraction réutilisables d'un langage d'interaction. Un troisième modèle, appelé modèle comportemental, permet de spécifier des recommandations quant à l'instanciation de ces composants. Nous proposons un algorithme permettant de générer des IHM adaptées au contexte et d'évaluer la qualité de celles-ci par rapport aux recommandations du modèle comportemental. Dans la troisième phase, nous avons implémenté une plateforme réalisant la vision soutenue par DAME. Cette implémentation est confrontée aux utilisateurs finaux dans une expérience de validation qualitative. De ces travaux ressortent des résultats encourageants, ouvrant la discussion sur de nouvelles perspectives de recherche dans le cadre de l'IHM en informatique ambiante.

Mots clés : Interaction homme-machine, intelligence ambiante, informatique ubiquitaire, génération d'interface, contexte d'interaction, multimodalité, multi-médias, programmation orientée composants.

ABSTRACT

This thesis deals with ambient intelligence and the design of Human-Computer Interaction (HCI). It studies the automatic generation of user interfaces that are adapted to the interaction context in ambient environments. This problem raises design issues that are specific to ambient HCI, particularly in the reuse of multimodal and multidevice interaction techniques. The present work falls into three parts. The first part is an analysis of state-of-the-art software architectures designed to solve those issues. This analysis outlines the limits of current approaches and enables us to propose, in the second part, a new approach for the design of ambient HCI called DAME. This approach relies on the automatic and dynamic association of software components that build a user interface. We propose and define two complementary models that allow the description of ergonomic and architectural properties of the software components. The design of such components is organized in a layered architecture that identifies reusable levels of abstraction of an interaction language. A third model, called behavioural model, allows the specification of recommendations about the runtime instantiation of components. We propose an algorithm that allows the generation of context-adapted user interfaces and the evaluation of their quality according to the recommendations issued from the behavioural model. In the third part, we detail our implementation of a platform that implements the DAME approach. This implementation is used in a qualitative experiment that involves end-users. Encouraging preliminary results have been obtained and open new perspectives on multi-devices and multimodal HCI in ambient computing.

Keywords: Human-Machine Interaction, ambient intelligence, ubiquitous computing, user interface generation, interaction context, multimodality, multidevice, component oriented programming.

REMERCIEMENTS

Je tiens à dire un grand merci…

A Yacine BELLIK, mon encadrant de thèse, qui m'a guidé et conseillé au quotidien. Tu m'as fait découvrir le domaine de l'Ambiant et plus généralement, le monde de la recherche. Je te remercie tout particulièrement pour la confiance et la liberté que tu m'as accordées dans mon travail, que ce soit dans mes choix théoriques ou pratiques.

A Christophe KOLSKI, professeur à l'université de Valenciennes, et à José ROUILLARD, maître de conférences à l'université de Lille 1, pour avoir accepté d'être mes rapporteurs. Votre analyse rigoureuse m'a permis d'apporter de nombreuses améliorations à ce manuscrit.

A Anne VILNAT, directrice adjointe du LIMSI-CNRS, pour avoir accepté de présider mon jury de soutenance. Je te remercie tout particulièrement pour les encouragements et le soutien que tu m'as prodigués au moment où j'en avais le plus besoin.

A Wolfgang MINKER, professeur à l'université d'Ulm, en Allemagne, pour avoir accepté de faire partie du jury de soutenance malgré la distance et les intempéries. J'ai beaucoup appris des années ATRACO sous ta coordination.

A Patrick Le Quéré, directeur du LIMSI-CNRS, pour son accueil dans le laboratoire et à Jean-Paul SANSONNET, directeur de recherche CNRS et responsable du groupe AMI, pour m'avoir permis de réaliser cette thèse.

Mais encore…

Je remercie également tous les collègues du groupe AMI (anciens ou récents) qui contribuent, chacun à leur façon, à l'ambiance exceptionnelle qui y règne. Je garderai toujours un souvenir ému de tous ces bons moments passés autour d'un repas, d'un café ou… sur un court de tennis.

Selon la coutume, je termine par les personnes qui ont le plus contribué à cet ouvrage… Je voudrais donc remercier mes proches, qui ont su m'insuffler le courage nécessaire à l'accomplissement d'un rêve d'enfant. Merci, en premier lieu, à Marie. Tes doutes ne sont rien face à la force de caractère, à la volonté et à l'attachement dont tu as fait preuve tout au long de cet éprouvant chemin. Tu m'as également offert le plus beau des cadeaux, de ceux qui changent notre perception du monde. Merci ainsi à Paul de m'avoir innocemment rappelé le goût des choses simples et le plaisir de vivre l'instant présent. A mes parents, pour avoir cultivé en moi le goût de la curiosité. A mes sœurs et à toute ma famille, pour votre inébranlable foi en moi. Et plus spécialement à ma belle-famille, pour m'avoir accueilli comme l'un des leurs : votre franchise m'est précieuse. Enfin, merci à mes amis de toujours placer la barre plus haut.

Les pages qui suivent sont aussi un peu les vôtres…

A Marie

A Paul

"Je te regarderai du coin de l'oeil et tu ne diras rien. Le langage est source de malentendus. Mais, chaque jour, tu pourras t'asseoir un peu plus près..."

Le petit prince, Antoine de Saint-Exupery

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

AVANT-PROPOS

La miniaturisation des dispositifs électroniques et leur connexion en réseau dans un contexte mobile favorisent leur intégration dans notre environnement quotidien. Les systèmes informatiques sont désormais embarqués dans tous les objets de notre environnement : des TV connectées aux systèmes embarqués dans les voitures, en passant par les programmateurs des machines à laver, un nombre grandissant d'objets de la vie courante est concerné. L'interconnexion de ces dispositifs, réalisable grâce aux progrès des télécommunications, change nos habitudes de vie. L'ordinateur, autrefois machine dédiée à réaliser une tâche spécifique dans un environnement connu et invariant, s'est transformé en calculateur universel devant simplifier nos vies quotidiennes. La mobilité actuelle des informations sur les réseaux et les coopérations entre dispositifs remettent en question la vision d'une machine isolée et indépendante. Un changement de paradigme s'opère vers les concepts de l'intelligence ambiante, où les frontières entre environnement physique et système informatique s'effacent au profit d'un environnement augmenté dont le comportement s'adapte aux utilisateurs et les assiste dans leurs tâches les plus quotidiennes.

Ces évolutions ont un impact sur les interfaces homme-machine (IHM). L'interaction dans les environnements ambiants devient contextuelle et distribuée. Elle est caractérisée par l'utilisation séquentielle ou simultanée d'un nombre important de périphériques d'interaction. En lieu et place d'une plateforme composée d'un ensemble de périphériques d'interaction prédéfinis, l'interaction ambiante est réalisée par une combinaison variable de périphériques. Ces situations d'interaction variées permettent l'introduction de nouvelles techniques d'interaction qui exploitent au mieux nos capacités sensorielles (par exemple : la synthèse vocale, les écrans tactiles, les avatars…). La complexité et la diversité des situations qui résultent de cette évolution font émerger un nouveau besoin : celui de fournir, à tout moment, des interfaces adaptées à la situation qui permettent de contrôler les services et applications supportés par les ressources disponibles dans l'environnement. Réaliser ce besoin implique de repenser la communication en fonction de l'utilisateur, des capacités du système et de la nature de l'environnement ambiant. Toutes ces composantes définissent un contexte d'interaction devant être désormais considéré et exploité dans sa globalité.

La gestion de l'interaction dans les environnements ambiants doit prendre en compte de nouveaux besoins qui ne sont pas supportés par les architectures actuelles. L'objectif de cette thèse est de proposer une démarche de conception d'un système d'interaction adapté aux problématiques de l'Ambiant. Nous étudierons notamment la capacité d'un tel système à combiner, de façon opportuniste, un ensemble de ressources disponibles et à instancier des techniques d'interaction multimodales. De plus, un tel système doit pouvoir s'adapter dynamiquement à l'environnement, interpréter et réagir aux changements du contexte tels que l'apparition ou la disparition de périphériques et/ou de services.

Notre travail autour d'un système de génération d'IHM dans un environnement ambiant s'articule en 3 parties, chacune divisée en plusieurs chapitres.

La première partie (Partie A) propose une vue d'ensemble des recherches menées depuis une vingtaine d'années sur les domaines de l'intelligence ambiante, de la conception d'IHM et des architectures logicielles d'IHM spécifiques à l'Ambiant. Nous commençons par cartographier le domaine de l'intelligence ambiante en étudiant son origine, sa terminologie et ses applications. Cette analyse nous permettra de mettre en exergue plusieurs problématiques soulevées par les environnements ambiants, et plus précisément l'impact de ces problématiques sur la conception d'IHM. Nous étudions ensuite les méthodes employées pour concevoir des IHM riches et innovantes, et notamment des IHM multimodales. A partir des problématiques de l'Ambiant et des méthodes de conception que nous aurons ainsi identifiées, nous pourrons comparer les architectures d'IHM existantes dans l'Ambiant. A l'issue de cette étude de l'état de l'art, nous extrayons un ensemble de besoins qu'un système d'interaction ambiant devrait idéalement satisfaire.

La seconde partie (Partie B) propose une démarche de conception d'IHM, intitulée DAME (Distributed Ambient Multimodal Environments), qui tente de répondre à ces besoins. Cette démarche repose sur une conception modulaire des IHM qui est représentée par un modèle de composants. Ces composants sont modélisés, *a posteriori*, dans 3 modèles : le modèle ergonomique, le modèle architectural et le modèle comportemental. Nous décrirons successivement chacun de ces 3 modèles. Le modèle ergonomique permet de spécifier le rôle des composants sur le plan ergonomique. Il décrit en particulier les modalités, les canaux de communication et les langages d'interaction qui sont implémentés par l'IHM. Le modèle architectural représente les contraintes structurelles et logicielles des composants. De plus, il définit 4 catégories de composants qui, organisés en couches, définissent les différents niveaux d'abstraction d'une IHM. Enfin, le modèle comportemental décrit les mécanismes de conception automatique d'IHM adaptées au contexte. Il repose sur la rédaction de recommandations, appelées critères, par les concepteurs ou les ergonomes du système. Nous terminons cette seconde partie par l'application de la démarche DAME sur un ensemble d'exemples qui illustrent la satisfaction des besoins exprimés à l'issue de l'étude de l'état de l'art.

La dernière partie (Partie C), enfin, décrit l'implémentation d'une plateforme qui suit l'approche DAME. Cette implémentation combine des technologies issues de domaines de recherche variés (système expert, ontologies et conteneur de composants) qui permettent d'instancier les modèles et comportements décrits dans la partie B. Elle s'accompagne également d'outils de spécification et de simulation destinés aux concepteurs de systèmes d'interaction ambiants. Une expérience auprès d'utilisateurs finaux a été conçue à partir de cette implémentation dans le cadre du projet de recherche européen ATRACO[1]. Nous présentons en dernier lieu, les résultats de cette étude qualitative qui nous permet de démontrer la faisabilité de notre approche.

Le mémoire se termine par une conclusion sur l'apport de nos travaux dans le domaine de l'interaction homme-machine dans les environnements ambiants et propose quelques perspectives de recherches pertinentes au regard de nos résultats.

---

[1] Projet de recherché européen intitule Adaptive and TRusted Ambient eCOlogies

http://www.uni-ulm.de/in/atraco/

Partie A
ETUDE DE LA CONCEPTION D'IHM DANS L'AMBIANT

Nous définissons, dans cette partie, les besoins qu'un système d'interaction ambiant doit satisfaire. Nous étudions pour cela les approches existantes en conception d'IHM ambiante.

Nous préciserons tout d'abord le contexte de notre étude en définissant l'intelligence ambiante dans le chapitre I. Nous définirons ce concept et ses caractéristiques et présenterons quelques applications qui illustrent l'utilité et la complexité du domaine. Nous explorerons les détails de cette complexité et de ses conséquences en termes d'interaction homme-machine dans le chapitre II. Dans le chapitre III nous montrerons par quelles techniques la conception des IHM s'adresse à ces problématiques. Nous étudierons à la fois la méthodologie de la conception en général mais également les techniques et métaphores qui sont spécifiquement conçues pour l'Ambiant. La complexité d'implémenter ces nouvelles formes d'interaction requiert des architectures logicielles spécifiques que nous étudions dans le chapitre IV. A l'issue de ce chapitre, nous proposons une comparaison des architectures existantes et en déduisons les limites des systèmes actuels.

L'ensemble de nos conclusions est résumé à la fin de cette partie, dans un bilan qui expose nos motivations pour la proposition d'une nouvelle approche de conception d'IHM dans l'Ambiant.

Chapitre I
L'INTELLIGENCE AMBIANTE

L'intelligence ambiante est un domaine de recherche vaste et aux applications variées. Nous apportons, dans ce chapitre, une analyse des différents champs d'application, ce qui nous permettra de mieux en cibler les contraintes et les besoins pour l'IHM. Nous présenterons, dans un premier temps, une introduction rapide aux fondements de ce domaine. Nous préciserons ensuite la terminologie que nous emploierons dans l'ensemble de ce document. Enfin, nous montrerons la richesse de l'Ambiant à travers ses nombreuses applications.

I.1 INTRODUCTION GENERALE

Entre les années 2000 et 2010, nous avons pu voir se multiplier le nombre de terminaux mobiles. Tant dans leurs variétés que dans leurs puissances, ces terminaux ont fait un saut technologique spectaculaire et sont devenus quasiment omniprésents. Le paysage informatique subit actuellement une profonde mutation centrée sur la mobilité : celle des utilisateurs, des périphériques et des données.

Pour bien différencier la situation que nous connaissions encore il y a 10 ans, nous parlerons d'ORDINATEUR PERSONNEL (en anglais *personal computer*, ou PC) et d'INFORMATIQUE CONVENTIONNELLE lorsque nous nous référerons à l'ordinateur « classique » composé d'un écran, d'un clavier, d'une souris et d'une unité de calcul.

Bien qu'existant sous une forme portable, l'ordinateur personnel n'est pas considéré comme mobile pour des raisons de poids, d'autonomie, mais également d'usage. En effet, c'est l'aspect immédiat de l'accès à l'information qui caractérise l'informatique mobile dans ces usages. Les terminaux mobiles sont rarement éteints. Ils restent en veille, disponibles dans la seconde pour nous assister en un seul geste. A contrario, si vous décidiez de regarder la météo sur votre ordinateur portable, vous seriez accaparés par cette tâche pendant quelques minutes. Bien plus que des considérations techniques (poids, puissance, autonomie) qui évoluent très rapidement, c'est finalement l'usage qui distingue l'informatique mobile de l'informatique personnelle.

Ce changement dans les usages n'en est qu'à ses balbutiements et les recherches en informatique ambiante cherchent à anticiper, évaluer et contrôler cette évolution. Par la création de nouveaux périphériques et de nouveaux moyens d'interagir, l'informatique ambiante explore les liens entre l'Homme et l'information. Mais puisque la relation homme-machine devient mobile, elle explore également les liens entre l'Homme et son environnement et donc également, les liens entre l'information et l'environnement (cf. Figure 1).

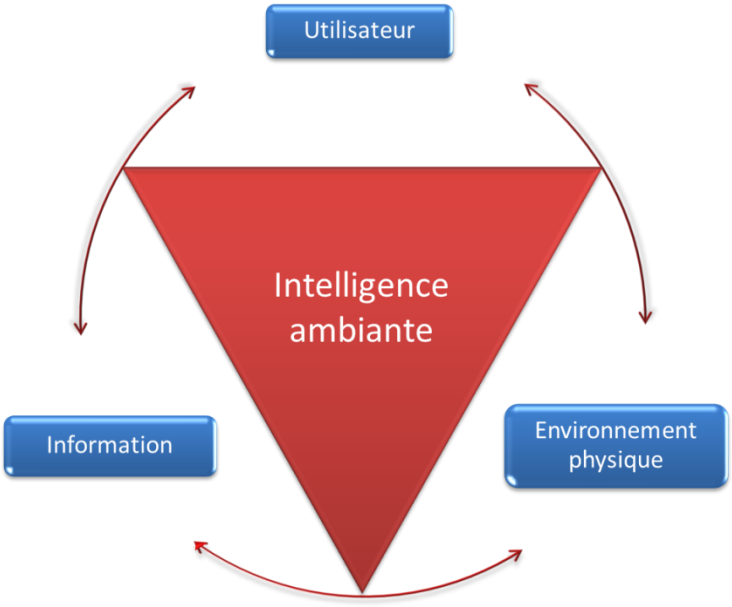


FIGURE 1 – L'INTELLIGENCE AMBIANTE ETUDIE LES LIENS ENTRE 3 ENTITES: L'UTILISATEUR, L'INFORMATION ET L'ENVIRONNEMENT

L'environnement dans lequel évolue l'utilisateur est rarement pris en compte dans l'IHM traditionnelle. C'est ce 3ème facteur qui caractérise et différencie l'informatique ambiante dans son approche de l'IHM, les deux premiers étant l'utilisateur et la tâche qu'il réalise. Car l'informatique ambiante ne se résume pas à l'informatique mobile telle qu'elle est accessible aujourd'hui. Si l'on extrapole les conjectures de Moore à propos de la miniaturisation des processeurs, on peut envisager un avenir proche dans lequel l'électronique sera intégrée dans tous les objets de la vie courante.

Selon l'importance que l'on donne aux différents facteurs entrant en jeu dans l'informatique ambiante, cette vision est déclinée en plusieurs thématiques que nous allons brièvement décrire.

I.1.1 L'INFORMATIQUE UBIQUITAIRE

L'INFORMATIQUE UBIQUITAIRE (Weiser 1993) (Kindberg & Fox 2002), également appelée informatique pervasive ou évanescente, se focalise sur la dissémination d'objets électroniques dans notre environnement au point que nous n'en aurions plus conscience. Ce domaine de recherche mise sur l'évolution des nanotechnologies et envisage la présence perpétuelle dans notre environnement quotidien, de capteurs discrets ou invisibles à l'œil nu.

En plus de ces capteurs, des périphériques pouvant agir sur l'environnement, appelés effecteurs, peuvent être intégrés. Un exemple de tels effecteurs est la commande numérique d'un volet électrique, ou bien encore la commande d'une climatisation.

Ces capteurs et ces effecteurs peuvent alors former un écosystème (Goumopoulos et al. 2011) auto-organisé ayant pour objectif d'adapter l'environnement aux besoins de l'utilisateur. Cette ubiquité pose cependant de nombreuses questions éthiques (cf. section II.3).

I.1.2 L'INTERNET DES OBJETS

L'INTERNET DES OBJETS (Van Kranenburg 2007) (Atzori, Iera & Morabito 2010) se focalise sur la conséquence la plus immédiate de cette ubiquité de l'informatique : chaque objet informatique ayant une représentation virtuelle, il découle de la vision ubiquitaire que les objets de la vie quotidienne auront une représentation virtuelle.

L'internet des objets pose donc la question de la nature de cette représentation, de ce qu'elle contiendra, mais aussi des moyens techniques de sa mise en œuvre. L'une des hypothèses fondatrices de l'internet des objets est de considérer que chaque objet sera accessible à tout instant par internet – grâce notamment au protocole d'adressage IPv6. On pourra ainsi connaitre son état et sa localisation à tout moment.

D'une certaine manière, dans l'internet des objets, l'objet physique et les informations qui lui sont associées ne font plus qu'un. Ce domaine se focalise surtout sur les conséquences sociétales et sur les changements d'usages impliqués par la cohabitation d'objets physiques de notre environnement avec leur représentation dans un monde virtuel (que celui-ci soit internet ou toute autre structure permettant le partage d'informations). Pour les entreprises, cela représente par exemple l'opportunité de mieux gérer ses stocks (Yan & Huang 2009). D'un point de vue social, l'usage de tels objets interconnectés à travers le web permettrait de mieux assister les personnes dépendantes. Il peut permettre aux personnes handicapées ou âgées de garder un contact social avec leur entourage et leur équipe soignante (Dohr et al. 2010). Il peut également aider dans le suivi purement médical du patient. En permettant la surveillance en temps réel des constantes vitales par exemple, des périphériques ont permis d'établir des profils personnalisés de patients diabétique (Jara, Zamora & Skarmeta 2011), ce qui a conduit à une meilleure adaptation de leur traitement.

I.1.3 SMART THINGS

Dans la continuité du thème de l'internet des objets, la notion de *smart things*, parfois traduite par objets intelligents ou bien objets augmentés, se focalise sur l'ajout de capacités d'auto-organisation et de coopération à des objets de la vie quotidienne. Grâce à l'augmentation d'autonomie énergétique des capteurs et unités de calculs embarquées, on peut envisager que les objets communiquent entre eux dans le but d'optimiser l'usage des ressources.

Cette vision s'applique à plusieurs catégories d'objets avec des objectifs différents. Ainsi, dans le cas des *smartbuilding* (Snoonian 2003) (Agarwal et al. 2010), l'objectif est d'utiliser la capacité d'auto-organisation des objets pour optimiser la consommation énergétique. Les *smarts furniture* représentent des appareils électroménagers augmentés. Ils concernent des objets au fonctionnement simple tels que des lampes qui détectent la présence des utilisateurs ou bien plus complexes tels que des réfrigérateurs intelligents capables de connaitre leur contenu et de réaliser automatiquement une liste de courses, voire de passer la commande de façon autonome. Enfin, les *smarts street furniture* ainsi que les *smart vehicles* visent à apporter une meilleure fiabilité, une meilleure sécurité, ainsi qu'un meilleur débit dans les modes de transports tant individuels que collectifs.

La notion de *smart things* est très proche de celle d'internet des objets car elle se focalise sur l'augmentation d'objets physiques avec des capacités digitales. Cependant, contrairement à l'internet des objets qui envisage une structuration à grande échelle, l'approche *smart things* se focalise plutôt sur l'auto-organisation et la communication des périphériques à une échelle locale.

Nous avons montré au travers de cette introduction, que l'intelligence ambiante recouvre différentes réalités. Chacune de ces réalités correspond à la mise en valeur d'un des facteurs de l'ambiant, mais toutes convergent sur un certain nombre de points communs. Afin d'éviter toute confusion dans la suite du document, nous présentons dans la section suivante, les termes que nous allons employer.

I.2 TERMINOLOGIE

Nous définissons, dans cette section, les termes employés dans le reste de ce document.

I.2.1 INTELLIGENCE AMBIANTE

Donner une définition claire et rigoureuse de l'intelligence ambiante n'est pas simple étant donné le large spectre de techniques et de théories que recouvre ce domaine. Plusieurs définitions ont été proposées. (Aarts & Ruyter 2009) positionne l'intelligence ambiante par rapport à l'informatique ubiquitaire et à l'informatique mobile en la différenciant par :

- la prise en compte du contexte et l'adaptation
- la personnalisation aux préférences de l'utilisateur
- la prise en compte de facteurs sociaux et une conscience de l'état courant de l'utilisateur

(Riva et al. 2005) place l'intelligence ambiante à l'intersection de l'informatique ubiquitaire, des systèmes de communication et des interfaces utilisateurs intelligent. L'Ambiant y est décrit[2] comme « le soutien efficace et transparent apporté à l'utilisateur dans la réalisation de ses activités grâce aux technologies de l'information et la communication. »

Nous retiendrons cette dernière définition qui caractérise l'Ambiant par l'efficacité et la transparence du soutien qu'elle apporte à l'utilisateur. C'est ce rapport à l'utilisateur qui fait de l'Ambiant un domaine plus riche que la simple réunion de ses parties. Ce rapport d'efficacité et de transparence se traduit par la vision d'un monde où l'ordinateur (ou plutôt l'informatique au sens large) parvient à s'intégrer dans le monde physique, et à participer à celui-ci. Le dictionnaire nous donne la définition suivante du mot ambiant :

« Qui constitue l'environnement où l'on vit, nous entoure de toutes parts. »

---

[2] *"AmI is the effective and transparent support to the activity of the subjects through the use of information and communication technologies".*

Cette définition justifie parfaitement l'emploi de l'expression **INFORMATIQUE AMBIANTE**. Ce domaine est donc en quelque sorte l'opposé de la réalité virtuelle[3] (Kalawsky 1993). Jusqu'alors, l'utilisateur d'un système informatique devait se plonger dans un environnement virtuel pour accéder à l'information. L'informatique ambiante nous fait la promesse de réaliser le chemin inverse, et d'intégrer l'information issue de traitements informatiques au sein même de notre monde physique. Cette intégration doit suivre une certaine logique pour être intuitivement et immédiatement saisissable par l'esprit humain. Elle doit être faite avec un certain degré d'intelligence. L'emploi de techniques issues de l'intelligence artificielle n'est donc pas rare dans le contexte de l'informatique ambiante, on parle alors d'Intelligence Ambiante.

Par la suite, nous emploierons le terme **AMBIANT** avec une majuscule pour traiter de l'Intelligence Ambiante.

I.2.2 ENVIRONNEMENT AMBIANT

Nous appellerons **ENVIRONNEMENT AMBIANT** tout espace physique et logique équipé d'un système logiciel et matériel visant à répondre aux objectifs et aux contraintes de l'Ambiant. Ainsi par exemple, une pièce équipée de capteurs et d'effecteurs, tous connectés au travers d'une plate-forme logicielle est un environnement ambiant.

Dans le reste de ce document, nous emploierons fréquemment le terme **ENVIRONNEMENT** en lieu et place d'environnement ambiant.

I.2.3 INFORMATIQUE CONVENTIONNELLE

Le système informatique de l'ambiant est déstructuré, il est distribué dans l'environnement au travers de plusieurs entités électroniques aux capacités variées. Il s'agit d'un changement de paradigme qui implique une mutation des architectures informatiques. Pour distinguer les architectures existantes jusqu'alors, nous emploierons le terme d'**INFORMATIQUE CONVENTIONNELLE**. Par opposition, les architectures sortant de ce cadre (et donc en particulier, celles dédiées à l'Ambiant) seront parfois qualifiées de **NON CONVENTIONNELLES**.

I.2.4 UNITE DE CALCUL ET PERIPHERIQUES

L'emploi du terme « ordinateur » s'étant généralisé, il recouvre diverses définitions. La plus couramment admise est celle correspondant à l'ordinateur de bureau, c'est-à-dire celle de l'ordinateur personnel (ou **PC**). Pour éviter toute ambigüité, nous nous efforcerons de ne pas utiliser ce terme en dehors de cette acception. La déstructuration de l'ordinateur en une multitude de composants électroniques qui collaborent justifie le retour à une terminologie qui différencie les différentes entités électroniques en présence.

Nous utilisons donc le terme « **UNITE DE CALCUL** » pour désigner les entités électroniques capables de réaliser des calculs et de communiquer entre elles à travers un réseau. L'ajout

[3] La réalité virtuelle est un domaine visant à immerger un utilisateur dans un environnement complètement virtuel en lui donnant la sensation que celui-ci est réel.

de capacités de communication à chacune des unités de calcul qui nous entoure est le fondement technologique de l'informatique ambiante.

A ces unités de calcul sont associés les **PERIPHERIQUES**. Ce sont des entités matérielles qui interagissent avec le monde physique et qui sont capables de communiquer avec une unité de calcul. Ces composants sont incapables de raisonner ou de communiquer à travers un réseau par eux-mêmes. Ils sont donc liés à des unités de calcul qui les contrôle. L'ambiant se caractérise par l'association d'unités de calcul de plus en plus petites et puissantes à des périphériques qui sont enfouis dans le monde physique. Cet enfouissement et cette distribution de la capacité de raisonnement du système permettent l'émergence d'un comportement qualifié d'adapté, voire d'intelligent.

La catégorie la plus répandue de périphériques est celle des **PERIPHERIQUES D'INTERACTION**, qui permettent à l'Homme d'interagir explicitement avec le système informatique. Il en existe cependant d'autres associées à la domotique, par exemple, qui permettent de capturer le contexte (capteur de luminosité par exemple) ou de modifier l'environnement (air conditionné par exemple).

I.2.5 CONCLUSION

Cet ensemble de définitions a l'avantage de recouvrir la totalité des types « d'ordinateurs » qui ont existé : depuis le supercalculateur partagé par une multitude d'utilisateurs dans les années 60 jusqu'au plus simple des capteurs miniaturisés qui soit apparu ces dernières années en passant par l'ordinateur personnel, le téléphone portable ou la tablette tactile. A partir de cette définition, on peut observer l'évolution des usages informatiques en fonction du ratio du nombre d'utilisateurs par unité de calcul. On distingue généralement 3 étapes principales : la première fut celle des centres de calcul dans lequel un unique ordinateur était partagé par plusieurs utilisateurs (ratio inférieur à 1) ; puis, vint l'ère de l'ordinateur personnel (ratio égal à 1) et enfin, l'émergence de l'Ambiant dans lequel un ensemble d'unités de calculs concourent au bien-être d'un ou plusieurs utilisateurs (ratio supérieur à 1). Bien entendu, en étudiant d'autres facteurs tels que l'ubiquité des systèmes informatiques, des étapes intermédiaires peuvent se dégager comme l'illustre la Figure 2. Il n'existe pas réellement de consensus sur les années charnières de ces étapes, tout particulièrement parce que celles-ci dépendent du fait que l'on se place du côté du chercheur (c'est le cas dans la Figure 2, ce qui explique que pour Waldner, l'intelligence ambiante soit déjà une réalité) ou bien du côté de l'utilisateur (pour lequel l'informatique ambiante n'existe pas encore).

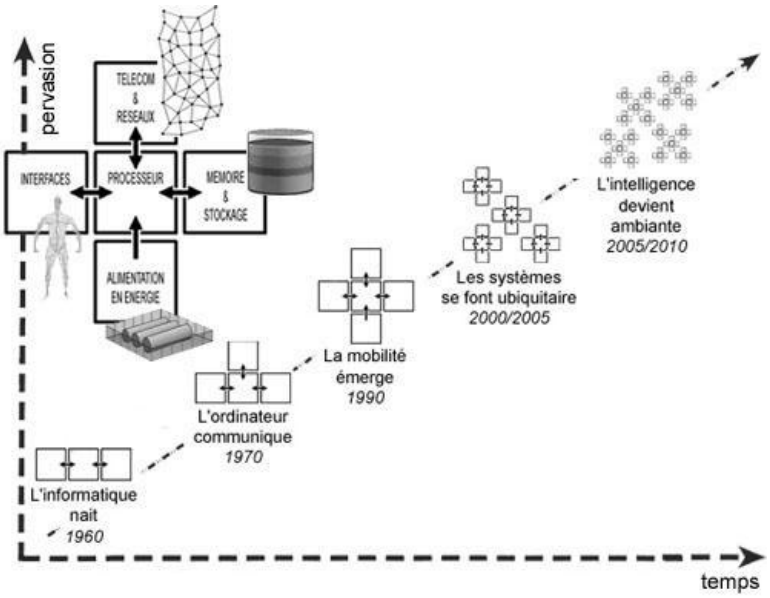


FIGURE 2 – DE L'INFORMATIQUE CENTRALE A L'INTELLIGENCE AMBIANTE, EXTRAIT DE (WALDNER 2007)

I.3 APPLICATIONS

Afin de justifier les problématiques soulevées par l'Ambiant, nous donnons un aperçu, dans cette section, de la variété de situations que ce domaine s'emploie à couvrir. Nous distinguons ces applications en fonction du soutien qu'elles apportent à l'utilisateur dans ses activités quotidiennes. Nous distinguons 4 principaux domaines d'application : l'informatique mobile, les activités effectuées à la maison, le domaine médical et la gestion du handicap.

I.3.1 L'AMBIANT EN SITUATION DE MOBILITE

Il est difficile d'imaginer un environnement qui soit Ambiant de façon générique. Il y a une infinité de déclinaisons possibles aux formes que peut prendre l'Ambiant en situation mobile, ces formes étant parfois incompatibles entre elles. Les applications proposés jusqu'à maintenant sont donc ciblées pour un type d'environnement donné, dans un secteur donné. Parmi ceux-ci, nous illustrerons ces applications dans le secteur commercial et dans le secteur culturel.

Dans le secteur commercial, l'intelligence ambiante peut se traduire par la mise en œuvre de systèmes de recommandation ubiquitaires, capable d'adapter le contexte du magasin en fonction des utilisateurs qui s'y trouvent et de leur localisation. Par exemple, dans (Keegan, O'Hare & O'Grady 2008), un système multi-agent est utilisé pour négocier automatiquement les achats de l'utilisateur. Ce dernier porte sur lui un PDA qui lui permet d'énumérer les objets qu'il désire acheter. Lors de ses déambulations dans le centre commercial, le PDA se charge de négocier automatiquement avec les commerces à proximité et de présenter les résultats de ces négociations à l'utilisateur qui conserve le pouvoir final d'effectuer l'achat ou non. Ce type d'interfaces permet également de proposer des produits similaires ou de suggérer des produits que l'utilisateur pourrait

souhaiter acquérir. Bien sûr, ces systèmes impliquent l'emploi de profils utilisateurs, qui en plus de poser des problèmes éthiques, sont difficiles à acquérir. Les clients ne souhaitent pas avoir à saisir leurs préférences manuellement pour chaque magasin qu'ils traversent. C'est pourquoi ces systèmes doivent être capables d'apprendre à partir des comportements des clients, et d'adapter un profil acquis dans un certain domaine pour un autre domaine (passer de gouts littéraires à des gouts musicaux par exemple). Dans (González et al. 2005), un modèle de l'utilisateur appelé *Smart User Model* est proposé pour résoudre ce problème en s'appuyant sur des méthodes d'apprentissage à base de noyaux (algorithmes de machines à vecteur support). Ce modèle a été mis en œuvre avec succès pour des systèmes de recommandation trans-domaines.

Ce type d'approches ne fait évidemment pas l'unanimité et pose notamment la question de la transparence des données acquises. Mais il est bien mieux accepté, et probablement plus justifié, lorsqu'il est appliqué à des secteurs éducatifs et culturels. Ainsi, dans le projet PEACH (Kuflik et al. 2004) (Stock et al. 2007), le profil utilisateur est appliqué pour générer un guide personnalisé lors de la visite de musées. Ce guide s'appuie sur les préférences utilisateurs ainsi que sur les données recueillies lors d'éventuels précédents parcours pour construire un parcours qui soit attractif. Un PDA est utilisé pour guider le visiteur tandis que les informations plus détaillées sur les œuvres sont fournies aux travers d'écrans et de haut-parleurs disséminés dans le musée. A l'approche du visiteur, des informations personnalisées lui sont proposées. Chaque utilisateur vit ainsi une expérience personnelle qui lui est destinée et qui peut varier d'une visite à l'autre. Un système similaire a été proposé, dans (Petersen & Kofod-Petersen 2006), pour la découverte interactive d'une ville, réalisant ainsi l'un des scénarios mis en avant par le groupe européen de conseil en nouvelles technologies ISTAG (Ducatel et al. 2001).

La difficulté de ces approches est de fournir des applications très spécifiques pour un secteur donné. Elles sont difficilement généralisables hors des limites de leur secteur d'application. Il existe cependant un domaine récurrent de l'Ambiant pour lequel des systèmes peuvent être conçus dans une approche plus globale et donc qui peuvent être comparés entre eux : celui de l'Ambiant à domicile.

I.3.2 L'AMBIANT DANS LES FOYERS

L'intelligence ambiante appliquée à la vie quotidienne d'un foyer est appelée *Smart Home*, maison intelligente ou bien encore maison interactive. Son objectif est d'appliquer les techniques d'intelligence ambiante pour améliorer le confort et la qualité de vie au domicile. En ce sens, elle se situe dans la continuité des innovations concernant nos modes de vie depuis l'émergence du marché de l'électroménager de ces 50 dernières années.

Cependant, la maison intelligente ne se résume pas au fantasme d'une automatisation des tâches ménagères. Ce concept va plus loin que la domotique. Il aborde le problème de la disparition des frontières entre le travail et le foyer, le repos et le divertissement. Les travaux concernant la migration de tâches (Sousa & Garlan 2002) (Pyla, Tungare & Pérez-Quinones 2006) en sont un bon exemple. Ils permettent de commencer une tâche à un endroit et de la poursuivre plus tard chez soi. Les travaux sur la collaboration à distance au travail (Elrod et al. 1992) (Hornecker & Buur 2006) peuvent également être appliqués dans le domaine privé pour maintenir un contact tangible avec la famille ou les amis. Le

sentiment d'être chez soi, dans sa sphère privée, peut donc être perçu différemment à mesure que les tâches deviennent mobiles. Bien que ce ne soit pas son ambition initiale, la maison intelligente s'annonce donc comme un changement profond de nos modes de vie.

Les objectifs de la maison intelligente sont :

- Amélioration du confort (Gallissot 2012) ressenti et de la qualité de vie

  Adaptation de l'environnement (température, luminosité…) aux envies de l'utilisateur mais aussi adaptation de l'ambiance (musique par exemple) à l'état affectif de l'habitant. L'Ambiant peut également être utilisé pour renforcer le lien social en mettant à disposition de façon transparente les technologies de communication à distance et en étudiant leur impact sur le sentiment de présence ressenti face à des représentations virtuelles du système (Tan et al. 2011) ou de personnes réelles.

- Automatisation des tâches de maintenance et d'entretien

  Il s'agit là des objectifs classiques de la simple domotique.

- Optimisation de la consommation énergétique

  Grâce à l'utilisation de systèmes de raisonnement automatique, les capteurs et effecteurs de la domotique peuvent permettre de prévoir et d'adapter la consommation énergétique, ce qui permet de faire des économies au niveau local et au niveau global (en équilibrant mieux les ressources du réseau).

- Amélioration de la sécurité

  Cela ne se limite pas aux alarmes et caméras antivol mais s'étend à la prévention d'accidents (chutes d'enfants ou de personnes âgées), à la réaction lorsque ces accidents se produisent, voire même au suivi médical des occupants.

- Maintien de personnes âgées ou handicapées à domicile

  Les capteurs et effecteurs de l'ambiant permettent de surveiller l'état d'un patient à distance, de détecter les situations à risque et de maintenir un lien permanent avec le patient souffrant de perte d'autonomie. Cette perte peut être mieux accompagnée à domicile, avec un meilleur confort physique et psychologique, dans le cadre de l'Ambiant.

En l'état actuel des connaissances cependant, le concept de *Smart Home* reste encore ancré dans le suivi et l'assistance de l'utilisateur et dans la prévention des risques. La maison intelligente fait intervenir de nouveaux périphériques (des capteurs et des effecteurs). Les premiers périphériques issus de la domotique (détecteur de fumée, sonde de température) sont vite devenus insuffisants et de nombreuses études se sont focalisées sur la création de nouveaux matériels pour mieux percevoir l'environnement. Mais dès lors, le besoin d'extraire de l'information à partir de ces capteurs est apparu. Au-delà du traitement du signal proposé dans la domotique, c'est l'ingénierie de la connaissance et l'intelligence artificielle qui permettent de donner du sens aux données perçues. Deux voies de recherche se sont alors ouvertes. La première est celle de l'apprentissage des préférences de l'utilisateur, dans le but de silencieusement adapter son environnement à son besoin de confort. Ainsi, dans (Hagras & Wagner 2011), la logique floue est utilisée pour apprendre les préférences en termes de luminosité et de température et pour restituer ces préférences dans un autre contexte (autre lieu ou autre moment). La

possibilité de mélanger ces préférences pour atteindre un compromis acceptable par plusieurs utilisateurs est également étudiée dans (Wagner & Hagras 2010). La seconde voie est celle de la reconnaissance de certains motifs dans les habitudes de l'utilisateur ou bien dans les situations à risque. La reconnaissance de ces motifs (ou bien la reconnaissance d'un motif inhabituel) peut être utilisée pour déclencher une action. Ainsi, la reconnaissance d'un risque d'incendie peut déclencher une alarme ou l'envoi d'un message aux pompiers (Augusto 2007) détaillant les circonstances telles que le nombre d'occupants en danger. De même le non-suivi d'une routine comme la prise quotidienne d'un médicament pourra générer un rappel à l'utilisateur (Becker et al. 2009). De par sa variété de situations et sa complexité, la maison intelligente propose un défi qui ne manquera pas de contribuer à l'innovation dans le domaine de l'intelligence artificielle (Augusto & Nugent 2006).

A l'aube de l'intelligence ambiante, la profusion de capteurs issus de la domotique permettant de contrôler les risques (incendie, monoxyde de carbone, chutes) a orienté les applications actuelles de la *Smart Home* vers le maintien à domicile de personnes en perte d'autonomie grâce à l'Ambiant, créant ainsi une confusion avec la notion d'*Ambient Assisted Living* (**AAL**). Nous avons cependant montré dans cette section que l'ambition de la Smart Home était plus générale et touchait à nos modes de vie. Nous allons maintenant étudier les spécificités de l'AAL et plus généralement, de l'usage de l'Ambiant dans le domaine médical.

I.3.3 L'AMBIANT DANS LE DOMAINE MEDICAL

Comme nous l'avons vu dans la section ci-dessus, l'application des concepts de l'Ambiant au domicile des personnes rend possible leur maintien à domicile même lorsqu'elles présentent une perte d'autonomie. Dans ce cadre, l'Intelligence ambiante vise avant tout à augmenter l'environnement pour recueillir des informations sur le patient. Ces informations sont intéressantes pour le personnel soignant car elles sont acquises dans le milieu quotidien du patient. Elles ne sont pas biaisées par la structure aseptisée de l'hôpital et s'étendent sur de plus longues durées. De cette façon, les médecins sont mieux à même de connaître les conditions dans lesquelles la maladie se déclare (présence d'allergènes, conditions de température, moment de la journée…), voire de repérer et prévenir des comportements dangereux (incompréhension du mode de prise du médicament, oubli ou non-respect de conditions d'hygiènes propres à certains traitements…).

L'accent dans ce type de recherches est mis sur la prévention (Haux 2006). L'intelligence ambiante apporte donc un meilleur environnement pour le diagnostic, le traitement, et la détection au plus tôt des affections graves ou chroniques. Dans (Jara, Zamora & Skarmeta 2011) par exemple, des patients diabétiques sont équipés de capteurs permettant d'analyser, à intervalles réguliers, leur taux de glycémie. Ces capteurs sont augmentés d'un système d'analyse des résultats qui permet d'évaluer la quantité d'insuline à prendre en tenant compte des habitudes de l'utilisateur. Il est de plus connecté avec les praticiens encadrant la prise d'insuline qui peuvent ainsi observer le comportement du patient et l'adéquation de son traitement à ses conditions de vie réelles.

L'espoir que portent ces techniques de surveillance médicale est d'améliorer la qualité de vie des personnes qui doivent à l'heure actuelle rester sous surveillance à l'hôpital et qui pourraient, demain, rester chez elle tout en continuant de surveiller leur santé.

I.3.4 L'INFORMATIQUE UNIVERSELLEMENT ACCESSIBLE

Le terme d'**ACCES UNIVERSEL** (ou *universal access* dans les pays anglo-saxons) correspond initialement à un ensemble de mesures prises par les politiques publiques pour améliorer l'accessibilité de leurs services. Partant du constat que tous ne pouvaient accéder aux services publics de façon égalitaire, un effort était nécessaire pour apporter la même qualité de service à toute la population. Cette prise de conscience a suscité une vague de normes pour l'accessibilité dans les pays occidentaux.

Au fur et à mesure que l'informatique devient ubiquitaire, il est devenu indispensable de la rendre accessible à tous et donc de lui appliquer les principes de l'accès universel. Dans les situations de mobilité, les utilisateurs moyens se retrouvent avec des capacités d'interaction limitées qui se rapprochent de celles de l'accès au service pour une personne handicapée (Carbonell 2006). On peut citer l'exemple de l'envoi de messages sur un téléphone portable : l'appareil étant dépourvu de clavier fonctionnel, l'utilisateur doit être assisté par des logiciels (correcteurs automatiques, complétion automatique…) pour la rédaction de message. Un autre exemple est celui du guidage GPS. La vision et l'attention de l'utilisateur étant accaparées par la conduite, le système se retrouve dans des conditions d'interaction similaires à l'interaction avec un malvoyant. Il doit alors utiliser d'autres techniques d'interaction telles que la parole et l'usage d'avertisseurs sonores pour alerter l'utilisateur de la présence d'une information urgente.

La problématique de l'*universal access* ne se limite donc pas à l'accès aux services par toutes les personnes quelles que soient leurs capacités physiques et cognitives, mais également à l'accès à ces services dans tous types de conditions (Stephanidis & Savidis 2001). Ce renversement de la problématique initiale de l'informatique ambiante pourrait même être, d'après (Carbonell 2006), l'un des moteurs de l'ouverture de l'informatique aux personnes handicapées.

I.4 CONCLUSION

En conclusion, l'intelligence ambiante est un domaine au carrefour des technologies de l'information, de la communication et de l'informatique mobile. Son objectif est d'intégrer l'informatique dans le monde réel au lieu d'impliquer l'utilisateur dans une interaction virtuelle qui le coupe de son environnement. Il se caractérise par l'usage de capteurs et d'effecteurs pour interagir avec l'environnement, ainsi que la dissémination d'unités de calcul miniaturisées organisées en réseau.

Après avoir introduit ces concepts, nous avons défini plus précisément les termes employés tout au long de cet ouvrage. Puis nous avons étudié l'impact que l'Ambiant pourrait avoir sur notre vie quotidienne en présentant ses applications. Nous avons identifié 4 domaines d'application : l'aide à l'utilisateur en situation de mobilité, le soutien dans les activités de

la vie quotidienne à domicile, le suivi de patients dans le domaine médical et enfin, l'accès universel aux services (informatiques ou autres) pour les utilisateurs handicapés.

Ces exemples d'application nous montrent que l'Ambiant se focalise sur une utilisation adaptée au contexte et intuitive pour l'utilisateur. Cette contrainte d'adaptation et ce besoin de transparence sont des problématiques qui ont des répercussions sur le plan de la conception des interfaces hommes-machines. Nous étudions ces répercussions dans le chapitre suivant.

Chapitre II
PROBLEMATIQUES ASSOCIEES A L'AMBIANT

Nous étudions, dans ce chapitre, les problématiques que pose l'Ambiant dans le domaine de la conception d'IHM. Nous commençons par présenter les contraintes technologiques imposées par l'Ambiant. En étudiant ces contraintes, nous identifions plusieurs conséquences pour l'IHM. Ces conséquences sont de l'ordre de la conception et de l'implémentation mais ne tiennent pas compte des facteurs sociaux. Bien que nos travaux ne se concentrent pas sur ces facteurs sociaux, nous présenterons cependant, en troisième section, quelques questions éthiques qui auront un impact sur l'approche que nous proposons dans cette thèse.

II.1 CONFIGURATION TECHNOLOGIQUE

L'Ambiant se distingue de l'informatique traditionnelle essentiellement par sa nature distribuée et son immersion dans un contexte physique qui est difficile à capturer. Ces deux barrières technologiques ont été étudiées, parfois même en dehors du contexte de l'informatique ambiante. Nous allons présenter dans cette section les outils qui ont été développés pour pallier ces problèmes techniques.

II.1.1 ARCHITECTURES DISTRIBUEES : ROLES DES COMPOSANTS ET DES SERVICES

En génie logiciel, les architectures orientées services (**SOA** pour *Service-Oriented Architecture*) sont une extension de la programmation orientée composant (**POC**). Cette dernière insiste sur la séparation des rôles des différentes entités composant un logiciel. Dans cette approche, le logiciel est décomposé en différents blocs de fonctionnalités appelés modules qui sont rendus les plus indépendants possible. Ce découplage entre les différentes briques logicielles est assuré par une modélisation de celle-ci sous forme de boites noires implémentant un certain contrat (on parle aussi d'interface). Chaque module n'a accès qu'au contrat des autres modules, sans pouvoir faire d'assertion quant à la façon dont celui-ci est implémenté. Cette technique de découpage du logiciel à plusieurs avantages, notamment celui de permettre à plusieurs équipes de travailler de façon plus ou moins indépendante par le respect de ces contrats, ainsi que la possibilité de réutiliser plus simplement certaines briques logicielles. L'approche POC a été facilitée et popularisée par l'introduction de la notion d'objet dans les langages de programmation.

L'approche SOA s'appuie sur les notions de base de l'approche POC. Elle en reprend les concepts de brique logicielle réutilisable définie par un contrat. Cependant, elle s'applique dans le contexte d'une distribution des unités de calcul à travers un réseau. En effet, la programmation orientée service consiste en l'implémentation de modules logiciels qui exposent leur contrat sur un réseau et dont les fonctionnalités peuvent être invoquées à distance. Les modules sont alors appelés services. L'approche SOA impose donc un cadre formel à la rédaction du contrat liant les services et permet la distribution du calcul sur plusieurs unités indépendantes. Elle diffère de l'approche POC selon plusieurs critères :

- L'absence de mémoire partagée entre les modules et donc le recours récurrent à la sérialisation des données. Puisque les services sont hébergés sur des hôtes différents, ils ne peuvent partager de l'information que par envoi de message. Plusieurs protocoles sont utilisés pour cela, le plus célèbre étant SOAP (*Simple Object Access Protocol*).
- Le service est « ouvert sur le monde ». Il expose son interface sur le réseau, permettant ainsi à un ensemble de machines d'en invoquer les fonctionnalités. Les formats utilisés pour spécifier ces contrats sont généralement dérivés de XML (c'est le cas, par exemple, de WSDL – *Web Service Description Language*).
- Les calculs peuvent se faire en asynchrone, de façon distribuée. Cependant, POC et SOA convergent de plus en plus sur ce point, au fur et à mesure de l'évolution des processeurs multi-cœurs.

Lorsque ces services utilisent des protocoles reposant sur les technologies du web (HTTP et REST par exemple), on les appelle **SERVICES WEB**. L'ubiquité du web en a fait un bon candidat comme substrat fondamental de la réalisation de l'approche SOA. Ainsi, les protocoles UPnP (*Universal Plug and Play*) et DPWS (*Device Profile for Web Services*) ont été créés dans le but d'unifier et de standardiser l'approche orientée service au-dessus d'une couche IP.

De par sa nature, cette structure colle parfaitement avec la distribution des fonctionnalités telle qu'on la retrouve dans l'Ambiant. Le problème de la distribution dans l'informatique ambiante a donc rapidement été étudié sous l'angle SOA. Grâce à ces formalismes, un certain nombre de problèmes liés à la distributivité dans l'Ambiant (Cervantes & Hall 2003) ont pu être identifiés. Par exemple, la notion de service introduit celle de cycle de vie. En informatique conventionnelle, les fonctionnalités sont à tout moment disponibles dans la mémoire (au besoin en chargeant une librairie dynamiquement). Les services au contraire, peuvent apparaitre et disparaitre du réseau à tout instant, au gré des changements d'état de leur hôte. Les logiciels s'appuyant sur des services doivent donc tenir compte de ce risque d'indisponibilité. Ils doivent être capables de s'adapter en choisissant à tout instant le service qui est le plus adapté au contexte. Si celui-ci vient à disparaître, ils doivent dynamiquement s'adapter en en utilisant un autre (Cervantes & Hall 2003). Ce problème de la **COMPOSITION DYNAMIQUE** des services est au cœur des architectures logicielles proposées pour l'informatique ambiante.

Cinq axes pour la composition de services ont retenu notre attention :

- **LA PROGRAMMATION PAR FLUX**. Le programmeur/concepteur peut spécifier dans un langage particulier comment les services doivent être composés pour en générer de nouveaux. Dans ce cas, le système requiert peu de complexité puisqu'il se contente d'appliquer un flot d'instructions généralement modélisée sous la forme d'un graphe tel que ceux du langage BPEL (*Business Process Execution Language*). Dans BPEL, les services sont composés en fonction de leurs types de donnée (i.e. en fonction de leur contrat), ce qui laisse peu de place à l'adaptation dynamique (un service similaire mais ayant une interface très légèrement différente ne pourra être utilisé en alternative). Cependant, des langages dérivés tels qu'ATRACO-BPEL (Goumopoulos et al. 2011) explorent la possibilité d'une description et d'une

composition des services basée sur leurs descriptions sémantiques et non sur leur contrats uniquement.

- **LA COMPOSITION PAR DEPENDANCE DE SERVICES**. C'est la méthode la plus immédiate pour automatiser la composition. Il s'agit, pour chaque service, de définir son graphe de dépendance, c'est-à-dire l'ensemble des services dont il dépend pour fonctionner. Une entité, appelée *Service Binder* (Cervantes & Hall 2003) a alors pour rôle de trouver sur le réseau ces services. Dans le cas où plusieurs services candidats pourraient être utilisés, le système doit faire un choix. Ce choix dépendra généralement du contexte (proximité physique de la ressource) ou de mesures de qualité de service (temps de réponse, bande passante…).
- **LA COMPOSITION DERIVEE DU MODELE DE TACHE**. (Rigole et al. 2007) introduit la notion de modèle de composition. Ce dernier est l'entité qui spécifie comment les compositions doivent être construites. Dans les autres approches, ce modèle de composition est statique ; un fichier WS-BPEL réalise par exemple l'un de ces modèles. Dans l'approche de (Rigole et al. 2007), un modèle de composition est défini pour chaque tâche que l'on souhaite réaliser. La création de ce modèle de composition repose sur une modélisation de la tâche (voir section IV.4.2.i). En associant chaque sous-tâche élémentaire à un service et en analysant les relations temporelles entre les tâches, la création d'un flux d'exécution de ces services devient possible.
- **LA COMPOSITION DERIVEE DES BESOINS UTILISATEURS**. Lorsque plusieurs services du même type (i.e. réalisant le même contrat) peuvent être utilisés, on se repose généralement sur le contexte pour faire un choix. Parmi les informations du contexte, la position de l'utilisateur est une information fréquemment utilisée car elle traduit les limites physiques des capacités d'interaction de celui-ci. Cependant, les préférences utilisateurs sont plus difficiles à utiliser car il est impossible d'établir des correspondances avec les spécifications des services (les domaines sont disjoints). Pour pallier cette limite, (Lopez-Velasco et al. 2005) propose d'étendre le domaine des langages de spécification de services. Elle propose une extension de WSDL intitulée AWSDL permettant de spécifier des informations contextuelles sur le service qui pourraient concerner l'utilisateur (langues disponibles, plateformes d'interaction…). Ainsi, le service peut être sélectionné par filtrage des préférences de l'utilisateur.
- **LA COMPOSITION AVEC DROITS D'ACCES**. Ce dernier point n'est pas réellement une méthode de composition mais plutôt une problématique transverse à toutes les autres méthodes : comment garantir un accès sécurisé aux services ? Comme nous l'avons détaillé plus haut, les services sont ouverts sur le monde, contrairement aux modules logiciels qui sont chargés dans la mémoire locale d'un processus. Par conséquent, le problème de la gestion des droits d'accès se pose. Quelle unité de calcul peut accéder à un service ? Au nom de quel utilisateur ? Et comment crypter et sécuriser ces échanges d'information. Il est démontré dans (Nguyen 2005) que les méthodes traditionnellement appliquées en gestion des droits d'accèse ne sont pas adaptées à l'ambiant car elles sont trop rigides et parce que la notion de privilèges accordés intemporellement à un utilisateur ne correspond pas à la réalité des usages. L'auteur propose donc un modèle de droit d'accès qui sont calculés dynamiquement en fonction de la tâche en cours et des rôles associés à l'utilisateur

dans cette activité. Les permissions sont attribuées de façon statique pour certains rôles dans certaines tâches. A chaque session d'une tâche, l'utilisateur se verra dynamiquement allouer les droits associés à son rôle.

Nous avons montré dans cette section que la distribution des fonctionnalités et des unités de calcul dans l'Ambiant présente de réels défis de génie logiciel. Ces problématiques ont été étudiées sous l'angle d'un nouveau paradigme de programmation : la programmation orientée services. Cette forme de programmation permet de structurer et de formaliser les problématiques liées à la distribution. Nous avons donc présenté plusieurs solutions tirant parti de la composition de ces services indépendants pour générer des entités logiques éphémères qui répondent aux besoins du système et de l'utilisateur. Nous pouvons remarquer que ces solutions reposent toutes sur une formalisation plus ou moins poussée des services disponibles, des tâches, voire même des utilisateurs. Le recrutement (c'est-à-dire la sélection et l'assemblage) de ces services passe également par l'usage d'informations contextuelles. Nous montrons dans la section suivante que le recueil de ces informations est par nature imprécis.

II.1.2 MATERIEL ET ACQUISITION DU CONTEXTE

Le problème de l'acquisition du contexte n'est pas propre à l'Ambiant, il se retrouve dans toutes les applications où un système informatique est immergé dans le monde réel et en interaction avec celui-ci. C'est le cas par exemple de l'informatique embarquée et de la robotique. On retrouve ainsi deux types de problématiques technologiques: celles liées au capteur et celles liées à l'association en réseaux de ces derniers.

La principale difficulté liée aux capteurs est leur marge d'erreur. Il est impossible d'acquérir les données avec une infinie précision et le bruit de l'acquisition va inévitablement se propager dans les couches supérieures du programme. Cette imprécision est propre à chaque capteur et immuable. Il faut donc apprendre à faire avec, soit en traitant l'information comme incertaine, soit en augmentant le nombre de capteurs et en fusionnant leurs données. Des algorithmes de filtrage peuvent également être appliqués (par exemple, le filtre de Kalman) pour adoucir les erreurs ponctuelles. Le second problème lié au capteur est l'autonomie. Comme le précise (Estrin et al. 2002), les capteurs dans l'ambiant sont voués à être nombreux et disséminés dans des endroits parfois difficiles d'accès. Ces contraintes laissent peu d'opportunité à une maintenance humaine, il faut donc que les capteurs soient autonomes. Or s'ils sont disséminés dans des endroits difficiles d'accès, il est fréquent qu'ils ne disposent pas de source d'alimentation ni de connexion à un réseau câblé. De tels capteurs doivent donc pouvoir communiquer sans-fil et produire leur propre énergie. La seule technologie actuellement disponible pour cela est l'énergie solaire dont l'usage, à cause de son faible rendement actuel, est réservé à des systèmes très peu énergivores. Enfin, le troisième frein technologique lié aux capteurs dans l'Ambiant est la dynamique du contexte. Si certains capteurs peuvent se contenter d'une fréquence d'acquisition faible (sonde de température, par exemple), la plupart devront fournir en temps réel au système un compte-rendu détaillé de la situation. Il est par exemple impensable, sur une chaine d'assemblage industriel, que le système de suivi de position ait des temps de réponse supérieurs à la demi-seconde. Or cette contrainte de taux de rafraichissement élevé va à l'encontre du besoin d'autonomie car il engendre une

surconsommation importante. Il faudra donc encore de nombreux efforts pour arriver à produire des capteurs qui allient autonomie, fiabilité et temps de réponses faibles.

Quand bien même de tels capteurs arriveraient sur le marché, la barrière technologique ne serait pas pour autant franchie. En effet, la seconde difficulté dans l'acquisition du contexte consiste à faire communiquer un réseau de capteurs (Estrin et al. 2002). Si la multiplication du nombre de capteurs est une étape indispensable à l'acquisition fiable des informations du contexte, elle a un impact sur les performances du réseau que ces capteurs utilisent. Ces problèmes de performances liés à la communication sont évoqués plus en détail dans (Macedonia & Zyda 1997). Il y est montré à travers de l'exemple UDP/TCP que la fiabilité des communications s'obtient au détriment de la rapidité de celle-ci. Le bon compromis entre fiabilité et bande passante doit être trouvé pour chaque application. Une taxonomie des réseaux de capteurs est donc proposée dans (Estrin et al. 2002).

Ces difficultés matérielles ont de lourds impacts sur la réalisation de prototypes pour l'Ambiant. Elles impliquent une intervention humaine pour l'étalonnage et la maintenance des capteurs. Dans les cas extrêmes de capteurs disséminés sur de grandes étendues, comme c'est le cas pour la prévention d'incendies en forêt (Yu, Wang & Meng 2005), le manque d'autonomie pourrait mener à une pollution de capteurs devenus obsolètes (Estrin et al. 2002). Par ailleurs, (Dey, Abowd & Salber 2001) montrent qu'à cause de leurs limites actuelles, le choix des capteurs conditionne fortement les applications qui sont généralement peu flexibles. Il conditionne également notre représentation mentale (en tant qu'utilisateur, chercheur ou concepteur) du contexte, ce qui bride la créativité de ces applications.

II.2 Consequences pour l'IHM

II.2.1 Des taches variees et ephemeres : une cible mouvante pour l'IHM

La première conséquence de la modularité apportée par le développement orienté services est que les applications sont conçues à la volée dans l'Ambiant, en fonction des services disponibles à un instant donné dans un lieu donné. La non prédictibilité des fonctionnalités disponibles rend la conception des tâches ardue, voire imprévisible. Deux approches sont possibles : générer des plans abstraits qui sont concrétisés à l'exécution ou composer dynamiquement des actions élémentaires pour générer un plan.

La première approche est réalisée dans (Mokhtar, Georgantas & Issarny 2005). Dans ces travaux, l'utilisateur est associé à un ensemble de tâches abstraites prédéfinies qui le concernent. Ces descriptions abstraites de tâches sont raffinées en fonction des services disponibles dans l'environnement pour créer une tâche concrète associée à certaines des fonctionnalités présentes. Le jeu de fonctionnalités choisies peut donc varier d'un environnement à l'autre.

La seconde approche est réalisée dans MAPPING (Gabillon, Calvary & Fiorino 2008). Il s'agit d'un démonstrateur appliquant des méthodes de planification pour générer, à l'exécution, des tâches interactives dans un environnement ambiant. Dans cette approche, l'objectif

utilisateur est formulé en langage naturel dans une IHM spécifique. Elle donne lieu au calcul de différents plans, traduits en tâches à suivre pour atteindre l'objectif recherché. De ces plans sont dérivés un noyau fonctionnel et une IHM conforme aux ressources disponibles.

Ces différentes approches proposent de construire le service dynamiquement mais ne résolvent pas la question de la conception de l'IHM qui l'accompagne. Que l'on se situe au niveau de la tâche ou au niveau des fonctionnalités (c'est-à-dire du noyau fonctionnel), il y a irrémédiablement besoin de générer automatiquement une interface adaptée à l'application (dans son sens le plus large). Ce problème s'accentue encore si l'on considère que les services constituant une application ont un cycle de vie indépendant de l'application. Il s'agit là d'un véritable changement de paradigme concernant la conception d'IHM, car on ne peut prévoir au moment de la conception l'ensemble des fonctionnalités et des tâches disponibles, ni garantir la continuité de service dans le temps. Ce problème n'est pas pris en compte par les architectures conventionnelles (cf. section IV.1.3), mais des méthodes non conventionnelles l'abordent (cf. Chapitre IV).

II.2.2 DISTRIBUTION DES COMPOSANTS D'INTERACTION

En situation de mobilité, la liste des périphériques qui sont disponibles peut évoluer au cours du temps. Ainsi le système doit permettre de détecter ces changements de la configuration matérielle et d'adapter l'interface homme-machine en conséquence.

Cela pose une première question : sur quels critères les périphériques doivent-ils être incorporés ou non à une IHM ? Il est par exemple évident qu'un écran qui se trouve dans une pièce ne pourra pas être utilisé pour interagir avec un utilisateur situé dans une autre pièce. De plus, le système ubiquitaire peut ne pas avoir de frontières clairement établies. Dans un monde où chaque objet peut être augmenté de capacités de communication, quelles sont les frontières d'un système interactif ? Doit-on se limiter à la pièce dans laquelle se situe l'utilisateur ? Doit-on considérer le bâtiment en entier ? Pour répondre à cette problématique, on peut s'appuyer sur les travaux de Jacquet (Jacquet 2006) pour circonscrire la zone d'interaction à une portion de l'espace correspondant aux capacités de perception de l'utilisateur. Bien sûr, cet espace varie en fonction des caractéristiques d'interaction du périphérique (voir la section IV.4.2.iii sur le modèle KUP) et du contenu qu'il véhicule. Par exemple, la zone de perception d'un texte sur un écran dépendra de la résolution de l'écran et de la taille de police utilisée. Différents algorithmes sont présentés pour implémenter une sélection des périphériques d'interaction adaptés en utilisant cette définition (Jacquet, Bellik & Bourda 2007).

II.2.3 FLEXIBILITE ET HETEROGENEITE DU CONTEXTE

En s'appuyant sur une analyse de cas d'utilisations, (Henricksen, Indulska & Rakotonirainy 2002) propose une représentation formelle du contexte dans l'Ambiant. Ce modèle a permis d'extraire quatre caractéristiques qui rendent le contexte difficile à modéliser :

- De nombreuses échelles de temps doivent être considérées.
- L'acquisition du contexte est imparfaite.
- Une même information doit être représentée sous différentes formes, à différents niveaux de granularité.

- Il existe de fortes corrélations/dépendances entre certaines variables du contexte.

Pour s'adapter à cette nature particulière du contexte dans l'Ambiant, le projet CoWSAMI (Athanasopoulos et al. 2008) a permis la définition d'une infrastructure logicielle qui permet d'intégrer dynamiquement de nouvelles sources de contexte. Cette implémentation réalise la vision du contexteur de (Dey 2000). Chaque composant du système exporte son propre contexte local, sous la forme d'un service web. Les contextes sont donc distribués et unifiés à la demande en une unique représentation. Cependant, cette architecture pose le problème de l'hétérogénéité des implémentations et des modèles de contexte. Ainsi, il faut quand même définir à la main des règles d'alignement entre les concepts présents dans les différents contextes locaux. Les différents constructeurs n'étant pas nécessairement prêts à adopter une standardisation de ces descriptions, la génération automatique de ces règles d'alignement demeure un problème ouvert.

Puisque l'information acquise à propos du contexte est par nature imparfaite, la plateforme Ecora (Padovitz, Loke & Zaslavsky 2007) se focalise sur une modélisation de l'incertitude concernant les informations du contexte et sur le raisonnement dans ces conditions d'incertitude. Il s'agit d'une solution multi-agents dans laquelle un serveur de contexte centralise les informations et fournit des services de raisonnement aux applications mobiles (les agents du système). Au niveau du raisonnement, une différence est faite entre le contexte (c'est-à-dire les valeurs données par les capteurs) et la situation (c'est-à-dire l'état réel du monde). Une théorie intitulée *Context Space Theory* est proposée par ces auteurs, dans laquelle un ensemble de situations plausibles sont générées à partir du contexte, pour ensuite se voir attribuer un indice de confiance.

II.3 QUESTIONS ETHIQUES

La vocation de l'Ambiant est d'être présent dans tous nos activités de la vie quotidienne, disséminés dans tous les espaces et de façon non invasive. Il est donc attendu de l'utilisateur qu'il s'habitue et perde conscience de la présence du système. Nous présentons dans cette section les deux inquiétudes majeures, sur le plan éthique, que cette approche engendre légitimement.

II.3.1 ROLE DU SYSTEME DANS LA PRISE DE DECISION

« Les technologies les plus abouties sont celles qui deviennent invisibles. Elles se fondent dans le décor de notre vie quotidienne, au point d'en devenir indiscernables. »

C'est ainsi que Mark Weiser[4] débuta son introduction à propos de l'ordinateur du 21ème siècle (Weiser 1991). Cette phrase a parfois été interprétée (à tort) comme la nécessité de faire physiquement disparaitre les composants du système. Avec la miniaturisation de l'électronique, il est devenu envisageable que les ordinateurs deviennent indiscernables,

[4] *"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it."*

voire même complètement invisibles à l'œil nu. Ainsi, ils pourraient être intégrés dans des vêtements (Mann 1997), dans l'architecture, les meubles ou dans tout autre objet de la vie courante (Ito et al. 2003). Il faut cependant comprendre que la vision de l'ordinateur évanescent (en anglais *disappearing computer* ou *calm computing*) n'est pas celle d'un ordinateur qui ne serait plus perceptible sur le plan physique, mais plutôt d'un système qui disparaîtrait sur le plan cognitif. Autrement dit, un système qui retiendrait un minimum l'attention de l'utilisateur et le laisserait se concentrer sur la tâche qu'il veut réaliser, plus que sur les moyens de l'accomplir.

Bien qu'à aucun moment Weiser ne rentre dans de telles considérations, une disparition du système sur le plan cognitif nécessite que celui-ci soit capable de raisonner et d'agir proactivement (Tennenhouse 2000). Le système doit abstraire la masse d'informations perçues par les capteurs, en tirer du sens, pour comprendre et anticiper les besoins/envies de l'utilisateur et lui proposer de façon non-intrusive des solutions.

Le fait de sortir ainsi l'utilisateur de la boucle de décision, ou tout du moins, de ne lui donner la possibilité d'agir qu'*a posteriori* sur celle-ci pose deux principaux problèmes (Rogers 2006). Tout d'abord, anticiper les besoins/envies d'un utilisateur est un problème d'intelligence artificielle très complexe, qui fait notamment intervenir la reconnaissance d'émotions et du contexte. Or, il a été démontré que les motivations et le contexte qui guide nos actions de la vie de tous les jours présentent une forte variabilité inter et intra-individus (Salvador et al. 2003). Mais surtout, même si l'analyse des habitudes et de la psychologie de l'utilisateur était possible, elle poserait un sérieux problème d'éthique, de confiance et de respect de la vie privée. Ces deux principaux arguments expliquent, d'après (Rogers 2006), que bien que de gros progrès aient été faits vers l'accomplissement de la vision de Weiser, il subsiste en l'état actuel des connaissances un écart important avec ce que nous sommes capables de réaliser de cette vision. Rogers propose donc de rediriger les efforts de la communauté vers une interaction avec le système qui mette l'homme au centre de la prise de décision.

II.3.2 LA PROTECTION DE LA SPHERE PRIVEE

Avec ses divers capteurs disséminés dans les moindres recoins de notre environnement, et ayant pour but affiché de « disparaitre » de notre conscience, l'Ambiant réveille en nous un certain nombre de craintes qui furent longuement explorées dans la littérature de science-fiction (Orwell 1949) (Bradbury 1953).

Ces craintes sont cependant fondées. Pour analyser ces risques, le projet de recherche intitulé **SWAMI** (*Safeguards in a World of AMbient Intelligence*) fut créé (Wright et al. 2008). D'après les quatre scénarios traités dans ce projet, on peut déduire que les risques éthiques concernant le respect de la vie privée s'articulent autour de quatre principales dérives (De Hert et al. 2009):

- La surveillance

  Chacune de nos actions dans un environnement digital est susceptible de laisser des traces. De la même façon, dans un monde physique augmenté de capacités digitales tel que le propose l'ambiant, chacune de nos actions aura des

répercussions (directe ou indirecte) sur l'état du système. A partir de ces traces, il est possible de retracer l'activité d'une personne.

- La tendance à une disparition de la frontière entre vie privée/publique

  L'Ambiant offre la possibilité de réaliser de multiples activités dans un même endroit. Dès lors, la vie privée n'est plus délimitée par des contraintes physiques. Dans l'un des scénarios élaborés dans le projet SWAMI, une personne travaille à domicile, chez elle, pour une entreprise de sécurité tout en s'occupant de ses enfants. Cette entreprise, pour des raisons de sécurité, doit contraindre et filtrer l'accès de son ordinateur à certaines données. Puisque la technologie permet de réaliser plusieurs tâches simultanément, libéré des contraintes géographiques, comment faire la distinction entre ce qui a trait à la vie privée et ce qui tient de son travail ? Comment protéger les données privées de cette personne au regard de la surveillance qu'exige son employeur alors qu'elle cumule au même moment dans un même endroit, des activités différentes ? Lorsque les barrières entre activités professionnelles et privées tombent, est-il possible de distinguer et de protéger les données privées ?

- Le profilage

  Le profilage (Brusilovsky, Kobsa & Nejdl 2007) est l'action de dresser un profil d'un utilisateur au moyen d'informations obtenues automatiquement par le système. Si ce dernier doit s'adapter automatiquement à l'utilisateur, le profilage est une étape nécessaire. On peut alors s'inquiéter du type d'information qui est généré et conservé. Et surtout, à qui servent ces informations ? Bien qu'elles soient généralement accumulées dans le but de satisfaire les besoins immédiats de l'utilisateur, il se pourrait qu'elles soient récupérées par des organismes gouvernementaux dans un but de surveillance ou par des entreprises commerciales dans le but d'un ciblage marketing.

- La convergence des technologies et la dépendance des données utilisateurs

  Les données accumulées par les capteurs peuvent être croisées pour découvrir des informations qui ne sont pas directement accessibles. Dès lors, si ces informations dérivées sont d'ordre privées, il faudrait que les informations qui permettent de les inférer soient elles aussi considérées comme privées.

Certaines de ces craintes dépendent uniquement d'une volonté politique. Elles sont donc malheureusement sujettes à modification dans le temps, y compris au sein d'instances démocratiques. Une politique protectionniste concernant les informations privées pourrait être changée, voire inversée au gré des élections. Les solutions techniques à ces problématiques peuvent donc se classifier en deux catégories. Les premières visent à simplifier la gestion et la classification des données qui sont alors physiquement présentes dans le système mais qui peuvent être filtrées par les différents acteurs du système, à condition que ceux-ci respectent la législation en vigueur. La seconde catégorie vise à fusionner et filtrer les données pour empêcher structurellement leur propagation dans le système. Ces dernières méthodes garantissent une indépendance de la protection des données vis-à-vis de toute autorité. Cependant en faisant de la perte de données une arme de protection, elles empêchent aussi la richesse des informations requises d'être utilisée à son plein potentiel.

L'IHM est également au cœur de ces interrogations car en rendant l'information accessible à son propriétaire, elle peut (par erreur ou par malveillance) divulguer cette information à d'autres personnes. Par exemple, (Want et al. 2002) présente une plateforme d'IHM totalement décentralisée intitulée : *The Personal Server*. Ce serveur personnel est un périphérique mobile qui permet à son propriétaire d'enregistrer des données et des applications et d'y accéder où qu'il soit. Au contraire des PDA, ce système ne dispose d'aucune interface homme-machine, il utilise les périphériques d'interaction disponibles dans l'environnement grâce à un protocole de communication sans-fil. L'usage de ces systèmes opportunistes pour produire une interface en reposant sur les périphériques qui sont disponibles dans l'environnement immédiat pose des problèmes de sécurité et de respect de la vie privée. Les utilisateurs pourraient voir divulguer sur des écrans et autres périphériques publics, des données qu'ils ne souhaitaient pas divulguer. Or l'aspect déstructuré et opportuniste du choix des éléments de l'interface rend la rédaction de règles d'accès et de sécurité pour le moins complexe. Des travaux ont été entrepris pour « développer des systèmes qui aident les autres utilisateurs à respecter la vie privée » (Langheinrich 2002) en s'appuyant sur leur bonne foi.

II.4 CONCLUSION

En conclusion, nous avons étudié, dans ce chapitre, les problématiques posées par l'Ambiant sous l'angle de l'implémentation technologique, sous celui de la conception d'IHM et enfin, sous celui de l'éthique.

Les barrières technologiques de l'Ambiant sont la distribution, l'hétérogénéité et l'aspect dynamique de ces architectures. Ces trois caractéristiques sont simultanément prises en considération par les architectures à base de composants ou orientées services. Ces architectures définissent les méthodes de description, de composition et de gestion du cycle de vie qui sont nécessaires à la flexibilité et la réutilisabilité des composants de l'Ambiant.

Cependant, la flexibilité des compositions permise par ces solutions techniques rend ardue la réalisation d'IHM. En effet, l'informatique conventionnelle considère comme statique la définition des trois entités suivantes : la tâche, le noyau applicatif, les médias. Or, dans l'Ambiant, ces trois entités ne sont pas connues au moment de la conception, elles sont déterminées à l'exécution, de façon opportuniste.

Pour faire face à ces inconnues, la démarche naturelle est de tenter de modéliser le contexte et d'automatiser les choix de conception à l'exécution. Cependant, cette vision soulève des questions éthiques. Cette adaptation automatique doit se faire dans le sens d'une protection de la sphère privée, un compromis doit donc être trouvé quant aux traces conservées à propos de l'utilisateur. Elle doit par ailleurs laisser le contrôle du système à l'utilisateur qui ne doit pas se sentir dépossédé de ce contrôle, ou à la merci d'un système potentiellement défaillant.

La conception d'interfaces homme-machine doit donc permettre l'apparition de nouvelles techniques d'interaction plus adaptées à l'Ambiant, permettant à l'utilisateur de se sentir en contrôle, tout en autorisant le système à adapter automatiquement certaines étapes du processus de conception. Nous présentons, dans le chapitre suivant, les méthodes classiques de conception d'IHM et les résultats qu'elles ont fourni en étant appliquées dans le contexte de l'Ambiant. Le chapitre suivant présentera les architectures capables d'automatiser certaines étapes de la conception.

Chapitre III
CONCEPTION D'INTERFACES HOMME-MACHINE DANS L'AMBIANT

L'Interaction Homme-Machine (IHM) est un important domaine de recherche depuis les années 60. D'abord cantonnée à l'étude de l'interaction à travers le clavier et la souris, l'IHM s'est portée, dès le début des années 80, sur les différentes autres manières de présenter des informations à l'utilisateur et d'interagir avec lui.

Du point de vue de la modélisation et de la conception, seules les fonctionnalités de l'application furent dans un premier temps représentées. L'interface entre ce noyau applicatif et l'utilisateur se faisait par l'intermédiaire de la ligne de commande. Ce type d'interfaces était donc réservé aux experts. Dès lors que le public s'intéressant aux ordinateurs s'élargit, il fallut trouver une solution pour rendre ces fonctionnalités plus accessibles. De ce besoin naquit un certain nombre de modèles permettant de représenter et d'étudier l'interaction entre l'homme et la machine.

Dans cette section, nous présenterons les méthodes utilisées pour concevoir des IHM. Ces méthodes sont essentielles pour la conception de nouvelles interfaces aptes à remplacer le paradigme d'interaction actuel. Elles reposent souvent sur l'emploi de métaphores pour concevoir de nouvelles techniques d'interaction. Nous introduisons, dans la seconde section, les métaphores et techniques d'interaction qui ont été développées spécifiquement pour l'ambiant. Nous verrons que la multimodalité est au cœur de ces techniques et nous présentons donc dans la dernière section, les concepts couramment utilisées pour décrire et raisonner sur la multimodalité.

III.1 METHODOLOGIE DE LA CONCEPTION

On distingue généralement la conception du noyau applicatif (également appelé noyau fonctionnel) de la conception de l'interface homme-machine à proprement parler. Le noyau applicatif est l'ensemble des fonctionnalités sur lesquelles reposent les tâches réalisées par le système. Ces fonctionnalités peuvent être réalisées de façon autonome. L'interface homme-machine, quant à elle, représente l'ensemble des moyens matériels et logiciels qui permettent à l'Homme de contrôler le système. Elle est donc chargée de faire l'interface entre l'homme et le noyau applicatif.

Plusieurs méthodes ont été proposées pour la conception du noyau applicatif. Les plus classiques sont des approches descendantes : les besoins de l'utilisateur sont analysés, puis spécifiés, formalisés et implémentés. L'utilisateur n'intervenant généralement que dans la phase d'analyse des besoins, ces méthodes se sont révélées peu efficaces pour les interfaces utilisateur, et plus particulièrement dans l'Ambiant. En effet, le nombre d'acteurs intermédiaires aidant, l'analyse des besoins peut mener à des divergences entre les besoins exprimés par les utilisateurs, ce qui est spécifié par les concepteurs et ce qui est

implémenté par les programmeurs. Ces incompréhensions peuvent mener à différents degrés de rejet par l'utilisateur final.

III.1.1 CONCEPTION ORIENTEE UTILISATEURS ET CONCEPTION PARTICIPATIVE

La conception centrée utilisateurs (*User-centered design* ou **UCD**) a été créée (Norman 1986) dans le but de pallier ces problèmes de communication en intégrant des utilisateurs finaux dans les différentes phases de la conception. Elle est plus formellement définie ainsi (Vredenburg et al. 2002) [5]:

« [Il s'agit] de la pratique des principes suivants: l'implication active des utilisateurs pour une compréhension complète des besoins utilisateurs et des tâches, la conception et l'évaluation itérative, et une approche multidisciplinaire ».

On applique généralement les principes de la conception orientée utilisateur pour construire l'interface et l'évaluer de façon itérative (Vredenburg et al. 2002). Cette approche permet d'obtenir de bons résultats dans les validations par les utilisateurs mais comporte deux inconvénients majeurs : un coût élevé et l'absence d'objectifs ou de repères stables permettant de mettre un terme aux itérations (Myers 1994). A chaque cycle (ou itération) de la conception, l'interface est évaluée par les utilisateurs finaux, ce qui peut amener à des modifications des spécifications initiales et le début d'un nouveau cycle. Chaque cycle est donc décomposé en 2 étapes principales :

- Le prototypage : Analyse et spécification des besoins, conception du prototype.
- L'évaluation : Evaluation de l'utilisabilité du prototype.

III.1.2 PROTOTYPAGE ET EVALUATION

A chaque cycle, la première phase de l'UCD est la conception d'un prototype. Ce travail peut reprendre les prototypes définis dans les précédentes itérations – on parle alors de raffinement. La phase de prototypage se décompose en 3 étapes intermédiaires : l'analyse et la spécification des besoins utilisateurs, la conception et l'implémentation de la solution et enfin l'évaluation de l'interface produite. Nous décrivons, dans cette section, un ensemble de méthodes couramment employées pour réaliser ces trois objectifs. Cette présentation n'est pas exhaustive, elle illustre les techniques les plus employées dans le domaine de l'Ambiant. Le lecteur pourra se référer à (Kolski 1995) pour une présentation plus détaillée de ces techniques.

La première étape d'analyse des besoins consiste à acquérir des informations sur l'utilisateur et sur ses usages du système. Cette étape repose sur le profilage des utilisateurs cibles et leur observation in situ.

---

[5] *"[it is] the practice of the following principles, the active involvement of users for a clear understanding of user and task requirements, iterative design and evaluation, and a multi-disciplinary approach".*

- **ANALYSE DES PROFILS UTILISATEURS**

  Cette étape consiste en la définition des différents utilisateurs de l'application, de leurs buts ainsi que de leurs caractéristiques : quelles sont leurs compétences en informatique ? Quel sont leurs niveaux d'expérience dans le domaine ?... De cette étude, on développe des cas d'utilisation qui décrivent dans quelles conditions un utilisateur active l'application et pour quelles fonctionnalités. Cette étape fait intervenir les utilisateurs finaux, qui sont sollicités par le concepteur pour définir des scénarios types ou être observés *in situ*.

- **OBSERVATION IN SITU**

  Cette étape préliminaire à la rédaction des besoins utilisateurs peut être utilisée comme alternative ou complément à l'usage de scénarios ou à l'interview des utilisateurs. Elle consiste à observer ces derniers en situation réelle, dans le but d'améliorer un système existant.

L'identification des besoins des utilisateurs dans l'Ambiant est plus complexe que dans les systèmes traditionnels. En effet, l'Ambiant est un univers nouveau, une vision du futur de l'informatique qui n'est pas encore réalisée dans la vie quotidienne des utilisateurs. Il leur est donc difficile de s'y projeter et d'imaginer les besoins que ces futurs systèmes pourraient satisfaire. Les concepteurs peuvent donc aider les utilisateurs à se projeter dans ce nouvel univers au travers de scénarios et de sessions de brainstorming.

- **STORYBOARD/SCENARIO**

  La rédaction de scénarios n'est pas toujours nécessaire. En effet, des résultats similaires peuvent être obtenus lorsqu'un corpus d'interaction est déjà disponible ou en analysant une application déjà existante. Cependant, les scénarios présentent plusieurs avantages (Carroll 2000) : flexibilité, possibilité d'explorer le même problème sous différents points de vue et différents niveaux de détails, et surtout la possibilité d'explorer de nouvelles formes d'interaction.

- **SESSION DE BRAINSTORMING**

  Ces sessions permettent d'imaginer l'interface de façon collaborative. Dans ces sessions, chaque partenaire partage ses idées sans autocensure. A l'issue de la session, il est nécessaire de classifier les idées obtenues, d'en sélectionner certaines et de se mettre d'accord sur une première idée générale de l'interface.

Une fois les besoins identifiés, l'étape de conception consiste à choisir quels artefacts feront partie de l'interface, comment ils interagissent et comment ils seront implémentés. Elle fait intervenir des méthodes issues de l'ingénierie logicielle, telle que l'utilisation de diagrammes UML (diagrammes de classe et de séquence notamment). Dans certains cas, il est possible de factoriser les développements à l'aide de boîtes à outils réutilisables.

- **BOITES A OUTILS**

  Les boites à outils sont utilisées pour simplifier l'implémentation de certaines parties de l'interface. Il peut s'agir de boites à outils graphiques telles que les *Windows Forms*, GTK+ ou Swing. Les fonctionnalités proposées par ces boites à outils ne se limitent pas nécessairement aux interfaces graphiques. Ainsi, certaines proposent la gestion de la parole (Traum & Larsson 2000), voire même le mélange de plusieurs styles d'interaction (Rousseau 2006).

Enfin, la dernière étape consiste à évaluer la qualité de l'interface conçue. Cette évaluation peut se faire sur un plan quantitatif (temps passé à effectuer une tâche, nombre d'erreurs commises lors de la réalisation d'une tâche…) ou sur un plan qualitatif (évaluation du ressenti subjectif de l'utilisateur). Dans l'Ambiant, il est parfois impossible ou très couteux d'implémenter un système interactif avant de l'évaluer. Dans ce cas, une pré-évaluation peut être réalisée avant toute implémentation à l'aide de la technique du magicien d'Oz, ce qui permet d'orienter les développements en amont de la phase d'évaluation.

- **La technique du Magicien d'Oz**

  Le Magicien d'Oz (Salber & Coutaz 1993) est une technique aidant au prototypage rapide. Elle permet de simuler le fonctionnement de parties de l'interface qui prendrait trop de temps à être implémentées ou qui sont inaccessibles en l'état actuel des connaissances. Un humain, appelé le magicien, est caché pendant l'expérience. Il pilote les réactions du système. L'utilisateur vit l'expérience comme si tout était implémenté tandis que le magicien compense les parties manquantes du système. Dans la Figure 3, l'utilisateur interagit en langage naturel avec un avatar affiché à l'écran dans une pièce ambiante. Le magicien d'Oz situé dans la pièce adjacente pilote les réactions de l'avatar à l'écran et surveille le comportement de l'utilisateur grâce à une caméra.

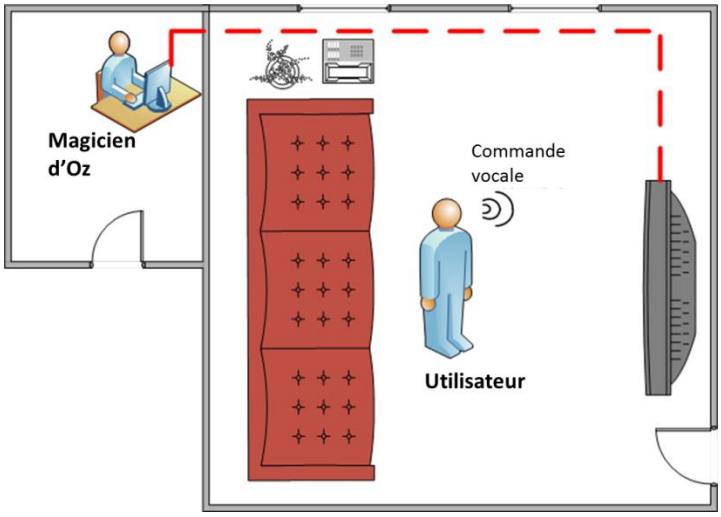


Figure 3 – Exemple d'application de la technique du magicien d'Oz

III.1.3 Metaphores d'interaction

A tout moment, le concepteur doit s'assurer que le système sera utilisable par l'utilisateur, c'est-à-dire que la manipulation des objets de l'interface doit se faire à un coût cognitif le plus faible possible. Pour permettre un meilleur apprentissage et une plus grande efficacité, les artefacts composants l'interface doivent être présentés à l'utilisateur de façon cohérente. C'est la raison pour laquelle des paradigmes d'interaction sont utilisés. Un paradigme d'interaction est un ensemble de règles qui décrit le comportement des éléments de l'interface. Le plus courant est le paradigme **WIMP** (*Window, Icon, Menu, Pointing device*) qui a été développé au Xerox PARC et qui est encore utilisé de nos jours.

L'intérêt d'un tel paradigme est qu'une fois que l'utilisateur a appris à se servir d'une application, il pourra réutiliser ces connaissances dans un autre contexte, dans une autre application, pourvu qu'elle obéisse au même paradigme d'interaction.

La notion de métaphore va même plus loin. D'après Blackwell (Blackwell 2006), il s'agit de la représentation mentale créée dans le but d'aider l'utilisateur à comprendre des fonctionnalités abstraites en les assimilant à des mécanismes qu'il connait déjà. Par exemple, la métaphore du bureau est utilisée dans la plupart des systèmes d'exploitation modernes (hors smartphones et tablettes). Dans cette métaphore, l'utilisateur a le sentiment de travailler sur un réel bureau. Par analogie, il peut donc s'attendre à pouvoir déplacer les fichiers ou à les déposer dans des dossiers… D'un point de vue cognitif, la métaphore est un guide permettant à l'utilisateur de se créer sa propre représentation mentale de l'interface et l'aide à découvrir de nouvelles façons d'interagir (Carroll & Thomas 1982). A travers le prisme de la métaphore, l'utilisateur perçoit le système, ses capacités et comprend les manipulations qu'il peut réaliser. Bien sûr, les métaphores ont leurs limites. Elles ne peuvent expliquer tous les concepts de l'interface, raison pour laquelle leur usage est sujet à controverses (Blackwell 2006). La notion de raccourci, par exemple, n'est pas expliquée par la métaphore du bureau.

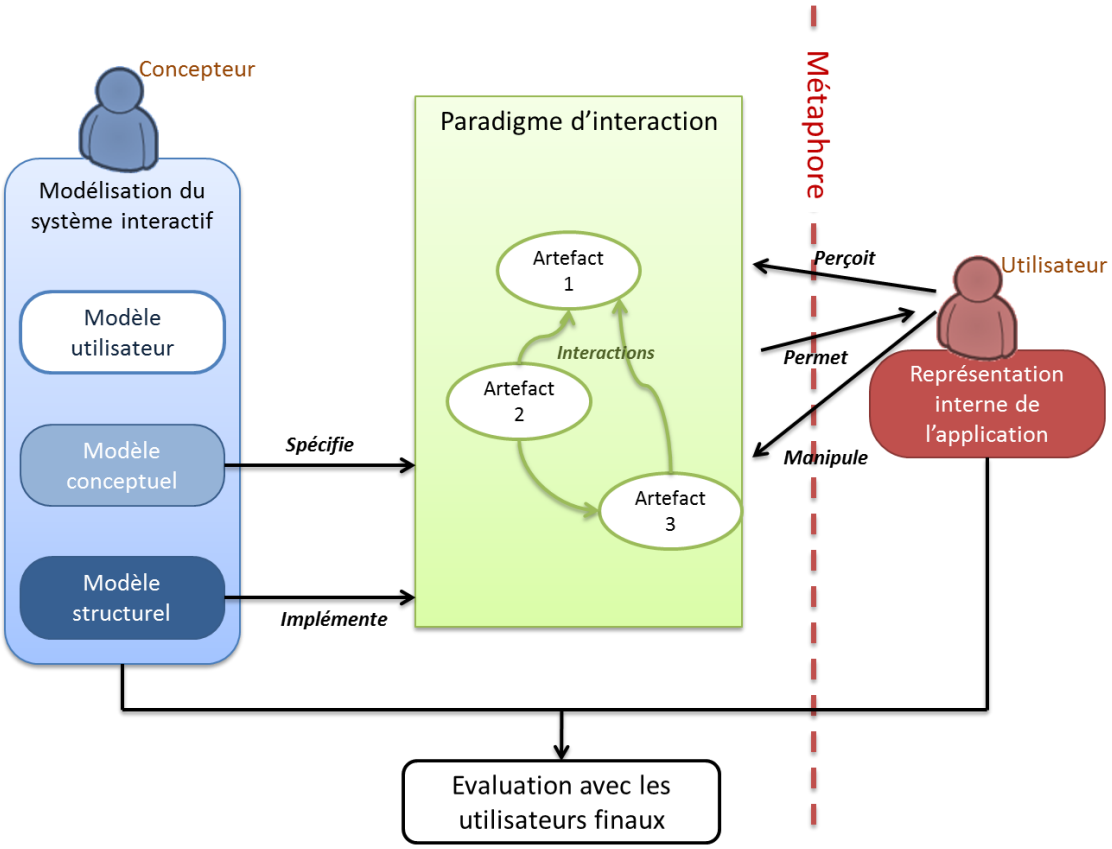


FIGURE 4 – RELATIONS ENTRE LES REPRESENTATIONS DU DESIGNER ET DE L'UTILISATEUR

Ces concepts issus de l'ergonomie et des sciences cognitives permettent au concepteur de structurer son travail. Les relations entre ces concepts sont montrées dans la Figure 4. Ils permettent d'élaborer et d'évaluer de nouvelles techniques d'interaction.

III.2 TECHNIQUES D'INTERACTION ET METAPHORES DE L'AMBIANT

En appliquant les techniques de design présentées ci-dessus au contexte de l'informatique ambiante, de nouvelles interfaces, qualifiées de **POST-WIMP** ont vu le jour. Le spectre de ces interfaces est très large, à l'aune de la richesse que permettent les environnements ambiants. Nous proposons donc de classer ces nouvelles approches selon les 4 axes qui sont définis ci-dessous.

III.2.1 INTERACTION IMPLICITE ET CALM COMPUTING

Weiser a développé la notion de « *calm computing* » (Weiser 1994). D'après lui, l'Ambiant est le contraire de la réalité virtuelle. Dans la réalité virtuelle, l'utilisateur est immergé dans un monde numérique. Au contraire, l'informatique ubiquitaire consiste à intégrer, de façon transparente, le monde digital dans le monde réel.

La notion d'interaction implicite (*seamless/implicit interaction* appelée également *iHCI* en anglais) est discutée dans (Chalmers & MacColl 2003). Elle est plus formellement définie dans (Schmidt 2005) à l'aide de la distinction entrées/sorties implicites:

« DEFINITION : Interaction Homme-Machine implicite

L'**iHCI** est l'interaction d'un être humain avec un environnement et des artefacts dans un but donné. Dans ce processus, le système acquiert des entrées implicites de l'utilisateur et peut lui présenter des sorties implicites.

DEFINITION : Entrée implicite

Les entrées implicites sont des actions et des comportements des êtres humains qui sont faits dans le but d'atteindre un objectif et non considéré de prime abord comme une interaction avec un ordinateur bien qu'ils soient capturés, reconnus et interprétés par le système comme une entrée.

DEFINITION : Sortie implicite

C'est la sortie d'un système qui n'est pas directement liée à une entrée explicite et qui est intégrée de façon transparente à l'environnement et à la tâche de l'utilisateur. »

L'interaction implicite suppose que l'utilisateur peut manipuler des objets sans faire de différences entre des entités numériques ou physiques. Par conséquent l'utilisateur peut se concentrer sur la tâche qu'il réalise et l'usage de l'outil devient transparent. La compréhension du fonctionnement de l'outil et son usage ne se mettent pas en travers de la tâche. C'est la raison pour laquelle ce paradigme d'interaction est parfois appelé l'interface transparente (Ishii 2004).

Cette vision est liée au concept de technologie calme (*Calm Technology*) (Weiser & Brown 1996). Les technologies calmes sont des dispositifs qui informent de façon passive l'utilisateur. Ce sont des périphériques dédiés aux sorties implicites telles qu'elles sont définies ci-dessus. Par exemple, la *dangling string* (Weiser & Brown 1996) est une corde qui ondule en fonction de l'activité du réseau informatique, permettant de garder un œil sur l'état du trafic sans faire l'usage d'un logiciel spécifique. Ces technologies sont à la fois

informatives et qualifiées de « reposantes » par les utilisateurs, elles sont souvent liées au design artistique.

Cependant, comme cela était souligné dans (Chalmers & MacColl 2003), l'interaction implicite ne signifie pas que la technologie soit nécessairement calme. La technologie calme n'est donc qu'une facette de l'interaction implicite. Dans l'interaction implicite, transparence ne veut pas dire invisibilité, elle signifie que la technologie doit être intuitive pour l'utilisateur.

III.2.2 INTERACTION TANGIBLE ET INTERACTION MIXTE

Le concept d'interface tangible est une évolution et une généralisation des travaux sur les « *graspable interfaces* » (Fitzmaurice, Ishii & Buxton 1995) et les « *tangible bits* » de Ishii et Ulmer (Ishii & Ullmer 1997). Il s'agit d'un paradigme dans lequel l'interaction repose sur la conception d'objets physiques que l'utilisateur peut prendre en main et manipuler.

Ainsi, comme le montre la Figure 5, des objets du monde réel (des « briques » interactives) peuvent interagir avec les objets du monde virtuel. L'idée derrière la notion de « briques » était de fournir à l'utilisateur la possibilité « d'attraper » l'interface, d'utiliser ses capacités intuitives à manipuler son environnement physique pour agir dans le monde virtuel. Pour peu que les objets soient conçus de façon à suggérer (de par leur forme ou leur usage habituel) l'interaction, on obtient un effet de levier pour diminuer la charge cognitive de l'interface. Cette suggestivité repose sur la théorie de l'**AFFORDANCE** établie par Gibson (Gibson 1977). On peut mieux observer la prégnance de cette notion d'affordance dans les travaux plus récents tels que la « bouteille » (cf. Figure 6) ou la « baguette magique » de Wilson et Shafer qui permet de désigner des objets du monde physique (Lampe, TV, chaine Hi-Fi) et de les contrôler par commande vocale (Wilson & Shafer 2003).

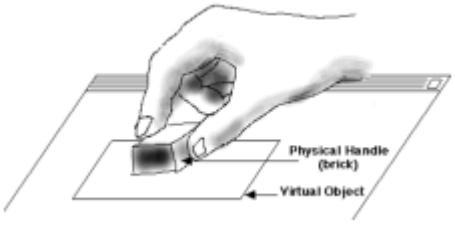


FIGURE 5 – DES BRIQUES POUR PILOTER LE MONDE NUMERIQUE, EXTRAIT DE (FITZMAURICE, ISHII & BUXTON 1995)

FIGURE 6 – UNE BOUTEILLE « CONTENANT » LES PREVISIONS METEO, EXTRAIT DE (ISHII 2004)

Ces interfaces présentent de bons résultats dans les tests d'utilisabilité pour plusieurs raisons. Tout d'abord, elles permettent à la fois une interaction continue et une exploration libre de l'espace des fonctionnalités par l'utilisateur. De plus, elles permettent de réifier des concepts ou des objets virtuels tels que des fichiers (Ullmer, Ishii & Glas 1998).

Cependant, elles ont l'inconvénient de reposer sur des périphériques qui sont souvent dédiés à la tâche pour laquelle ils sont conçus. Elles ne proposent donc généralement pas une grande flexibilité ce qui contraint leur réutilisabilité. On peut citer par exemple l'interface *Urp*, proposée dans (Ullmer & Ishii 2000) qui permet de représenter une ville pour des urbanistes. Dans cette interface, les bâtiments sont représentés par leur maquette que l'urbaniste peut déplacer librement tandis que le système projette par-dessus une simulation des ombres à différents moments de la journée. Il est clair qu'une telle interface ne serait pas réutilisable dans un contexte autre que celui de l'urbanisme. Bien que plusieurs voies soient explorées pour l'élaboration de métaphores d'interaction pour les interfaces tangibles (Svanaes & Verplank 2000), cette spécialisation de l'interface est un frein à l'émergence d'une métaphore cohérente et générique.

Malgré cela, l'analyse de plusieurs systèmes de cette catégorie a permis l'élaboration d'une taxonomie proposée dans (Fishkin 2004) ainsi que d'un langage descriptif permettant de spécifier les interactions possibles (Shaer et al. 2004), appelé **TAC** (*Tokens And Constraints*).

Les systèmes interactifs mixtes (**SIM**) (Charfi 2009) sont des systèmes où coexistent des objets physiques et des représentations d'objets virtuelles. Ce domaine étant issu des travaux sur la réalité augmentée (Feiner, Macintyre & Seligmann 1993), les travaux le concernant se concentrent généralement sur un contexte dans lequel l'utilisateur est équipé d'un matériel permettant de superposer des informations virtuelles au monde réel. L'usage de tels matériels intrusifs va à l'encontre de la définition de l'Ambiant. Cependant, les travaux sur la modélisation des SIM visent à unifier les différentes formes d'interaction mixte, ils incluent donc la modélisation de l'interaction tangible. De fait, les modèles qui sont proposés pour les SIM peuvent donc être adaptés et réutilisés dans le contexte de l'informatique ubiquitaire. Ainsi, le modèle ASUR (Dubois 2001) est une notation permettant de décrire l'interaction d'un utilisateur avec un système interactif mixte au cours d'une tâche donnée. Une analyse comparative des méthodes de conception des SIM est proposée dans (Charfi 2009).

III.2.3 INTERFACES UTILISATEURS ORGANIQUES ET INTERACTION KINETIQUE

Les interfaces organiques (*Organic User Interfaces* ou OUI) (Vertegaal & Poupyrev 2008) sont dans la continuité des interfaces tangibles. Il s'agit d'interfaces dont la forme même peut varier et contribuer à l'interaction. Cette discipline se focalise sur l'interaction au travers de surfaces d'affichage non planes et éventuellement déformables. A cause de limites matérielles, les « tokens » de l'interaction tangibles peuvent rarement être actualisés par le système (tant en matière de position que de forme). Par conséquent, l'interaction tangible est un domaine qui s'est généralement focalisée sur la conception d'interfaces physiques en entrée avec une sortie projetée sur une surface fixe. Dans (Holman & Vertegaal 2008), Holman et Vertegaal comparent l'interaction homme-machine avec l'interaction avec des objets de la vie de tous les jours. Selon eux, les deux facteurs qui différencient ces deux formes d'interaction sont :

- **L'ABSENCE DE VARIETES DE FORMES DES INTERFACES HOMME-MACHINE** qui, à cause de contraintes matérielles, convergent toutes vers une surface plane rectangulaire.

- **L'IMPOSSIBILITE POUR L'UTILISATEUR DE DEFORMER L'INTERFACE**, comme il pourrait le faire avec une feuille de papier par exemple.

Grâce à l'avancée des technologies OLED et E-Ink, ces deux limites pourraient être surmontées dans un avenir proche. De la même façon que pour les TUI (*Tangible User Interface*), les auteurs s'appuient sur le concept d'affordance pour garantir la simplicité de la prise en main et de l'expérimentation par l'utilisateur. L'optimisation de cette affordance passe par l'emploi de la surface déformable à la fois comme entrée et comme sortie du système. L'unicité des entrées/sorties du système dans un même objet d'interaction donne l'illusion d'avoir le matériau qui réagit de lui-même aux sollicitations physiques de l'utilisateur, de telle sorte que celui-ci est perçu comme vivant, d'où l'emploi du terme organique (Holman & Vertegaal 2008). Par rapport aux TUI, les OUI représentent donc un changement dans la façon de considérer l'interface dans lequel les entrées et les sorties ne font physiquement plus qu'un.

La Figure 7 illustre une telle interface. Dans ce système, l'interface est une surface librement malléable que les utilisateurs peuvent sculpter à la main pour obtenir le relief voulu. Le relief est perçu par le système à l'aide d'un balayage laser, ce qui lui permet d'adapter la topographie affichée dessus en conséquence. L'interaction est directe pour l'utilisateur et le comportement du système est organique puisqu'il évolue par mimétisme avec un système physique similaire. Par exemple, créer une dépression dans le sol permet aux utilisateurs de dévier un cours d'eau. Dans la Figure 8, une feuille de papier sert de surface de projection à un browser web. La navigation repose sur les déformations de la feuille. Ainsi, tordre le coin droit ou gauche de la feuille permet d'aller d'avant en arrière dans l'historique, l'orientation verticale permet de scroller dans la fenêtre courante. Des actions permettent également d'interagir avec d'autres OUI. Ainsi, la Figure 8 illustre le mouvement de frottement de deux feuilles, qui permet de copier le contenu de l'une dans l'autre. Contrairement au système *illuminating clay*, les interactions proposées ne sont pas « naturelles », les OUI peuvent donc également supporter un jeu d'actions liées symboliquement à des déformations du medium sans qu'il n'y ait de sens physique à ce lien.

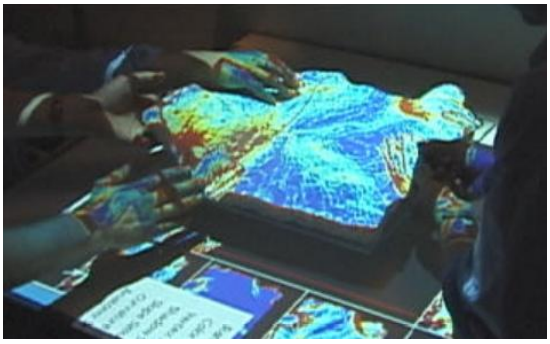


FIGURE 7 – ILLUMINATING CLAY, ISSU DE (PIPER, RATTI & ISHII 2002)

FIGURE 8 – PAPERWINDOWS, ISSU DE (HOLMAN ET AL. 2005)

Ces deux systèmes reposent sur des techniques de projections vidéo, ce qui suppose un montage lourd, sensible aux occultations et particulièrement difficile à calibrer. Cependant, les avancées en termes d'écrans souples reposant sur les technologies OLED et E-Ink permettent d'envisager une application débarrassée de ces contraintes matérielles.

La conception des interfaces organiques repose sur 3 principes de design :

- **UNICITE DES ENTREES/SORTIES**. Les entrées et sorties sont réalisées sur le même objet, il y a unicité, dans le périphérique d'interaction, de la manipulation et du rendu de l'information.
- **UNICITE DE LA FONCTION ET DE LA FORME**. C'est la notion d'affordance qui consiste à proposer une forme qui suggère la fonction qu'elle propose et qui lui soit ergonomiquement adaptée.
- **LA FORME SUIT LE FLUX D'ACTIVITES**. Le système doit être capable d'adapter sa forme par lui-même pour s'adapter à l'état des informations qu'il représente et au contexte plus général dans lequel il est utilisé.

La forme et son changement sont au cœur de l'interaction organique. Ces travaux sont donc fortement liés aux travaux sur les interfaces kinétiques (**KUI** ou *Kinetic User Interface*) (Parkes, I. & I. 2008). Les KUI se penchent sur le troisième des principes de design qui sont proposés ci-dessus. Ils utilisent le mouvement comme mode de communication. Cette vision devient possible grâce à la miniaturisation des effecteurs (notamment des moteurs).
Dans (Parkes, I. & I. 2008), une esquisse de taxonomie des types d'interaction kinétique est proposée. Elle identifie 4 expressions du mouvement :

- **LE MOUVEMENT D'ACTUALISATION DES PERIPHERIQUES D'INTERACTION**. Le mouvement du matériel d'interaction peut permettre de maintenir à jour la cohérence de la représentation de l'état du système. Ainsi, dans PICO (Patten & Ishii 2007), la position des entités d'une interface tangible qui servent de périphériques d'entrée peut être mise à jour par le système grâce à des électro-aimants.
- **LE MOUVEMENT INCARNANT UNE INFORMATION**. C'est l'exemple des mouvements utilisés dans le Calm computing pour représenter une information externe au système. On peut par exemple citer l'exemple des moulins à vent en papier dont la vitesse de rotation dépend de la météo locale (Dahley, Wisneski & Ishii 1998).
- **LE MOUVEMENT INCARNANT UN GESTE**. Certains systèmes sont dotés d'une mémoire kinétique, c'est-à-dire qu'ils peuvent enregistrer un mouvement qui leur est imposé et le rejouer à loisir (Raffle, Parkes & Ishii 2004).
- **LE MOUVEMENT CREATEUR DE FORMES**. Ces mouvements consistent à changer la forme même de l'objet d'interaction. La nouvelle forme permet de représenter un message, de façon symbolique ou représentative.

III.2.4 LE COUPLAGE

La notion de couplage est déjà présente dans les IHM conventionnelles. Il s'agit de l'association de différentes ressources d'interaction dans un même espace de travail. Dans l'informatique conventionnelle, l'espace de travail se compose d'un écran, d'une souris et d'un clavier situés à proximité les uns des autres. L'action du couplage permet de remodeler cet espace de travail en mixant plusieurs écrans, souris et clavier, voire de nouvelles ressources d'interaction.

Ce couplage n'est donc pas nouveau, mais l'Ambiant lui donne une nouvelle dimension. En effet, à travers la mobilité, l'utilisateur est amené à modifier les configurations figées de

l'informatique conventionnelle. Le système doit dont être capable de réassocier de façon spontanée les couplages de périphériques (Kindberg & Zhang 2003).

Une analyse comparative du couplage dans les IHM conventionnelles et non-conventionnelles est proposée dans (Barralon, Lachenal & Coutaz 2004). On y distingue le couplage de surfaces (i.e. les périphériques de rendu graphique), le couplage d'instruments (i.e. les périphériques d'entrée) et les couplages hybrides instruments-surfaces. Cette analyse décrit également des techniques de couplages réalisées par l'utilisateur grâce à des gestes synchronisés, comme ceux proposés pour associer plusieurs tablettes en les entrechoquant (Hinckley 2003). Le couplage de périphériques durant l'exécution nécessite de vérifier que certaines conditions soient vérifiées telles que la disponibilité des périphériques ou leur compatibilité mutuelle. Un automate est par exemple proposé dans (Barralon, Lachenal & Coutaz 2004). Il permet de décrire le cycle de vie des couplages de périphériques. Cet automate est enrichi d'une métaphore permettant aux utilisateurs de comprendre les règles d'association en établissant un parallèle avec l'assemblage de molécules (Barralon & Coutaz 2008).

De plus, la création de nouveaux couplages peut avoir des effets de bords sur les couplages déjà existants. Par exemple, si une souris est couplée à un écran et que ce dernier est couplé à un second écran, il faut adapter le comportement de la souris pour englober les deux écrans qui constituent la surface. Pour traduire ces conséquences et analyser de potentiels conflits, deux notations sont proposées dans (Barralon, Coutaz & Lachenal 2007). L'une est une notation graphique qui représente les connexions et fournit une représentation visuelle des contraintes de couplage. La seconde est une notation algébrique traduisant les propriétés de composition du couplage. Ces notations permettent une analyse systématique (et donc automatisable) du couplage de périphériques.

Pour que cette approche soit valide, les conséquences du couplage doivent être observables, prédictibles, traçables et contrôlables par le système. Cela suppose qu'un système informatique surveille l'état du contexte, conserve une trace des couplages actifs et applique les règles de couplage. Un tel système fait partie d'une vision plus globale qui consiste à fournir à l'utilisateur des moyens d'interagir avec l'interface pour en modifier sa forme, sa distribution ou son niveau de détails. Cette vision appelée extra-IHM ou méta-IHM est décrite dans (Calvary & Coutaz 2007).

III.2.5 Conclusion

Les techniques présentées dans cette section font toutes état de l'usage de méthodes d'interaction innovantes combinant le geste, parfois la parole et des techniques de visualisation non conventionnelles pour mieux s'intégrer dans l'environnement physique de l'utilisateur.

Le contexte d'utilisation variable de l'informatique ambiante suscite le besoin de créer des interfaces qui reposent sur des techniques d'interaction utilisant au mieux les capacités perceptuelles de l'utilisateur. Ces techniques d'interaction sont qualifiées de multimodales car elles permettent d'interagir selon différentes modalités d'interaction. Nous définissons, dans la prochaine section, les termes qui font la théorie de la modalité.

III.3 TERMINOLOGIES DES APPROCHES MULTIMODALES

Les techniques d'interaction présentées dans la section précédente ont pour point commun de mettre à profit la diversité des périphériques du système ambiant pour fournir des moyens d'interaction plus élaborés et plus riches. La théorie des modalités décrit les concepts qui permettent de structurer la forme de l'interaction homme-machine. Elle décrit les capacités logiques ou physiques de communication du système et ses liens avec le contenu de l'interaction.

En fonction des auteurs et des disciplines, les concepts mis en avant peuvent recouvrir différentes définitions qui sont parfois contradictoires. Des études approfondies des différentes significations acceptées pour ces termes sont fournies dans (Bellik 1995), (Truillet 1999) et (Rouillard 2008). Nous nous focaliserons donc sur les 2 acceptions les plus communes de ces termes, en analysant l'approche de (Bellik 1995), qui est orientée utilisateur, et celle de (Nigay & Coutaz 1995), qui est orientée système.

III.3.1 TERMINOLOGIE CENTREE SUR L'UTILISATEUR

L'approche centrée utilisateur est issue des études en ergonomie, physiologie et psychologie. Elle consiste à définir les termes de mode, modalité et média en fonction des capacités de perception et d'action des êtres humains.

D'après (Bellik 1995), le **MODE** fait référence au système sensoriel que l'homme utilise pour traiter l'information. Il existe deux catégories de mode :

- Les modes en entrée, qui permettent à l'être humain de percevoir l'information. Il peut s'agir des modes visuels, auditifs, olfactifs et gustatifs (associés aux organes correspondant). Enfin, le dernier mode en entrée, appelé TPK (Tactilo-proprio-kinesthésique), est associé aux récepteurs qui sont disséminés dans la peau et les muscles et qui permettent de percevoir la température, la pression, la texture, la forme ou le poids d'objets à travers la perception de notre propre corps.
- Les modes en sortie, qui permettent à l'être humain de produire l'information. Il s'agit du mode oral associé au système vocal et du mode gestuel associé au système musculaire.

La **MODALITE** est définie par la structure de l'information qui est perçue par l'utilisateur (par exemple du texte, une sonnerie, une vibration…). Elle ne doit pas être confondue avec la structure de données utilisée dans le système pour représenter l'information. Par exemple, un même texte peut être représenté par différentes structures de données alors que les utilisateurs caractériseront ces différentes informations de « texte ». Il s'agit donc d'une abstraction de la structure de l'information qui est adaptée à la perception de l'utilisateur. D'après (Truillet 1999), un même mode peut supporter plusieurs modalités. Ainsi, par exemple, le mode auditif peut supporter des modalités sonores verbales et non verbales. Ces modalités sont organisées en une taxonomie par Bernsen (Bernsen 1994) qui décrit les principales caractéristiques de leur structure.

Un **MEDIA** est un périphérique physique qui supporte l'expression d'une ou plusieurs modalités : un écran, un clavier, des haut-parleurs… Par exemple, le média écran peut exprimer les modalités « images » et « texte ». On emploiera le terme **MULTI-MEDIAS** pour

discuter des approches combinant plusieurs médias (Rouillard 2008). Ce terme est à différencier du terme multimédia qui, passé dans le langage courant, désigne un support combinant plusieurs modalités (vidéo, texte et son).

On appelle **PLATEFORME** un ensemble de médias qui sont couplés de façon statique et qui constituent une cible pour l'adaptation de l'interaction. Par exemple, une interface ne sera pas rendue de la même façon (i.e. avec les mêmes modalités) sur un PC et sur un smartphone car ce dernier repose sur le mode tactile et par ailleurs la morphologie de son écran n'est pas la même que celle du PC.

La **MULTIMODALITE** étudie l'usage de plusieurs modalités dans un même système. Elle est structurée par les propriétés du modèle CARE (Coutaz & Nigay 1994) qui permettent de combiner les modalités. Ces propriétés qualifient la multimodalité selon la capacité du système à distribuer l'information entre les modalités. Elles sont au nombre de 4 :

- Complémentarité

  L'information peut être divisée en informations élémentaires qui sont convoyés par différentes modalités de façon complémentaire. Elles sont ensuite fusionnées. Cette complémentarité peut avoir lieu de façon simultané (synergique) ou alternée.

- Assignation

  L'information ne peut être échangée qu'au travers d'une unique modalité.

- Equivalence

  L'information peut être convoyée par plusieurs modalités sans perte de qualité.

- Redondance

  Deux modalités qui sont équivalentes pour une information et qui seraient utilisées en même temps (ou séquentiellement dans une fenêtre de temps réduite) définissent un usage redondant.

III.3.2 TERMINOLOGIE CENTREE SUR LE SYSTEME

L'approche centrée sur le système vise à décrire les concepts d'interaction multimodale en établissant un lien avec l'implémentation et non avec ce que l'utilisateur en perçoit. (Nigay 1994, p. 69) définit les concepts élémentaires de cette approche de la façon suivante :

« Un **LANGAGE D'INTERACTION** *est un système conventionnel structuré de signes qui assure une fonction de communication entre un système informatique et un utilisateur. De manière plus formelle, un langage d'interaction se définit par un vocabulaire d'éléments terminaux et une grammaire*[6].

[6] L'auteur fait ici allusion à la typologie des grammaires de Chomsky. Cependant, elle ne précise pas à quelle classe de complexité le langage d'interaction appartient.

Un **DISPOSITIF PHYSIQUE** *est un transducteur de propriétés physiques : un dispositif physique capte des mouvements de l'environnement (et notamment les actions de l'utilisateur) ; un dispositif de sortie produit des mouvements perceptibles par l'environnement (et notamment par l'utilisateur). »*

Ces définitions sont reprises dans (Bouchet & Nigay 2004) pour introduire la notion de modalité d'interaction. Une **MODALITE D'INTERACTION** y est définie comme l'association d'un dispositif physique d et d'un langage d'interaction L. La modalité d'interaction est donc définie par le couple <d, L>.

Cette approche permet de mieux coller aux contraintes technologiques en offrant une modélisation plus formelle et moins ambigüe (car elle évite l'usage de concepts issus des sciences sociales dont la formalisation en langue naturelle peut laisser place à l'ambigüité). Le dispositif physique est sémantiquement équivalent à la notion de média. La notion de mode quant à elle, n'a pas d'équivalence.

L'inconvénient de cette définition est que la modalité est dépendante du périphérique qui l'implémente. Par exemple, si l'on utilise le langage d'interaction de l'entrée de texte, il peut être implémenté par un clavier ou bien sur un écran (grâce à un widget de clavier virtuel). Les couples <langage d'interaction, périphérique> étant différents dans les deux cas, il s'agit dans cette approche de deux modalités différentes alors que sur le plan cognitif, elle fait intervenir les mêmes connaissances et mêmes symboles. Cette approche est donc utile pour informer le système de comment instancier telle ou telle autre modalité, mais elle n'est pas adaptée à la rédaction de recommandations générales à propos de l'usage ergonomique des modalités.

En contrepoids des propriétés CARE qui sont associées à la perception utilisateur de la multimodalité, les propriétés CASE (Nigay & Coutaz 1993) ont été définies pour décrire, dans une approche orientée système, la nature de l'implémentation de la multimodalité. Ces quatre propriétés sont définies en fonction de 2 axes :

- L'usage des modalités de façon séquentielle ou parallèle
- L'existence ou non d'un mécanisme de fusion/fission

		Usage des modalités	
		Séquentiel	Parallèle
Forme de Fusion-Fission	Combinée	Alterné	Synergique
	Indépendante	Exclusif	Concurrent

FIGURE 9 – L'ESPACE MULTIMODAL ORIENTE SYSTEME CASE

III.3.3 TECHNIQUES DE FUSION/FISSION

En fonction de la nature de la multimodalité, celle-ci peut faire intervenir (mais ce n'est pas obligatoire) des processus de fusion pour les entrées, et de fission pour les sorties. Ces mécanismes supposent une fissibilité de l'information échangée entre le système et l'utilisateur. Elles impliquent donc une structuration hiérarchique de l'information qui doit être accessible au système pour que celui-ci puisse opérer un découpage au niveau sémantique. Ce degré d'abstraction différencie la multimodalité des approches multimédias. Le contenu même de l'information que l'interface permet de manipuler est découpé en unités sémantiques qui sont traitées de façon multimodale en utilisant les propriétés CARE ou CASE.

Différentes approches génériques ont été proposées pour la fusion multimodale : approches à creusets (Nigay 1994), à automates (Bellik 1995) ou bien à règles dans FAME (Duarte & Carriço 2006). La fission multimodale est implémentée à base de règles dans (Rousseau 2006) et à base d'arbres de pondération dans (Jacquet 2006).

Qu'il s'agisse de multimodalité en entrée ou en sortie, les approches proposées ne s'appliquent qu'à des systèmes dans lesquels l'information peut-être facilement et uniformément divisée en sous-unitées de façon à la structurer de façon linéaire ou hierarchique. Dans tous les cas, il appartient au concepteur, dès la phase de conception, de figer ce découpage de l'information. C'est la raison pour laquelle ces systèmes ne sont actuellement utilisés qu'au cas par cas, sur des domaines bien ciblés, et sans possibilité de portage automatique d'un domaine à l'autre.

Un état de l'art comparatif des méthodes de fusion/fission multimodale est proposé dans (Dumas, Lalanne & Oviatt 2009). Ces méthodes n'étant pas automatisables, nous avons décidé de ne pas les placer au cœur de notre recherche. Nous démontrerons cependant dans le chapitre IX que notre approche est compatible avec la fusion/fission lorsque celle-ci est disponible.

III.4 CONCLUSION

Dans ce chapitre, nous avons décrit les outils et méthodes de conception d'IHM qui sont les plus couramment employées. Ces méthodes suivent une approche participative qui implique l'utilisateur dans l'analyse et la spécification de ces besoins dans le système. L'utilisateur est impliqué au travers de différentes techniques (entretiens, expériences en magicien d'Oz…) qui ont été appliquées à l'Ambiant ces dernières années.

Les résultats extraits de la littérature ont ensuite été analysés et catégorisés. Nous avons retenu 4 techniques d'interaction dans la 2ème section que nous avons décrites en illustrant le caractère innovant de chacune d'entre elle.

Ces techniques proposent une adaptation de l'IHM au contexte variable de l'Ambiant par l'utilisation de modalités d'interaction variées. La combinaison de ces modalités est appelée multimodalité. Il s'agit d'une forme d'interaction riche qui permet d'échanger de

l'information en combinant l'usage de plusieurs sens du système perceptif humain. Nous avons défini les deux terminologies qui font autorité en termes de théorie de la modalité.

L'implémentation de ces systèmes multimodaux requiert une architecture adaptée. Nous étudions donc, dans le chapitre suivant, les différentes formes d'architectures logicielles qui permettent de produire des IHM multimodales et qui répondent aux problèmes spécifiques posés par l'Ambiant que nous avions identifiés dans le chapitre II.

Chapitre IV
MODELES D'ARCHITECTURES POUR LES SYSTEMES INTERACTIFS

Le génie logiciel consiste en une démarche de rationalisation des processus de développement de logiciels informatiques. Il consiste en l'analyse des processus de conception qui permettent de produire des systèmes fiables et ergonomiques. Dans ce processus d'analyse rationnelle, l'architecture logicielle décrit de manière symbolique et schématique les différents éléments d'un système informatique ainsi que leurs interactions respectives. L'analyse des besoins est un première étape du génie logiciel qui permet de spécifier ce que le système doit faire, tandis que l'architecture logicielle permet de décrire comment réaliser ces objectifs.

Dans ce chapitre, nous analysons les architectures d'IHM existantes et déterminons leur apport par rapport à l'Ambiant. Nous étudions, dans un premier temps, les architectures utilisées traditionnellement dans l'IHM conventionnelle. Cela nous permet d'identifier les manques et défis architecturaux qui nous permettent de sélectionner 3 types d'architectures qui sont ensuite analysées au cas par cas : les architectures multi-médias, les architecture orientées services et les architectures d'IHM adaptatives.

IV.1 LES ARCHITECTURES POUR L'IHM CONVENTIONNELLE

Le rôle de l'interface utilisateur est de présenter les fonctionnalités du noyau applicatif d'une façon adaptée à l'utilisateur. Or si le noyau applicatif est développé isolément de l'IHM, il peut s'éloigner structurellement des besoins de celle-ci. Pour cette raison, il est souvent difficile d'implémenter l'IHM qui a été conçue durant les étapes préliminaires de l'analyse centrée utilisateur. Le principe de séparation de l'IHM et du noyau applicatif est communément acquis (Myers 1994), notamment dans un objectif de réutilisabilité des fonctionnalités. Cependant, il n'existe pas un unique consensus sur la délimitation des frontières entre les différents éléments constituant le programme.

Nous présentons dans cette section quelques modèles classiques d'architecture qui proposent une délimitation du rôle de chacun des composants : le noyau applicatif, l'interface utilisateur et le moyen dont ces deux entités communiquent. Ces architectures sont de deux formes : elles peuvent être organisées en couches ou en agents. Une comparaison de ces architectures est proposée dans (Bellik 1995).

IV.1.1 ARCHITECTURES EN COUCHES

Le premier modèle à avoir établi une disjonction entre le noyau applicatif et l'interface utilisateur est le modèle de Seeheim (UIMS 1983) (Pfaff 1985), établi lors de la conférence UIMS de 1983 à Seeheim. Il décrit l'interface utilisateur comme étant composée de 3 parties (voir Figure 10) :

- L'interface avec le noyau applicatif
- Le contrôleur de dialogue : Il gère le dialogue avec l'utilisateur en analysant les séquences d'entrées reçues par le composant de présentation. Il fait le lien entre les éléments concrets de l'interface utilisateur et les fonctionnalités du noyau applicatif.
- Le composant de présentation : Il gère les entrées/sorties des périphériques d'interaction

Le lien entre le composant de présentation et l'interface avec le noyau applicatif montre qu'il faut parfois court-circuiter le contrôleur de dialogue lorsque les informations à afficher sont trop complexes ou qu'elles nécessitent un taux de rafraichissement élevé. Cela montre à quel point l'indépendance totale entre l'interface utilisateur et le noyau applicatif est difficile à atteindre.

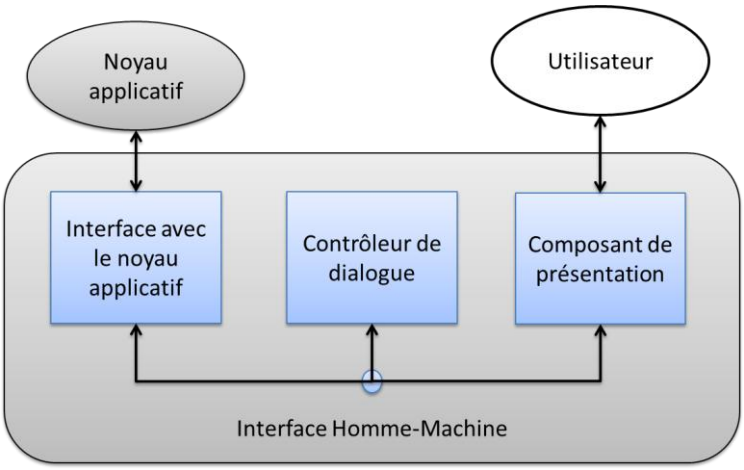


FIGURE 10 – LE MODELE SEEHEIM

Le modèle ARCH (Bass et al. 1992) propose une approche plus proche de l'implémentation. Il s'appuie sur le fait que dans la plupart des cas, le noyau applicatif et les boites à outils pour l'interaction existent hors du développement de l'interface finale. Il s'agit du point de départ et ils apparaissent au niveau le plus bas de la Figure 11. Des adaptateurs sont ensuite introduits pour transformer ces objets en objets du domaine ou en objets de présentation qui sont les entités que le contrôleur de dialogue peut manipuler.

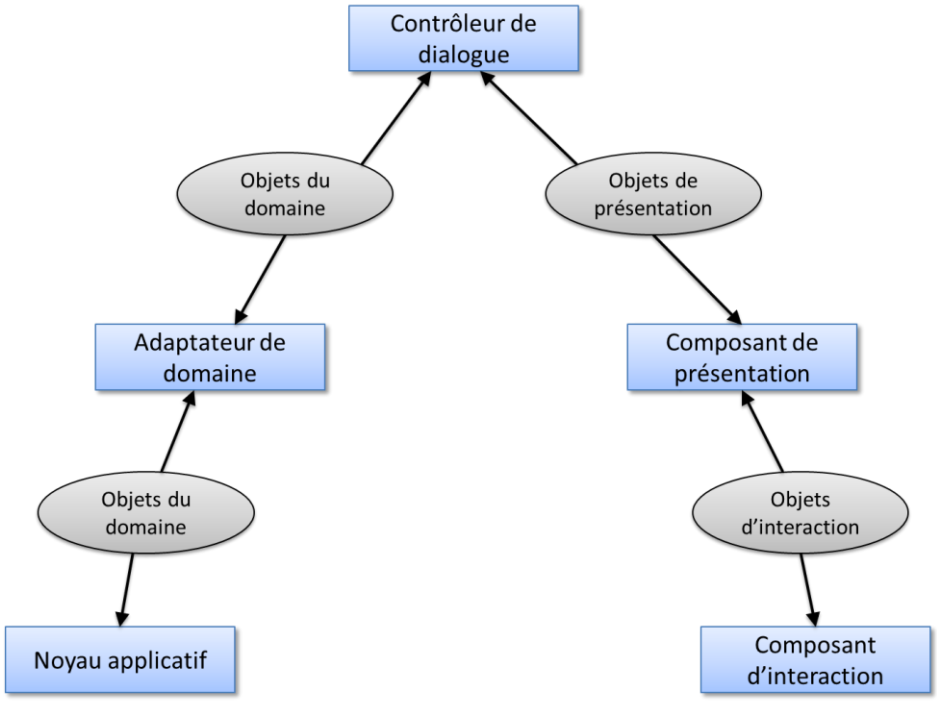


FIGURE 11 – LE MODELE ARCH

Le modèle ARCH est un raffinement du modèle de Seeheim dans lequel le composant de présentation a été divisé en deux parties : le composant de présentation et le composant d'interaction. Cette division permet de distinguer les boites à outils réutilisables qui interfacent les médias avec le contrôleur de dialogue de l'interface. Ce modèle identifie et nomme les objets qui réalisent l'interface entre chaque niveau, et illustre ainsi le besoin de découplage entre les composants d'une application.

IV.1.2 ARCHITECTURES MULTI-AGENTS

Le modèle MVC (Krasner & Pope 1988) est apparu avec Smalltalk. Dans ce modèle, une interface utilisateur est un ensemble d'agents, chacun composé de 3 composants :

- Le modèle : Il définit les fonctionnalités du domaine
- La vue : C'est la partie qui est perçue par l'utilisateur. Elle représente, parfois partiellement, l'état courant du modèle.
- Le contrôleur : Il reçoit les entrées de l'utilisateur et envoie des messages au modèle pour l'informer des actions de l'utilisateur. Il envoie également des messages à la vue pour mettre celle-ci à jour.

Le modèle PAC (Coutaz 1987) représente une hiérarchie d'agents, chacun ayant 3 composants : la présentation, le contrôle et l'abstraction. L'abstraction représente les fonctionnalités du noyau, cette décomposition rappelle donc celle du modèle de Seeheim. La hiérarchie d'agents peut être interprétée de plusieurs façons mais le modèle ne propose pas de méthodologie pour identifier à quel niveau doivent se placer les agents et sur quels critères les subdiviser. En lieu et place, la démarche PAC est appliquée à d'autres modèles comme ARCH par exemple (voir PAC-Amodeus (Nigay 1994)), et démontre l'intérêt de l'usage d'une hiérarchie d'agents sur les infrastructures existantes pour leur apporter

flexibilité et réutilisabilité. Ces différentes approches ont été appliquées à différents domaine. Une analyse détaillée de leurs différentes variantes est proposée dans (Kolski et al. 2009).

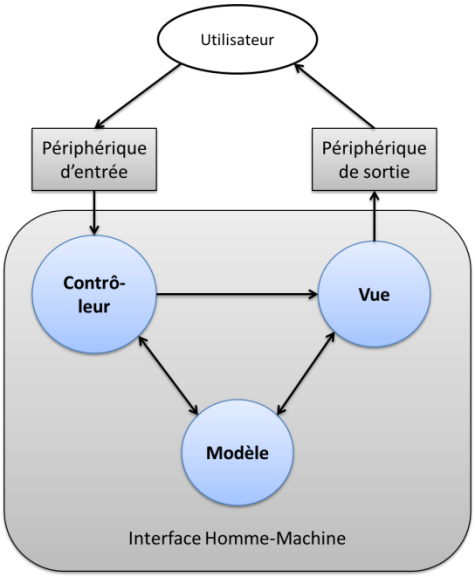


FIGURE 12 – LE MODELE MVC

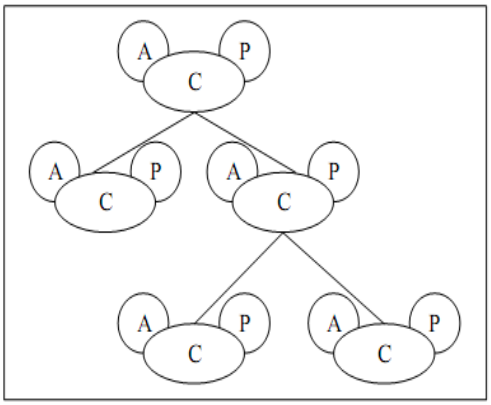


FIGURE 13 – LE MODELE PAC

IV.1.3 LIMITES DE CES ARCHITECTURES

Ces architectures, longtemps utilisées en informatique conventionnelle, ont trouvé leurs limites avec l'informatique ambiante. Elles reposent toutes sur une vision centralisée de l'application et de l'interaction et ne peuvent donc répondre à aucun des problèmes présentés dans la section II.2.

En effet, la variabilité des tâches implique que le noyau fonctionnel soit lui aussi variable, d'une exécution à l'autre, mais également parfois durant l'exécution. Or, ces modèles reposent sur l'hypothèse d'un noyau fonctionnel stable et rigoureusement défini au moment de la conception de l'IHM. De plus, même si ces modèles ne sont pas incompatibles avec la distribution de l'IHM, ils ne rendent pas celle-ci observable au travers des modules logiciels qui y sont définis. Enfin, le contexte est absent de ces architectures.

Des architectures, que nous qualifierons de non-conventionnelles, ont été conçues pour répondre à ces limites. Selon leur approche de la multimodalité et la problématique qu'elles se proposent de résoudre (variabilité de contexte, de plateformes d'interaction, distribution et multiplicité des périphériques…), ces architectures peuvent s'inspirer des approches conventionnelles ou les remettre complètement en question.

Nous présentons dans la prochaine section les architectures multi-médias. Nous rappelons que ce terme définit les approches combinant plusieurs médias pour l'interaction. Il fait référence au terme anglais de multidevice et ne doit pas être confondu avec le terme multimédia (sans tiret) qui désigne un contenu combinant audio, vidéo et texte (Rouillard 2008).

IV.2 ARCHITECTURES POUR LES IHM MULTI-MEDIAS

Dans le but de dépasser les limites des approches classiques, on peut chercher à intégrer de nouveaux périphériques dans la structure des applications interactives. Le paradigme de la programmation par flux a engendré plusieurs systèmes visant à traiter les périphériques d'interaction comme des sources d'événements. Ces systèmes proposent tous une représentation graphique du flux d'évènements qui permet de visualiser et d'éditer le comportement d'une interface.

IV.2.1 ICON

ICON signifie *InputCONfigurator*, il s'agit d'un outil permettant aux programmeurs de composer graphiquement des médias en entrée et de les assembler en techniques d'interaction. L'architecture est décrite dans (Dragicevic & Fekete 2001), elle décrit l'association de composants élémentaires appelés modules.

La méthode de composition employée est celle des flux de données (illustrée par la Figure 14). Un module est un composant de type boite noire qui est caractérisé par ses entrées et ses sorties. Une entrée/sortie est définie par un label et un type de données. L'association entre des modules se fait en reliant les sorties d'un module sur les entrées d'un autre. Cette association n'est possible que si les types sont compatibles. La plateforme se charge ensuite d'interpréter ces diagrammes en temps réel et de synchroniser les échanges d'évènements, ce qui libère le programmeur d'une grande partie de la complexité de la gestion des évènements.

Les médias d'entrée sont représentés par des modules exposant les évènements qu'ils génèrent. Les médias de sortie ne sont pas directement représentés. L'application cible doit exposer les modules de sortie qui permettent de manipuler ses données et d'en faire le rendu. Enfin, des modules représentant des techniques d'interaction ou des outils intermédiaires peuvent être insérés pour faire la jonction entre les deux. La Figure 14 illustre l'utilisation d'ICON pour réaliser une interaction bimanuelle qui permet de tracer des lignes sur une tablette.

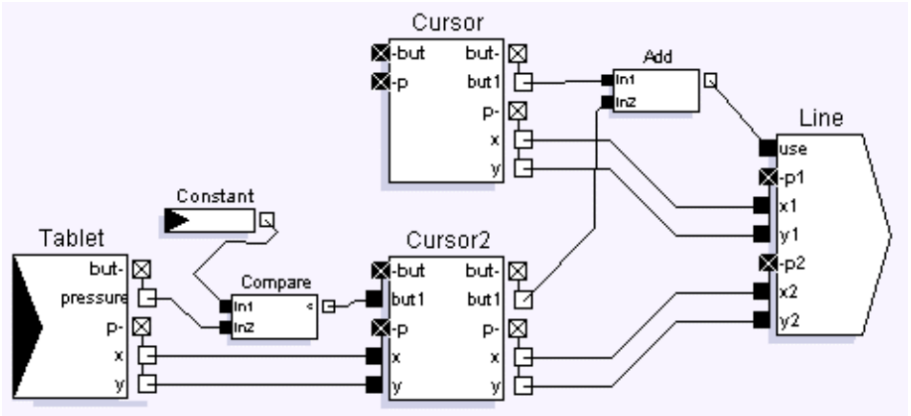


FIGURE 14 – EDITION D'UNE CONFIGURATION MULTI-MEDIAS DANS ICON, EXTRAIT DE (DRAGICEVIC & FEKETE 2001)

Ces modules sont dynamiquement découverts au chargement du programme et peuvent être librement manipulés durant l'exécution, ce qui permet de tester en continu (i.e. sans

redémarrage du système) plusieurs configurations. L'implémentation proposée dans (Dragicevic & Fekete 2001) s'intègre dans un environnement Java Swing mais pourrait être adapté à d'autres boites à outils.

Cette approche a été développée dans le cadre de la plasticité des IHM (Thevenin 2001). Son application dans l'Ambiant est limité par deux facteurs :

- Les périphériques sont recherchés localement et doivent être implémentés en Java. Cette approche répond au problème de la multitude de médias disponibles sans adresser celui de leur distribution et de leur hétérogénéité.
- La composition par flux est un paradigme de programmation qui ne permet pas de représenter simplement des états, des transitions d'états ou des boucles. Or, les techniques d'interaction modernes requièrent des structures de ce type.

IV.2.2 OPEN INTERFACE

Le projet Open Interface (Serrano et al. 2008) s'inspire des principes d'ICON et résout le problème de la distribution et de l'hétérogénéité des médias.

Ce projet a pour vocation de rationaliser et d'unifier les méthodes de conception par association de composants (Lawson et al. 2009). Pour cela, la plateforme propose une base de données de composants gérant des médias d'entrée courants (clavier, souris…) ou moins conventionnels (wiimote, webcam…).

Cette base de données peut être augmentée de nouveaux périphériques hétérogènes (i.e. implémentés dans des langages divers tels que le C++, le Java ou Matlab). Chaque média est décrit dans un langage de description de composants[7] appelé CIDL (*Component Interface Description Language*) qui décrit la structure du composant média indépendamment de son langage d'implémentation. La distribution des périphériques peut également être adressée par l'usage de protocoles réseaux permettant d'échanger les évènements issus des médias. La haute fréquence d'acquisition des médias d'entrée requiert pour cela l'usage de protocoles réseaux et formats de donnés spécifiques tels qu'OSC[8].

De même que dans ICON, les médias de sorties ne sont pas directement représentés, ils sont implicitement contrôlés par des modules de sortie spécifiques à une technique d'interaction. Le modèle de composants d'Open Interface est plus évolué que celui d'ICON et offre la possibilité de sortir du cadre de la programmation par flux pure en intégrant la possibilité d'utiliser des callbacks. Cependant, il ne permet pas non plus de rendre visible les états et transitions qui composent une technique d'interaction.

[7] Cette technique n'est pas sans rappeler le langage IDL (Interface Description Language) défini dans l'architecture de composants hétérogènes CORBA.

[8] Open Sound Control est un format de donnés adapté à l'encodage de flux de données : http://opensoundcontrol.org/

IV.2.3 FLOWSTATES

Flowstates (Appert et al. 2009) est une boite à outils unifiant deux boites à outils antérieures : ICON et SwingStates. Cette dernière, décrite dans (Appert & Beaudouin-Lafon 2008), intègre un modèle de machines à état à la librairie Swing de Java.

Les passerelles entre flots de données et machines à états développées dans Flowstates offrent un grand pouvoir d'expression et une meilleure flexibilité pour le prototypage de l'interaction. L'usage des machines à états de SwingStates est particulièrement approprié à la spécification de la logique de l'interaction tandis que les flots de données et la programmation visuelle d'ICON, se prêtent tout à fait à la spécification de relations continues, parallèles et dynamiques.

IV.2.4 CONCLUSION

Les architectures multi-médias permettent l'implémentation de techniques d'interaction innovantes. Elles sont adaptées à la distribution et à l'hétérogénéité des médias de l'Ambiant et offrent aux développeurs des outils graphiques permettant de rapidement prototyper une nouvelle technique d'interaction. Elles permettent de plus de factoriser les développements en favorisant une approche à base de composants réutilisables.

Cependant, ces architectures ne satisfont pas tous les besoins de l'Ambiant. Tout d'abord, la conception des interfaces y est abordée de manière statique, c'est-à-dire que le concepteur doit développer une interface en composant un ensemble de médias prédéfini. La liste de ces médias n'est pas dynamiquement actualisée et ces approches ne proposent aucun mécanisme d'adaptation automatique des connexions entre composants. De plus, la représentation des médias est limitée aux médias d'entrée. Les médias de sortie sont abstraits au sein de composants qui représentent les techniques d'interaction ou les données du domaine. La composition de médias d'entrée est rendue flexible à travers les connexions de composants alors que la composition de médias de sortie est, si ce n'est impossible, du moins cachée au sein des composants de haut de niveau, ce qui interdit toute flexibilité pour les sorties.

IV.3 ARCHITECTURE POUR LA GENERATION D'IHM ORIENTEES SERVICES

Par opposition à une conception de l'IHM conventionnelle qui est généralement orientée tâches, la programmation orientée services a encouragé, dans certains travaux de recherche, un retour vers une conception de l'IHM guidée par les fonctionnalités disponibles. Dans ce type d'interface, la tâche que l'utilisateur doit/peut réaliser n'est pas nécessairement modélisée ; la conception se focalise sur la présentation des fonctionnalités offertes par l'application (i.e. le noyau fonctionnel) et laisse à la charge de l'utilisateur le soin d'identifier, parmi les fonctionnalités exposées, celles qui satisfont ses besoins. Pour faciliter cette recherche, l'accent est mis sur la cohérence de l'organisation des fonctionnalités (cohérence des menus, filtrage des fonctionnalités…) et la consistance de l'interface produite.

Bien que ces approches aient été un temps délaissées au profit de la conception orientée tâche, elles sont à nouveau étudiées sous le prisme de l'Ambiant. Puisque dans l'Ambiant, l'application est composée de plusieurs services (Rouillard, Vantroys & Chevrin 2007) qui sont décrits de façon standard, il est possible de se servir de ces descriptions pour construire automatiquement une IHM qui expose ces services et qui soit adaptée à la plateforme multimodale disponible. Nous présentons dans un premier temps les résultats de travaux établissant un lien entre la composition de services et celle d'IHM. Nous montrerons que ces travaux, focalisés sur le noyau fonctionnel, ne sont pas applicables tels quels à notre problématique ambiante. Cependant, ils démontrent l'intérêt de l'approche centrée sur les services et inspirent les trois approches que nous présentons par la suite. Ces trois approches ont été séléctionnées pour leurs différences concernant leur modélisation du noyau fonctionnel.

IV.3.1 INTRODUCTION : COMPOSITION FONCTIONNELLE ET LIENS AVEC L'IHM

Nous présentons dans cette section des systèmes permettant la composition fonctionnelle et étudions comment ils permettent d'établir un lien avec la composition d'IHM.

WComp (Tigli et al. 2009) est un framework à composants reposant sur le modèle de composants LCA (Lightweight Component Architecture). De la même façon que pour les approches présentées dans la section précédente (IV.2), les composants représentent des modules logiciels réalisant un ensemble de fonctionnalités formalisées par des contrats.

Dans (Joffroy 2011), une méthodologie est proposée pour l'assemblage d'IHM pré-éxistantes. Cet assemblage d'IHM est guidé par l'assemblage des composants proposant des services du noyau fonctionnel. Ainsi, un noyau fonctionnel est conçu par composition et, de la même façon, une IHM est conçue en assemblant les IHM associées à chacun de ces composants. La sélection des éléments de l'IHM à conserver et à assembler peut être assistée par des outils tels que OntoCompo (Brel & Renevier-Gonin 2011) (Brel et al. 2011). Ces outils permettent au concepteur de naviguer entre les fonctionnalités du noyau fonctionnel et les éléments de l'IHM au moyen d'annotations sémantiques. Ainsi, le concepteur peut identifier les éléments d'IHM liés à une fonctionnalité et vice-versa, ce qui lui permet de réaliser sa sélection et son association.

Ces travaux explorent de façon interessante le lien entre composition fonctionnelle et composition d'IHM. Ils fournissent des outils pour guider le concepteur dans sa conception d'une composition applicative (i.e. fonctionnalités+IHM) mais ne permettent cependant pas l'automatisation du processus de génération de l'IHM. Par ailleurs, ces approches se limitent à l'assemblage d'interfaces graphiques traditionnelles, sans tenir compte d'autres modalités ou de l'introduction de nouveaux périphériques d'interaction. Elles ne couvrent donc pas les besoins idéntifiés dans le chapitre précédent mais ouvrent la voie à de nouvelles approches reposant sur la description des fonctionnalités fournies pour produire une IHM. Nous présentons dans les sections suivantes 3 de ces approches automatisées.

IV.3.2 PUC ET HUDDLE

Le PUC (*Personal Universal Controller*) est un environnement qui permet de générer automatiquement une interface utilisateur en s'appuyant sur une description logique des périphériques et services disponibles dans l'environnement (Nichols 2006). Le domaine

d'application ciblé est celui de l'ambiant à domicile (multimédia, contrôle de l'électro-ménager, contrôle de la domotique).

PUC propose un langage de description des services décrit dans (Nichols et al. 2004). Comme l'illustre la Figure 15, ce langage décrit les variables d'état ainsi que les commandes du service. Il ajoute à cela des informations sémantiques. En l'occurrence, il s'agit de labels permettant de décrire pour l'utilisateur le rôle de la variable d'état ou de la commande ainsi qu'une structure hiérarchique permettant de regrouper les fonctionnalités du service en fonction de leur proximité sémantique.

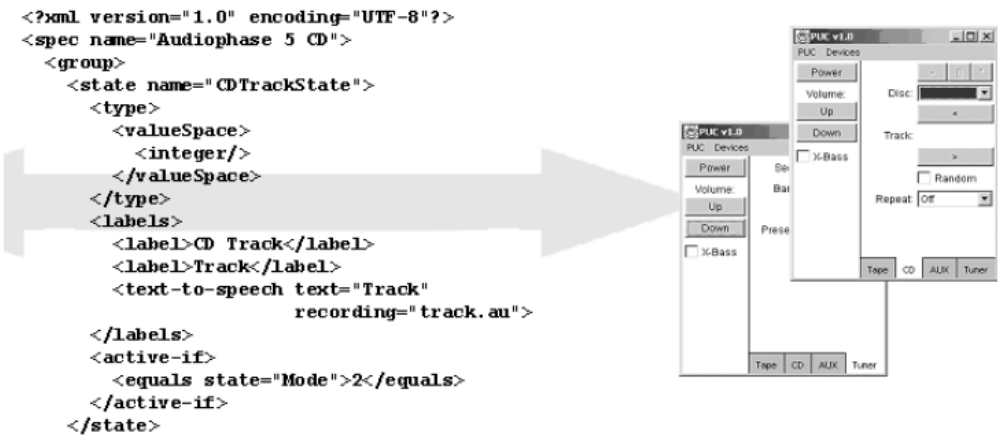


FIGURE 15 – PRINCIPE DE FONCTIONNEMENT DE PUC (EXTRAIT DE (NICHOLS ET AL. 2004))

Cette structure est ensuite raffinée, à l'aide de règles de transformation encodées dans un arbre de décision, pour créer l'interface ainsi que son agencement spatial (si c'est une interface graphique qui est générée). Un moteur de rendu spécifique à chaque modalité d'interaction doit être implémenté et un historique des précédentes instanciations est conservé et réutilisé pour maintenir la consistance des interfaces.

Dans Huddle (Nichols et al. 2006), le système est étendu à la génération d'interfaces combinant différents services qui répondent de façon synergique à un besoin utilisateur. On retrouve ici la notion de tâche qui requiert l'usage de plusieurs services. La navigation entre les services de la même tâche peut être rendue par deux méthodes : une représentation graphique des flux connectant les périphériques, ou bien une simple agrégation graphique (par exemple avec un panneau muni d'onglets).

IV.3.3 SUPPLE

Contrairement au système précédent, SUPPLE (Gajos & Weld 2004) décrit la génération d'une interface adaptée au service comme la résolution d'un problème d'optimisation numérique.

Ce système propose une description des fonctionnalités sous la forme d'un arbre décrivant les fonctionnalités du système et les organisant selon une hiérarchie qui correspond à l'agencement physique. Chaque fonctionnalité est représentée par une variable d'état, c'est-à-dire un type de données ainsi qu'un ensemble de validité pour ces données. Les cibles du rendu sont exclusivement des écrans. Ces derniers sont modélisés par des widgets permettant le rendu des variables d'état et des widgets permettant la navigation dans

l'interface, ainsi que des métriques évaluant les widgets en fonction des types de données à représenter. Les associations <variable d'état, widget> sont toutes considérées et évaluées par les métriques fournies. La solution sélectionnée est l'assemblage qui fournit le meilleur score pour la représentation des variables d'état et la navigation au sein de l'interface.

Diverses métriques ergonomiques peuvent être prises en compte pour évaluer ce score. Les développeurs de SUPPLE définissent ainsi des métriques permettant de prendre en compte (entre autres) : le coût de la navigation entre 2 contrôles, la consistance des interfaces pour un même noyau fonctionnel généré sur différentes plateformes et les capacités physiques (et éventuels handicaps) de l'utilisateur.

L'approche SUPPLE permet de gérer automatiquement les compromis dans la conception d'une interface en explorant l'espace de conception dans son ensemble. Cette approche est donc, en quelque sorte, plus flexible que l'approche à base de règles de transformations dans PUC. Cependant, sa complexité croît exponentiellement avec le nombre de variables et de widgets considérés. Ainsi, les performances de SUPPLE se dégradent lorsque la richesse des interfaces utilisateur augmente.

IV.3.4 CONCLUSION

Dans les architectures d'IHM basées sur les spécifications des services, le concepteur se focalise sur la description des fonctionnalités qu'il souhaite voir s'ajouter à l'environnement ambiant. Il n'a pas besoin d'envisager toutes les configurations de périphériques possibles car c'est le système qui, à partir des fonctionnalités, génère une interface.

Cependant, ces approches ne tiennent pas compte de la notion de tâche, qui est au cœur de la conception moderne des IHM. La conception orientée tâche permet de filtrer, sélectionner, et organiser (dans l'espace et le temps), les fonctionnalités du système qui sont interfacées avec l'utilisateur. Le manque de sémantique liant les services à leurs usages empêche leur liaison automatique avec un formalisme de tâches quelconque. Par conséquent, le risque est grand, dans ces approches, de produire une interface qui soit confuse pour l'utilisateur, car peu ciblée sur ses besoins du moment.

IV.4 ARCHITECTURES POUR LA GENERATION D'IHM ADAPTATIVES

Les IHM adaptatives constituent une classe de systèmes d'interaction homme-machine qui sont capables de s'adapter dynamiquement à un ensemble d'états identifiables du contexte. Lorsque cette adaptation peut se faire au cours d'une même session d'interaction, on parle d'adaptation en ligne (*online adaptation*), ce que nous appellerons également **ADAPTATION INTRA-SESSION**. A contrario, lorsque cette adaptation à lieu entre les sessions d'interaction, nous parlerons d'**ADAPTATION INTER-SESSIONS** (*offline adaptation*).

Nous présenterons tout d'abord, dans cette section, les différentes dimensions identifiées pour l'adaptation des IHM. Puis nous présenterons des modèles d'architecture permettant

une adaptation inter-session. Enfin, nous aborderons le sujet des architectures qui permettent une adaptation intra-session.

IV.4.1 DIMENSIONS DE L'ADAPTATION

L'adaptation d'une interface homme-machine consiste à adapter la forme que celle-ci prend en fonction du contexte d'interaction (Tarpin-Bernard 2006). Le **CONTEXTE D'INTERACTION** est défini par (Dey, Abowd & Salber 2001) 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. »*

Cette définition peut sembler générale mais elle illustre la diversité des situations dans lesquelles l'adaptation peut être nécessaire, diversité qui implique une très grande variété de contextes à prendre en compte. Cependant, on distingue généralement 3 domaines dans le contexte d'interaction (Calvary et al. 2003) (Vanderdonckt et al. 2005) :

- **L'UTILISATEUR**

  Il est décrit par ses capacités (perceptuelles, motrices et cognitives) et ses préférences ou ses caractéristiques culturelles (comme sa langue maternelle par exemple).

- **LA PLATE-FORME**

  Il s'agit de l'ensemble des matériels et logiciels qui prennent part à l'interaction.

- **L'ENVIRONNEMENT**

  Il 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. Dans la suite de ce mémoire, nous y ferons référence à travers les termes d'environnement ou de **CONTEXTE ENVIRONNEMENTAL**.

Tarpin-Bernard (Tarpin-Bernard 2006) précise que l'on ajoute généralement à ce triplet, bien que parfois de façon implicite, une prise en compte de l'activité de l'utilisateur à travers les notions d'activités ou de tâches. La **TACHE** utilisateur traduit les objectifs que cherche à atteindre l'utilisateur lorsqu'il se sert du système. Différentes méthodes et notations ont été proposées pour la décomposer et la structurer (cf. section IV.4.2.i). L'**ACTIVITE** est l'ensemble des actions qui sont réellement effectuées par l'utilisateur. L'activité est en quelque sorte la tâche effective tandis que la tâche est la tâche prévue. En d'autres termes, l'activité représente CE QUI EST FAIT tandis que la tâche désigne CE QUI DOIT ETRE FAIT.

(Tarpin-Bernard 2006) propose également un espace de conception de l'adaptation qui s'articule autour de 6 questions : Pourquoi ? Où ? Qui ? Quand ? Quoi ? Comment ? Cet espace de conception est représenté dans la Figure 16. La figure est séparée en deux parties par l'axe où-qui : au-dessus, se situent les informations qui peuvent susciter l'adaptation, en dessous se situent les différentes solutions qui peuvent y être apportées. Ces travaux sont inspirés de (Malinowski et al. 1992) dans lequel quatre niveaux sont définis pour structurer le processus d'adaptation :

- **INITIATION** : décision d'un acteur de suggérer une adaptation
- **PROPOSITION** : élaboration de suggestions ou recommandations
- **SELECTION** : processus de choix parmi les propositions d'adaptation
- **EXECUTION** : mise en œuvre de l'adaptation choisie

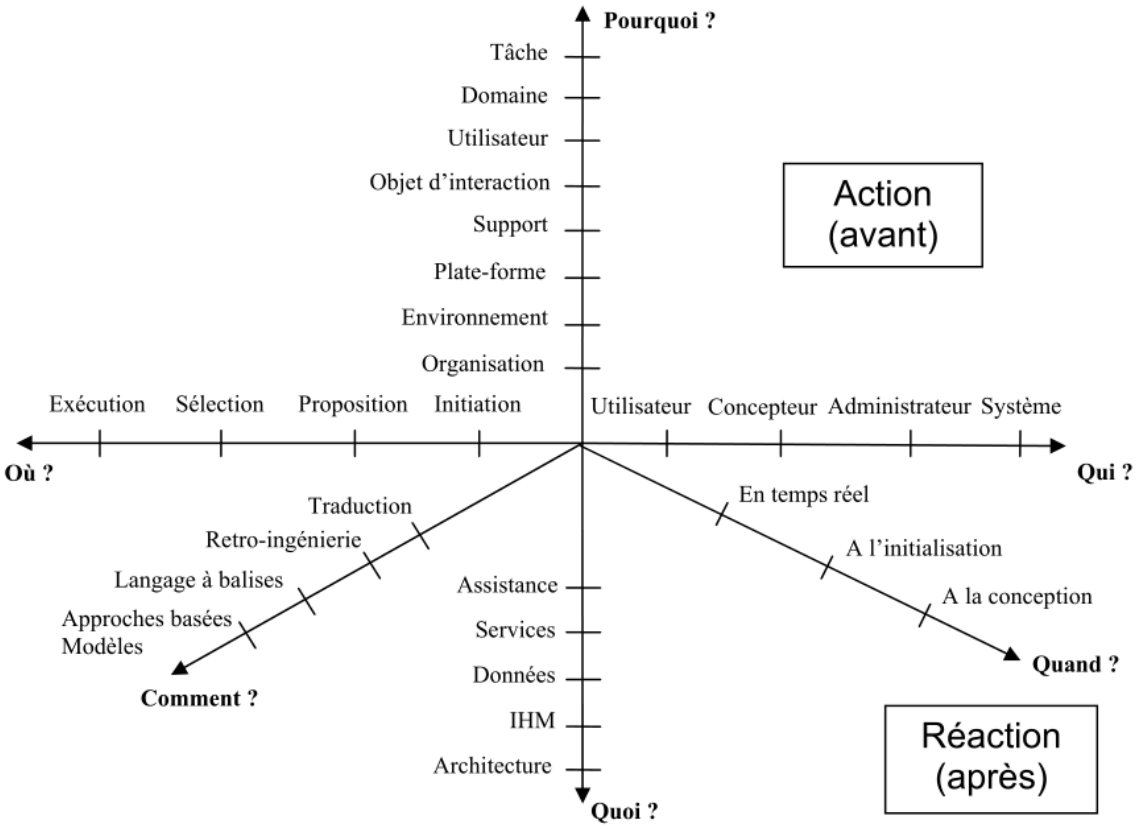


FIGURE 16 – ESPACE DE CONCEPTION POUR LA PRISE EN COMPTE DU CONTEXTE, EXTRAIT DE (TARPIN-BERNARD 2006)

IV.4.2 INGENIERIE DES MODELES ET GENERATION D'IHM (ADAPTATION INTER-SESSIONS)

Nous présentons dans cette section les modèles d'architectures qui permettent une adaptation offline en adoptant une approche de génération automatique (ou partiellement automatique) de l'IHM. Ces approches utilisent une modélisation des différents aspects de l'interaction ainsi que des transformations entre modèles pour générer une interface. Cette méthodologie n'est pas spécifique à l'IHM, on parle plus généralement d'ingénierie dirigée par les modèles (**IDM**).

Dans les architectures d'IHM employant l'IDM, on suit généralement une conception orientée tâches, c'est-à-dire que les interfaces proposées doivent répondre à un besoin de l'utilisateur, par opposition avec une interface conçue pour exposer un certain nombre de fonctionnalités prédéfinies. Dans ces architectures, on identifie différentes catégories de méta-modèles qui définissent une cible pour l'adaptation : tâche, domaine, noyau fonctionnel, composants de l'interface… Ces méta-modèles sont ensuite instanciés en modèles du contexte. Des transformations génériques définies sur les méta-modèles sont

ensuite appliquées entre modèles pour produire, par raffinement successifs, une interface adaptée à la cible (i.e. au contexte d'interaction).

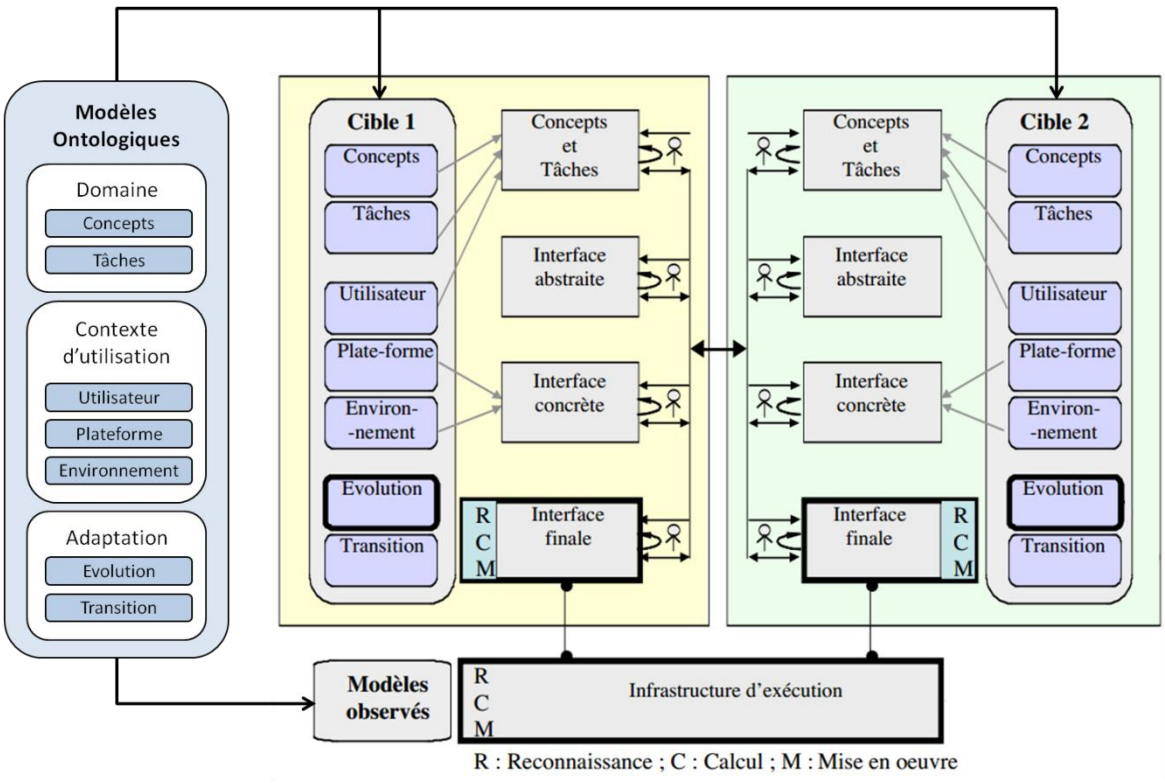


FIGURE 17 – MODELE CAMELEON POUR L'APPROCHE IDM DE L'IHM ADAPTATIVE, ADAPTE DE (CALVARY ET AL. 2003)

Le nombre et la nature des méta-modèles fait débat. Le modèle CAMELEON (Calvary et al. 2003) tente d'harmoniser les approches et les notations en décrivant les différents méta-modèles (appelés modèles ontologiques) qui interviennent dans le processus de conception de l'interface. Elle articule la génération autour de 4 types de modèles :

- Les modèles de tâches et éventuellement du domaine de la tâche

  Le modèle de tâche décrit la tâche comme un ensemble structuré de sous-tâches. Le modèle du domaine décrit les concepts du domaine d'application de la tâche et les données qui lui sont associées.

- L'interface utilisateur abstraite **AUI** (*Abstract User Interface*)

  Le modèle d'interface abstraite décrit des composants d'interface canoniques qui sont indépendants des composants disponibles sur la plateforme. Ces composants représentent des actions courante de l'IHM (par exemple, sélectionner un élément dans une liste) sans préjuger de leur rendu (ni sur le plan de la modalité, ni sur celui de la structure).

- L'interface utilisateur concrète **CUI** (*Concrete User Interface*)

  Ce modèle décrit les composants de l'interface qui vont être réellement utilisés. Ils s'expriment dans un contexte spécifique de modalité et d'apparence générale. Ces composants ne sont pas instanciables, ils représentent les composants que

l'on retrouve généralement dans une boîte à outil (par exemple, les widgets courants tels qu'une case à cocher).

- L'interface utilisateur finale **FUI** (*Final User Interface*)

  L'interface finale est le module logiciel qui va être exécuté sur la plateforme. Il peut s'agir des composants d'une librairie (par exemple, le widget case à cocher de la librairie GTK). Ce modèle décrit des structures fortement dépendantes de la plateforme cible.

Le passage d'un niveau d'abstraction à un autre s'opère par raffinements successifs à partir des informations qui décrivent la cible de l'interaction. Ces informations sont contenues dans des méta-modèles (les modèles ontologiques) dont les instances varient pour chaque cible. Ces raffinements peuvent être automatisés ou bien faire intervenir le concepteur.

La modélisation de la tâche et les langages de description de l'interface sont au cœur de cette famille d'approches. Nous allons donc nous intéresser dans un premier temps aux méthodes de modélisation de la tâche, puis aux langages de description d'IHM, avant de présenter des exemples d'infrastructures d'exécution permettant d'implémenter cette méthode.

IV.4.2.i MODELISATION DE LA TACHE

Les premiers modèles de tâches sont issus du domaine de l'ergonomie. Un des modèles les plus anciens de cette catégorie est GOMS (John & Kieras 1996). GOMS spécifie 4 catégories de concepts : les buts de l'utilisateur (Goals), les Opérateurs, les Méthodes et les règle de Sélection. Ce modèle est une approche cognitiviste permettant d'évaluer la charge cognitive de la réalisation d'une tâche. Il permet de modéliser les règles de transition entre les différents états cognitifs de l'utilisateur mais ne permet pas de modéliser les variables du domaine de la tâche. Ce modèle n'est donc pas utilisable par la machine, son objectif se limite à l'évaluation et l'optimisation d'une interface durant sa phase de conception. Cependant, ce modèle a posé les bases d'une représentation des tâches sous la forme d'une hiérarchie permettant l'abstraction d'un but en un ensemble de sous-buts et pose la question des relations qui existent entre les différentes tâches.

Cette question est plus formellement détaillée dans les modèles K-MAD (Baron et al. 2006) et CTT (Mori, Paterno & Santoro 2002) qui proposent une représentation du domaine de la tâche permettant de représenter formellement les variables et les transitions d'états du système. Ces modèles proposent une représentation de l'utilisateur très limitée ne faisant pas intervenir son état cognitif. Cette simplification se justifie par la finalité différente de ces modèles qui est d'automatiser certaines phases de la conception de l'interface. La représentation des tâches dans ces deux modèles est une hiérarchie permettant d'abstraire les tâches et de les raffiner en un ensemble de tâches élémentaires liées par des opérateurs temporels. Les deux principales différences entre ces modèles sont :

- K-MAD permet d'assigner des préconditions et des post-conditions à ses tâches. Ce qui permet de modéliser ce que doit réaliser la tâche, et qui peut aider à la construction automatique de réseaux de tâches. CTT n'offre pas un tel mécanisme.
- CTT propose un modèle d'enchainement des tâches basé sur une relation binaire entre deux tâches. Cette relation binaire s'ajoute à « l'arbre » des tâches dont la

structure est donc plutôt un graphe acyclique orienté. Par comparaison, dans K-MAD, ces opérateurs temporels permettant de lier les tâches entre elles sont des attributs des nœuds s'appliquant sur les enfants de la tâche. K-MAD mélange donc les relations « abstraction/raffinement » et « enchainement temporel » dans la même structure tandis que CTT utilise deux modes de représentation indépendants, offrant ainsi une plus grande flexibilité.

Dans les travaux qui sont présentés par la suite, l'usage du modèle CTT est prédominant. On pourra se référer à (Charfi 2009) pour une présentation plus détaillée de K-MAD.

La Figure 18 donne un exemple de représentation d'une tâche en CTT. Il s'agit d'un extrait[9] d'une tâche consistant à retirer de l'argent dans un distributeur. On peut y distinguer trois types de tâches supportées : les tâches systèmes (représentées par un ordinateur), les tâches d'interaction (représenté par un utilisateur face à un écran) et une tâche abstraite (représentée par un nuage). La structure hiérarchique permet de décomposer une tâche tandis que les liaisons horizontales décrivent les relations temporelles entre tâches d'un même niveau. On peut observer, par exemple, que la tâche « valider montant » ne peut être réalisée qu'après la tâche « demander montant » à cause de l'opérateur séquentiel qui les lie (>>). Les tâches « prendre argent » et répondre question » peuvent, quant à elles, être réalisées simultanément (opérateur |||). Dans le cas où plusieurs tâches sont liées entre elles par différents opérateurs temporels, un ordre de précédence est appliqué permettant de garantir l'unicité de l'interprétation de ceux-ci.

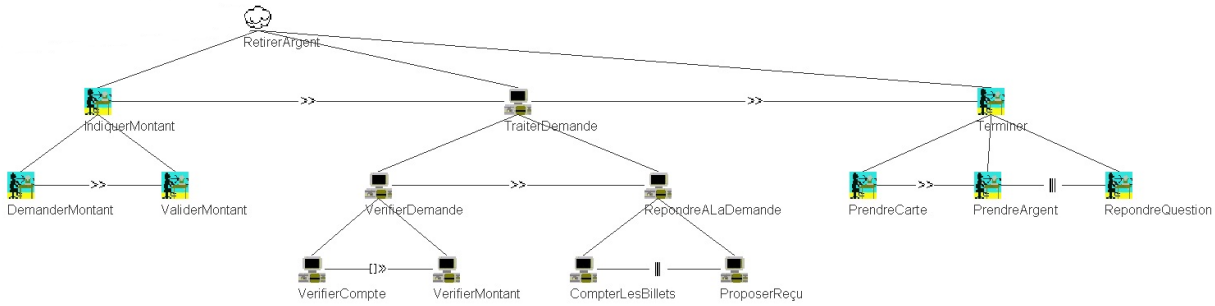


FIGURE 18 – EXEMPLE D'ARBRE CTT POUR LE RETRAIT D'ARGENT A UN DISTRIBUTEUR

IV.4.2.ii LANGAGES DE DESCRIPTION DE L'INTERFACE

Les langages de description d'interface utilisateur (**UIDL** : *User Interface Description Language*) sont apparus il y a une quinzaine d'années dans l'objectif de fournir un moyen formel de décrire des interfaces utilisateurs qui soient indépendantes de la plateforme de rendu. Cette indépendance se traduit différemment en fonction du langage, en s'appliquant à différents niveaux allant du plus concret (indépendance vis-à-vis du système d'exploitation ou au langage de programmation) au plus abstrait (indépendance vis-à-vis

[9] Cette tâche devrait être précédée d'une tâche d'identification visant à sécuriser la transaction par l'entrée d'un code secret. Pour conserver la lisibilité de la figure, nous ne faisons pas apparaître cette étape.

des modalités mises en œuvre pour orchestrer l'interaction). Un UIDL définit un format de fichier (le plus généralement dérivé d'XML) pour décrire les différents composants de l'interface. Ces descriptions sont ensuite interprétées par une plateforme logicielle qui les concrétise.

Nous avons choisi de présenter ici un aperçu de trois de ces langages qui sont représentatifs des UIDL. Il existe cependant d'autres langages et nous invitons le lecteur à se référer à (Stanciulescu 2008, p. 32) pour une analyse comparée des langages existants.

UIML

UIML (*User Interface Markup Language*) (Abrams et al. 1999) est un métalangage de description d'interfaces utilisateur. Ce métalangage propose un squelette de balises indépendantes de la plateforme système, du rendu et du langage de programmation de l'application. Le concepteur doit alors étendre ce vocabulaire à l'aide d'un ensemble de balises spécifiques aux besoins de l'application, ce qui permet au moteur de rendu de faire le lien entre les balises génériques du langage et la plateforme cible.

En ne faisant aucune hypothèse quant au contenu présenté où à la nature de la plateforme, ce langage reste ouvert aux évolutions. La description de l'interface est décomposée en cinq parties : sa composition, sa structure, ses données, son rendu et ses événements. Un moteur de rendu doit bien entendu être implémenté pour chaque plate-forme cible (de la même façon que pour HTML) et des outils tels que TIDE (Ali et al. 2002) ont été développés pour faciliter le développement d'interfaces UIML en affichant en temps réel le rendu du fichier édité.

Contrairement aux langages qui suivent, ce langage n'offre qu'UN NIVEAU D'ABSTRACTION UNIQUE : il permet d'abstraire les widgets. Il ne permet cependant pas de modéliser la tâche, l'application ou encore de généraliser les composants de l'interface au point de permettre un rendu multimodal.

TERESAXML

TeresaXML (Berti et al. 2004) a pour but de décrire des interfaces utilisateur multiplateformes. Il couvre les modèles abstraits et concrets (AUI et CUI) de la démarche de conception présentée en introduction. Ce langage est fréquemment utilisé en conjonction avec le langage CTT pour la modélisation de la tâche. Il a été conçu dans le cadre de l'approche Teresa qui est décrite dans la prochaine section.

USIXML

UsiXML (*User Interface eXtensible Markup Language*) (Stanciulescu et al. 2005) permet de représenter 3 niveaux d'abstraction de l'interface: l'interface utilisateur abstraite (AUI), l'interface utilisateur concrète (CUI) et l'interface utilisateur finale (FUI).

Ce langage ajoute également deux niveaux d'abstraction qui correspondent à la description du noyau fonctionnel de l'application : les modèles de la tâche et du domaine. Cette décomposition confère au langage une indépendance vis-à-vis de la plateforme, des modalités employées et du contexte d'utilisation. Le concepteur peut ainsi spécifier l'ensemble de l'interface adaptative en utilisant chacun de ces niveaux.

Les UIDL pourraient sembler, à première vue, être de bons candidats comme modèles de l'interface puisqu'ils sont une abstraction de celle-ci. Cependant, les UIDL permettent seulement de décrire le contenu de l'interface. Or, on attend d'un modèle d'interface la capacité de raisonner sur les modalités entrant en jeu, sur les métaphores et techniques d'interaction appliquées.

La différence principale avec les modèles conceptuels de l'interaction réside donc dans la granularité à laquelle ils s'appliquent. Les UIDL sont de bons formats pour représenter la structure d'une interface. En tant que tels, ce seront de bons candidats lorsqu'il s'agira d'établir un lien entre les raisonnements faits sur le modèle d'interaction et leur instanciation concrète, comme nous allons le voir dans la section suivante à propos des plateformes de génération d'interfaces.

IV.4.2.iii PLATEFORMES D'EXECUTION ET OUTILS

TERESA/USIXML

TeresaXML et UsiXML sont des langages de description des interfaces qui s'appuient sur la structure proposée dans Cameleon. A ces langages de description sont associés un ensemble d'outils d'édition graphiques qui permettent de spécifier les liens entre les différents niveaux des modèles et leurs raffinements.

Cette suite d'outils s'appelle Teresa pour TeresaXML, elle est décrite dans (Mori, Paterno & Santoro 2002). Ces outils visent à fournir au concepteur une méthode de navigation entre les modèles de Cameleon, que ce soit dans le sens du *forward engineering* ou du *reverse engineering*. Ces outils définissent avant tout une méthodologie de conception qui permet au concepteur de construire des IHM adaptées qui conservent une certaine cohérence d'une instanciation à l'autre. L'automatisation n'en est pas son but premier bien que des extensions qui vont dans ce sens, telles que MARIA (Paterno, Santoro & Spano 2011), aient récemment vu le jour.

La suite logicielle équivalente pour UsiXML est décrite dans (Stanciulescu et al. 2005). L'approche UsiXML est résolument orientée vers l'automatisation de la génération des IHM adaptées. Comparée à Teresa, elle ajoute pour cela deux éléments essentiels :

- Une modélisation du domaine de la tâche (et donc du noyau fonctionnel à interfacer).
- Un mécanisme de réécriture des modèles permettant de passer automatiquement d'un modèle à l'autre. Ce mécanisme repose sur des règles de réécritures qui sont fournies par le concepteur lors de la phase de modélisation du système.

UsiXML a été utilisé pour générer des interfaces monomodales (graphiques) et multimodales (graphique+parole). Cependant, les résultats de ces générations automatiques sont limités à une interaction stéréotypée, sous forme de formulaires. Elle ne convient pas à la génération d'interfaces contenant des techniques d'interaction non conventionnelles.

CAMELEON-RT

Cameleon-RT (Balme et al. 2004) est une architecture visant à l'implémentation du modèle CAMELEON. Elle se destine à 3 formes d'adaptation : la distribution, la migration et la plasticité. Pour définir ces notions, il est d'abord nécessaire d'introduire la notion d'espace interactif. Un **ESPACE INTERACTIF** regroupe 3 types d'entités :

- L'espace physique dans lequel l'interaction a lieu
- Les ressources informatiques pour l'interaction, le calcul et les communications réseau
- Les services et fonctionnalités du monde digital qui s'appliquent dans cet espace

A partir de cette définition, Balme définit une **PLATEFORME** comme l'entité qui gère et contrôle les ressources d'interaction d'un espace interactif. Le nombre d'ordinateurs impliqués dans la plateforme permet de la qualifier d'**ELEMENTAIRE** (un seul ordinateur) ou de **CLUSTER** (plusieurs ordinateurs).

Ainsi, on peut définir les termes « distribuées », « migrantes » et « plastiques » de la façon suivante :

- Une **INTERFACE DISTRIBUEE** est une interface qui utilise des ressources d'interactions issues d'un même cluster. Si une interface est rendue au moyen d'une unique plateforme élémentaire, elle n'est pas distribuée.
- La **MIGRATION D'UNE INTERFACE** est le transfert d'une partie ou la totalité du rendu de l'interface sur d'autres ressources d'interaction (que celles-ci appartiennent ou non à la plateforme en cours).
- Une **INTERFACE PLASTIQUE** est une interface ayant la capacité de s'adapter aux changements de l'espace interactif tout en préservant son utilisabilité.

Le modèle Cameleon-RT propose un découpage de l'interface graphique et des composants génériques fournissant les services nécessaires à son exécution. Sur la Figure 19, cela correspond à la couche DMP (*distribution, migration, plasticity*). Cette couche est située entre la couche de la plateforme et celle du système interactif (qui est en fait un système à base de composants). Elle contient l'infrastructure nécessaire à la gestion du contexte, à son interprétation et à sa synthèse. Les situations identifiées déclenchent, au sein d'un moteur d'évolution, le raisonnement automatique pour la génération de solutions d'adaptation de l'interface.

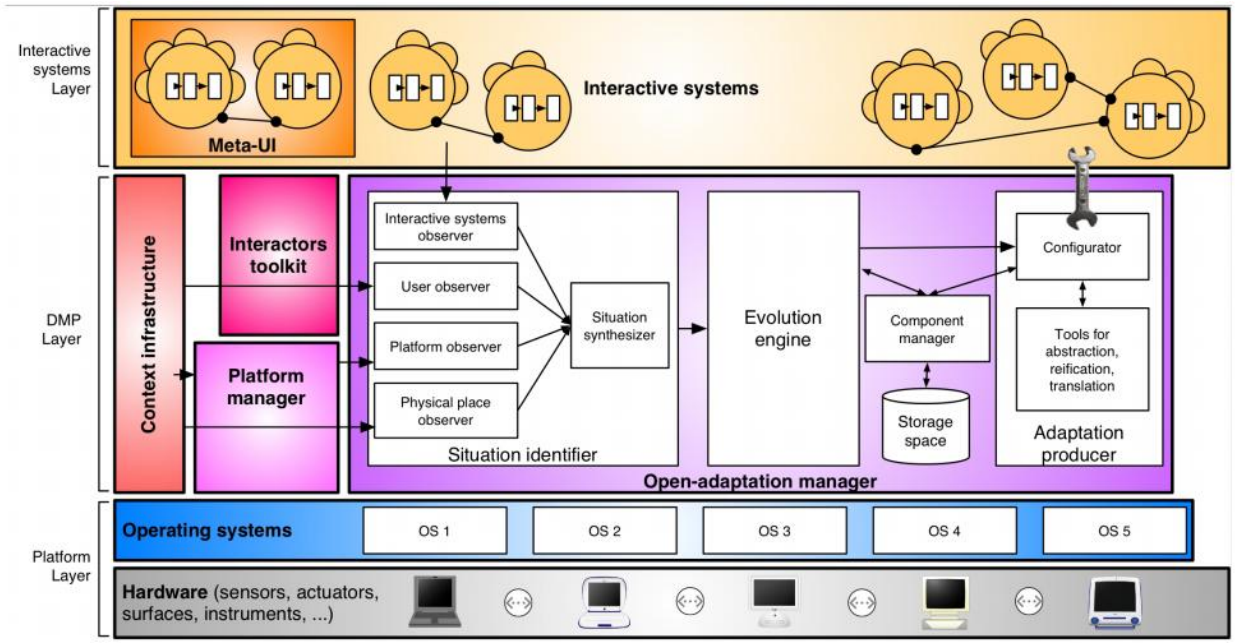


FIGURE 19 – ARCHITECTURE DE LA PLATEFORME CAMELEON-RT, EXTRAIT DE (BALME ET AL. 2004)

IV.4.2.iv CONCLUSION DE LA GENERATION D'IHM

Les systèmes de génération automatique étudiés ci-dessus se divisent en deux catégories : ceux qui sont orientés tâches et ceux qui sont orientés fonctionnalités.

Les premiers permettent de répondre aux besoins réels de l'utilisateur. Ils offrent à celui-ci une composition des services qui s'oriente vers un objectif bien défini. Malheureusement, ils reposent sur une modélisation préalable détaillée des différentes étapes du processus. Or, de la même façon qu'il est impossible de prévoir quels seront les matériels disponibles dans un environnement, il est impossible d'anticiper toutes les tâches que l'utilisateur pourrait vouloir réaliser avec. Ainsi, ces approches se restreignent à un ensemble de scénarios prédéfinis. Enfin, elles requièrent une étape d'identification et d'association des services qui doivent réaliser les tâches élémentaires, dont l'implémentation n'est pas triviale.

Les systèmes orientés fonctionnalités, quant à eux, sont susceptibles de présenter à l'utilisateur une interface qui ne coïncide pas avec ce qu'il cherche à faire. Identifier et sélectionner les fonctions utiles à un but particulier parmi l'ensemble des fonctionnalités qu'offre un certain matériel a un coût cognitif élevé. De plus, d'un périphérique à l'autre, la structure des fonctions n'est pas la même, l'expérience acquise par l'utilisateur n'est donc pas transposable. Malgré tout, les approches d'analyse de traces permettent de faire un choix dans les fonctions à présenter en priorité.

Du point de vue de l'interaction en elle-même, la génération automatique n'atteint pas encore la richesse permise par l'intelligence ambiante. Aucune des techniques d'interaction innovantes présentées dans la section III.2 ne peut être instanciée, à l'heure actuelle, par un système de génération automatique. Ces derniers restent souvent prisonniers de la modalité graphique et, même lorsqu'ils proposent de la multimodalité, se limitent à des présentations sous forme de formulaires graphiques et vocaux. Ces limites résultent de l'absence d'un méta-modèle de dialogue qui soit modulable (i.e. permettant d'ajouter de

nouveaux modèles de dialogue adaptés aux nouvelles techniques d'interaction) et multimodal (i.e. qui permette de tisser des liens entre les différents modèles de dialogues).

IV.4.3 EVOLUTION DE L'IHM EN COURS DE SESSION (ADAPTATION INTRA-SESSION)

Nous présentons dans cette section, différentes approches ayant recours à une modélisation du comportement adaptatif dynamique (i.e. en cours de session) de l'interface. Dans le cas général, ces modèles sont appelés **MODELES COMPORTEMENTAUX** et peuvent se présenter sous différentes formes (règles, arbres de pondération…).

FAME

L'architecture FAME (Duarte & Carriço 2006) est une architecture multimodale qui propose à la fois des mécanismes de fusion et de fission. Le concept central est celui de composant adaptable. Un composant adaptable réalise un besoin de l'utilisateur (par exemple, la navigation dans un fichier) et propose différentes implémentations reposant sur différents médias. Chacune de ces implémentation est appelée configuration. Un même composant adaptable est donc susceptible d'être rendu selon différentes modalités en fonction des médias disponibles.

FAME ne propose pas de méthode de génération automatique, mais utilise les différentes configurations des composants adaptables pour permettre une adaptation dynamique au contexte. A chacun de ces composants est associée une **MATRICE COMPORTEMENTALE** qui définit les règles qui gouvernent la configuration du composant adaptable. Chaque dimension de la matrice représente une des propriétés du composant (modalité de sortie utilisée, représentation simple ou détaillée des données…). A chaque case est donc associée une configuration particulière. Chaque case contient un triplet qui indique si la configuration est permise, combien de fois elle a été activée, et un seuil d'activation. Le nombre d'activations précédentes de la configuration permet de conserver une trace qui peut être utilisée, *a posteriori*, pour analyser la fréquence des situations. Le seuil d'activation est une fonction booléenne qui peut faire intervenir des informations provenant du contexte et permet de désactiver certaines configurations lorsque les conditions sont défavorables. Ce seuil peut également faire intervenir la trace des activations précédentes de la configuration, ce qui permet de modéliser des phénomènes d'apprentissage ou de gérer des transitions entre configurations par exemple.

L'approche FAME reste cependant assez abstraite quant à l'implémentation du système de fusion/fission. Par ailleurs, la gestion des conflits entre règles n'est pas abordée.

WWHT

Le modèle WWHT (*What, Which, How, Then*) (Rousseau 2006) est une approche de conception de présentations multimodales de l'information. Ce sont des interfaces qui ne comportent que des sorties. Ce modèle propose l'emploi d'un **MODELE COMPORTEMENTAL** à base de règles pour définir l'adaptation de la présentation en cours d'exécution.

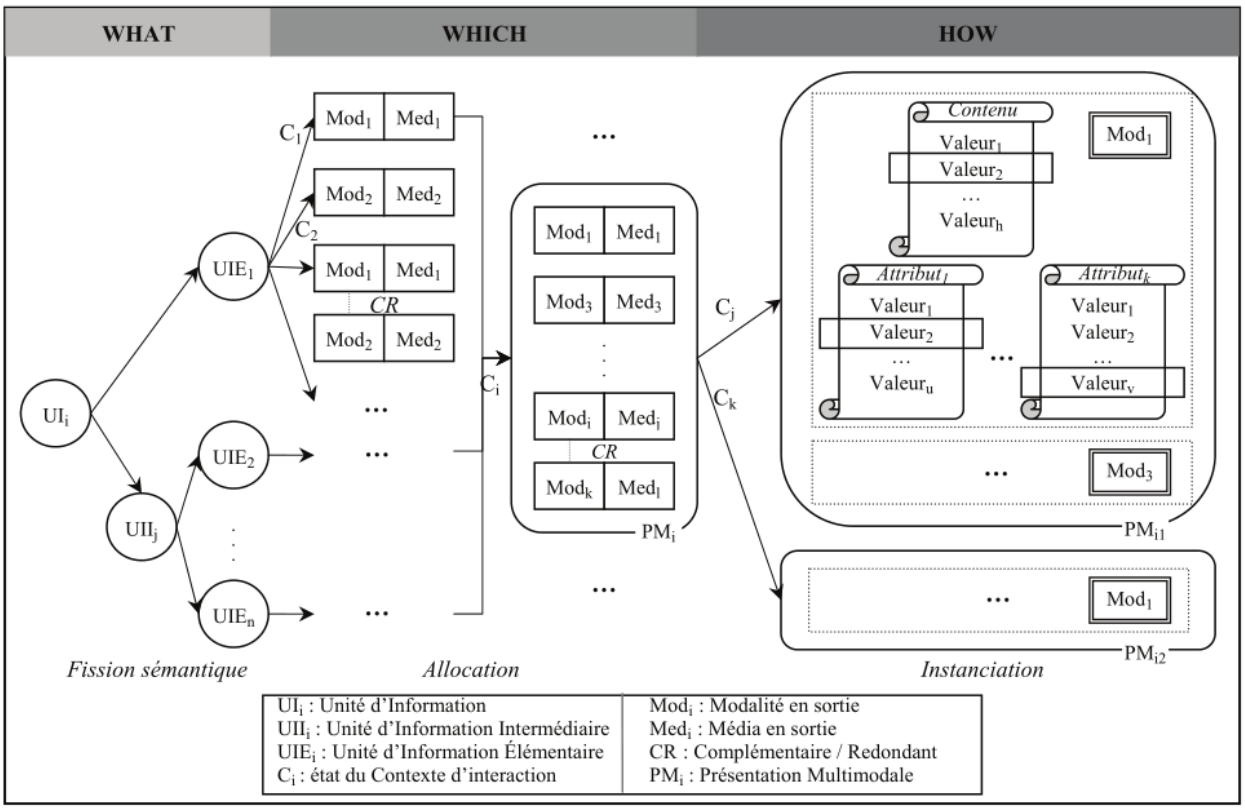


FIGURE 20 – LES TROIS PREMIERES ETAPES DE WWHT, EXTRAIT DE (ROUSSEAU 2006)

WWHT s'appuie sur un moteur de division sémantique de l'information à présenter en une hiérarchie d'informations élémentaires. Il produit ensuite une présentation adaptée au contexte en articulant la procédure autour de 4 questions qui sont résolues à l'exécution (cf. Figure 20) :

- *What*: Quelle information présenter?

  Il s'agit de la phase de **FISSION SEMANTIQUE** qui consiste à diviser l'information à présenter en unités d'informations élémentaires.

- *Which*: Quelle présentation multimodale choisir?

  Il s'agit de la phase d'**ALLOCATION**, qui sélectionne, pour chaque unité sémantique, la présentation adaptée. Une présentation est définie comme l'association d'une unité sémantique, d'une modalité et d'un média. Les triplets sont sélectionnés sur la base d'une évaluation effectuée par un ensemble de règles appelé modèle comportemental. Un exemple de tels modèles est proposé dans la Figure 21, à propos de la présentation de l'information « réception d'un appel ».

  Le système de règles gère les conflits entre règles par une méthode d'élection (chaque règle apporte des votes à une solution candidate). Après allocation des unités informationnelles, un second tour est organisé pour définir les candidats qui pourraient être instanciés en complémentarité ou en redondance des candidats primaires. Ce tour n'implique que les règles de type « composition » (cf. Figure 21) et permet l'usage automatique des propriétés CARE dans la génération de l'interface.

- *How*: Comment instancier cette présentation?

  Il s'agit de la phase d'**INSTANCIATION** : des attributs morphologiques sont sélectionnés pour chaque présentation (taille et couleur de texte, niveau sonore d'une sonnerie…). Pour cela, un ensemble de feuilles de styles prédéfinies définissent les valeurs requises en fonction des variables d'état du contexte. Les feuilles de styles adaptées au contexte courant sont sélectionnées et fusionnées pour fournir les attributs qui caractérisent la présentation à instancier.

- *Then*: Comment faire évoluer cette présentation?

  L'approche WWHT définit 2 types d'évolution d'une présentation : le raffinement et la mutation. Le premier type d'évolution intitulé « raffinement » ne change pas les composants d'interaction (modalités, médias) utilisés par la présentation mais leurs instanciations (i.e. leurs attributs morphologiques). Le second type intitulé "mutation" est un changement radical des modalités et/ou médias composant la présentation (remise en cause de la phase d'allocation).

  Ces évolutions sont déclenchées par un système de gestion du contexte. Dans (Rousseau et al. 2006), Rousseau propose deux implémentations différentes de systèmes de gestion de contexte, l'une centralisée et l'autre distribuée.

Id	Description en langage naturel	Type
R1	**Si** l'utilisateur est malvoyant **Alors** ne pas utiliser le mode Visuel	Contextuel
R2	**Si** le haut-parleur est occupé **Alors** ne pas utiliser la modalité Sonnerie	Contextuel
R3	**Si** le niveau sonore est supérieur à 90 dB **Alors** ne pas utiliser le média Haut-parleur	Contextuel
R4	**Si** l'UIE est un « appel » **Alors** favoriser la modalité Vibration **Et** recommander la modalité Sonnerie	Contextuel
R5	**Si** l'UIE est un « appel » **Et** le téléphone est en mode renforcé **Alors** utiliser une redondance	Composition
R6	**Si** l'UIE est un « appelant » **Alors** favoriser les modalités fortement Analogiques	Critériel
R7	**Si** l'UI est un « appel de X » **Et** le niveau de batterie est faible **Alors** éviter le média Vibreur	Contextuel

FIGURE 21 – EXEMPLE DE MODELE COMPORTEMENTAL DANS WWHT, EXTRAIT DE (ROUSSEAU 2006)

KUP

L'architecture KUP[10] (*Knowledge, User, Presentation*) (Jacquet 2006) est une architecture à agents qui vise à présenter de façon opportuniste des informations dans l'Ambiant. De même que dans WWHT, la génération et l'adaptation des interfaces se limitent ici à des modalités de sortie.

[10] Cet acronyme est traduit par son auteur par : Information, Utilisateur, Présentation.

Dans le modèle KUP, chaque agent dispose d'un **ESPACE DE RAYONNEMENT**, zone géographique dont la forme et la distance dépend de l'agent et qui correspond à la possibilité de cet agent d'interagir avec un autre agent. Lorsque l'agent représente un média par exemple, l'espace de rayonnement est l'ensemble des points de l'espace où ce média peut être perçu (avec le sens adapté au mode employé par le média). Par exemple, l'espace de rayonnement d'un écran se propage devant lui mais pas derrière car un utilisateur placé derrière ne pourrait percevoir ce qui est affiché à l'écran. L'espace dual de l'espace de rayonnement est appelé **ESPACE PERCEPTUEL**. Il s'agit de l'ensemble des points de l'espace, décrits par rapport à un agent (par exemple l'utilisateur), où ce dernier peut percevoir une autre entité.

Le modèle KUP permet de séparer la phase de fourniture d'une information de sa phase de présentation. Lorsqu'un utilisateur (U) pénètre dans l'espace de rayonnement d'une source d'informations (K), celle-ci lui fournit une ou plusieurs unités sémantiques pertinentes. Il est possible qu'au moment où l'utilisateur reçoit ces unités sémantiques, aucun dispositif de présentation (P) ne soit à proximité (au sens de l'espace de rayonnement). Cependant vu que les utilisateurs sont mobiles, il est possible qu'ultérieurement, un dispositif de présentation pénètre dans l'espace de perception de l'utilisateur. Ceci provoquera alors un événement de proximité qui aura pour effet de déclencher le processus de présentation (ou d'évolution) des unités sémantiques de l'utilisateur sur le dispositif en question.

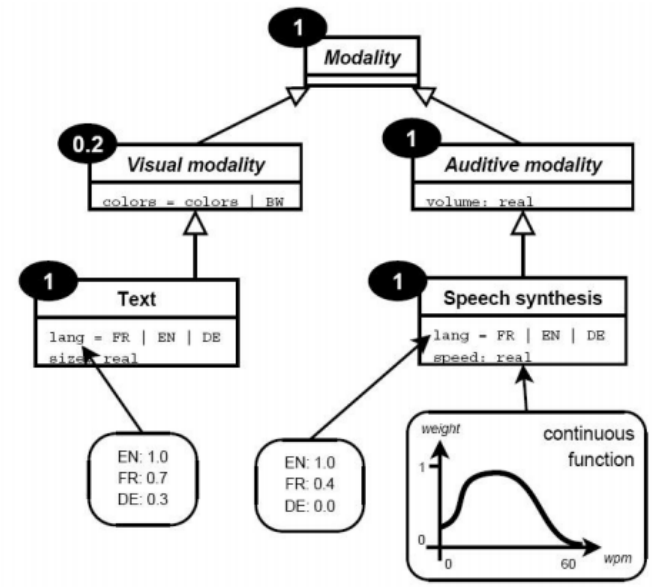


FIGURE 22 – PROFIL PARTIEL DU MODELE KUP, EXTRAIT DE (JACQUET 2006)

Dans ce modèle, chaque agent peut exprimer des contraintes et des préférences différentes en termes de modalités. Les agents partagent une même définition de l'espace de conception sous la forme d'un arbre taxonomique des modalités. Les modalités sont hiérarchisées dans cet arbre dont les nœuds et les feuilles sont étiquetés par les attributs morphologiques qui impactent l'instanciation de la modalité. Chaque agent définit indépendamment ses préférences sous la forme d'un ensemble de fonctions de pondérations associées à chaque attribut morphologique d'une modalité. La Figure 22 illustre les préférences d'un utilisateur américain, traduisant que pour les modalités textuelles et de synthèse vocale, il préfère la langue anglaise. En fonction du type des

attributs morphologique, la fonction de valeurs peut être à valeur discrète ou continue (c'est le cas de la vitesse de synthèse vocale dans la Figure 22).

Lorsqu'un agent utilisateur (U) portant un ou des agents d'information (K) pénètre l'espace de rayonnement d'un agent de présentation (P), les profils présentés sous forme d'arbres taxonomique pondérés sont fusionnés selon l'algorithme exposé dans (Jacquet, Bellik & Bourda 2007). A l'issue de ce processus, on obtient un arbre pondéré dont le nœud de poids maximum est la modalité choisie pour instancier l'information à présenter. Les attributs de ce nœud décrivent les valeurs des attributs morphologiques permettant d'instancier la modalité choisie.

CONCLUSION DE L'ADAPTATION INTRA-SESSION

L'évolution des IHM adaptatives est un problème complexe. Nous avons vu qu'un système capable de générer des interfaces ne pouvait pas nécessairement les faire évoluer en cours d'interaction. L'adaptation dynamique est assez rarement adressée dans la littérature, et le plus souvent, en termes généraux.

La première difficulté réside dans la surveillance du contexte pour détecter les situations qui provoqueront une adaptation. Cette identification est réalisée en termes de surveillance de variables spécifiques du contexte. Ces variables sont identifiées grâce à des outils de conception tels que la matrice comportementale ou des règles. Elles peuvent également être implémentées de façon *ad-hoc* dans le système (règle de proximité dans KUP).

La seconde difficulté réside dans l'application d'une stratégie d'adaptation. Celle-ci reprend généralement la méthode de génération de l'interface depuis sa phase initiale. Il n'existe à notre connaissance aucun système multimodal ambiant permettant d'adapter l'interaction automatiquement par modification progressive d'une interface déjà existante.

IV.5 COMPARAISON DES ARCHITECTURES D'IHM ET CONCLUSION

Nous comparons, dans cette section, les architectures d'IHM précédemment présentées. Nous avons montré dans la section IV.1.3 que les architectures conventionnelles ne satisfaisaient aucun des besoins en adaptation de l'Ambiant. Nous écartons donc ces modèles de notre étude puisque la plupart des critères utilisés pour comparer les solutions n'auraient pas de sens sur les architectures conventionnelles.

Notre analyse comparée s'étend sur 3 axes, chacun divisé en plusieurs questions :

- La structure de l'architecture (Tableau 1)
  - Comment cette architecture réalise-t-elle l'abstraction du noyau fonctionnel ?
  - Comment sont assemblées les interfaces (selon quel paradigme) ?
  - Est-ce que les entrées/sorties sont explicitement gérées par l'architecture ? Le fait que la gestion des entrées/sorties soit explicite permet une certaine

flexibilité de la distribution des interfaces. Cela permet au concepteur d'explorer plusieurs associations de périphériques.

- L'adaptation de l'interface proposée (Tableau 2)
  - A quel niveau de dynamicité se situe le système ? Le système peut être adaptable (c'est-à-dire à l'initiative du concepteur) ou adaptatif (c'est-à-dire à l'initiative du système). Nous distinguerons, dans les systèmes adaptatifs, ceux qui le sont en cours de session de ceux qui le sont entre sessions.
  - Quelle est la cible de l'adaptation ? L'adaptation visée par le système peut concerner la plateforme, l'utilisateur, l'environnement, les fonctionnalités, la tâche ou son domaine.
  - Quelle est la stratégie d'adaptation suivie? Repose-t-elle sur la multimodalité ou sur l'usage du multi-médias ? Cette adaptation supporte-t-elle la distribution ? Si oui, alors le système doit permettre de concevoir une interface qui soit distribuée sur plusieurs plateformes.
  - Comment la plateforme identifie-t-elle les changements d'états du contexte qui requièrent une adaptation ? Cette information n'est précisée que pour les plateformes adaptatives intra-session et n'a pas de sens pour les autres plateformes.

- L'évolutivité de la plateforme dans le temps (on considère ici plusieurs échelles de temps allant de l'exécution à la réutilisation de composants sur plusieurs cycles de conception) (Tableau 3)
  - Les composants développés lors d'un cycle de conception peuvent-ils être réutilisés dans un autre cycle de conception ? En particulier, lorsque la réutilisation est théoriquement possible, est-elle guidée par un découplage clair entre le noyau fonctionnel et l'IHM ?
  - Comment la plateforme peut-elle être étendue ? L'extension de la plateforme est un élément primordial pour l'exploration de futures techniques d'interaction innovante. En fonction des plateformes, l'ajout de nouvelles techniques d'interaction peut être plus ou moins limité par la nécessité de modéliser ce qui n'est qu'expérimental.
  - La plateforme permet-elle de personnaliser le comportement adaptatif proposé ? Cette personnalisation peut être effectuée par le concepteur ou par l'utilisateur. Il s'agit de définir un modèle comportemental qui ne soit pas figé mais sujet à la définition de fonctions d'évaluations arbitraires. Nous nommons cette capacité à personnaliser le comportement adaptatif le profilage.

En résumé, les systèmes actuels permettent de concevoir des interactions non conventionnelles innovantes, mais ils sont peu adaptés à l'Ambiant.

En effet, si les systèmes permettent la multimodalité, celle-ci est limitée soit aux entrées dans les systèmes multi-médias, soit aux sorties dans KUP et WWHT. Les systèmes orientés tâche contiennent une description de la plateforme (entrées et sorties) mais ne permettent pas, pour le moment, de générer des interfaces non conventionnelles. En effet, on peut observer dans le Tableau 3 que pour étendre les capacités de la plateforme, il faut modifier

des modèles à plusieurs niveaux d'abstraction et proposer également des transformations entre modèles. Cela est possible sur des techniques conventionnelles pour lesquelles nous avons suffisamment de recul, mais plus difficile pour des techniques d'interaction non conventionnelles pour lesquelles nous ne connaissons pas forcément de représentation canonique dans le modèle des interfaces abstraites (AUI).

De plus, l'interfaçage avec le noyau fonctionnel suppose, dans ces approches, une simplification de celui-ci. Le noyau fonctionnel est considéré, pour la plupart des architectures, comme un ensemble de variables d'état associées à des types de données sur lesquels on agit par des commandes simples (approches multi-médias, orientées service, WWHT et KUP). Dans les approches orientées tâche, le noyau fonctionnel est complètement absent de la modélisation et est représenté dans le modèle de tâche au travers des tâches systèmes. L'association automatique de ces « tâches systèmes » au noyau fonctionnel n'est pas aisée, elle peut reposer sur une modélisation des objets du domaine (comme dans UsiXML) mais fait alors de nouveau intervenir une certaine simplification du noyau fonctionnel.

L'extensibilité de la plateforme est un autre besoin primordial dans l'Ambiant. Ce dernier est le lieu de convergence de plusieurs équipes de conception et de plusieurs facteurs (technologiques, sociologiques ou psychologiques). Les développements dans l'Ambiant peuvent mener à des besoins très variés, ce qui requiert une certaine flexibilité de l'architecture et la capacité de personnaliser les comportements du système. Toutes les architectures que nous avons présentées ici sont extensibles, c'est-à-dire que l'on peut ajouter de nouveaux comportement aux interfaces générées. Cet ajout consiste dans certains cas à inclure de nouveaux composants logiciels (c'est le cas des architectures multi-médias et orientées services). Dans d'autres cas, une phase de modélisation et d'édition de règles comportementales est nécessaire (approches orientées tâches et à adaptation intra-session). La personnalisation du comportement du système est obtenue, lorsque cela est possible, par l'édition d'un modèle comportemental composé de règles ou de fonctions de pondération. Enfin, Le but d'une architecture commune étant de factoriser les développements, il faut que les composants du système soient réutilisables, ne serait-ce que partiellement. FAME est la seule architecture à ne pas proposer de mécanisme de réutilisabilité, ce qui est lié à son manque d'abstraction du noyau fonctionnel. Les approches multi-médias ont une réutilisabilité partielle car les composants peuvent représenter aussi bien des médias en entrée ou des techniques d'interaction (réutilisables) que des éléments du noyau fonctionnel (non réutilisables). Les autres approches considèrent la génération d'une interface comme l'assemblage de composants d'interaction découplés du noyau fonctionnel (généralement des widgets graphiques). Ces composants sont donc tous réutilisables.

Enfin, la colonne « Identification des situations à adapter » du Tableau 2 détermine comment le système identifie qu'un changement de situation nécessite une adaptation de l'IHM produite. Hormis KUP qui utilise une méthode *ad-hoc*, les systèmes adaptatifs intra-sessions reposent tous sur la surveillance de variables du contexte. La définition de ces variables ainsi que les seuils impactant l'adaptation est à la charge du concepteur. Cette situation n'est pas satisfaisante car elle suppose du concepteur une certaine connaissance du contexte d'exécution qui est supposé inconnu dans l'Ambiant. Cependant, il n'existe à l'heure actuelle aucune autre méthodologie à notre connaissance.

Plateforme	Abstraction du noyau fonctionnel	Assemblage de l'interface	Gestion des Entrées	Gestion des Sorties
Icon	Modules intégrés au flux de données	Flux de données	Explicite	Implicite
Open Interface	Composants (variables d'état + évènements) intégrés au flux de données	Flux de données	Explicite	Implicite
FlowStates	Modules intégrés au flux de données	Flux de données + Machines à états	Explicite	Implicite
PUC/Huddle	Langage à balises contenant : variables d'état, commandes, labels et information de regroupement	*Ad-hoc* par un générateur (plugin)	Implicite	Implicite, graphique ou vocale
Supple	Arbre de variables d'état	Construction d'un arbre de widgets	Implicite	Implicite, graphique uniquement
Teresa	Tâches « système » dans le modèle de tâches	Construction d'un arbre de widgets	Explicite	Explicite
UsiXML	Tâches « système » dans le modèle de tâches + Modèle du domaine	Construction d'un arbre de widgets ou équivalent VoiceXML	Explicite	Explicite
Cameleon-RT	Tâches « système » dans le modèle de tâches	Construction d'un arbre de widgets	Explicite	Explicite
FAME	Absente	Composants exécutés en parallèle de façon indépendante	Implicite	Implicite
WWHT	Unités sémantiques d'information	Choix de <unité sémantique, modalité, média> + CARE	Absente	Explicite
KUP	Unités sémantiques d'information	Choix de <unité sémantique, modalité, média>	Absente	Explicite

TABLEAU 1 – COMPARAISON DES STRUCTURES DES ARCHITECTURES D'IHM

Plateforme	Dynamicité	Cible	Stratégie	Identification des situations à adapter
Icon	Adaptable	Médias en entrée + Techniques d'interaction	Combinaison multi-médias	--
Open Interface	Adaptable	Médias en entrée + Techniques d'interaction	Combinaison multi-médias	--
FlowStates	Adaptable	Médias en entrée + Techniques d'interaction	Combinaison multi-médias	--
PUC/Huddle	Adaptatif inter-sessions	Fonctionnalités	Arbre de décision	--
Supple	Adaptatif inter-sessions	Fonctionnalités + Coût cognitif de la navigation + Utilisateur	Optimisation dans un espace solution fini muni de métriques ergonomiques	--
Teresa	Adaptable	Tâche + Plateforme	Transformation de modèles manuelle	--
UsiXML	Adaptatif inter-sessions	Domaine + Tâche + Plateforme	Transformation de modèles automatisée par des règles	--
Cameleon-RT	Adaptatif intra-session	Tâche + Plateforme	Transformation de modèles	Variables du contexte à surveiller (définies par le concepteur)
FAME	Adaptatif intra-session	Environnement + Utilisateur	Règles encodées dans la matrice comportementale	Variables du contexte à surveiller (définies dans la matrice comportementale)
WWHT	Adaptatif intra-session	Unité sémantique + Plateforme +	Règles portant sur les modalités	Variables du contexte à surveiller (définies par le concepteur)
KUP	Adaptatif intra-session	Préférences utilisateur + Plateforme + Unité sémantique	Profils de préférences portant sur les modalités	Recalculé à chaque pénétration dans l'espace de rayonnement

TABLEAU 2 – COMPARAISON DES CAPACITES D'ADAPTATION DES ARCHITECTURES D'IHM

Plateforme	Réutilisabilité des composants	Extension de la plateforme	Profilage du comportement automatisé
Icon	Partielle	Ajout de composants	--
Open Interface	Partielle	Ajout de composants	--
FlowStates	Partielle	Ajout de composants	--
PUC/Huddle	--	Ajout de générateurs	Le générateur est *ad-hoc*, fourni par le concepteur.
Supple	Totale	Ajouts de widgets Ajouts de métriques	Métriques personnalisables
Teresa	Totale	Modification des modèles d'interface (abstraite, concrète et finale)	--
UsiXML	Totale	Modification des modèles d'interface (abstraite, concrète et finale)	Règles de transformation de modèles
Cameleon-RT	Totale	Modification des modèles d'interface (abstraite, concrète et finale)	Règles de changement d'état (migration, plasticité)
FAME	Aucune	Ajout de composants et de leur matrice comportementale	Matrice comportementale
WWHT	Totale	Ajouts de nouvelles modalités au moteur de rendu + règles	Règles d'élection
KUP	Totale	Extension de l'arbre taxonomique des modalités	Arbres taxonomiques pondérés

TABLEAU 3 – COMPARAISON DES CAPACITES D'EVOLUTION DES ARCHITECTURES D'IHM

CONCLUSION DE LA PARTIE A

RESUME

Au cours de cette partie, nous avons présenté les défis associés à la conception d'interfaces homme-machine dans un contexte d'intelligence ambiante. Nous avons dans un premier temps étudié la nature même de l'Ambiant (Chapitre I). Ce domaine d'application constitue une vision ambitieuse et complexe du futur de l'informatique. Cette vision envisage un système d'information omniprésent et qui est pourtant capable de se faire le plus discret et non intrusif possible. Ces applications concernent notre vie de tous les jours, que ce soit à la maison ou dans le domaine médical, celui des transports, de la gestion du handicap… Cette vision pose cependant des défis matériels, logiciels et éthiques importants (Chapitre II). Sur le plan logiciel, la nature distribuée, dynamique et hétérogène du milieu ambiant empêchent une analyse « *a priori* » des ressources et fonctionnalités disponibles. Des choix qui étaient traditionnellement faits à la phase de conception du logiciel sont repoussés à la phase d'exécution. Le concepteur doit dont déléguer son expertise à un système logiciel qui l'applique à l'exécution.

Nous avons par la suite étudié les cycles de conception de l'IHM dans l'Ambiant (Chapitre III). Contrairement au développement du noyau applicatif, le développement de l'IHM fait intervenir de nombreux acteurs autres que les programmeurs. Ainsi, nous avons montré comment les utilisateurs, de par leurs besoins et préférences, prennent une place prépondérante dans le cycle de développement. La richesse du milieu ambiant et les techniques de conception participatives ont permis de créer des techniques et des métaphores d'interaction originales et créatives. La mise en œuvre cohérente de ces techniques est un défi pour le développeur qui doit tenir compte de la multiplicité des périphériques d'entrées/sorties, de leur hétérogénéité et de leur distribution sur un réseau.

Ainsi, il est nécessaire de proposer des guides permettant de structurer le développement de l'IHM à travers des modèles d'architecture (Chapitre IV). Nous avons étudié trois types de modèles d'architecture : les modèles conventionnels, les modèles dédiés aux environnements multi-périphériques et les modèles dédiés à l'adaptation automatique. La première catégorie n'est malheureusement pas utilisable dans l'Ambiant car elle repose sur des hypothèses (invariabilité du noyau fonctionnel et des périphériques d'interaction, par exemple) qui sont remises en cause par la nature même de notre objet d'étude. La seconde catégorie est intéressante parce qu'elle tient compte de la distribution de l'interface sur une combinaison non-standard de périphériques et permet le prototypage et l'implémentation de techniques d'interaction innovantes. Cependant, ces techniques n'offrent pas la souplesse nécessaire à l'Ambiant car elles ne tiennent pas compte de la variation des configurations matérielles en cours d'exécution, elles nécessitent l'intervention d'un concepteur pour reconfigurer les combinaisons de médias. La génération automatique semble être la seule alternative. Celle-ci peut se baser sur une expression des fonctionnalités ou bien des tâches à effectuer. Les méthodes dérivant l'interface à partir des fonctionnalités nous apprennent que le noyau fonctionnel est une

cible primordiale de l'adaptation. Cependant, le découplage entre interface et noyau fonctionnel qu'elles réalisent est insuffisant et ne permet pas de produire des interfaces prenant en compte les considérations ergonomiques spécifiques à une tâche. Au travers des IHM adaptatives et de l'ingénierie des modèles, nous avons étudié l'abstraction du noyau sous forme de tâche et la transformation de ce modèle en une interface abstraite puis concrète. Cette méthode de génération est séduisante car elle prend en compte les différentes problématiques de l'Ambiant. Cependant, les interfaces générées par IDM sont restreintes, dans les faits, à des structures de dialogue simples (interaction par formulaire) et limitées aux modalités graphiques et vocales. La réalisation de techniques d'interaction plus élaborées se heurte à la difficulté de la modélisation d'une interface abstraite compatible. En effet, l'approche IDM suppose que les différentes techniques d'interaction aient des points communs qui soient factorisables dans une interface abstraite. Or, nous disposons de trop peu de recul sur les techniques d'interaction non conventionnelles exposées en III.2 pour conclure quant à la validité de cette hypothèse. C'est la raison pour laquelle ces architectures sont, en l'état actuel des connaissances, inadaptées à la production automatique de techniques d'interaction non conventionnelles. Enfin, les architectures de présentation d'information introduisent la notion de modèle comportemental qui guide l'évolution de l'interface dans le temps. Ces modèles comportementaux doivent être personnalisables et représenter un ensemble de choix ergonomiques faits par le concepteur. Ils consistent à surveillant le contexte et à appliquer les consignes du concepteur lorsqu'une adaptation est requise.

PERSPECTIVES

A la lumière de cette étude de l'existant, nous concluons que l'Ambiant modifie notre rapport à la conception des IHM par deux facteurs principaux :

- L'émergence d'une modélisation formelle

  La conception d'IHM conventionnelle fait appel à des modèles de l'utilisateur ou du système qui peuvent être abstraits. Ils s'accompagnent de recommandations générales qui sont interprétées par des êtres humains lors de la conception. En repoussant les choix de conception à l'exécution, on transmet la responsabilité de ces choix au système. Il est donc nécessaire de pouvoir lui transmettre des recommandations sous la forme d'un langage qu'il peut interpréter. C'est la raison pour laquelle les approches de génération font toutes appel à divers modèles pour formaliser l'espace problème.

- L'émergence de nouvelles techniques d'interaction

  Dans la conception d'IHM ambiantes, on sort du carcan des interfaces graphiques pour explorer de nouvelles voies d'interaction. La création de ces techniques requiert une compréhension de la communication humaine qui dépasse les capacités des systèmes de raisonnement actuels. La conception et l'évaluation de ces nouvelles techniques est un processus créatif qui ne peut donc se faire qu'avec la participation d'un être humain.

Ces deux approches peuvent sembler contradictoires car elles vont à la fois dans le sens d'une plus grande spécification de l'interface et d'une prise de responsabilité du système dans la conception ; mais elles demandent également une plus grande implication du concepteur qui met à l'épreuve sa créativité. Cette contradiction n'est qu'apparente. Elle traduit, selon nous, la nécessité de redistribuer les rôles dans la conception d'IHM. Le rôle du concepteur n'est plus de concevoir des interfaces mais de concevoir des techniques d'interaction sous la forme de composants réutilisables qu'il accompagne de recommandations ergonomiques. C'est le système qui, ensuite, combinera ces composants en appliquant les recommandations ergonomiques du concepteur.

Nous pensons que le besoin de modélisation ne se situe donc pas dans la **spécification ou l'abstraction des techniques d'interaction** (comme c'est le cas avec les approches IDM actuelles) car :

- la spécification repose sur un langage prédéfini qui contraint l'implémentation aux concepts existant (par exemple, les balises du modèle TeresaXML). Même lorsque ce langage comporte une grammaire générative, il permet rarement de couvrir l'ensemble des langages d'interaction possibles.
- l'abstraction contraint les concepts implémentés à se rattacher à un ensemble de racines communes (par exemple, les concepts du modèle AUI dans Cameleon).

Ces approches contribuent donc à limiter le pouvoir d'innovation du concepteur/développeur. Partant de ce constat, nous pensons que pour autoriser son extension par des techniques d'interaction innovantes, l'architecture d'IHM ambiante doit laisser le plus de liberté possible au concepteur/développeur dans la réalisation des composants de l'interface.

Les travaux réalisés dans le cadre de WComp (Joffroy 2011) (Brel & Renevier-Gonin 2011) ont ouvert cette voie en proposant au concepteur de construire une IHM à partir de morceaux choisis d'IHM existantes. Cependant, le choix des morceaux à assembler n'est pas automatisable en l'état actuel des connaissances. Il résulte d'un travail réalisé manuellement par le concepteur, à l'aide d'outils proposés par le système. Par ailleurs ces approches se limitent à l'interaction graphique, sans explorer de nouvelles techniques d'interaction.

Les systèmes multi-médias s'inspirent de ces approchent à composants en se focalisant sur l'introduction de nouveaux médias et des nouvelles techniques d'interaction associées. Néanmoins, à notre connaissance, aucune architecture ne combine l'automatisation de l'assemblage des composants de l'interface avec une approche à base de composants librement développés. Nous proposons donc, dans la partie B, une approche intitulée DAME pour la conception d'IHM qui permet :

- La **génération automatique d'une IHM** par assemblage de composants.
- L'extension du système par de **nouvelles techniques d'interaction librement implémentées** (nous verrons ce que ce terme signifie exactement dans la section V.2).
- La prise en compte de **recommandations ergonomiques** et du contexte pour l'assemblage des composants qui forment une interface.

Partie B
PROPOSITION D'UNE DEMARCHE DE CONCEPTION DES IHM AMBIANTES

A partir des besoins identifiés dans la partie A, nous présentons DAME, une démarche de conception des IHM adaptée aux environnements ambiants.

Nous présentons, dans le chapitre V, notre vision d'un système interactif ambiant. Nous prenons soin de clarifier les hypothèses et le cadre de conception et d'exécution que cible notre démarche. Nous concluons par la définition d'un cycle de conception qui fait intervenir trois modèles qui sont successivement détaillés : un modèle des propriétés ergonomiques du système (chapitre VI), un modèle de la structure des composants permettant d'instancier une IHM (chapitre VII) et enfin, un modèle comportemental permettant au concepteur de définir des recommandations ergonomiques (chapitre VIII).

A l'issue de la formalisation des modèles de DAME, nous appliquons notre démarche sur un ensemble d'exemples dans le chapitre IX. Cela nous permettra d'illustrer comment les besoins identifiés dans le chapitre V sont satisfaits par notre approche.

Chapitre V
MOTIVATIONS ET DEMARCHE DE CONCEPTION

Les méthodes de conception participative décrites dans la section III.1 ont permis d'imaginer de nouvelles techniques d'interaction dédiées à l'Ambiant (cf. section III.2). Cependant, il n'existe pas de structure ni de démarche encadrant la réalisation de ces techniques d'interaction qui sont généralement implémentées de façon *ad-hoc* et ne sont donc pas réutilisables.

Nous avons identifié, dans le chapitre II, les caractéristiques de l'Ambiant qui empêchent la réalisation d'une démarche flexible et réutilisable. Pour résumer, le problème fondamental est que l'ambiant est un « monde ouvert »[11] : de nouveaux périphériques peuvent entrer et sortir en permanence du système ; de même, la nature des fonctionnalités proposées varie dans l'espace et dans le temps. Cette diversité des situations implique que les techniques d'interaction et les modèles ergonomiques sur lesquels elles reposent peuvent également varier.

Ce chapitre décrit les défis techniques de la réalisation d'une plateforme qui puisse s'adapter à de nombreuses techniques d'interaction, et qui soit suffisamment flexible pour supporter la définition des futures techniques d'interaction.

Ce chapitre est composé de 3 sections. La première (section V.1) décrit notre vision d'un système ambiant qui tient compte de l'hypothèse du monde ouvert. Cette vision, que nous avons participé à développer dans le projet de recherche européen ATRACO (Heinroth & Minker (eds.) 2011), fournira un cadre de référence à l'ensemble de notre démarche. Nous présenterons ensuite, dans la section V.2, les besoins du système que nous avons extraits à partir de l'analyse de l'état de l'art. En réponse à ces besoins, nous proposons une approche innovante de la conception des IHM dans l'Ambiant. Cette approche se distingue par un usage des modèles différent de celui couramment proposée en IDM pour la génération d'interfaces. Notre approche offre une certaine liberté aux différents acteurs du développement d'espaces ambiants et utilise les modèles comme langue commune permettant de partager les descriptions des propriétés structurelles et ergonomiques du système, ainsi que les recommandations ergonomiques concernant leur composition dynamique. Nous justifions les décisions prises dans cette approche dans la section V.3.

[11] L'hypothèse du monde ouvert existe également en logique. Dans ce contexte, un monde ouvert est un monde dans lequel les agents ont une connaissance incomplète de la véracité des propositions logiques. Par conséquent, les agents ne peuvent s'appuyer que sur les propositions logiques dont ils connaissent la véracité pour raisonner. Par analogie, l'hypothèse du monde ouvert en IHM ambiante indique que les agents réalisant l'IHM ne connaissent pas l'ensemble des situations possibles à l'avance. Ils sont soumis à la non-monotonicité de leur environnement.

V.1 NOTRE VISION DE L'AMBIANT

Les architectures logicielles ambiantes sont orientées services (cf. section II.1), elles reposent sur une forme de programmation appelée programmation orientée composant (POC). La composition dynamique de ces composants permet d'apporter la flexibilité nécessaire aux logiciels ambiants. Ce concept étant central, nous présentons dans un premier temps une formalisation de la notion de composant réutilisable, ce qui nous permettra ensuite d'introduire les concepts fondamentaux de la vision de l'Ambiant que nous avons adopté dans ATRACO.

V.1.1 FORMALISATION DE LA NOTION DE COMPOSANT

La conception UML fait intervenir un type de diagramme utilisé pour structurer l'architecture logicielle : le diagramme de composants. Ce diagramme permet d'identifier, dans le système, des briques élémentaires spécialisées et réutilisables. L'objectif est de pouvoir diviser le travail entre différentes équipes et de réutiliser des développements antérieurs pour construire plus rapidement et à moindre coût de nouvelles applications.

Un composant est une entité logicielle dont le développement suit une logique de « boite noire ». Cela signifie que le contenu du composant est parfaitement inconnu des autres entités logicielles. C'est ce mécanisme qui garantit la réutilisabilité des composants. Pour être utilisable malgré son aspect « boite noire », le composant définit les fonctionnalités principales par lesquelles il interagit avec d'autres composants. Ces fonctionnalités sont regroupées en ensemble de fonctions dans des interfaces. Chaque composant requiert un ensemble d'interfaces (les fonctionnalités fournies par d'autres composants) et expose de nouvelles interfaces (les fonctionnalités qu'il implémente).

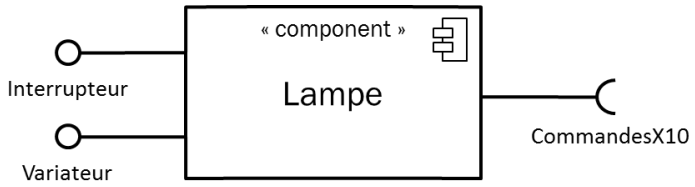


FIGURE 23 - UN COMPOSANT LOGICIEL REPRESENTANT UNE LAMPE CONTROLEE PAR LE PROTOCOLE X10

La Figure 23 illustre la notation UML2 pour un composant de base : une lampe contrôlée par le système X10[12]. Cette lampe expose 2 interfaces permettant de la contrôler : l'interrupteur et le variateur de lumière. Elle requiert la présence d'un composant fournissant l'implémentation du protocole X10. L'assemblage de plusieurs composants s'illustre au travers de diagrammes de composants tels que celui de la Figure 24.

Les interfaces peuvent indifféremment être considérées comme des fonctionnalités mises à disposition ou bien comme des messages échangés entre composants. Une interface peut

[12] X10 est un protocole de communication qui permet de controler des prises électriques à distance en s'appuyant sur la technologie des courants porteurs en ligne. Il est généralement implémenté par un boitier branché sur un port série.

recourir à des objets intermédiaires partagés par les composants qui l'exposent ou la requièrent. Ainsi, le fait que ce soit le composant A ou le composant B qui expose l'interface ne présume pas de la direction des échanges de messages. Une même interface peut permettre l'échange de messages dans les deux sens. Dans les exemples proposés dans les chapitres suivants, nous simplifierons parfois la notation en ne faisant apparaitre qu'une interface là où il est clair qu'un échange à double sens a lieu. Cette simplification permettra d'alléger les figures en ne représentant que les interfaces les plus significatives.

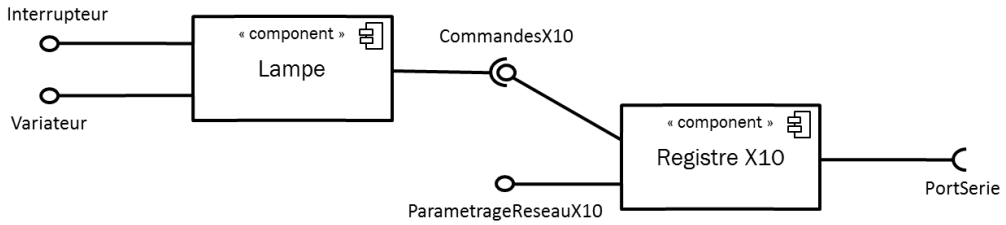


FIGURE 24 - ASSOCIATION DE DEUX COMPOSANTS LOGICIELS

La notation utilisant des ronds (ou *lollipop*) pour représenter les interfaces exposées par le composants et des arcs (ou *sockets*) pour représenter les interfaces requises est appelée socket-lollipop. Parmi les nombreuses notations spécifiées en UML2 pour les diagrammes de composants, celle-ci nous semble la plus concise, nous n'utiliserons donc que celle-ci dans le reste de ce document.

L'association de plusieurs composants existants permet de créer de nouveaux composants récursivement. Cette récursivité est illustrée par la notation UML utilisée dans la Figure 25. Bien que des algorithmes de composition automatique aient été proposés (Urbieta et al. 2008), ce type d'association, appelée composite est généralement réalisée manuellement par le concepteur. L'Ambiant repose donc sur la vision d'un système où la composition serait partiellement voire totalement automatisée. Les applications disponibles disparaissent au profit d'une écologie de composants qui s'assemblent et se désassemblent dynamiquement au gré des besoins et des capacités du système. Cette vision cadre parfaitement avec les caractères hétérogène et dynamique des services qui sont implémentés dans l'Ambiant. Les services (services webs, composants CORBA ou autres) représentent les différents composants de cette écologie ambiante.

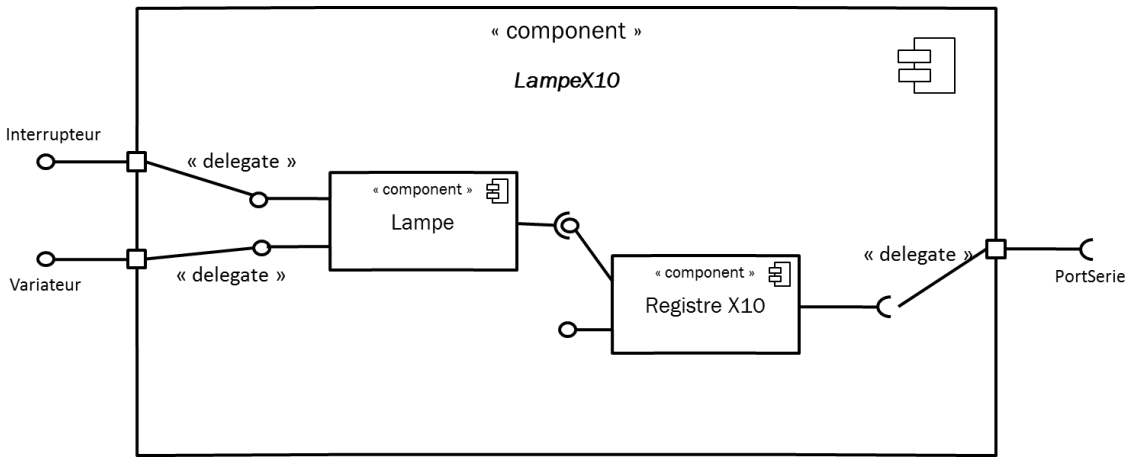


FIGURE 25 - CREATION D'UN NOUVEAU COMPOSANT PAR ASSOCIATION

La transition des systèmes classiques vers les systèmes ambiants se traduit par la disparition d'une conception monolithique de l'environnement et des applications. Le système devient un écosystème de services distribués qui permettent, par agrégation, de créer de nouveaux services offrant des fonctionnalités d'un niveau plus élevé. La gestion de cet écosystème de services distribués requiert la définition d'une nouvelle architecture logicielle. Le projet de recherche européen ATRACO[13] vise à définir une telle architecture.

V.1.2 NOTRE CADRE DE REFERENCE : ATRACO

V.1.2.i LA SPHERE D'ACTIVITE

La métaphore de l'écosystème ambiant a été introduite dans (Goumopoulos & Kameas 2008). Un écosystème ambiant y est défini comme l'entité qui conceptualise un espace physique peuplé de périphériques, de services et de personnes qui sont liés les uns aux autres et à l'environnement et qui supporte les activités de la vie quotidienne d'un utilisateur.

Le projet ATRACO fournit une plateforme conceptuelle ainsi qu'une architecture logicielle qui supporte la réalisation d'écosystèmes ambiants adaptatifs et fiables qui sont assemblés à l'exécution pour satisfaire les besoins des utilisateurs. La notion d'activité est au cœur du système. Contrairement à la « tâche » qui cherche à modéliser ce que l'utilisateur devrait faire pour atteindre son objectif, « l'activité » désigne ce que l'utilisateur fait réellement. Dans ATRACO, les buts de l'utilisateur sont supposés déduis de son activité. C'est donc son activité (et non une tâche définie *a priori*) qui guide le processus de recrutement des ressources de l'écosystème. Ce sous-ensemble de ressources qui collaborent dans un espace et dans un but donné est donc appelé sphère d'activité.

Cette sphère d'activité est réalisée par la collaboration d'agents logiciels qui seront aptes à recruter les composants et services de l'écosystème ambiant, à les assembler et orchestrer leur fonctionnement dans le but d'accompagner l'activité de l'utilisateur. Les sphères d'activités sont donc des bulles[14] éphémères de collaboration entre périphériques et services recrutés dans l'écosystème, elles s'assemblent et se dissolvent dynamiquement au gré des activités de l'utilisateur. La Figure 26 illustre cette architecture. Les mécanismes d'initialisation et de dissolution de ces sphères ne seront pas abordés dans ce document. Pour plus de détails sur l'architecture ATRACO, on pourra se référer à (Heinroth & Minker (eds.) 2011).

---

[13] Le projet de recherche européen ATRACO a été financé par l'Union Européenne dans le contexte du 7ème projet cadre (FP7), en tant que projet membre de l'initiative FET Pervasive Adaptation (PERADA).

[14] Cette métaphore repose sur l'analogie avec le concept de « bulle » utilisé par le psychologue Robert Sommer (Sommer 1959) pour décrire un espace dont les limites sont temporairement définies et qui permet de distinguer les informations entrantes des informations sortantes.

Nous présentons dans la section suivante une vue d'ensemble d'ATRACO et des agents logiciels qui sont évoqués dans la Figure 26. Cela nous permettra de délimiter le cadre dans lequel l'interaction homme-machine prend place. Nous mettrons notamment en valeur le lien entre les besoins utilisateurs, la tâche et l'interface homme-machine.

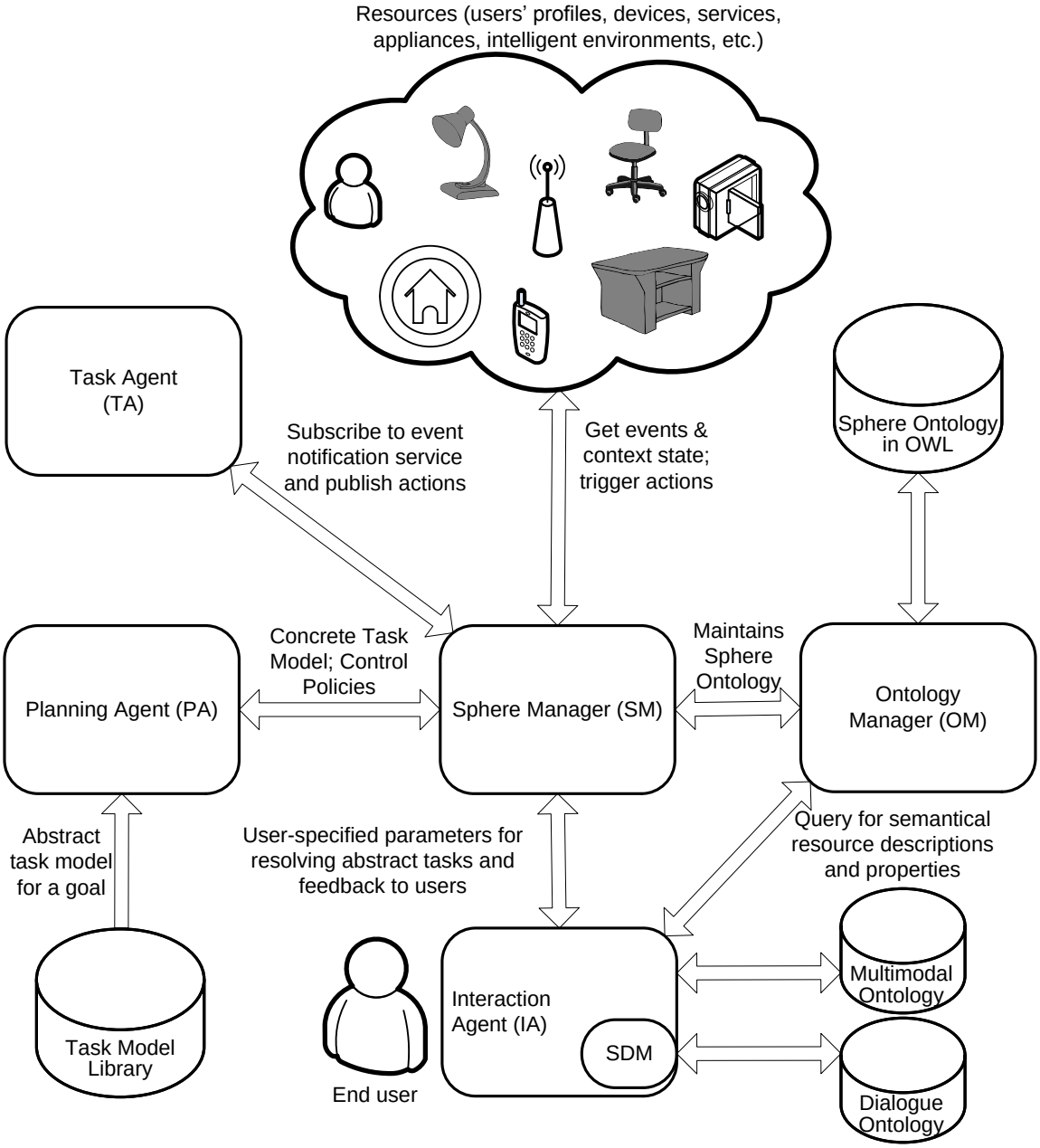


FIGURE 26 – ARCHITECTURE D'UNE SPHERE D'ACTIVITE DANS ATRACO (EXTRAIT DE (HEINROTH ET AL. 2011))

V.1.2.ii LES AGENTS COMPOSANT UNE SPHERE D'ACTIVITE

L'hypothèse fondatrice du système ATRACO est que chacun des composants de l'écosystème ambiant est autonome, c'est-à-dire que chaque entité est en mesure de décrire ses propriétés, ses fonctionnalités et le cas échéant, ses objectifs d'une façon standard. Il s'agit de la couche SOA dans la Figure 27. Chacune de ces descriptions est alignée sur un modèle défini dans (Heinroth & Minker (eds.) 2011) qui permet de fusionner

les descriptions des services disponibles dans l'écosystème, et plus particulièrement de ceux qui sont employés par la sphère d'activité.

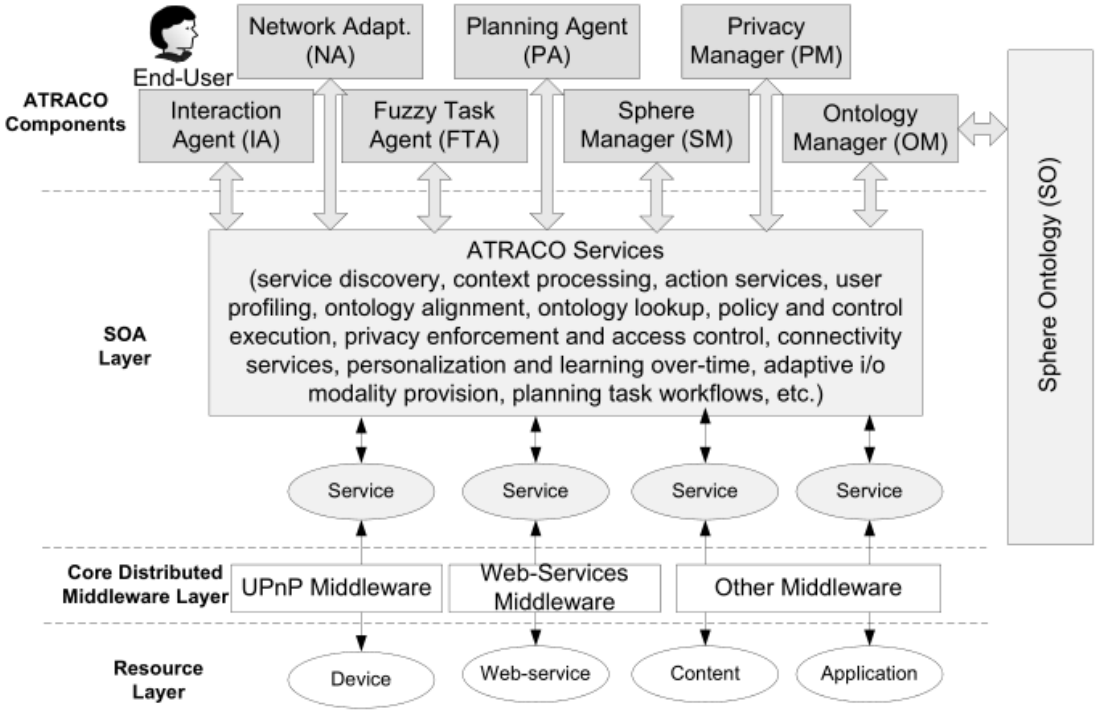


FIGURE 27 - L'ARCHITECTURE SYSTEME ATRACO, EXTRAIT DE (GOUMOPOULOS ET AL. 2010)

Le standard choisi pour réaliser cette fusion d'information est celui des ontologies. Il s'agit d'une description formelle d'un ensemble de connaissances sous la forme d'un réseau de concepts qui sont structurés par des relations binaires. Ce système de représentation symbolique des connaissances a plusieurs avantages :

- Il est naturellement adapté au monde ouvert. Sa simplicité permet de facilement l'étendre avec de nouveaux concepts et de nouvelles relations. Par ailleurs, de nombreux travaux ont été réalisés sur l'alignement ou la fusion de différents modèles décrits dans ce formalisme.
- Sa structure proche de la logique du premier ordre permet de l'interfacer simplement avec des systèmes de raisonnement automatisés reposant sur l'inférence logique.
- Il est bien outillé et supporté par la communauté scientifique. Il dispose ainsi d'un ensemble d'outils graphiques pour l'édition et la transformation de modèles, ainsi que de librairies permettant la lecture, la sauvegarde et le raisonnement sur les modèles.

Une sphère est associée à chaque activité de l'utilisateur. Cette sphère se compose d'un ensemble d'agents logiciels qui se partagent les rôles concernant la bonne orchestration de l'activité. A ces composants actifs est associée une ontologie qui est le résultat de la fusion des descriptions des différents services et des agents qui composent la sphère. Cette ontologie de sphère permet l'échange d'information entre les différents composants de celle-ci.

A l'initialisation de la sphère, trois entités sont sollicitées :

- L'agent de planification (Planning Agent) : Il permet d'établir des modèles de tâches pour remplir les objectifs de la sphère. Il fournit un plan abstrait au contrôleur de sphère. Ces plans sont abstraits dans le sens où ils ne font aucune hypothèse sur les services qui seront recrutés pour le réaliser. Il indique les étapes à réaliser et laisse au contrôleur de sphère le soin de réaliser ce plan en fonction des ressources disponibles.
- Le contrôleur de sphère (Sphere Manager) : Il recueille les plans établis par l'agent de planification et sélectionne les services de l'écosystème qui permettront de les instancier dans les meilleures conditions. Il a également pour rôle, une fois que cette instanciation est faite, de contrôler l'exécution de la tâche en orchestrant les différentes étapes et en gérant les signaux échangés avec les autres agents (notamment l'agent d'interaction).
- Le contrôleur d'Ontologies (Ontology manager) : Il recueille les informations des différents services disponibles et les fusionne. Il fournit ainsi un langage commun aux différents agents pour partager de l'information.

La sphère d'activité (à ne pas confondre avec l'activité effective de l'utilisateur) peut être raffinée en plusieurs « buts » dont chacun peut être décrit par un modèle de tâche. Ces modèles de tâches sont généralement fournis au système par les designers, il incombe à l'agent de planification de les transformer en plans qui sont transmis et exécutés par le moniteur de sphère. La Figure 28 illustre le modèle structurant le domaine de la sphère d'activité.

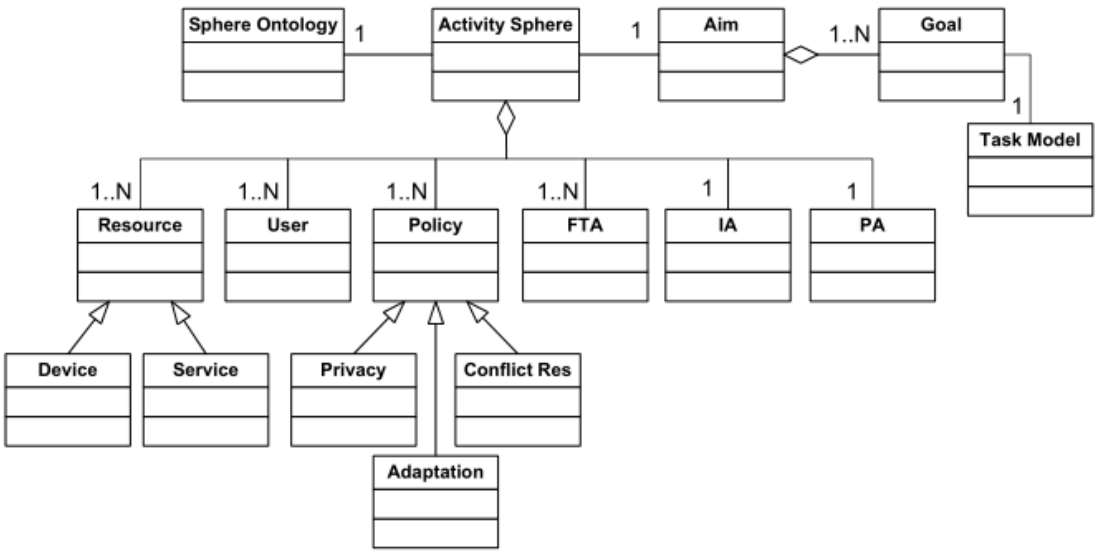


FIGURE 28 - MODELE DE DOMAINE DE LA SPHERE D'ACTIVITE, EXTRAIT DE (GOUMOPOULOS ET AL. 2010)

Une fois un flux de fonctionnalités instancié dans l'écosystème, les autres agents (et en particulier l'agent d'interaction) entrent en jeu pour garantir une adaptation automatique

du système aux préférences de l'utilisateur, tant en termes de comportements du système (température, modes d'économies d'énergie…) qu'en termes de fiabilité ou de respect de la vie privée. Dans la phase d'exécution de la tâche, l'agent d'interaction a pour rôle de fournir des interfaces adaptées au contrôle et au suivi de la tâche par l'utilisateur. Dans la section suivante, nous montrons comment les IHM peuvent être instanciées dans la métaphore des sphères d'activités.

V.1.3 NOTRE CONTRIBUTION A ATRACO : REALISATION DE L'IHM DANS LES SPHERES D'ACTIVITES

ATRACO propose une vision de l'Ambiant qui fait intervenir 5 mécanismes d'adaptation. Parmi eux, l'adaptation de l'interaction est un élément majeur qui conditionne l'utilisabilité du système et donc son acceptation par l'utilisateur final.

Ainsi, au sein de l'architecture d'ATRACO, l'agent d'interaction a pour charge de (cf. Figure 29) :

- Recruter les périphériques qui seront utilisés pour interagir.
- Produire une interface qui soit adaptée aux périphériques disponibles.
- Faire le suivi de la tâche interactive et interagir avec le moniteur de sphère pour (a) l'informer des actions de l'utilisateur et (b) être informé des évolutions du système (qui peuvent avoir un impact sur la présentation interactive).

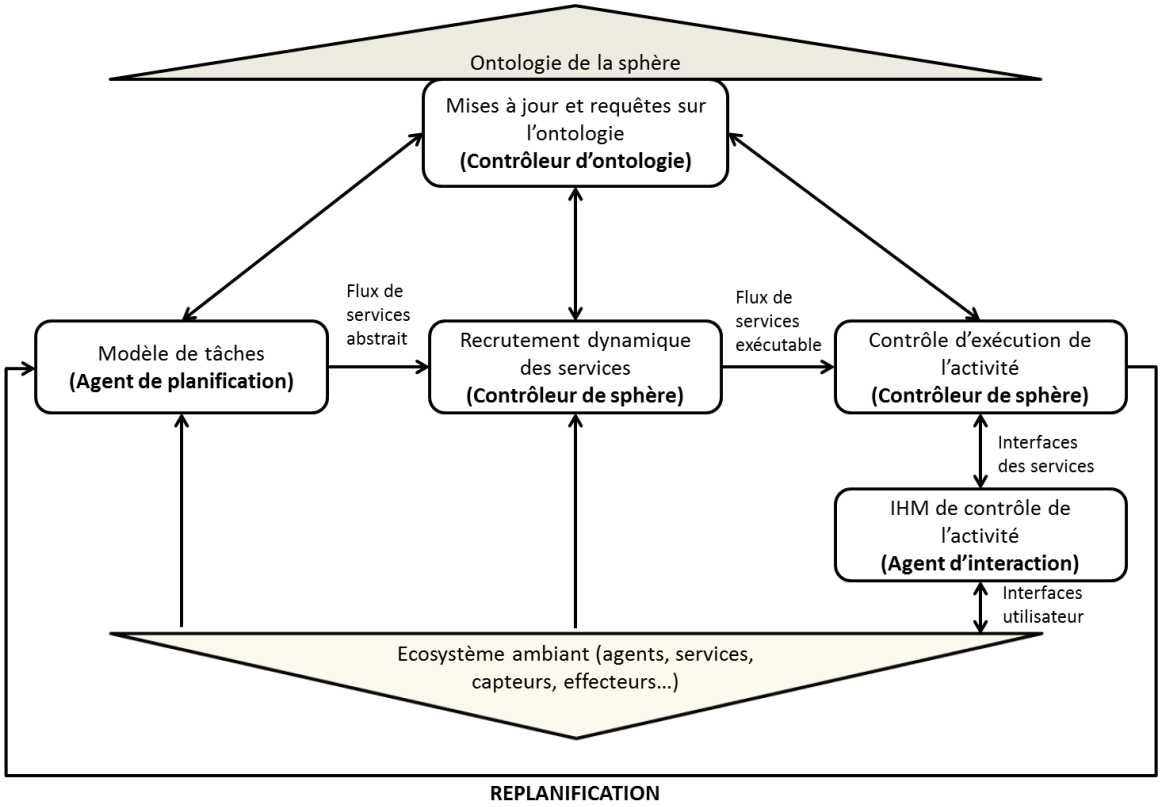


FIGURE 29 - ROLE DE L'AGENT D'INTERACTION AU SEIN DE L'ARCHITECTURE ATRACO

Le projet ATRACO repose sur une vision de l'Ambiant qui met l'accent sur les problématiques de composition et de décentralisation. L'apport de la notion de sphère

d'activité constitue un changement de paradigme vis-à-vis des systèmes traditionnels. On notera que concernant les IHM, ce changement est double : il faut à la fois s'adapter à une variabilité de la tâche interactive mais également à une variabilité du noyau fonctionnel.

A la lumière de ce cadre de référence, nous avons illustré le rôle que doit avoir un système interactif dans les futurs systèmes ambiants. Bien que l'accomplissement de cette vision demeure conditionnée (elle est notamment soumise à des progrès en termes de web sémantique, d'alignement d'ontologie et de standardisation des services et des modèles de tâches), elle nous permet d'identifier les besoins des futures IHM distribuées dans un environnement ubiquitaire.

V.2 EXPRESSION DES BESOINS POUR L'IHM AMBIANTE

A partir de l'analyse de l'existant que nous avons réalisé dans la partie A et de la vision de l'informatique ambiante que nous avons décrit ci-dessus, nous définissons les besoins suivants que doit satisfaire un système d'IHM ambiant.

(a) GESTION DES MORPHOLOGIES MULTI-MEDIAS DISTRIBUEES EN ENTREE ET EN SORTIE

Le système interactif doit pouvoir faire usage de médias de diverses natures qui sont disséminés dans l'environnement physique mais connectés à l'écosystème ambiant. Nous souhaitons également que le couplage de médias ne se limite pas aux médias d'entrée, comme c'est le cas des approches existantes, mais intègre également les sorties.

(b) FLEXIBILITE, REUTILISABILITE ET EVOLUTION DES TECHNIQUES D'INTERACTION

Les concepteurs et ergonomes doivent pouvoir proposer de nouvelles techniques d'interaction riches qui bénéficient de la multiplicité des périphériques. La définition de telles techniques doit être réutilisable d'un environnement physique à un autre (pour peu que les conditions du contexte s'y prêtent).

(c) AUTOMATISATION DE LA CREATION D'UNE INTERFACE ADAPTEE A LA TACHE ET AUX CONTRAINTES ERGONOMIQUES

Le système interactif doit pouvoir produire des interfaces adaptées au contexte selon le principe d'utilisabilité. Il doit avoir une représentation interne du contexte et des concepts soutenant l'analyse ergonomique d'une interface et doit pouvoir raisonner dessus pour instancier l'interface la plus adaptée.

Le besoin (b) fait écho au principe de « monde ouvert » présenté en introduction. Nous avons montré que ce principe a un impact sur les systèmes à l'exécution et comme il est pris en charge par les architectures proposées pour l'Ambiant. Cependant il n'existe, à notre connaissance, aucune approche qui tienne compte de ce principe lors des phases de conception. La gestion de la multiplicité des périphériques a été abordée pour des architectures paramétrées manuellement (Icon, Open Interface). Des automatisations de ces pratiques ont été proposées dans le cadre de la présentation d'information (KUP, WWHT) mais ces solutions se limitent aux sorties du système interactif. Notre approche

vise à combler cette limite en tenant compte des entrées et des sorties de façon adaptée au contexte ergonomique (c).

Nous avons montré, en conclusion de la partie A que les approches existantes ne satisfaisaient pas l'ensemble de ces besoins. Nous proposons donc notre démarche de conception appelée DAME, dans la section suivante. Cette démarche sera formalisée dans les chapitre VI, VII et VIII. Nous analyserons la satisfaction des besoins exprimés ci-dessus par notre approche dans le chapitre IX.

V.3 DAME : UNE METHODE DE CONCEPTION POUR L'ASSOCIATION DE COMPOSANTS RESPECTANT DES RECOMMANDATIONS ERGONOMIQUES

En s'appuyant sur le cadre de réflexion proposé dans les 2 sections précédentes, nous proposons une approche pour l'interaction homme-machine ambiante intitulée **DAME** - **DISTRIBUTED AMBIENT MULTIMODAL ENVIRONMENTS**. DAME traduit une vision de la conception dans l'Ambiant qui cherche à satisfaire les besoins exprimés dans la section précédente. Cette vision a été initialement conçue dans le cadre du projet de recherche européen ATRACO.

Son point de départ est la volonté d'offrir aux concepteurs un espace de développement encadré, mais ouvert aux extensions par l'apport de nouvelles techniques d'interaction non prévues à l'origine. Cette possibilité d'extension est atteinte par l'usage de la programmation orientée composant. Nous définissons dans un premier temps un type de composants particuliers, les composants d'interaction dédiés, qui sont les briques élémentaires de notre approche. Nous étudierons en particulier comment ces briques sont rattachées aux fonctionnalités existantes (i.e. aux services) et aux modèles de tâche. Nous analyserons ensuite les cycles de conception des composants d'un système interactif Ambiant. Cela nous amènera à la définition d'un ensemble de modèles nécessaires à la réutilisation de ces briques élémentaires dans des contextes variés et à la prise en compte, à l'exécution, de recommandations faites par les experts du domaine (ergonomes ou concepteurs).

V.3.1 LES COMPOSANTS D'INTERACTION DEDIES (CID)

PRESENTATION

Pour satisfaire aux besoins définis dans la section précédente, nous introduisons un nouveau concept : le **COMPOSANT D'INTERACTION DEDIE** (**CID**). De façon informelle, nous pouvons définir[15] le CID comme étant la brique élémentaire de l'interaction. Il s'agit d'un composant logiciel qui décrit une interaction entre le système et l'utilisateur. Le CID se concentre uniquement sur un besoin particulier pour lequel le système et l'utilisateur

[15] Une définition plus formelle du CID sera donnée dans le chapitre VII.

doivent interagir et sur l'implémentation d'un dialogue d'interaction qui réalise ce besoin, raison pour laquelle il est qualifié de dédié.

Ce composant est au cœur de notre approche, il apporte la satisfaction des besoins (b) et (c). En effet, de par sa nature, il permet la flexibilité nécessaire à la créativité (car le composant est programmé de manière libre par le concepteur), la réutilisabilité (c'est un composant, il peut s'appuyer sur d'autres composants pour factoriser les fonctionnalités communes) et l'adéquation à la tâche (il est dédié et non générique). C'est en ce sens qu'il s'agit de la brique élémentaire de l'interaction et le rôle du système interactif est de sélectionner le (ou les) CID(s) qui permet(tent) de réaliser une tâche donnée dans les conditions ergonomiques optimales.

Liens entre CID, tache et services

Nous abordons, dans cette section, la question du référentiel utilisé pour concevoir les CID. Ces derniers pourraient être conçus pour réaliser une tâche ou bien pour exposer les fonctionnalités d'un service, comme c'est le cas des approches présentées dans la partie A. Cependant, nous montrons qu'aucune de ces solutions ne convient réellement. Par conséquent, nous proposons une 3ème alternative : le lien avec les cas d'utilisation.

La Figure 30 illustre les liens entre les notions de tâche, de CID, et de services. La composition de services élaborée à partir d'un modèle de tâche (relation A1) est l'une des étapes du processus proposé dans ATRACO. Dans (Gabillon, Calvary & Fiorino 2008), un algorithme basé sur la composition d'interfaces homme-machine est proposé. Dans cette étude, chaque service apporte sa propre interface. Les interfaces sont ensuite composées en suivant le schéma de composition des services (relation A2). L'une des limites de cette approche est l'absence de spécificité de l'interface. Si une interface est conçue pour un service, elle n'est pas conçue pour un usage particulier mais couvre l'ensemble des usages possibles de ce service. Après composition, l'interface obtenue mélange donc tout un ensemble de fonctionnalités qui n'ont aucun rapport avec la tâche en cours.

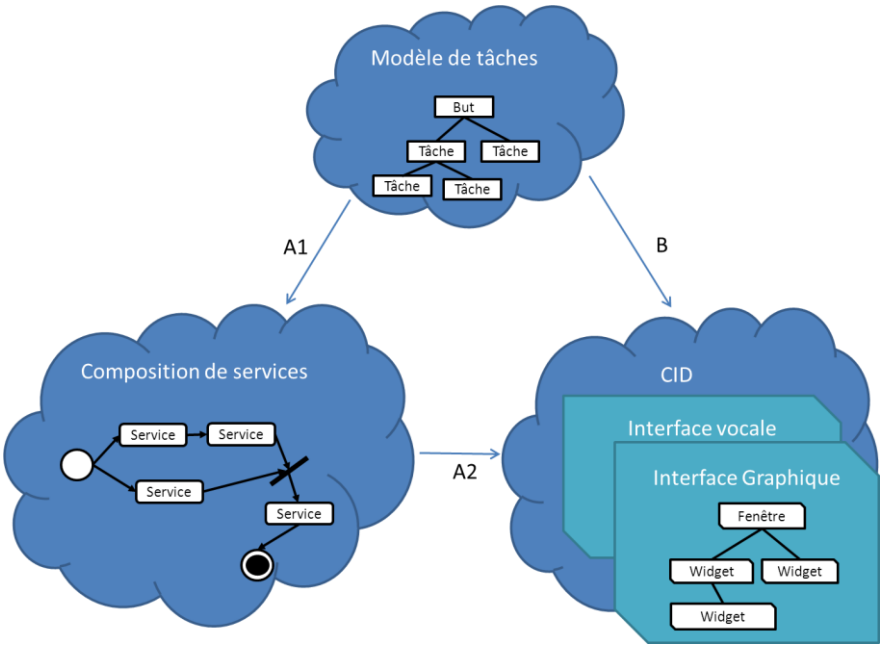


Figure 30 - Lien entre CID, Services et Taches

Pour dépasser cette limite, on pourrait être tenté de lier le composant d'interface à la tâche qu'il implémente (relation B dans la figure). Cependant, le modèle de tâche se décompose en sous-tâches qui sont décrites pour un contexte donné (celui de la tâche en cours). Elles ne sont pas réutilisables d'une tâche à une autre. Par exemple, dans (Gabillon et al. 2011), les tâches générées par planification sont spécifiques à une activité bien précise (cf. Figure 31). Par ailleurs, une tâche n'est associée à aucun service en particulier, un CID qui serait associé à une tâche n'aurait donc *a priori* aucune connaissance sur les fonctionnalités du noyau fonctionnel auxquelles il peut accéder.

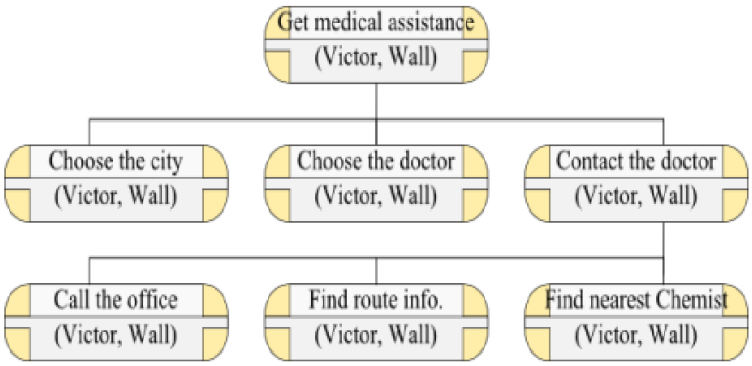


FIGURE 31 – EXEMPLE DE MODELE DE TACHES ISSUES D'UN ALGORITHME DE PLANIFICATION, EXTRAIT DE (GABILLON ET AL. 2011)

Pour répondre à ce problème, on pourrait décider d'associer le CID à la fois à la tâche ainsi qu'aux interfaces des services permettant de réaliser cette tâche. Il s'agit là d'associer le CID à des interfaces et non à des implémentations de celles-ci. Le CID serait donc bien découplé du noyau fonctionnel (i.e. des services qui implémentent ces interfaces). Cependant, pour chaque tâche, plusieurs plans de composition de services peuvent être envisagés, il faudrait alors implémenter des CID pour chaque couple <tâche, {composition de services}> possible. La combinatoire rend donc cette approche irréalisable en pratique. De plus, une telle spécialisation des CID irait à l'encontre du principe de réutilisabilité puisqu'il faudrait couvrir chaque plan de composition par un CID différent.

Si l'on associe le CID au service, on rencontre un problème de flexibilité et d'adéquation à la tâche. Si, par contre, on l'associe aux tâches (élémentaires ou abstraites) du modèle des tâches, on perd la réutilisabilité du composant. Nous proposons donc de faire intervenir un autre concept permettant de rendre le CID réutilisable tout en l'associant aux besoins de l'utilisateur et non aux fonctionnalités offertes par le noyau fonctionnel.

CID ET CAS D'UTILISATION

Lorsque l'on conçoit des IHM, on s'intéresse aux besoins de l'utilisateur (cf. user-centered design, section III.1.1). Ainsi, les IHM conçues sont associés à des besoins et non à des tâches ou à des services. Il en va de même pour le CID dont nous avions précisé en le définissant qu'il était dédié à un besoin utilisateur.

Pour répondre aux problèmes soulevés dans la précédente section, (Almendros-Jiménez & Iribarne 2008) propose une approche de modélisation des tâches principales par un diagramme de cas d'utilisation. En UML, l'analyse des besoins s'effectue à l'aide de diagrammes de cas d'utilisation qui énumèrent les attentes des différents acteurs du

système envers ce dernier. Un cas d'utilisation représente une unité discrète d'interaction entre un utilisateur (humain ou machine) et un système. Il est une unité significative de travail. La Figure 32 présente un exemple de tels diagrammes. On peut y constater que des liens peuvent être établis entre deux cas d'utilisation. Ils portent les stéréotypes « include » et « extend » et leur sémantique permet de structurer l'espace des cas d'utilisation.

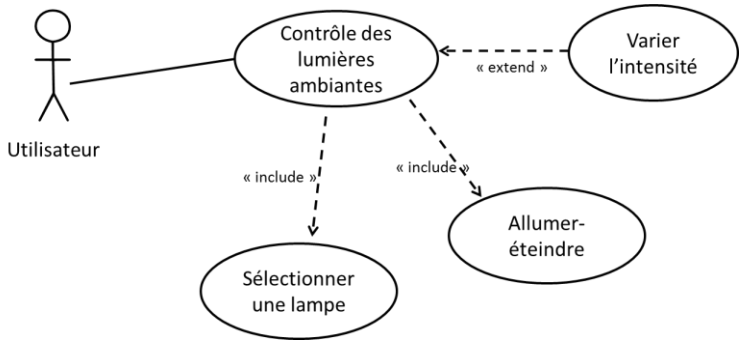


FIGURE 32 - UN EXEMPLE DE DIAGRAMME DE CAS D'UTILISATION POUR LE CONTROLE DE LA LUMINOSITE

Les cas d'utilisation ont l'avantage d'être à la base de la conception des services comme de leurs IHM. Toute démarche de conception s'appuie sur cette analyse des besoins avant d'entamer l'implémentation. Les concepteurs s'attachent donc à créer des cas d'utilisation qui soient réutilisables (Almendros-Jiménez & Iribarne 2008). On notera en particulier qu'un service est conçu pour implémenter un nombre fini de cas d'utilisation alors qu'un nombre infini de tâches peuvent faire appel à lui. Ainsi, le cas d'utilisation est :

- **dédié** à l'accomplissement d'un besoin utilisateur
- **réutilisable** à condition que les concepteurs de services et d'IHM respectent certaines règles de conception qui sont exposées dans (Almendros-Jiménez & Iribarne 2008).
- **dénombrable** car l'on sait dès la conception d'un service quels sont les cas d'utilisation qu'il implémente.

Le cas d'utilisation est donc le parfait candidat pour faire le lien entre la conception de CID, les services et les tâches de façon à la fois dédiée et réutilisable. La Figure 33 illustre ce fonctionnement. Le CID doit avoir une connaissance des fonctionnalités du noyau qu'il peut invoquer. Pour représenter ces fonctionnalités tout en découplant le CID du service auquel il sera associé, nous nous appuyons sur la notion d'interface fournie par un composant. Ainsi, un CID est un composant associé à un ensemble « d'interfaces requises » qui seront fournies par les services instanciés dans le noyau fonctionnel (relation C2). D'un autre côté le CID est associé au(x) cas d'utilisation qu'il implémente (relation C1).

La Figure 33 fait apparaitre les liens entre le modèle de tâche et une composition de services d'une part (relation T2) et les cas d'utilisation d'autre part (relation T1). Ces liens ont fait l'objet de plusieurs études. La relation T2 est traitée dans (Urbieta et al. 2008) qui propose un état de l'art des différentes solutions existantes. La relation T1 est abordée dans (Sinnig 2008) qui propose une méthodologie pour automatiser le passage d'un modèle de tâche sous la forme CTT à un modèle des besoins utilisateurs sous la forme d'un diagramme de cas d'utilisation. Nous admettrons donc que ce lien peut être établi soit de

façon statique (au moment du design), soit de façon dynamique (à la transformation du modèle de tâche en composition de services) par l'agent de planification.

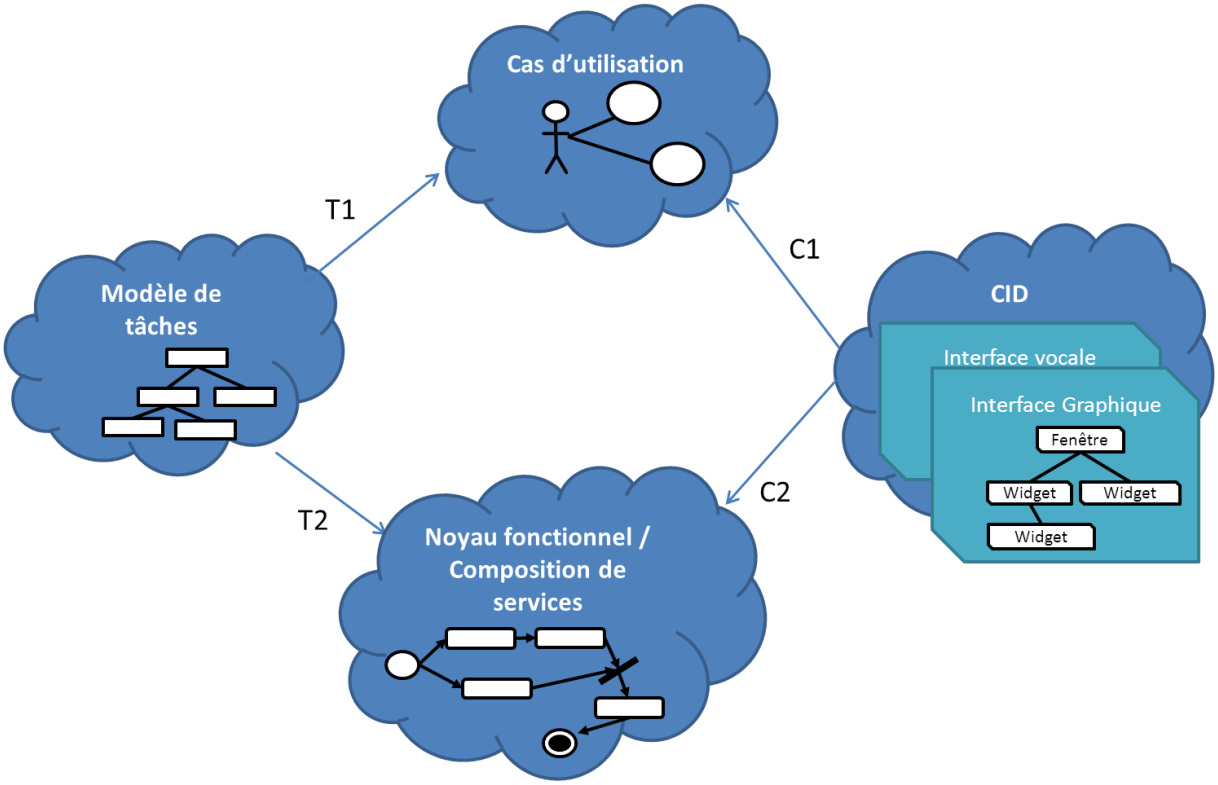


FIGURE 33 – LE CAS D'UTILISATION AU COEUR DE LA RELATION TACHE/SERVICE/CID

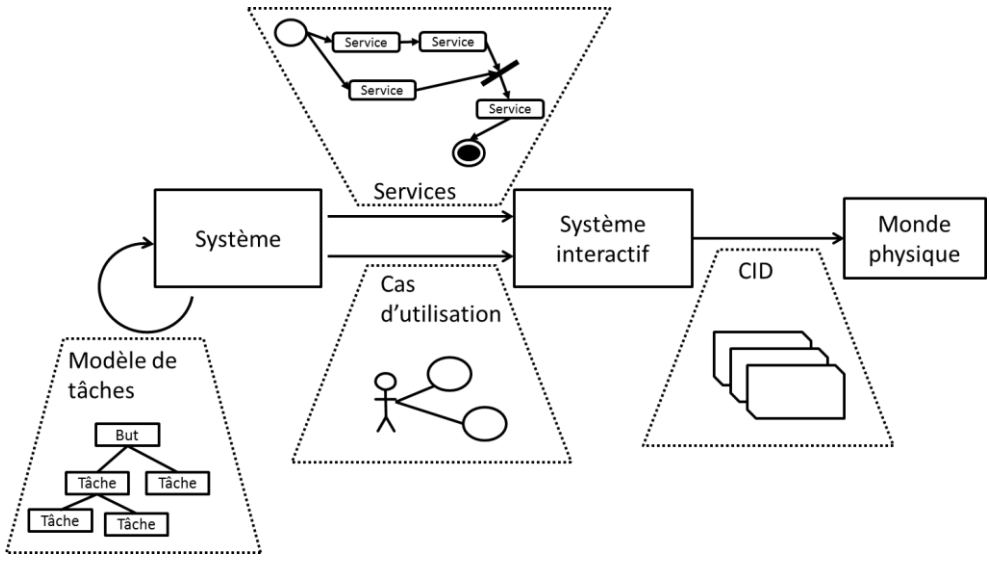


FIGURE 34 - ROLE DU SYSTEME INTERACTIF DANS L'AMBIANT

La Figure 34 illustre notre vision d'un système interactif ambiant, dérivée de la vision proposée dans ATRACO. Celui-ci reçoit en entrée un ensemble d'interfaces du noyau fonctionnel (interface des services instanciés), et un ensemble de cas d'utilisation à réaliser en s'appuyant sur lesdits services. Il dispose d'un ensemble de CID qui implémentent

certains cas d'utilisation. A charge au système interactif d'identifier les CID qui permettront l'implémentation des cas d'utilisation associés au noyau fonctionnel fourni et d'évaluer parmi les différentes solutions possibles, celle qui est la plus adaptée au contexte.

Cette approche nécessite l'utilisation de modèles représentant les liens entre services, cas d'utilisation et CID, mais également entre les CID et le contexte ergonomique. Bien que notre approche repose sur l'utilisation de modèles, elle va à contre-courant des approches traditionnellement « descendantes » (top-down) de l'ingénierie des modèles dans l'Ambiant. En effet, celles-ci partent d'un modèle de tâche et le raffinent successivement pour générer une interface. Ne permettant pas au concepteur d'intervenir dans la phase de création et de l'IHM (les composants graphiques/vocaux assemblés sont prédéfinis et monomodaux), cette forme de génération limite la richesse des interfaces produites. Notre approche est « ascendante » (bottom-up) car elle repose sur l'association de composants logiciels préalablement développés. Nous partons du principe que la créativité des concepteurs et des ergonomes est indispensable à la conception d'interfaces riches et leur offrons un cadre permettant de définir librement de nouvelles techniques d'interaction et de nouveaux composants d'interaction : les CID. Pour répondre au problème de la variabilité de l'Ambiant, ces composants doivent être développés selon une méthodologie qui garantisse leur réutilisabilité. Le concepteur doit notamment fournir des informations sur le composant qu'il produit qui permettent d'identifier les contextes dans lesquelles celui-ci est adapté. Les prochaines sections décrivent cette méthodologie et proposent deux modèles permettant de décrire l'adéquation des CID sur les plans logiciels et ergonomiques.

V.3.2 LES ACTEURS DE LA DEMARCHE DE CONCEPTION DE DAME

Le développement logiciel fait traditionnellement intervenir quatre catégories d'acteurs:

- Les concepteurs : Ils conçoivent et spécifient le comportement du système.
- Les développeurs : Ce sont les experts en développement de logiciels. Ils implémentent sous forme d'instructions machine les spécifications données par le développeur.
- Les évaluateurs : Ils sont en charge de valider le logiciel développé en le comparant au cahier des charges.
- Les utilisateurs finaux : Ce sont les cibles du logiciel, ils peuvent avoir des degrés d'expertise variés dans le domaine métier de l'application.

La construction d'une interface utilisateur est un cas particulier de développement logiciel dans lequel les concepteurs sont des experts en IHM qui conçoivent et spécifient le comportement de l'interface. Le produit fini est alors évalué par des experts en facteurs humains, souvent des ergonomes, qui s'appuient sur l'expérimentation auprès d'utilisateurs finaux. Le lecteur pourra trouver dans (Kolski 1995) (Kolski, Ezzedine & Abed 2001) plus de détails sur les processus de conception des interfaces (modèle en V, en spirale, etc.).

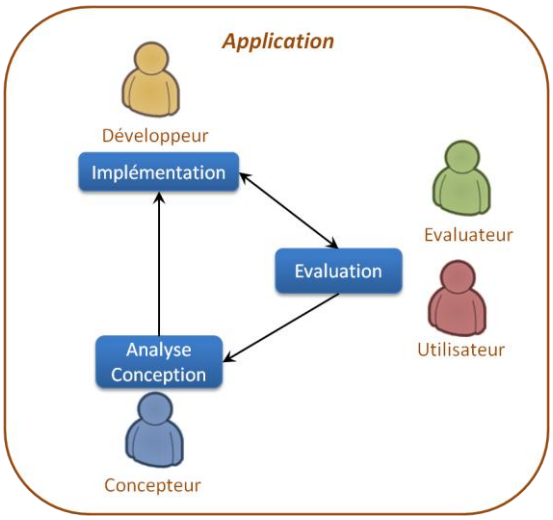


FIGURE 35 –ACTEURS ET CYCLE DE CONSTRUCTION D'UNE INTERFACE

La Figure 35 illustre le rôle de chacun des intervenants dans le cycle de construction d'une interface. Ce schéma peut aussi bien s'appliquer de façon générale à l'ensemble de l'application qu'à l'un de ces modules spécifiques. De fait, depuis l'industrialisation du modèle MVC, les méthodes de développement requièrent une séparation des modules d'IHM et du noyau fonctionnel dès la conception. Ces développements se font en parallèle et nécessitent une grande communication entre les rôles de concepteur/développeur d'IHM et ceux de concepteur/développeur du noyau fonctionnel, ce qu'illustre la Figure 36. Ces rôles peuvent par ailleurs être endossés par la même personne physique. Enfin, l'utilisateur final et l'évaluateur sont en contact direct avec les concepteurs et développeurs lors des phases de conception et d'évaluation.

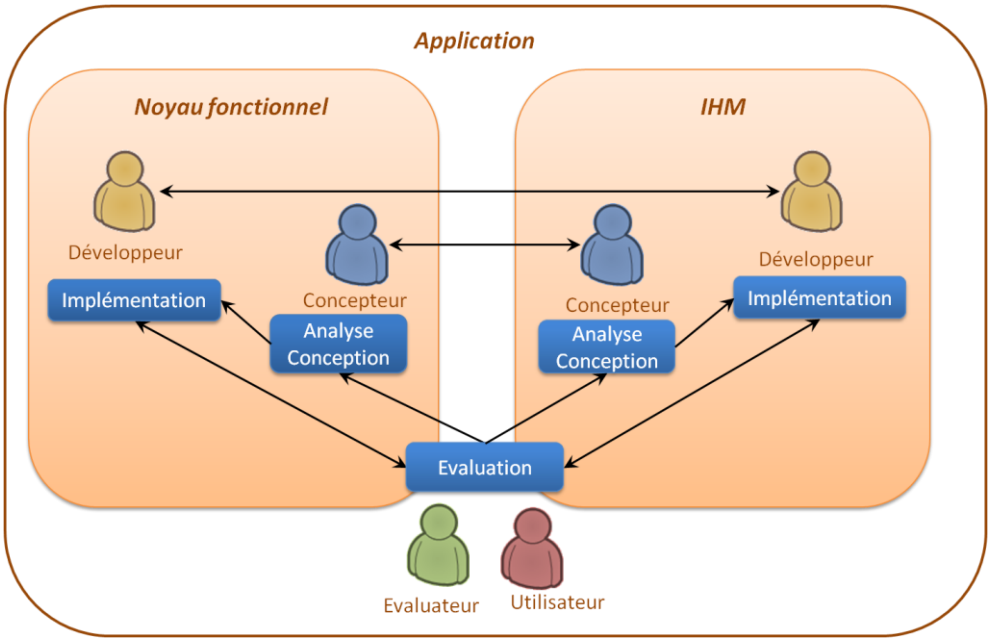


FIGURE 36 – ACTEURS ET CYCLES DE CONSTRUCTION D'UNE APPLICATION DIVISEE EN MODULES

Dans le cadre de l'informatique ambiante, ce schéma est perturbé par la production indépendante des modules qui forment le système. La notion d'application n'a plus de sens

dans l'Ambiant qui ne reconnait que la notion de composant (on parlerait de module dans les architectures traditionnelles) et celle de plateforme ambiante (on parlerait d'environnement d'exécution).

Le remplacement de la notion d'application par celle de composant a les conséquences suivantes :

- Le domaine n'est que partiellement connu au moment de la conception.
- Les composants logiciels sont développés de façon indépendante, selon des cycles de production qui ne communiquent pas entre eux. En ce qui concerne l'IHM, nous distinguons 3 catégories de modules :
  - Les composants représentant les noyaux fonctionnels
  - Les composants implémentant une technique d'interaction. Ces modules sont génériques, ils implémentent les symboles et règles de composition d'un langage d'interaction, mais ne se limitent pas à un dialogue en particulier.
  - Les composants d'interaction dédiés (i.e. les CID). Ces composants s'appuient sur les précédents pour fournir un dialogue spécifique à un cas d'utilisation particulier.
- La conception de chacun de ces modules fait intervenir des équipes (concepteur, développeur, ...) distinctes et indépendantes.
- Le seul acteur à partager le cycle de vie de tous les composants est l'utilisateur final.

La réalisation d'une plateforme ambiante pose donc le problème de l'utilisabilité du système lorsque celui-ci est produit par des entités indépendantes qui ne communiquent pas directement. Pour résoudre cette problématique, nous proposons de faire apparaitre de nouveaux acteurs :

- Le concepteur d'ambiant : Il conçoit l'architecture du système en identifiant la nature des briques élémentaires qui seront nécessaires ainsi que leurs modes de communication.
- L'administrateur d'ambiant : Tout environnement ambiant est en constante évolution. Bien plus qu'un simple système d'exploitation, il requiert une maintenance et une personnalisation de chaque plateforme. De la même façon qu'un administrateur réseau intervient sur un réseau, un administrateur ambiant définit les paramètres de la plateforme et s'assure de la compatibilité des éléments qui la composent.
- L'ergonome d'ambiant : Pour garantir l'utilisabilité du système, l'ergonome doit pouvoir exprimer des règles génériques au niveau de la plateforme ambiante.

Les flèches entre les acteurs de la Figure 37 représentent les échanges d'information nécessaires entre les acteurs du développement des plateformes ambiantes. Le concepteur d'ambiant a pour rôle de définir la nature des composants du système. Par conséquent, il fournit aux concepteurs un cadre de définition abstrait des composants. Ces derniers fournissent en retour une description concrète de leur composant, de ses attributs et de ses capacités de communication avec d'autres composants. De façon similaire, l'ergonome

d'ambiant fournit un cadre descriptif des composants sur le plan ergonomique. Il met en place un certain nombre de règles qui définissent la cohérence de l'ensemble du système interactif ambiant. Ces règles ergonomiques ne concernent pas directement les évaluateurs de noyaux fonctionnels qui font de la validation logicielle. Elles concernent cependant les évaluateurs de techniques d'interaction et de composants d'interaction qui doivent fournir une description de l'ergonomie de leur composant dans le cadre descriptif fourni par l'ergonome d'ambiant.

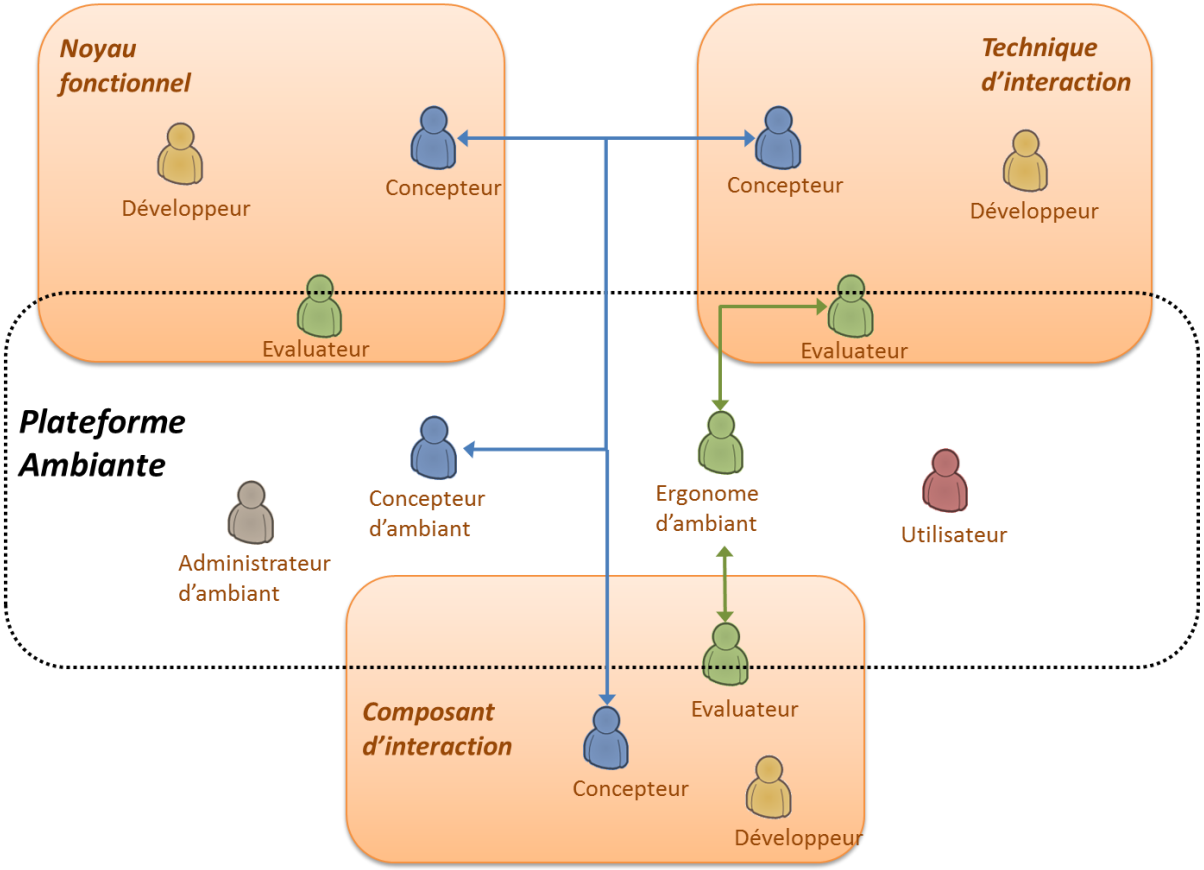


FIGURE 37 - ACTEURS DE LA CONSTRUCTION D'UNE PLATEFORME AMBIANTE

Le but de DAME est de fournir un environnement ambiant permettant aux ergonomes et aux concepteurs de faire converger leurs développements en partageant l'expression de certaines contraintes. Ces contraintes peuvent être des recommandations ergonomiques ou bien des contraintes structurelles s'appliquant à l'infrastructure logicielle (par exemple, des règles ou recommandations concernant l'association de composants). La démarche de DAME consiste à fournir à ces 3 nouveaux acteurs, un langage descriptif commun leur permettant de déclarer les propriétés des composants et les recommandations qu'ils souhaitent voir appliquées. Notre démarche de conception consiste à ajouter, à l'issue du développement d'un composant, une phase de modélisation de celui-ci *a posteriori*.

Suivant cette approche, les propriétés et recommandations concernant les composants d'une architecture DAME sont scindées en 3 composantes, chacune représentée par un modèle :

- Le modèle ergonomique : ce modèle définit les propriétés qui caractérisent l'IHM sur le plan ergonomique. Ce modèle permet aux acteurs de l'Ambiant de spécifier des comportements génériques en terme de modalités.
- Le modèle architectural : Un modèle d'architecture qui structure l'architecture logicielle d'une plateforme ambiante, décrit ses composants élémentaires et leurs attributs. Ce modèle permet au concepteur d'ambiant et à l'administrateur d'ambiant de spécifier des comportements génériques à propos de l'assemblage des composants.
- Le modèle comportemental : Un algorithme permettant d'automatiser l'application de ces règles à condition que les composants de la plateforme soient décrits dans les deux modèles précédents.

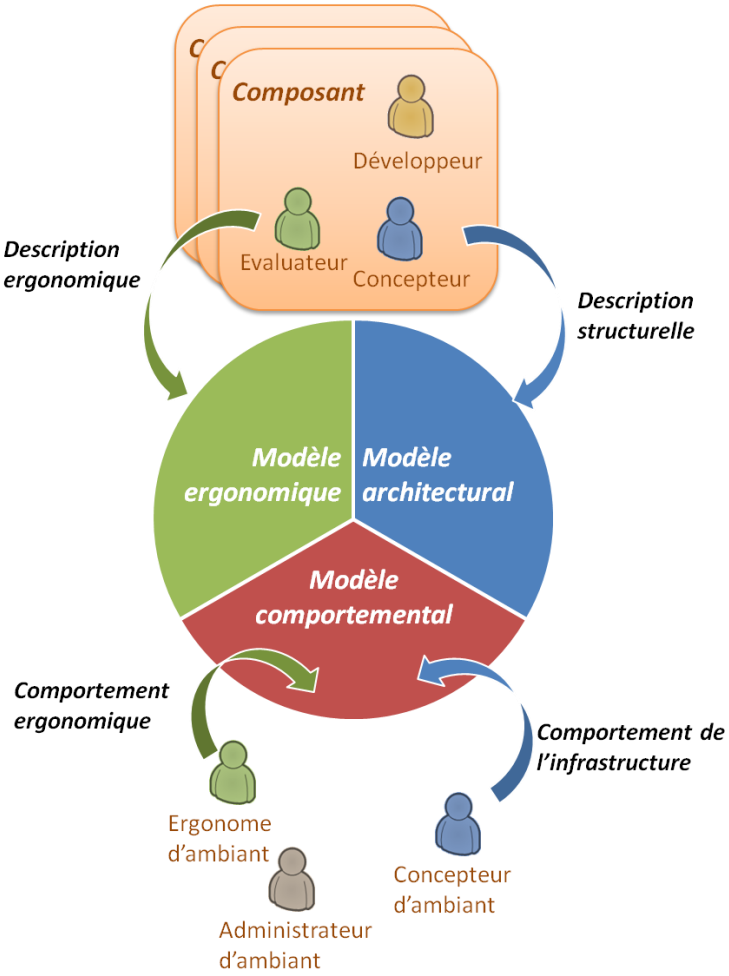


FIGURE 38 - ROLE DES MODELES ERGONOMIQUE ET D'ARCHITECTURE DANS DAME

A partir des modèles d'architecture et ergonomique, nous envisageons le processus de développement logiciel dans l'ambiant de la façon suivante (cf. Figure 38) :

- Des composants sont développés de façon indépendante les uns des autres dans des cycles de production privés. Chacun de ces composants est livré dans la plateforme avec une description de ses aspects structurels (dans le modèle architectural) et de ses aspects ergonomiques (dans le modèle ergonomique).

- Le concepteur, l'administrateur et l'ergonome d'ambiant expriment leurs recommandations dans le modèle comportemental en s'appuyant sur les concepts déclarés dans les 2 autres modèles.
- A l'exécution, un algorithme permet de réaliser les besoins utilisateurs en sélectionnant les CID qui permettent le meilleur compromis entre les considérations ergonomiques et d'architecture.

Cette démarche peut s'appliquer à divers modèles, ce sont les ergonomes et concepteurs de l'espace ambiant qui ont la charge de définir les concepts sur lesquels ils souhaitent pouvoir établir leurs recommandations. Nous proposons, dans la suite de cette thèse, notre vision de ce que ces modèles devraient contenir pour réaliser les objectifs que nous nous sommes fixés. Nous terminerons donc ce chapitre en définissant quelques conventions et notations concernant la formalisation des modèles que nous proposons par la suite.

V.3.3 FORMALISME DES MODELES

Nous définirons les modèles de DAME en utilisant une approche formelle qui permettra de lever les ambiguïtés éventuelles dues au langage naturel. Cette section a pour objectif de définir les notations et les conventions mathématiques que nous utiliserons dans le reste du document.

Dans la mesure du possible, nous formaliserons nos concepts en apportant dans un premier temps une définition informelle en langue naturelle, que nous complèterons ensuite par une formalisation mathématique. Cette formalisation a 3 objectifs :

- GARANTIR LA COHERENCE DU MODELE. La formalisation mathématique permet de clarifier les liens entre concepts en évitant les ambiguïtés et en permettant une vérification du caractère bien formé du modèle lors de la conception et/ou à l'exécution.
- DECOUPLER LE MODELE DE SON IMPLEMENTATION. Le modèle ainsi présenté pourra être implémenté dans le standard qui convient (langage objet, ontologie…) par la suite mais ne sera lié à aucune implémentation en particulier. Il permettra cependant de proposer des algorithmes en pseudo-code qui s'appuient sur la structure des concepts proposés.
- DONNER UN CADRE FORMEL A L'EXPRESSION DE REGLES S'APPLIQUANT SUR LE MODELE. Ces règles peuvent être des contraintes structurelles des modèles ergonomique et architectural ou bien des recommandations formulées par le concepteur ou l'ergonome d'Ambiant dans le modèle comportemental.

Notre modèle repose sur les outils mathématiques liés à la logique du premier ordre. Nous utiliserons des **prédicats unaires** pour typer les instances du modèle en fonction du concept qu'elles instancient. Nous utiliserons les notations traditionnelles de ce domaine et distingueront en particulier les notions d'**intension** et d'**extension** des prédicats. L'extension d'un prédicat P sera notée $\mathcal{E}_P$, il s'agit de l'ensemble des instances vérifiant le prédicat P.

Les liens entre objets du modèle seront représentés par des **relations binaires**. Une relation binaire associe des éléments d'un ensemble de départ à ceux d'un ensemble d'arrivée. Contrairement à une fonction, la cardinalité de l'image ou de l'antécédent d'un élément par une relation binaire peut être multiple, ce qui justifie son emploi dans notre modèle. Pour

définir une relation binaire, il faut définir son ensemble de départ, son ensemble d'arrivée et son graphe. Le graphe représente l'ensemble des couples qui sont en relation. Ainsi, si l'on note A l'ensemble de départ et B l'ensemble d'arrivée de la relation binaire R, et $\mathcal{G}_R$ son graphe, alors $\mathcal{G}_R \subset A \times B$. Ces notations seront particulièrement utilisées dans le chapitre VI.

Dans le chapitre VII, nous emploierons des diagrammes mettant en scène les différents types de composants. La correspondance de ces diagrammes avec les notations mathématiques (prédicats et relations binaires) est définie dans l'annexe A.

V.4 CONCLUSION

Dans ce chapitre, nous avons exposé nos motivations pour proposer une nouvelle démarche de conception des IHM qui soit adaptée à l'Ambiant. L'étude des systèmes existants que nous avons faite dans la partie A a mis en exergue leurs limitations. Ces limites, que nous avons détaillées dans la section V.2, résultent de l'incompatibilité apparente des approches de composition (telles que les approches multi-médias) et des approches adaptatives à base de modèles (telles que les approches de génération automatique).

Nous estimons qu'en automatisant totalement la conception de l'IHM, les approches issues de l'IDM brident la créativité des concepteurs. Nous proposons donc l'approche DAME qui place la composition automatique au cœur de la génération des IHM. Cette approche utilise des modèles formels pour fournir un langage déclaratif des propriétés structurelles, ergonomiques et comportementales des composants et du système.

Dans DAME, les cycles de conception se focalisent sur la construction de composants dédiés (les CID) à un besoin utilisateur. Ces composants sont réutilisables au sein de plusieurs tâches et sont donc définis par rapport à des cas d'utilisation. Chaque CID est conçu par rapport au cas d'utilisation qu'il réalise, et non à la tâche dans laquelle il pourra être impliqué. Cette approche est argumentée dans la section V.3.2, elle permet de définir une granularité variable des fonctionnalités représentées dans le composant d'interaction élémentaire. Le libre choix de la granularité de l'implémentation du CID permet au concepteur d'effectuer différents compromis entre réutilisabilité et intégration de nouvelles techniques d'interaction.

Les cycles de conception sont réalisés par des acteurs différents sans qu'il y ait nécessairement concertation. Cependant, les composants développés doivent collaborer à l'exécution. Cette collaboration requiert le partage de conventions communes. Nous avons donc étudié les rôles des différents acteurs de la conception dans l'Ambiant et proposé la distinction de 3 nouveaux rôles : le concepteur d'Ambiant, l'évaluateur d'Ambiant et l'administrateur d'Ambiant. Ces 3 catégories d'acteurs sont responsables de la définition des conventions qui permettent aux composants créés par des équipes indépendantes de collaborer efficacement. Ils interviennent à 3 niveaux, que nous caractérisons par 3 modèles descriptifs : le modèle des propriétés ergonomiques, le modèle d'infrastructure et le modèle du comportement du système. Ces trois modèles sont appelés respectivement :

modèle ergonomique, modèle architectural et modèle comportemental. Les équipes de conception qui développent pour l'Ambiant doivent décrire leurs composants selon ces 3 aspects afin de pouvoir l'inclure dans un environnement existant.

En terme de démarche de conception, il est possible de définir plusieurs jeux de modèles pour l'Ambiant, chaque triplet de modèles définira alors des systèmes ambiants aux capacités différentes (et potentiellement incompatibles). Nous proposons, dans les chapitres qui suivent, un jeu de modèles qui illustre la faisabilité de notre démarche et permet de satisfaire les besoins exprimés dans la section V.2.

Chapitre VI
LE MODELE D'ANALYSE ERGONOMIQUE

Le modèle d'analyse ergonomique permet de décrire les propriétés ergonomiques des composants du système. Nous présentons dans cette section les briques indispensables à la réalisation de notre vision. L'Ambiant étant un monde ouvert, prétendre proposer un modèle exhaustif couvrant toutes les situations possibles est hors de portée. Nous proposons donc un modèle canonique qui pourra être étendu par l'ergonome d'ambiant si besoin est.

Ce modèle se compose de trois sous-domaines principaux que nous présentons dans la première section. Les sections VI.2 à VI.4 décrivent ensuite chacun de ces aspects.

VI.1 LES TROIS DIMENSIONS DU MODELE

La Figure 39 présente une vue d'ensemble informelle des liens entre les concepts du modèle ergonomique. Ces différents concepts sont définis en détail dans les sections qui suivent. Nous nous attarderons particulièrement à décrire les liens entre ces concepts, liens qui sont représentés par des flèches sur la figure.

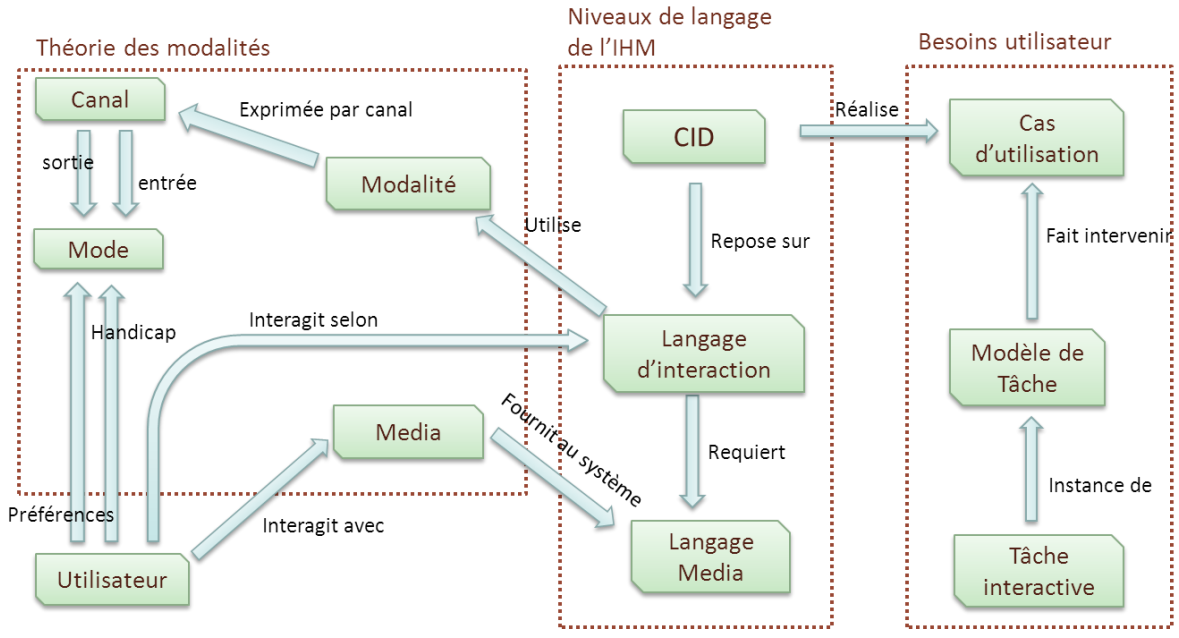


FIGURE 39 - VUE D'ENSEMBLE DU MODELE ERGONOMIQUE

Cette figure illustre clairement les 3 considérations principales du modèle ergonomique :

- La description des besoins de l'utilisateur
- Les niveaux de langage de l'IHM
- La théorie des modalités

Certains des liens qui sont établis entre les concepts du modèle ergonomique ne peuvent être décrits directement dans ce modèle. Nous verrons dans le chapitre VII que ces liens peuvent cependant se déduire à partir de concepts définis dans le modèle d'architecture. Par exemple, le fait qu'un CID soit rattaché à un langage d'interaction particulier peut être déduit des dépendances de ce dernier vis-à-vis d'un composant qui implémente une technique d'interaction particulière.

Les trois prochaines sections décrivent chacune des trois dimensions du modèle.

VI.2 LES NIVEAUX DE LANGAGE DE L'IHM

Nous commencerons notre analyse par la description des niveaux de langage qui permettent de décrire la structure de l'IHM. Nous proposons une classification en 3 niveaux dont le but est d'identifier les différents degrés d'abstraction auxquels se rapportent les composants du système. Ces niveaux d'abstraction correspondent à une complexité plus ou moins importante du langage.

Pour représenter ces différents niveaux de complexité, nous nous appuierons sur la hiérarchie de Chomsky. Cette classification permet de définir des contraintes sur les langages en termes d'expressivité sans présumer de la nature ou du contenu de celui-ci. Pour rappel, le Tableau 4 liste les classes de langages proposées par Chomsky. On notera que cette hiérarchie est inclusive, c'est-à-dire que $\mathcal{C}_3 \subset \mathcal{C}_2 \subset \mathcal{C}_1 \subset \mathcal{C}_0$. Chacune de ces classes de langages peut être reconnue par un automate de complexité grandissante.

Notation	Classe de langages	Type de grammaire générative	Automate associé
$\mathcal{C}_3$	Langages rationnels	Grammaires régulières	Automate fini
$\mathcal{C}_2$	Langages algébriques	Grammaires non contextuelles	Automate à pile non déterministe
$\mathcal{C}_1$	Langages contextuels	Grammaires contextuelles	Automate linéairement borné
$\mathcal{C}_0$	Langages récursivement énumérables	Grammaires générales	Machine de Turing

TABLEAU 4 - HIERARCHIE DE CHOMSKY

Un **LANGAGE DE MEDIA** est un ensemble d'actions physiques pouvant être effectuées et/ou perçues par le système à travers un média. Le langage de média est caractérisé par la perception que l'utilisateur a de ses interactions. Il peut être traduit en langage machine par un ensemble de messages qui peuvent être générés par une grammaire régulière et reconnus par des automates finis. Cette catégorie de langage décrit les interactions physiques entre le système et l'utilisateur.

Un **LANGAGE D'INTERACTION** est un ensemble de symboles abstraits associés à une grammaire riche qui véhicule des concepts de haut niveau (i.e. à même de représenter et manipuler le domaine de la tâche). Cette catégorie de langage décrit les interactions conceptuelles entre le système et l'utilisateur, il correspond aux représentations mentales que l'utilisateur se fait de l'interaction (cf. Figure 4, page 53). Les différents symboles de ce langage sont générés et associés grâce aux actions élémentaires effectuées dans le langage de média.

Le CID réalise un dialogue qui respecte un certain langage d'interaction. Ce lien ne sera pas représenté directement dans le modèle car il peut être déduit à partir des informations contenues dans le modèle architectural. Le CID sera cependant représenté à travers son lien avec les cas d'utilisation qu'il réalise (cf. section VI.4).

On note que ces définitions ne tiennent pas compte de l'implémentation mais de la perception de l'utilisateur. Un même langage de média peut être implémenté de plusieurs façons différentes, c'est-à-dire en utilisant des protocoles différents. Une définition orientée système de ces mêmes concepts sera présentée dans le chapitre VII. Nous prenons dans ce modèle une approche orientée utilisateur qui permettra de définir des comportements identiques pour plusieurs implémentations qui réaliseraient des techniques d'interaction semblables. Ces langages sont utilisés dans la suite de ce chapitre pour caractériser certains concepts, nous nommons donc leurs ensembles ainsi :

Définition
Nous définissons :
$\mathcal{L}_m$ – L'ensemble des langages de média
$\mathcal{L}_i$ – L'ensemble des langages d'interaction
Les langages de media sont voués à être interprétés de façon simple par le système, en s'appuyant sur des automates finis déterministes. Ce sont donc des langages appartenant à la classe des langages rationnels :

$$\mathcal{L}_m \subset \mathcal{C}_2$$

Le langage d'interaction, en revanche, est libre d'appartenir à n'importe quelle classe de langage puisqu'il peut désigner des interactions conceptuelles complexes avec l'utilisateur (comme le langage naturel, par exemple) :

$$\mathcal{L}_i \subset \mathcal{C}_0$$

Les éléments de $\mathcal{L}_m$ sont identifiés dans le modèle par le prédicat $LangageMedia$.
Les éléments de $\mathcal{L}_i$ sont identifiés dans le modèle par le prédicat $LangageInteraction$.

En conclusion, la conception d'une plateforme multimodale interactive consiste à réaliser une transformation permettant de passer d'un ensemble de langages de médias donné à un langage d'interaction donné. Bien sûr, cette transformation n'est pas triviale et nécessite l'implémentation de structures *ad-hoc*, propres au langage d'interaction considéré. Le modèle d'architecture présentera les éléments intermédiaires permettant de réaliser cette transformation. Enfin, bien que les détails d'implémentation soient abstraits dans le modèle ergonomique, la structure du langage, sur le plan ergonomique, sera étudiée dans la section VI.3.4.

VI.3 LA THEORIE DES MODALITES

D'après Bernsen (Bernsen 1994), le problème fondamental que cherche à résoudre la théorie des modalités est le suivant : « *Etant donné un ensemble d'informations devant être échangées entre l'utilisateur et le système pendant l'accomplissement d'une tâche dans un contexte donné, identifier les modalités d'entrées/sorties qui constituent une solution optimale à la représentation et à l'échange de ces informations ».*[16]

Le modèle analytique a pour objectif de permettre aux ergonomes de résoudre ce même problème d'optimisation des ressources d'interaction en tenant compte du contexte. Aussi semble-t-il pertinent d'y intégrer les éléments de base de la théorie des modalités, de sorte qu'un raisonnement fait sur la base des notions de mode ou de modalité puisse être traduit en conséquences sur la représentation de l'interface, et donc sur les composants instanciés dans le modèle d'architecture.

Nous introduisons donc, dans cette section, les notions de mode, média et modalité. Puisque nous souhaitons pouvoir établir des recommandations ergonomiques à partir de ce modèle, nous suivrons une démarche orientée utilisateur (par opposition à la terminologie orientée système), inspirée des travaux décrits dans la section III.3.1. Nous présentons ensuite notre définition d'un canal de communication, notion primordiale qui nous permettra de caractériser les langages d'interaction.

VI.3.1 MODE

Nous partirons, dans un premier temps, de la définition de Bellik, que nous limiterons aux êtres vivants :

> **Définition** (extrait de (Bellik 1995))
> *« Dans une communication entre deux entités (humains, animaux…), le **mode de communication** mis en jeu se rapporte, pour chacune de ces entités, à l'*ORGANE *(ou au système d'organes) utilisé pour percevoir ou produire des informations.*
> *Cela nous amène par conséquent à distinguer tout d'abord et pour chaque communiquant, le mode d'entrée du mode de sortie. »*

Cette définition fait clairement apparaitre le caractère physiologique du mode. Il s'agit de décrire les capacités « matérielles » dont dispose une entité pour communiquer avec une autre, sans se préoccuper de la structure ou de la nature de l'information.

Le mode renvoie à un système d'organes. Il serait donc tentant, dans le cas des machines, d'associer le mode au périphérique qui permet de produire ou de percevoir l'information : le media. Cependant, cela nous conduirait à une multiplication de variantes très similaires

[16] "Given any particular set of information which needs to be exchanged between user and system during task performance in context, identify the input/output modalities which constitute an optimal solution to the representation and exchange of that information."

d'un même mode. Il y a en effet quelque chose d'universel à interagir avec une souris, que celle-ci dispose de 2 ou de 6 boutons. Le problème ne se pose évidemment pas chez l'Homme, puisque tous possèdent les mêmes organes. Pour éviter cette dérive, nous prenons soin de restreindre la définition précédente de mode aux êtres humains. Nous proposons ci-après une extension de cette définition aux machines. Bien que cette définition laisse place à une interprétation subjective, elle permet d'éviter une confusion entre le mode et l'outil qui l'implémente.

Définition

Les **modes de communication** d'un système informatique sont des classes de médias permettant de percevoir ou d'agir sur le monde réel. L'appartenance de deux médias à la même classe (au même mode) dépend du mode utilisé par l'utilisateur pour percevoir ou produire de l'information avec ce média. Elle ne dépend pas de leur implémentation technique ni de la structure de l'information transmise.

Ces modes se divisent en deux catégories : les modes en entrées qui permettent au système de perçoir l'information et les modes en sortie qui lui permettent de la transmettre.

Il est impossible de fournir une liste exhaustive des modes de communication de la machine, puisqu'elle est en perpétuelle évolution. Dans le but de fournir une certaine cohérence à la taxonomie des modes, nous prenons la convention suivante lorsqu'il s'agit d'établir ou d'étendre la liste des modes machines :

Convention

Les intitulés des modes sont définis par analogie avec ceux d'un système vivant. En priorité par rapport aux modes humains lorsque cela est possible. Lorsque cela n'est pas possible, ils seront rattachés aux modes présents dans la nature chez d'autres espèces animales, en s'appuyant sur les taxonomies de la physiologie animale. Dans le cas d'un mode de communication totalement nouveau, un nouvel intitulé pourra être créé.

En nous appuyant sur ces définitions, nous proposons la formalisation suivante :

Définition

Nous noterons $\mathcal{M}$ l'ensemble des modes et $\mathcal{M}_e, \mathcal{M}_s$ ses restrictions respectives aux modes d'entrée et de sortie.

$$\mathcal{M} = \mathcal{M}_e \cup \mathcal{M}_s \;\; et \;\; \mathcal{M}_e \cap \mathcal{M}_s = \emptyset$$

Les éléments de $\mathcal{M}_e$ sont représentés dans le modèle par le prédicat ModeEntree.

Les éléments de $\mathcal{M}_s$ sont représentés dans le modèle par le prédicat ModeSortie.

Exemple

Les modes dont l'Homme est naturellement doté sont notés :

$\mathcal{M}_e^h = \; Visuel, Auditif, Olfactif, Gustatif, TPK$

$\mathcal{M}_s^h = \{Oral, Gestuel\}$

Le mode TPK est l'abréviation de tactilo-proprio-kinesthésique. Il regroupe l'ensemble des perceptions (en entrée) issues des organes permettant la perception de son propre corps en interne et en surface. Les organes concernés sont, pour la plupart, non localisés. Il s'agit

de récepteurs distribués dans les tissus de notre peau ou de nos muscles. Ce mode regroupe les sensations de toucher, d'équilibre, de faim, de température mais également la sensation kinesthésique (sensation de déplacement volontaire ou involontaire d'un muscle). Ce mode est en fait constitué d'un ensemble de sens distincts dans certaines taxonomies de physiologie. Cependant, cette distinction n'est pas utile dans le cadre de l'IHM, raison pour laquelle ils sont regroupés en un mode unique. Enfin, ce mode d'entrée est présent dans la machine de façon plus primitive grâce au clavier, à la souris et aux écrans tactiles. Il est en passe de gagner en précision et en richesse avec la production de matériaux électroniques souples permettant de percevoir les textures, la pression ou la température.

Les modes de sortie sont moins nombreux que ceux d'entrée mais ils sont plus expressifs. Ils couvrent en effet nombre d'interactions inconscientes de notre part. Par exemple, concernant le mode gestuel, il ne s'agit pas uniquement de gestes à visée de communication tels que ceux utilisés dans la langue des signes. Il s'agit de toute action musculaire, consciente ou non, qui soit perceptible de l'extérieur. Ainsi, les mouvements des muscles du visage peuvent traduire notre état émotionnel à travers le mode gestuel sans que ce geste participe à aucune action. De même, concernant le mode oral, il ne se limite pas à son utilisation la plus évidente : la parole. Il englobe l'activité de tous les muscles permettant la production de sons, que ceux-ci soient langagiers (parole) ou non (sifflement).

On remarquera que notre énumération des modes se limite aux modes humains. Les modes disponibles pour une machine sont variables et peuvent couvrir des sens inaccessibles à l'homme. Nous proposons dans le Tableau 5 une liste non exhaustive des modes actuellement utilisables par la machine, à charge aux ergonomes d'ambiant d'enrichir le modèle avec les modes existant sur leur plateforme.

	Mode machine	**Exemples de périphériques**	**Canaux de communication**
En Entrée	TPK	Souris, clavier, dalle tactile, gant numérique, trackball, joystick, occulomètre	<Gestuel, TPK>
	Visuel	Caméra, caméra stéréoscopique	<Gestuel, Visuel>
	Auditif	Microphone	<Oral, Auditif>, <Gestuel, Auditif>
En Sortie	Gestuel	tablette braille, bras haptique, vibreur	<Gestuel, TPK>, <Gestuel, Auditif>, <Gestuel, Visuel>
	Oral	haut-parleurs	<Oral, auditif>
	Affichage2D	Ecran, Imprimante	<Affichage2D, Visuel>

TABLEAU 5 –MODES ET PERIPHERIQUES D'ENTREES/SORTIES D'UNE MACHINE ET EXEMPLES DE CANAUX DE COMMUNICATION CORRESPONDANTS

Exemple

Nous cherchons à énumérer les modes d'un ordinateur classique (souris, clavier, écran). Nous nous reposons sur la convention décrite ci-dessus et établissons donc que :

$$\mathcal{M}_e^{pc} = TPK$$
$$\mathcal{M}_s^{pc} = \{Affichage2D\}$$

Par comparaison avec les modes humains, les entrées proposées par un écran tactile, une souris ou un clavier sont des entrées proprioceptives (TPK). En sortie, le mode employé est l'affichage (en 2D). Ce mode n'existe pas chez l'Homme. Il correspond à la présentation d'une information sur une surface (écran ou vidéoprojection). Notons que le 2D se rapporte à la structure des médias associés au mode et non à la structure de l'information rendue (qui elle, peut être en 3D projective).

Lorsqu'il s'agit d'interaction avec l'utilisateur (à l'opposé de l'interaction avec une autre machine ou avec l'environnement), le mode employé doit être compatible avec l'un des modes humains. Pour illustrer ce lien entre mode d'entrée et de sortie, nous introduisons la notion de canal de communication.

Définition

On appelle **canal de communication** un élément de $\mathcal{M}_s \times \mathcal{M}_e$. Les canaux sont identifiés par la relation binaire $Canal$.

Exemple

Le langage parlé fait intervenir le canal de communication (Oral, Auditif).

Le canal de communication est une entité qui sera réutilisée pour caractériser les modalités dans la section VI.3.3. La définition générale proposée ci-dessus couvre toutes les combinaisons de communication faisant intervenir l'Homme ou la machine (Homme-Machine mais aussi Homme-Homme et Machine-Machine). Cependant, nous nous limiterons à la description des canaux Homme-Machine, c'est-à-dire le sous-ensemble $\mathcal{M}_s^h \times \mathcal{M}_e^m \cup \mathcal{M}_s^m \times \mathcal{M}_e^h$. Le Tableau 5 illustre les canaux de communication les plus fréquemment employés dans l'informatique conventionnelle.

VI.3.2 MEDIA

Comme nous le soulignions en introduction, les études sur la multimodalité ont pour concept central celui de modalité mais intègrent généralement 2 autres concepts que sont le media et le mode. Or, des trois concepts, celui de media est celui qui trouve des définitions les plus variées.

En effet, pour (Blattner & Dannenberg 1990), il désigne tout véhicule d'information physique ou logique. Ainsi, un courrier est un média, de même qu'un e-mail, une onde porteuse… Ce concept est trop général pour notre approche. Dans la théorie des modalités de Bernsen (Bernsen 1994), le terme média est associé aux qualités perceptuelles de l'être humain. Or, nous cherchons à représenter les qualités perceptuelles et actionnelles de l'entité communicante. Nous utiliserons donc la définition suivante :

Définition (extrait de (Bellik 1995))
En informatique, le terme **media** est utilisé pour *« identifier le dispositif physique permettant à la machine d'acquérir de l'information (capteur, périphérique d'entrée) ou de la diffuser (effecteur, périphérique de sortie)*. »

Cette définition englobe à la fois les entrées et les sorties mais se limite à celles de la machine (les périphériques d'interaction). On notera cependant la symétrie avec la notion d'organe du corps humain qui permet d'acquérir de l'information ou de la diffuser dans le monde physique. La différence s'arrête cependant au fait que pour acquérir ou diffuser de l'information, un être vivant fait généralement intervenir un groupe d'organes travaillant en coordination, un organe étant rarement mis seul à contribution. La modélisation des organes du corps humain permettant la perception/action selon un certain mode n'est pas primordiale dans l'étude ergonomique d'une interface, nous évitons donc de la représenter pour ne pas alourdir le modèle.

Dans ce contexte, le fait que le média se limite à la représentation des périphériques d'interaction simplifie la représentation de celui-ci dans le modèle. Nous caractérisons un média par un identifiant et son association à un langage de média.

Définition
On note $\boldsymbol{P}$ **l'ensemble des médias**[17]. Ses éléments sont identifiés par le prédicat $Media$.
On associe à un média son langage de média par la relation binaire $realiseLM$.

Exemple
Une imprimante intitulée $imp1$, utilisant le langage de media $postscript$ sera représentée par $Media\ \ imp1\ \ \wedge realiseLM(imp1, postscript)$

Cette définition ne fait pas apparaitre le lien entre le média et le (ou les) modes qu'il incarne. Nous ajoutons pour cela les relations suivantes :

Définition
Soient $p \in P$ un média et $m \in \mathcal{M}$ un mode, la relation entre p et m peut prendre 3 valeurs :
- **Primaire** : p a été conçu pour être perçu ou s'exprimer à travers m.
- **Secondaire** : p n'a pas été conçu pour être perçu ou s'exprimer à travers m. Cependant, p agit sur m par effet de bord.
- **Inopérante** : Le media ne peut être perçu (ou exprimer de l'information) à travers le mode m.

On définit ainsi les relations binaires :

[17] La notation P est utilisée pour rappeler le fait qu'un media est un Périphérique d'interaction. Elle a été choisie pour éviter les confusions avec l'ensemble $\mathcal{M}$ qui décrit les modes.

- $Primaire$ de l'ensemble P vers l'ensemble $\mathcal{M}$ dont le graphe $\mathcal{G}_{Primaire}$ met en relation un media avec ses modes primaires.
- $Secondaire$ de l'ensemble P vers l'ensemble $\mathcal{M}$ dont le graphe $\mathcal{G}_{Secondaire}$ met en relation un media avec ses modes secondaires.

Exemple

Reprenons l'exemple précédent et l'imprimante $imp1$. Celle-ci a pour mode primaire le mode visuel. Cependant, ce modèle d'imprimante étant bruyant, il a un impact sur le mode auditif. Cet effet de bord (i.e. non désiré par le concepteur) implique que $imp1$ ait pour mode secondaire le mode auditif.
Nous noterons donc

$$Primaire\ \ imp1, Visuel$$
$$Secondaire\ \ imp1, Auditif$$

On remarque qu'il est pertinent d'associer cette relation d'effet de bord $Secondaire$ au média et non à la modalité ou au langage de media car il s'agit bien d'une spécificité d'une implémentation particulière dudit média. En effet, une imprimante d'un autre modèle utilisant le même langage de média (PostScript) pourrait être suffisamment silencieuse pour n'avoir aucun effet de bord auditif. Dans cet exemple, le mode secondaire ne véhicule pas d'information, mais il y a des cas où le message transmis peut être partiellement (voire même totalement reconstruit) grâce à l'usage des modes secondaires.

La description des modes primaires et secondaires d'un media permettra de raisonner sur la pertinence d'employer tel ou tel média pour implémenter un langage d'interaction donné (cf. chapitre VIII).

VI.3.3 MODALITE

La notion de canal de communication permet d'illustrer l'adéquation entre des modes d'entrée et de sortie. Cette adéquation, d'origine anatomique, a permis à l'Homme de développer des langages d'interaction riches supportés par certains canaux. Ces langages reposent sur un ensemble de conventions qui structurent l'échange, tant dans son contenu que dans sa forme. Ainsi donc, au sein d'un même canal d'interaction, différentes structures de l'information peuvent être employées. Nous appelons ces structures les modalités de l'interaction.

Définition (extrait de (Bellik 1995))
[…] La modalité de communication est définie par la STRUCTURE *des informations échangées, telle qu'elle est perçue par l'être humain.*

Cette notion faisant appel à la subjectivité de l'utilisateur, il n'est pas évident de structurer l'espace des modalités existantes. (Bernsen 1994) propose une taxonomie reposant sur des critères qui caractérisent sa structure :

- Le critère Langagier : Indique si la modalité utilisée repose ou non sur la préexistence d'un lexique plus ou moins important, d'une syntaxe plus ou moins complexe et sur des représentations sémantiques et pragmatiques.
- Le critère analogique : Indique si les informations véhiculées tiennent leurs représentations de l'entité signifiée (i.e. par analogie).
- Le critère arbitraire : Indique si la modalité repose sur la préexistence de représentations sémantiques connues par les utilisateurs (non arbitraire) ou si elle nécessite l'apprentissage de conventions (arbitraire).
- Le critère dynamique : Indique si le temps intervient explicitement dans la structure des informations.

Définition

On définit $\mathbb{M}$ **l'ensemble des modalités**. Une modalité est identifiée dans le modèle par le prédicat $Modalite$. On associe à cette représentation une caractérisation selon la taxonomie de Bernsen :

On définit les relations binaires suivantes définies sur $\mathbb{M}$:
$critereLangagier$ à valeurs dans $\{langagier, nonLangagier\}$
$critereAnalogique$ à valeurs dans $\{analogique, nonAnalogique\}$
$critereArbitraire$ à valeurs dans $\{arbitraire, nonArbitraire\}$
$critereDynamique$ à valeurs dans $\{statique, dynamique\}$

De même que pour les média, une modalité peut être perçue à travers différents modes. Bernsen et Bellik associent la modalité à son mode d'expression. A la différence du média qui est attaché aux modes sur lesquels il a un impact, nous associons la modalité aux canaux de communication qui permettent son échange. En effet, la modalité est la structure de l'information qui part d'un mode de sortie pour être perçue par un mode d'entrée. Il nous semble donc plus cohérent qu'elle soit associée à la fois aux modes d'entrée et de sortie du canal.

Définition

Puisque le canal permet de concrétiser le concept abstrait de modalité, on dit que le canal $c \in \mathcal{M}_s \times \mathcal{M}_e$ **réifie** la modalité $m \in \mathbb{M}$, ce que l'on notera $C(m, c)$
On définit la relation binaire $reification$ de l'ensemble $\mathbb{M}$ vers l'ensemble $\mathcal{M}_s \times \mathcal{M}_e$ dont le graphe $\mathcal{G}_C$ met en relation une modalité avec les canaux au travers desquels elle peut s'exprimer.

Exemple

Le canal de communication $< Affichage2D, Visuel >$ supporte diverses modalités de communication dont les modalités texte et pictogramme. Les interfaces graphiques modernes reposent sur un subtil mélange de ces deux modalités.
On notera donc
$reification\ \ texte, Canal\ \ Affichage2D, Visuel$
$reification\ \ pictogramme, Canal\ \ Affichage2D, Visuel$

Les figures suivantes illustrent les liens entre les modalités fréquemment rencontrées en interaction homme-machine et les canaux de communication qui les convoient.

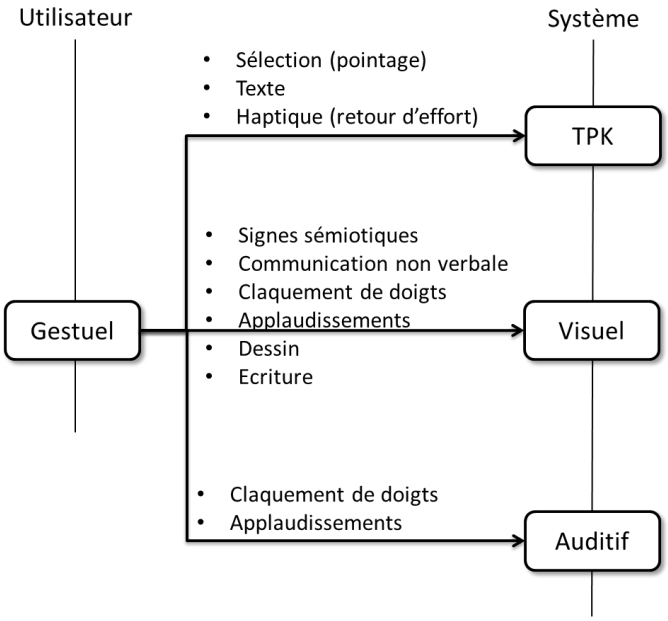


FIGURE 40 – MODALITES ASSOCIEES AUX CANAUX <GESTUEL, _> DANS LE SENS HOMME -> MACHINE

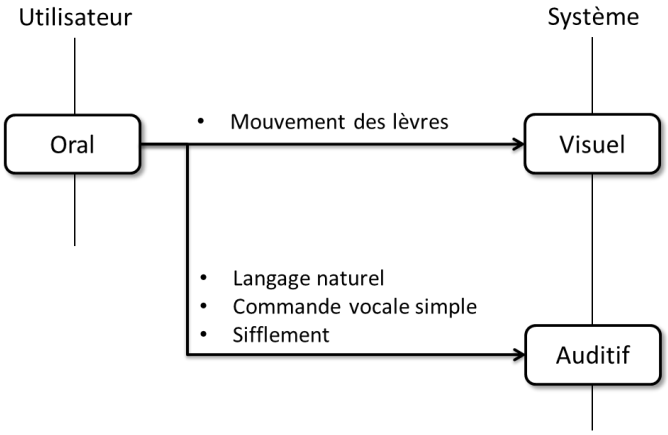


FIGURE 41 – MODALITES ASSOCIEES AUX CANAUX <ORAL, _> DANS LE SENS HOMME -> MACHINE

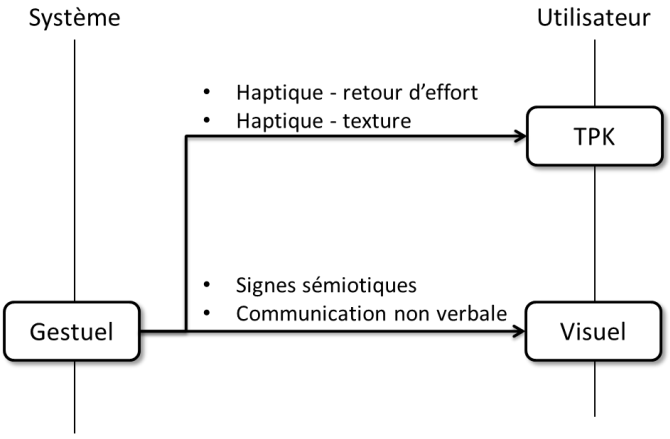


FIGURE 42 – MODALITES ASSOCIEES AUX CANAUX <GESTUEL, _> DANS LE SENS MACHINE -> HOMME

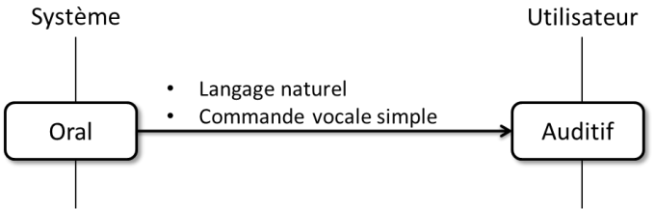


FIGURE 43 – MODALITES ASSOCIEES AU CANAL <ORAL, AUDITIF> DANS LE SENS MACHINE -> HOMME

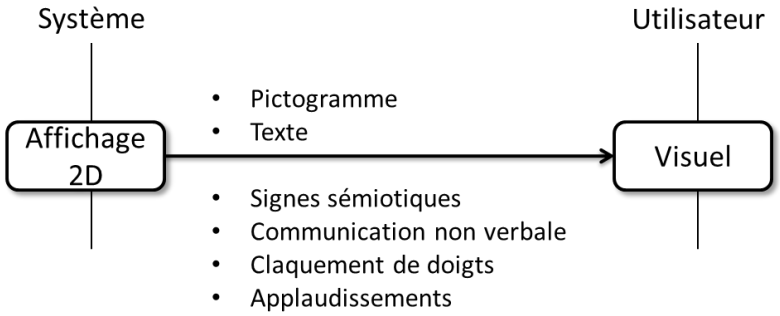


FIGURE 44 – MODALITES ASSOCIEES AU CANAL <AFFICHAGE, VISUEL> DANS LE SENS MACHINE -> HOMME

Dans la Figure 42, le canal <Gestuel, Visuel> fait référence à la communication entre un robot humanoïde et un être humain. La Figure 44 illustre les modalités de communication en sortie les plus fréquemment utilisées par la machine sur le mode Affichage2D. On peut remarquer que les modalités de la communication gestuelle (« signes sémiotiques », « communication non verbale », …) sont associés au canal <Affichage2D, Visuel>. Cette association est rendue possible par l'usage d'un avatar qui apparaitrait à l'écran.

On le voit par ces exemples, l'association entre une modalité et un canal de communication particulier ne peut se déduire automatiquement. Cette association ne peut se faire qu'à condition que certains composants logiciels soient présents pour interpréter/représenter les structures de l'information véhiculée par la modalité. Dans le cas de la communication

non verbale sur le canal <Affichage2D, Visuel> par exemple, ce composant logiciel serait un avatar. La structure de ces composants logiciels, leur description dans le modèle architectural et leurs liens avec les concepts du modèle ergonomique sont étudiés dans la section VII.5.

VI.3.4 THEORIE DES MODALITES ET LANGAGES D'INTERACTION

La hiérarchie de Chomsky nous donne des informations sur la complexité du langage et sur la nature des algorithmes à employer pour les analyser. Cependant, les informations qu'elle nous donne ne sont pas directement utilisables sur le plan ergonomique, car elle ne décrit pas les liens entre les éléments de la grammaire (les symboles) et leur représentation physique (i.e. comment ces symboles sont manipulés par l'utilisateur). C'est la théorie des modalités que nous avons présentée qui nous permet de caractériser la structure d'un langage d'interaction.

Commençons tout d'abord par définir la notion de langage d'interaction sur le plan ergonomique.

> **Définition** (issue de (Nigay & Coutaz 1995)[18])
> *Un langage d'interaction définit un ensemble d'expressions bien formées (i.e., une association de symboles suivant une certaine convention) qui portent un sens.*

Nigay ajoute[19] que *la génération d'un symbole ou d'un groupe de symboles résulte d'actions sur des périphériques d'interaction*. Nous généralisons un peu cette analyse en considérant que ce sont les modalités qui constituent les leviers d'action sur les symboles d'un langage d'interaction. Bien que les médias jouent un rôle dans la réification de ces modalités, c'est l'action effectuée sur ces médias au travers des modalités (pointage, saisie de texte…) qui permet de manipuler les symboles d'un langage d'interaction. Cette définition est plus générale que celle fournie dans la terminologie de la multimodalité orientée système (cf. section III.3.2), qui limite le langage d'interaction à l'interaction avec un seul média.

Ainsi, nous considérons qu'un langage d'interaction est caractérisé par les modalités qui permettent de le structurer et par les canaux de communication employés pour réifier ces modalités. La Figure 45 illustre une caractérisation du langage d'interaction WIMP par ses modalités et ses canaux de communication. A noter que selon nous, un langage d'interaction couvre aussi bien les échanges en entrées qu'en sortie des entités communicantes. Il couvre également plusieurs modalités.

---

[18] "An interaction language defines a set of well-formed expressions (i.e., a conventional assembly of symbols) that convey meaning."

[19] "The generation of a symbol, or a set of symbols, results from actions on physical devices."

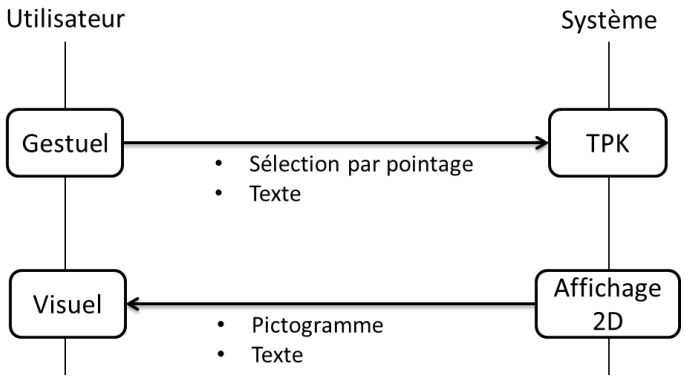


FIGURE 45 - DESCRIPTION DU LANGAGE D'INTERACTION WIMP

Définition

On appelle **schéma d'interaction** une partie de $\mathbb{M} \times \mathcal{M}_s \times \mathcal{M}_e$.

On associe à chaque langage d'interaction ses schémas d'interaction par l'intermédiaire de la relation binaire $Structure$ qui associe les éléments de l'ensemble $\mathcal{L}_i$ à ceux de $\mathbb{M} \times \mathcal{M}_s \times \mathcal{M}_e$.

Les schémas d'interaction d'un langage, identifiés par , permettent d'identifier les structures du langage en terme de modalités, de canaux de communication et donc de mode. Par extension, ils permettent de lier les langages aux médias. Un récapitulatif des liens entre la théorie des modalités et notre définition de langage d'interaction est proposé dans la Figure 46.

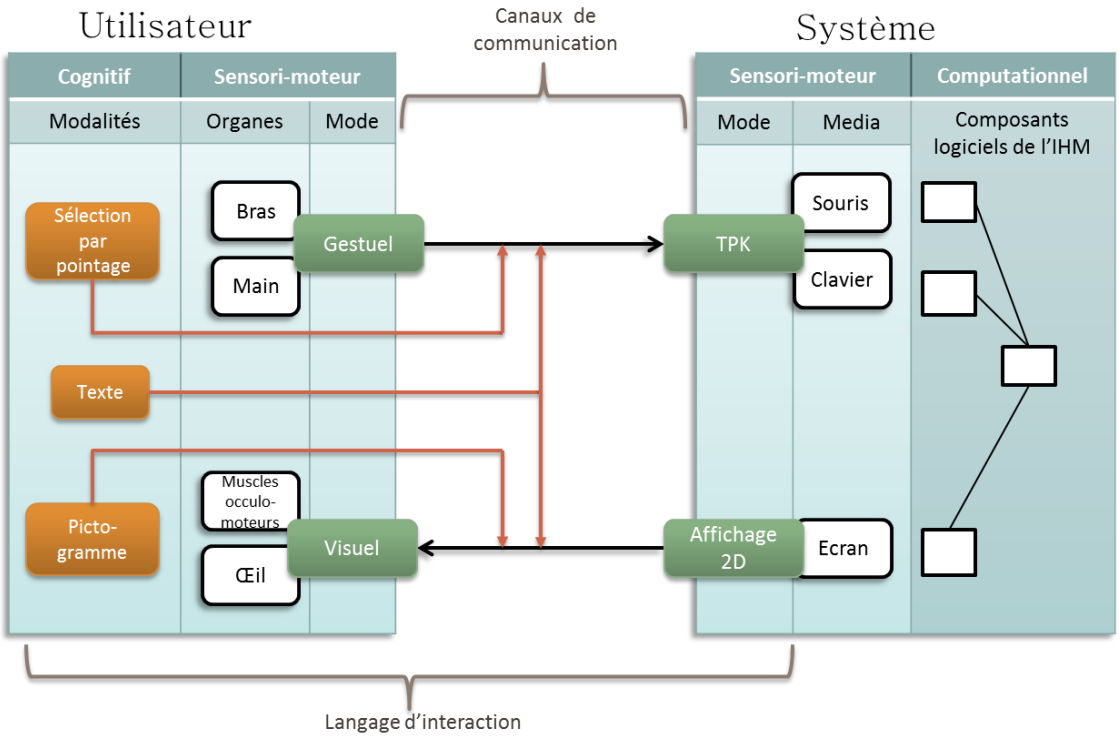


FIGURE 46 - LA BOUCLE DE L'INTERACTION HOMME-MACHINE, CAS DU LANGAGE WIMP

La Figure 46 illustre la boucle de l'interaction homme-machine sous l'angle ergonomique. Elle montre comment le modèle ergonomique permet de raisonner sur la notion de langage d'interaction. Les composants du modèle architectural apparaissent dans la partie droite du diagramme. Nous verrons dans la section VII.4 comment l'architecture logicielle proposée permet de faire le lien avec le modèle ergonomique et notamment, quel impact un composant logiciel peut avoir sur la boucle d'interaction.

A titre d'exemple, nous fournissons une description d'un langage de navigation vocale qui s'apparente à celui des formulaires web. Ce langage, que nous appelons « Navigateur Vocal » permet la représentation de formulaires vocaux. Ces formulaires peuvent être décrits sous la forme de fichiers respectant le format VoiceXML (Rouillard 2004) qui constituent le langage de modélisation des CID associés.

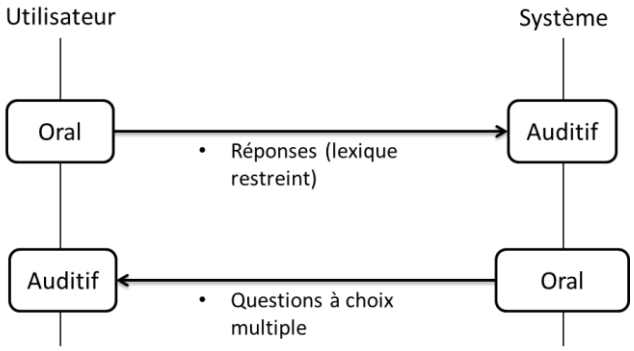


FIGURE 47 - DESCRIPTION DU LANGAGE D'INTERACTION "NAVIGATEUR VOCAL" (VOICEXML)

Notre modèle permet également de représenter des langages d'interaction qui sortent du cadre traditionnel de l'IHM. Ainsi la Figure 48 illustre le cas de l'interaction entre un humain et une machine selon la langue des signes.

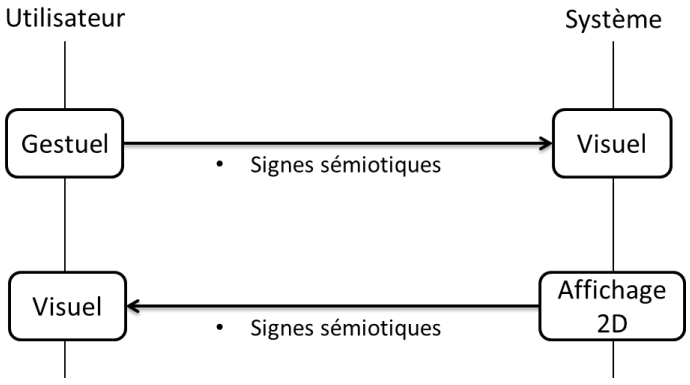


FIGURE 48 - DESCRIPTION DU LANGAGE D'INTERACTION "LANGUE DES SIGNES FRANÇAISE (LSF)" RENDU PAR UN AVATAR

On remarque que le canal machine->homme utilise le mode affichage en sortie tandis que le canal homme->machine utilise le mode gestuel. On imagine dans ce cas que l'on utilise un écran pour afficher un avatar et une caméra pour reconnaitre les gestes. L'analyse de ce diagramme fait clairement apparaitre un biais ergonomique qui peut diminuer la qualité de l'interaction : l'utilisation du mode affichage (en 2D) pour transcrire la modalité « signes sémiotiques » fait perdre la notion de profondeur à l'utilisateur qui perçoit le geste. Une

possibilité pour obtenir un rendu plus fidèle serait de remplacer le canal machine->homme comme dans la Figure 49. Ce remplacement implique que l'on dispose d'un robot humanoïde pour pratiquer les signes.

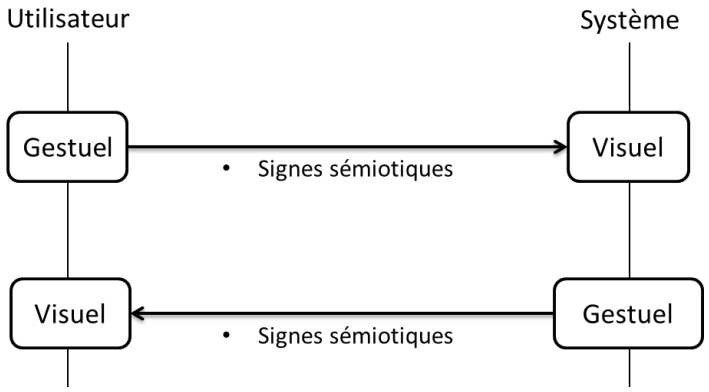


FIGURE 49 - DESCRIPTION DU LANGAGE D'INTERACTION "LANGUE DES SIGNES FRANÇAISE (LSF)" RENDU PAR UN ROBOT

Dans cet exemple, on comprend que le modèle ergonomique permettra au système d'énumérer toutes les solutions d'interaction envisageables et que le modèle d'architecture permettra de spécifier lesquelles sont instanciables en fonction des périphériques et composants logiciels disponibles. L'application du modèle comportemental permettra ensuite d'évaluer chacune de ces solutions.

VI.4 LES BESOINS DE L'UTILISATEUR

Les besoins de l'utilisateur sont représentés dans notre modèle par un graphe de cas d'utilisation. Cette approche a été expliquée et justifiée en V.3.1. Les cas d'utilisation sont définis dans les spécifications UML2 fournies par l'Object Management Group (OMG 2011) de la façon suivante :

> **Définition** (adaptée de (OMG 2011))[20]
> Les cas d'utilisation sont des moyens de spécifier les usages d'un système. Ils sont habituellement utilisés pour capturer les besoins d'un système, c'est-à-dire ce que le système est censé faire.

[20] Page 597 : *Use cases are a means for specifying required usages of a system. Typically, they are used to capture the requirements of a system, that is, what a system is supposed to do.* Page 606 : *Use cases define the offered behavior of the subject without reference to its internal structure. These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional behavior and error handling*.

Les cas d'utilisation définissent le comportement proposé par un système sans faire référence à sa structure interne. Ces comportements, impliquant des interactions entre utilisateurs et système, peuvent provoquer des changements dans l'état du système ainsi que des communications entre celui-ci et son environnement. Un cas d'utilisation peut inclure des variations de son comportement de base, tels que des comportements optionnels ou la gestion des erreurs.

Les cas d'utilisation définis par l'OMG sont organisés en réseau représentés par un diagramme de cas d'utilisation. Les relations qui existent entre les cas d'utilisation sont de 3 types : « include », « extend » et « inherit » :

- Inherit :

  Un cas d'utilisation peut hériter d'un autre. Dans ce cas, le cas d'utilisation héritant (i.e. le cas fils) implémente le cas d'utilisation hérité (le cas père) d'une façon spécialisée (i.e. adaptée à un contexte spécifique). C'est bien l'ensemble du comportement du cas père qui est importé et remanié dans le cas fils, contrairement aux deux autres relations qui décrivent des sous-parties de comportements plus généraux.

- Extend :

  La relation d'extension (à ne pas confondre avec l'héritage) indique qu'un cas d'utilisation représente un comportement additionnel et optionnel d'un autre cas d'utilisation. Cette relation permet d'identifier plusieurs variantes d'un même cas d'utilisation en extrayant les composantes qui varient d'une version à l'autre.

  UML2 va plus loin en définissant des « points d'extension » qui représentent la partie du comportement qui est étendue par la relation d'extension. Pour ne pas alourdir le modèle, nous ne représenterons pas les points d'extension car ils n'impactent pas directement l'instanciation des CID.

- Include :

  La relation d'inclusion indique que le comportement décrit par un cas d'utilisation est requis (ou inclus) dans le comportement décrit pas un autre cas d'utilisation. Cette relation peut servir à factoriser les comportements communs à plusieurs cas d'utilisation ou à énumérer les différents comportements intermédiaires d'un même cas d'utilisation.

Ces relations permettent de structurer les cas d'utilisation en un ensemble de comportements du système qui sont réutilisables. Au concept de cas d'utilisation, nous associons la modélisation suivante:

Définition

On définit le prédicat CU qui identifie les extensions du concept « cas d'utilisation ». L'ensemble de ces extensions est noté $\mathcal{E}_{CU}$.

Les relations structurelles entre cas d'utilisation sont représentées par les relations binaires suivantes :

$$InheritsCU : \mathcal{E}_{CU} \times \mathcal{E}_{CU}$$
$$IncludesCU : \mathcal{E}_{CU} \times \mathcal{E}_{CU}$$
$$ExtendsCU : \mathcal{E}_{CU} \times \mathcal{E}_{CU}$$

Exemple

Le diagramme de la Figure 50 représente les cas d'utilisation d'un système multimédia. La modélisation des CU « observer le média », « rendu sonore » et « rendu vidéo » serait la suivante :

$CU\ observerMedia$
$CU\ renduSonore$
$CU\ renduVideo$
$IncludeCU\ observerMedia, renduSonore$
$ExtendsCU(renduVideo, observerMedia)$

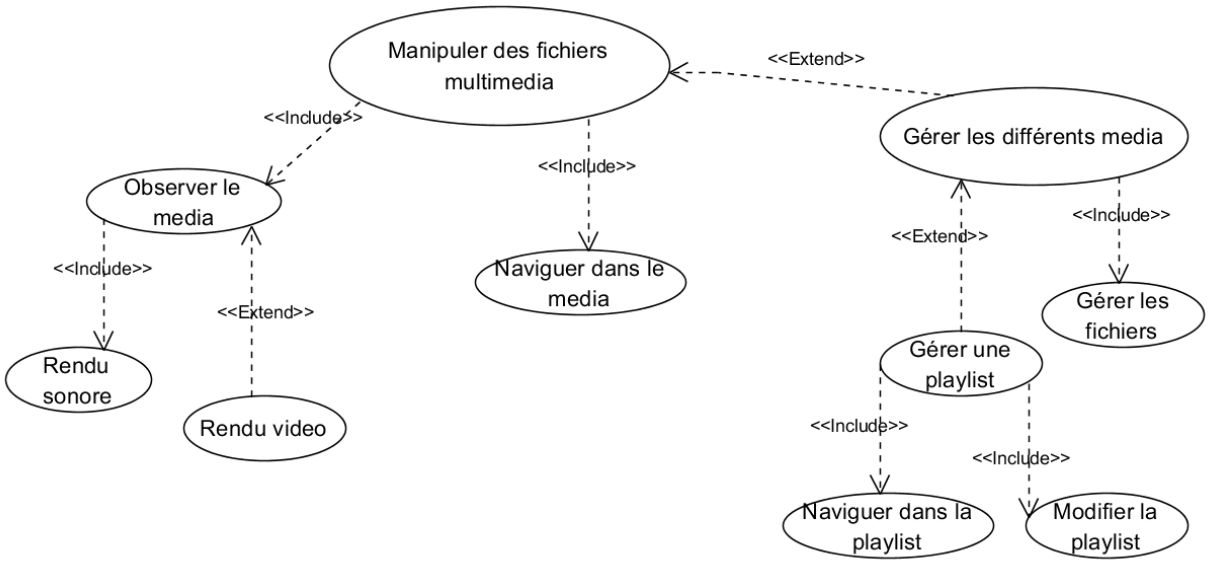


FIGURE 50 - DIAGRAMME DE CAS D'UTILISATION D'UN SYSTEME MULTIMEDIA

VI.5 CONCLUSION

Nous avons décrit dans ce chapitre notre vision d'un modèle ergonomique canonique. Ce dernier contient les concepts nécessaires à la description des aspects ergonomiques multimodaux d'une interface homme-machine. Il permet au développeur de techniques d'interaction et de CID de décrire leurs composants en des termes qui sont compréhensibles par un ergonome, ce qui facilite le partage de recommandations.

Notre modèle reprend la terminologie de la multimodalité proposée dans la section III.3.1 et propose une définition innovante de la notion de canal et du lien entre la modalité et les

modes (par l'intermédiaire d'un canal). Il contient également une description des différents niveaux de langages qui constituent une IHM : le langage de média, le langage d'interaction et enfin le dialogue qui est implémenté par un CID. Enfin, il comprend également une représentation des besoins de l'utilisateur sous la forme de cas d'utilisation.

Le lien entre les 3 domaines que nous venons d'exposer et qui constituent le modèle ergonomique ne sont pas explicites. En effet, le fait qu'un cas d'utilisation puisse être exprimé à travers tel ou tel autre langage d'interaction n'est pas représenté. Il en va de même du lien entre un média et les modalités qu'il permet d'instancier sur un canal. Ces liens ne peuvent être connus à l'avance car ils dépendent de leur implémentation sous la forme d'un composant logiciel. Nous proposons donc, dans le chapitre suivant, un découpage de l'architecture d'une IHM ambiante. A l'issue de cette description, dans la section VII.4, nous montrerons comment, en décrivant ces composants en termes du modèle ergonomique, des ponts entre les domaines de ce dernier peuvent être inférés.

Chapitre VII
LE MODELE D'ARCHITECTURE

Le modèle d'architecture (appelé également modèle architectural) définit l'infrastructure logicielle du système. Il définit les règles qui permettent la composition des modules et structure la construction d'une interface en différentes couches logicielles aux rôles bien définis.

Nous allons dans un premier temps présenter le modèle de composition que nous avons adopté. Nous présenterons ensuite une vue d'ensemble du modèle et des différentes catégories de composants qui constituent une IHM. Nous définirons ensuite plus précisément chacune de ces catégories. Tout au long de cette présentation, nous illustrerons nos définitions par des exemples. Nous ferons appel à un exemple issu de l'informatique conventionnelle qui sera notre fil conducteur : l'IHM d'un centre multimédia permettant de gérer une bibliothèque de fichiers multimédias et de contrôler leur rendu. La section 0 résumera la modélisation architecturale de cette exemple et en analysera les bénéfices. Enfin, la section VII.5 proposera une mise en relation du modèle architectural avec le modèle ergonomique.

VII.1 COMPOSANTS ET COMPOSITES

La combinaison de plusieurs composants pour produire un nouveau composant est au cœur des architectures ambiantes. Nous adoptons la formalisation suivante, issue des spécifications UML :

> **Définition** (adaptée de (OMG 2011))
> *Une **interface** [21] représente la déclaration d'un ensemble de fonctionnalités et d'obligations formant un tout public et cohérent. Une interface spécifie un contrat qui doit être respecté par l'entité qui réalise l'interface (i.e. l'implémente).*
> *Un **composant**[22] représente une partie modulaire du système qui encapsule son contenu et que l'on peut remplacer sans modifier l'impact sur son environnement. Un composant définit son comportement en termes d'interfaces requises et exposées. Un composant peut donc se substituer à un autre seulement si leurs types sont conformes.*

---

[21] Page 86 : *An interface […] represents a declaration of a set of coherent public features and obligations. An interface specifies a contract; any instance of a classifier that realizes the interface must fulfill that contract.*

[22] Page 149 : *A component represents a modular part of a system that encapsulates its contents and whose manifestation is replaceable within its environment. […] A component defines its behavior in terms of provided and required interfaces. […] One component may therefore be substituted by another only if the two are type conformant. Larger pieces of a*

La Figure 51 propose un modèle des interfaces et des composants.

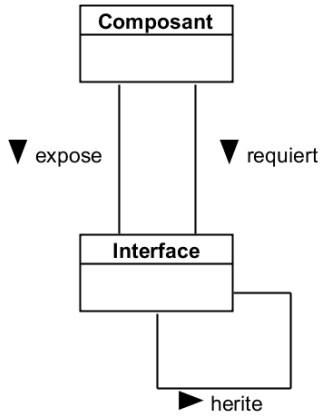


FIGURE 51 - LIENS ENTRE LES CONCEPTS D'INTERFACE ET DE COMPOSANT

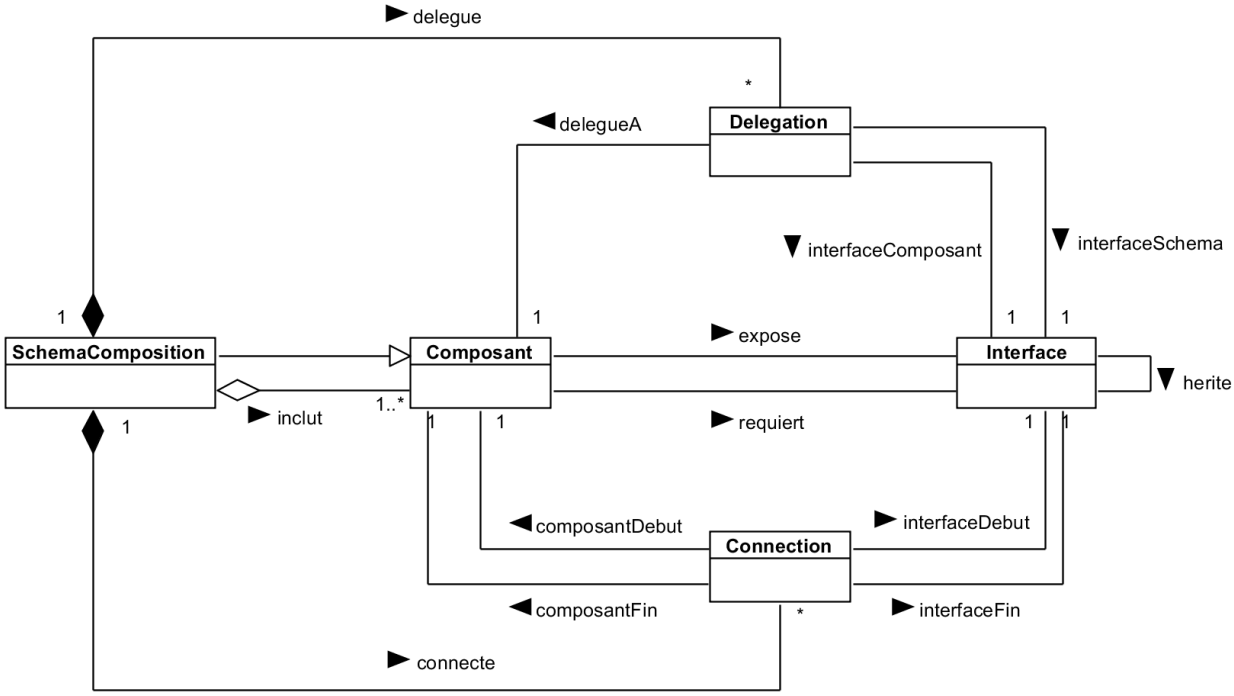


FIGURE 52 - MODELE DU DOMAINE DE LA COMPOSITION DE COMPOSANTS

Cette définition nous permet de définir ci-dessous les entités décrivant une association de composants, telle que celle-ci a été définie dans la section V.1.1. Pour rappel, la traduction implicite des diagrammes OMT dans le langage mathématique est spécifiée dans l'annexe

---

system's functionality may be assembled by reusing components as parts in an encompassing component or assembly of components, and wiring them together.

A. Cette spécification nous permet de rédiger formellement des contraintes structurelles telles que celles de la page suivante.

> **Définition** (adaptée de (OMG 2011))
> *« Des fonctionnalités de plus haut niveau dans le système peuvent être fournies en réutilisant des composants comme morceaux d'un composant de plus haut niveau ou bien en assemblant et **câblant** plusieurs composants entre eux.* »
>
> *On appelle **schéma de composition** le diagramme représentant ce câblage.*

Le schéma de composition est modélisé dans la Figure 52, à laquelle nous associons les contraintes suivantes :

- Les connexions sont bien formées
  - Elles ne concernent que des composants inclus (pas de composants externes)
  - Les types sont compatibles

$$\forall co, c_1, c_2, i_1, i_2 \ \ connecte\ \ sc, co$$
$$\wedge\, composantDebut\ \ co, c_1$$
$$\wedge\, composantFin\ \ co, c_2$$
$$\wedge\, interfaceDebut\ \ x, i_1$$
$$\wedge\, interfaceFin\ \ x, i_2$$
$$\Rightarrow$$
$$inclut\ \ sc, c_1\ \ \wedge inclut(sc, c_2) \qquad (1)$$
$$\wedge\, herite\ \ i_2, i_1\ \ \wedge requiert\ \ c_1, i_1\ \ \wedge \qquad (2)$$
$$expose(c_2, i_2)$$

- Les délégations sont bien formées

1. Elles ne concernent que des composants inclus (pas de composants externes)
2. Les types sont compatibles
3. Une interface requise (resp. exposée) par le schéma est déléguée à une interface requise (resp. exposée) par le composant interne

$$\forall sc, d, c, i_1, i_2 \ \ delegue\ \ sc, d\ \ \wedge delegueA\ \ d, c$$
$$\wedge\, interfaceSchema\ \ d, i_1\ \ \wedge interfaceComposant(d, i_2)$$
$$\Rightarrow$$
$$inclut\ \ sc, c \qquad (1)$$
$$\wedge\, herite\ \ i_2, i_1 \qquad (2)$$
$$\wedge\quad requiert\ \ sc, i_1\ \ \wedge requiert\ \ c, i_2$$
$$\vee\ \ expose\ \ sc, i_1\ \ \wedge expose(sc, i2) \qquad (3)$$

- Toutes les interfaces exposées ou requises par le schéma sont déléguées

$$\forall sc, i\ \ SchemaComposition\ \ sc\ \ \wedge\ \ requiert\ \ sc, i\ \ \vee expose\ \ sc, i$$
$$\Rightarrow \exists d\ \ delegue\ \ sc, d\ \ \wedge interfaceSchema\ \ d, i$$

La Figure 25 (cf. page 102) illustre le schéma de composition d'un composant « LampeX10 » qui vérifie les conditions ci-dessus. Les connexions entre composants internes y sont reflétées par la notation graphique socket-lollipop et les délégations (lien entre composant interne et externe) par un carré blanc suivi d'un lien en trait plein. L'intérêt du modèle proposé est de pouvoir représenter en interne, les diagrammes de composants introduits en section V.1.1. Cette représentation formelle permet au système de raisonner sur les liens constituant le schéma et ainsi de les instancier automatiquement.

VII.2 VUE D'ENSEMBLE DE L'ARCHITECTURE DANS DAME

Dans les modèles d'architecture classiques, on peut dégager 3 types d'entités distinctes :

- Le système fonctionnel

  Dans notre cas, il s'agit du système gérant la sphère d'activité. Il fournit au système interactif un noyau fonctionnel dont la structure peut varier au cours du temps.

- Le système d'interaction

  Il s'agit du composant qui réalise l'interface entre le système et l'utilisateur via les médias. Il instancie les composants d'interaction dédiés (CID).

- Le média

  Il s'agit de l'intermédiaire physique du système dans le monde réel.

La Figure 53 illustre la place qu'occupent ces 3 entités dans l'Ambiant. Elle fait notamment apparaitre un aspect important de l'Ambiant : les tâches, les composants d'interaction et les périphériques interactifs sont extérieurs au système. En effet, les tâches et les CID sont le résultat de développements faits par des équipes extérieures qui sont ensuite installées dans le système, répertoriés par des librairies de composants disponibles. De même, les périphériques d'interaction sont des entités imposées par le monde physique, le concepteur du système d'interaction ne sait pas à l'avance lesquels seront disponibles.

Les flèches affichées avec un trait épais représentent la chaine d'interaction traditionnelle. Cette chaine, aussi appelée « arche » dans le modèle ARCH est implémentée de façon monolithique et figée dans les systèmes d'exploitation actuels. Le système d'interaction ambiant doit être en mesure de la construire dynamiquement. Nous emploierons pour cela des composants logiciels qui pourront être associés pour interfacer le noyau fonctionnel, le CID et les médias. Les couches logicielles réalisant une IHM sont illustrées dans la Figure 54. On appelle l'ensemble des composants liant le CID aux périphériques d'interaction le **MEDIATEUR** car il réalise une médiation entre les périphériques d'interaction et les besoins utilisateurs instanciés par un CID. L'association de composants qui permet d'interfacer le noyau fonctionnel avec l'utilisateur est appelée **CHAINE D'INTERACTION**.

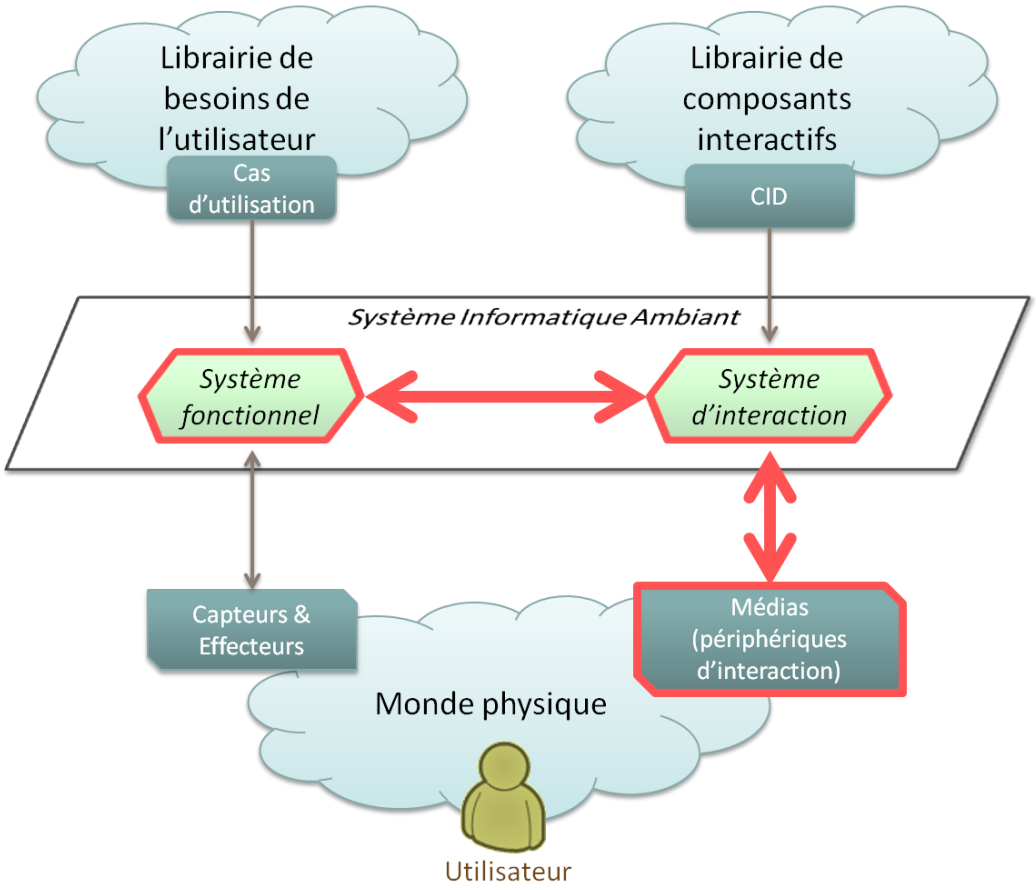


FIGURE 53 – ROLES DES COMPOSANTS CLASSIQUES D'UNE IHM DANS L'AMBIANT

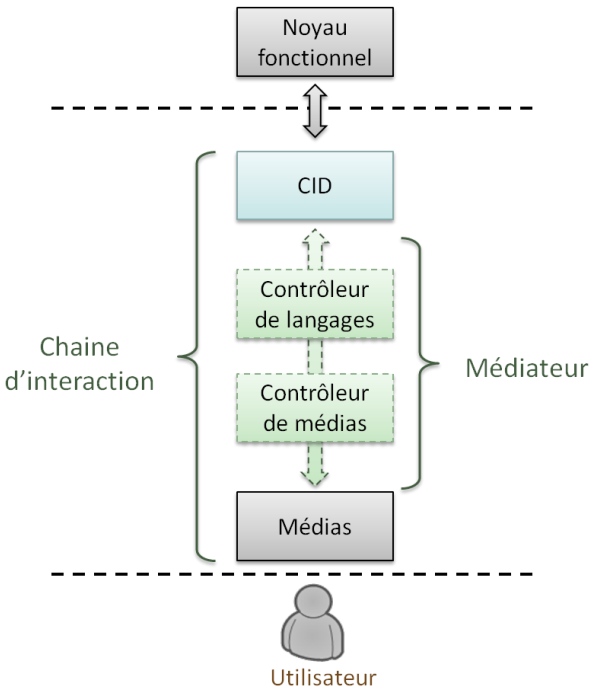


FIGURE 54 – LES COUCHES LOGICIELLES IMPLEMENTANT UNE IHM

Dans la Figure 54, le noyau fonctionnel et les périphériques d'interaction sont des entités imposées au système interactif. Celui-ci a pour rôle de proposer l'accomplissement de

certains besoins utilisateurs en sélectionnant les composants d'interaction qui seront les plus adaptés. Cette sélection est réalisée en tenant compte de contraintes structurelles

- L'ensemble des fonctionnalités requises par le CID doit être disponible dans le noyau fonctionnel.
- Les périphériques d'interaction disponibles doivent permettre l'instanciation du noyau fonctionnel.

L'application de ces contraintes permet de construire dynamiquement les chaines d'interaction. L'algorithme réalisant cette construction fait partie du modèle comportemental, il est présenté dans le chapitre VIII. Le but du modèle d'architecture est d'assister à la construction automatique de cette chaine, en répondant notamment aux questions suivantes :

(a) Un CID donné peut-il être instancié au regard des fonctionnalités offertes par le noyau fonctionnel ?
(b) Un CID donné peut-il être instancié au regard des médias disponibles ?

La réponse à la question (a) requiert une définition du noyau fonctionnel et du CID qui fasse explicitement apparaitre les fonctionnalités (i.e. les interfaces) par lesquelles ils communiqueront. La question (b) est plus subtile car le lien entre le CID et les médias n'est pas direct. En effet, un CID exprime un dialogue interactif reposant sur un langage d'interaction spécifique. Nous introduisons donc deux composants intermédiaires permettant de réaliser la liaison entre le CID et les médias. Le **CONTROLEUR DE MEDIAS** réalise une abstraction des médias présents et propose au reste du système un ensemble de structures de données qui permettent la réalisation de certaines modalités. Le **CONTROLEUR DE LANGAGES** s'appuie sur ces structures de données pour construire un (ou des) langages d'interaction cohérents. Il expose une API permettant au CID d'instancier un dialogue spécifique (i.e. dédié à un besoin utilisateur) dans ce(s) langage(s) d'interaction. Ainsi, la définition du langage d'interaction ne repose pas sur un ensemble de médias prédéfinis, elle est adaptable à plusieurs jeux de périphériques. Cette flexibilité sera étudiée plus en détails dans le chapitre IX.

Le schéma présenté dans la Figure 54 peut être plus formellement représenté par la Figure 55 qui définit les 5 catégories de composants présents dans le modèle architectural. Ces composants sont présentés plus en détail dans la section suivante.

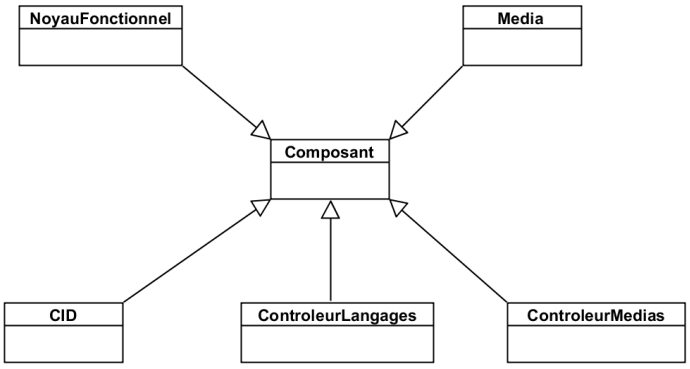


FIGURE 55 - CATEGORIES DE COMPOSANTS DU MODELE ARCHITECTURAL

Notre démarche consiste donc à proposer un schéma de composition type qui permet l'élaboration de la chaîne d'interaction. Nous envisageons que les équipes travaillant sur le système ambiant développeront des composants qu'elles attribueront à chacune de ces catégories. Le système interactif disposera donc d'une bibliothèque de composants pour chacune des 3 catégories suivantes : CID, Contrôleur de langages et Contrôleur de médias. Les catégories Noyau fonctionnel et Médias étant imposées par le monde extérieur, le système interactif devra sélectionner dans ses bibliothèques, les composants qui permettent de réaliser le chainage du noyau fonctionnel aux médias.

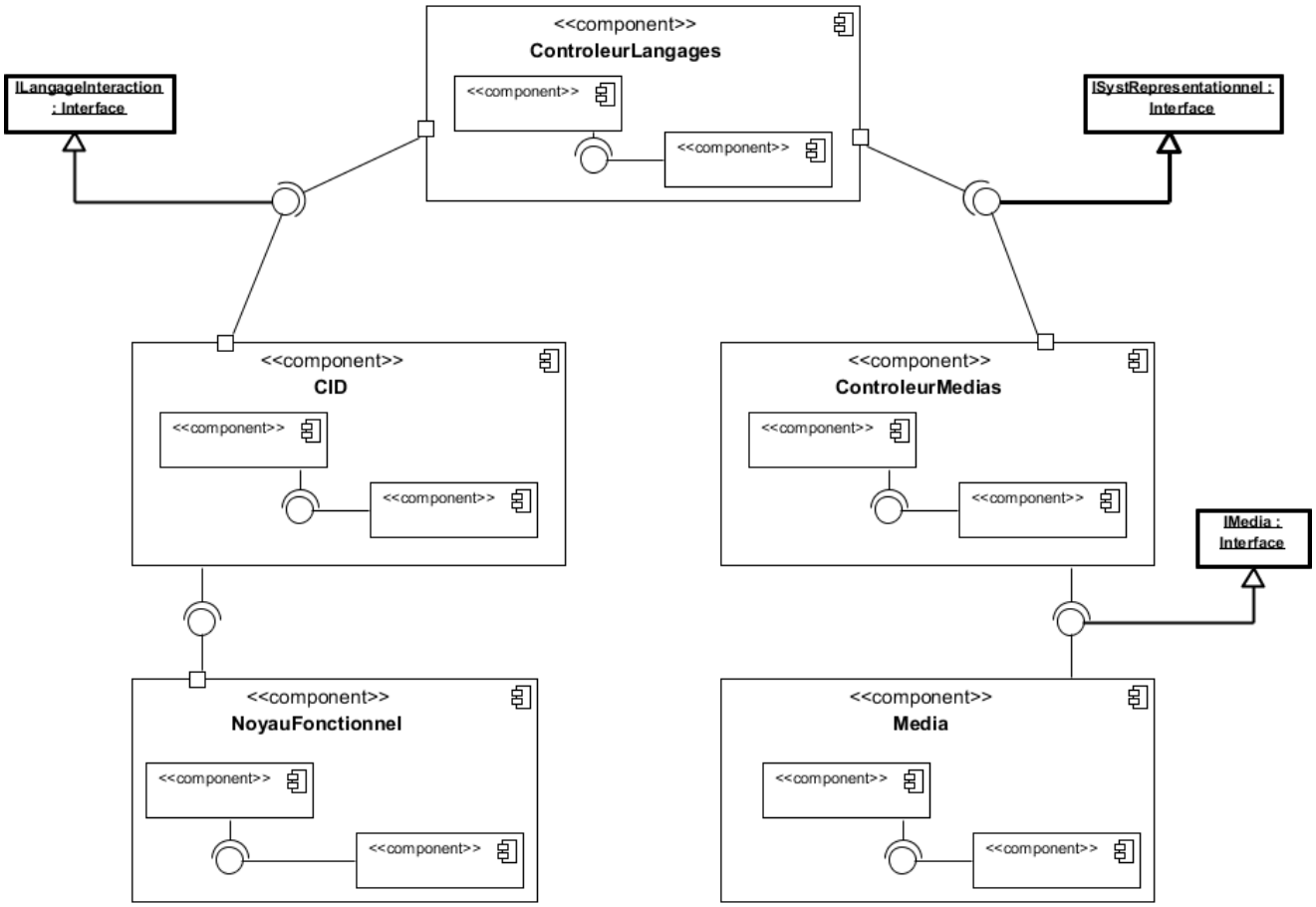


FIGURE 56 – VUE COMPOSITE DU MODELE ARCHITECTURAL D'UNE CHAINE D'INTERACTION

A ce niveau d'analyse, les dépendances entre les différentes catégories de composants ne sont pas établies, elles varieront d'une implémentation à l'autre de chaque composant. Cependant, nous typons les liens entre les différentes catégories de composant en introduisant les interfaces abstraites suivantes : *IMedia*, $ISystRepresentationnel$, *ILangageInteraction*.

Plus formellement[23] :

$$IMedia, ISystRepresentationnel, ILangageInteraction \subset \mathcal{E}_{Interface}$$

[23] Pour rappel, $\mathcal{E}_{Interface}$ désigne l'extension du prédicat $Interface$ défini implicitement par la Figure 51, page 45.

Chaque composant pourra lui-même faire intervenir une association de sous-composants. Une représentation plus précise de la Figure 54 serait donc celle proposée dans la Figure 56. Les interfaces abstraites nous permettront alors de veiller au caractère bien formé de la structure générale et nous servira notamment durant le processus de formation de cette application composite.

Dans la section suivante, nous allons détailler le rôle de chacune de ces catégories de composants.

VII.3 DETAIL DES COMPOSANTS

Cette section donne une définition plus précise de chaque catégorie de composants du modèle architectural, accompagnée d'exemples illustratifs. L'exemple développé tout au long de cette présentation sera celui d'un centre multimédia. Nous avons présenté une vue d'ensemble des cas d'utilisation associés dans la Figure 50, page 139. A l'issue de cette section, nous disposerons d'un modèle complet d'une IHM conventionnelle permettant de réaliser ces besoins.

VII.3.1 NF – LE NOYAU FONCTIONNEL

Le noyau fonctionnel d'une tâche ambiante est construit dynamiquement, sur la base d'une composition de services réalisée par le système ambiant (i.e. hors du système d'interaction). Il expose et requiert des interfaces représentant les fonctionnalités du système. Ces interfaces sont réalisées par association de composants de complexité inférieure, eux-mêmes résultats d'éventuelles compositions. On peut ainsi structurer récursivement le noyau fonctionnel fournit au système interactif par son schéma de composition.

Définition

Nous définissons le **noyau fonctionnel** (**NF**) comme étant une catégorie particulière de composants qui fournit les fonctionnalités du domaine applicatif de la tâche en cours. C'est ce noyau fonctionnel qui permet de satisfaire les besoins de l'utilisateur et qui doit pour cela être interfacé.
Le noyau fonctionnel est représenté par le prédicat $NoyauFonctionnel$.

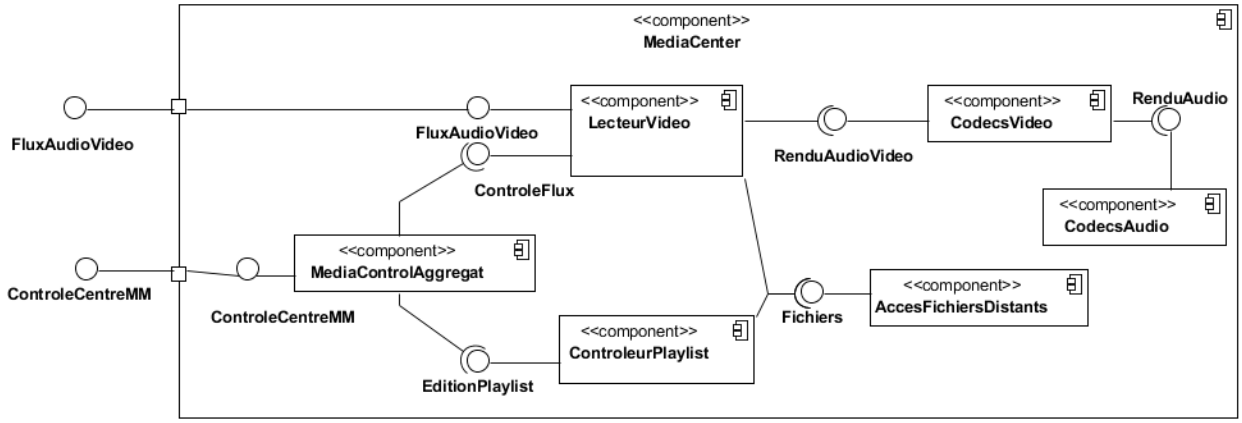


FIGURE 57 – EXEMPLE DE NOYAU FONCTIONNEL D'UN SYSTEME DE RENDU MULTIMEDIA

Exemple

La Figure 57 illustre un exemple de noyau fonctionnel qui couvre les fonctionnalités d'un système multimédia. Ce NF expose une interface de contrôle appelée `ControleCentreMM` qui agrège les fonctions de contrôle du rendu audio-vidéo avec celles de contrôle de la liste de lecture. Cette interface permet également d'être averti des changements d'état des variables d'état concernant la lecture du fichier (position dans la piste lue, position dans la liste de lecture…). Il expose une seconde interface nommée `FluxAudioVideo` qui permet de récupérer le flux multimédia en cours de lecture.

VII.3.2 LE PERIPHERIQUE D'INTERACTION

Le média est un périphérique qui permet à l'utilisateur d'interagir avec le système selon un ou plusieurs langages de média. Il a été défini dans le modèle ergonomique par le prédicat $Media$. Nous étendons le média en tant qu'objet du monde physique par sa représentation logicielle sous forme de composant du système.

Définition

Un **média** est une catégorie de composants qui établissent une passerelle entre les messages échangés dans le système et le monde physique. Les médias logiques sont chargés de transformer les symboles en action du monde physique (pour les médias en sortie) ou d'interpréter des actions du monde physique en symboles interprétables par le système (pour les médias en entrée).

Par ailleurs, les médias occupent une certaine place dans la hiérarchie des composants du modèle architectural. Cette position est modélisée par le fait que les interfaces qu'il expose/requiert sont de nature à être connectées à un contrôleur de médias :

$\forall m, i \; Media\ m \ \wedge \ expose\ m, i \ \vee \text{requiert m, i} \ \Rightarrow herite(i, IMedia)$

Exemple

La Figure 58 montre comment un clavier et un écran tactile sont représentés dans le modèle architectural. Ces composants seront rarement décomposables en sous-composants car leur implémentation est très liée au matériel. Les interfaces qu'ils exposent sont les protocoles de communication qui permettent au système d'échanger des messages avec eux. Pour le clavier, nous avons choisi une interface reposant sur le format de messages OSC[24]. Pour l'écran tactile, il s'agit de primitives OpenGL pour l'affichage et du protocole TUIO[25] qui décrit des évènements tactiles gérant le *multitouch*.

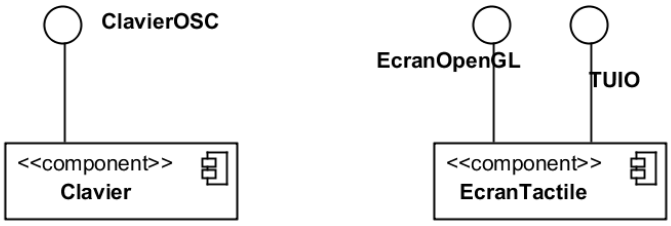


FIGURE 58 – EXEMPLES DE MEDIAS COURANTS

VII.3.3 CM – LE CONTROLEUR DE MEDIAS

Le contrôleur de medias a pour rôle d'abstraire un groupe de périphériques au regard du module qui implémente le langage d'interaction. Plus concrètement, le contrôleur de medias propose des structures de données élémentaires sur lesquelles les langages d'interaction sont construits et se charge de faire l'interface entre ces structures de données et les périphériques. (Bernsen 1994) et (Bouchet & Nigay 2004, p. 105) appellent ces structures de données le « système représentationnel ». Ce sont ces structures de données qui permettent la réalisation d'une modalité qui en est une abstraction.

Définition

Nous définissons le **contrôleur de médias** (**CM**) comme étant une catégorie particulière de composants qui réalise une abstraction des évènements des médias d'entrées et de sortie sous la forme d'une structure de données permettant la réalisation d'une ou plusieurs modalités. Les contrôleurs de médias sont définis par le prédicat $ControleurMedias$.

Par ailleurs, les contrôleurs de médias occupent une certaine place dans la hiérarchie des composants du modèle architectural. Cette position est modélisée par le fait que les interfaces qu'il expose/requiert sont de nature à être connectées soit à un Média, soit à un contrôleur de langage, ce qui se formalise ainsi :

$$\forall m\ ControleurMedias\ cm\ \wedge\ requiert\ cm,i\ \vee \text{expose cm,i}$$
$$\Rightarrow herite\ i,IMedia\ \vee\ herite\ i,ISystRepresentationnel$$

[24] http://opensoundcontrol.org/

[25] http://www.tuio.org/

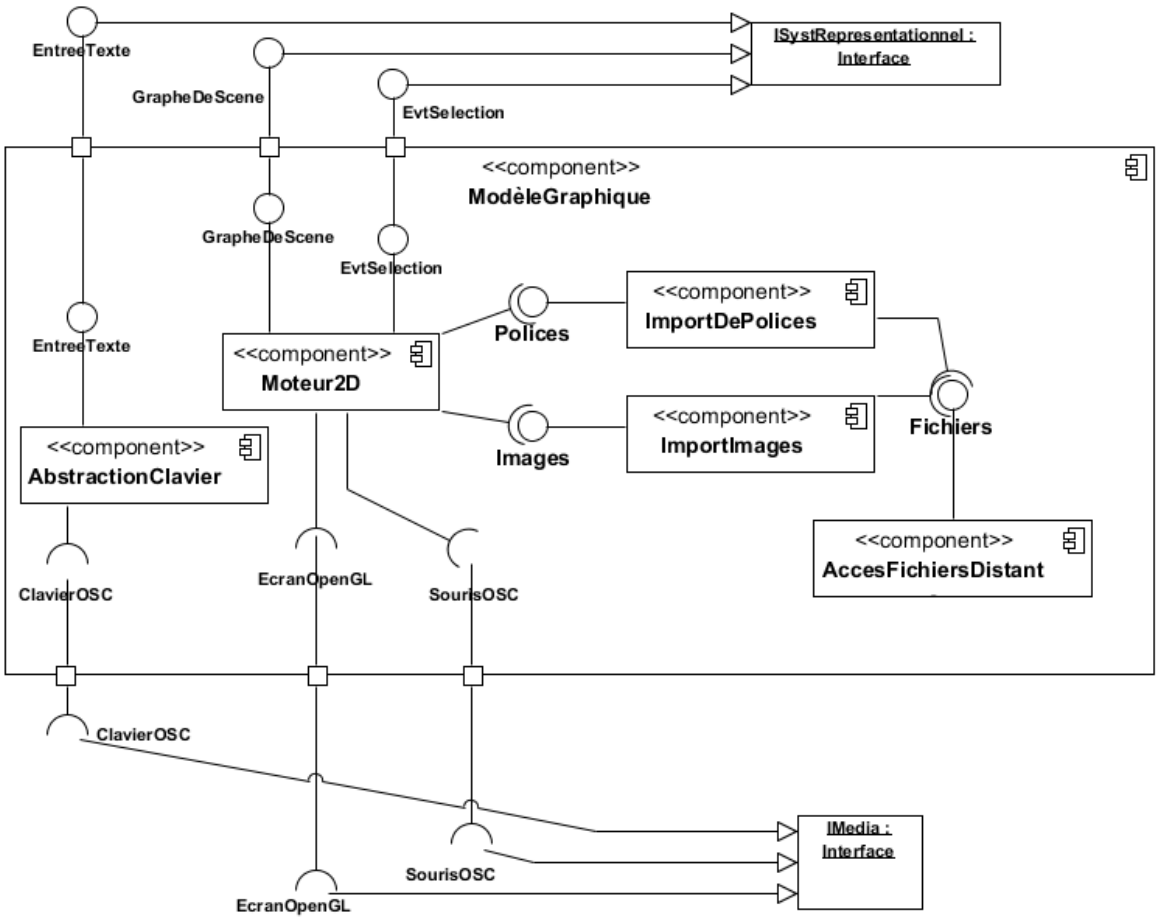


FIGURE 59 - EXEMPLE DE CONTROLEUR DE MEDIA REALISANT LES STRUCTURES TRADITIONNELLES DES INTERFACES GRAPHIQUES

Exemple

La Figure 59 illustre un contrôleur de média qui se connecte à une souris, un écran et un clavier et qui expose une structure de données décrivant un graphe de scène et les évènements de sélection des nœuds du graphe[26] associés.

Les médias sont découplés du contrôleur de médias par les interfaces EcranOpenGL, SourisOSC et ClavierOSC. Ces interfaces décrivent une implémentation concrète de langages de médias abstraits (Ecran, Souris et Clavier). Elles héritent toutes de l'interface IMedia, ce qui permet au système de faire la distinction avec les interfaces GrapheDeScene, EvtSelection et EntreeTexte qui héritent de ISystRepresentationnel. La structure de graphe de scène est obtenue par composition des langages de médias de l'écran et de la souris. Cependant, les évènements du clavier sont traités à part car ils ne permettent pas d'interagir avec le graphe de scène. Ces modifications seront définies dans le langage

[26] Ces évènements de sélection correspondent à la fonctionnalité de « *peeking* ». Le peeking est l'action de retrouver l'objet qui a été sélectionné lors d'un clic à la souris dans une image 2D ou 3D.

d'interaction, grâce à des fonctionnalités plus évoluées telles que le focus. La différence entre ClavierOSC et EntreeTexte réside dans le degré d'abstraction de ces deux interfaces. L'une décrit les informations brutes (quelle touche a été enfoncée, pendant combien de temps…) tandis que l'autre fournit une représentation indépendante du protocole utilisé et de la configuration du clavier (i.e. des symboles).

VII.3.4 CL – LE CONTROLEUR DE LANGAGE

Le contrôleur de langage implémente un langage d'interaction en définissant les structures génériques de dialogue entre le système et l'utilisateur (i.e. la grammaire de l'interaction). Il s'appuie pour cela sur le système représentationnel fourni par le contrôleur de média (i.e. les structures de données qui permettent la réalisation des modalités).

> **Définition**
> Nous définissons les **contrôleurs de langages** (**CL**) comme étant une catégorie particulière de composants qui implémentent les structures génériques (i.e. réutilisable) du dialogue homme-machine. Il fournit une abstraction de la grammaire d'interaction qui peut ensuite être utilisée par les CID pour générer des dialogues interactifs spécifiques. Les contrôleurs de langage sont définis par le prédicat $ControleurLangage$.

Par ailleurs, les contrôleurs de langage occupent une certaine place dans la hiérarchie des composants du modèle architectural. Cette position est modélisée par le fait que les interfaces qu'il expose/requiert sont de nature à être connectées soit à un CID, soit à un contrôleur de médias, ce qui se formalise ainsi :

$$\forall m\ ControleurLangage\ cl \ \wedge\ requiert\ cl, i\ \vee \text{expose cl, i}$$
$$\Rightarrow herite\ i, ISystRepresentationnel$$
$$\vee\ herite\ i, ILangageInteraction$$

Exemple
Si l'on reprend l'exemple précédent, un contrôleur de langage associé à un contrôleur de médias offrant les structures graphe de scène, évènements de sélection et entrée texte pourrait instancier le langage d'interaction WIMP, c'est-à-dire l'interaction graphique traditionnelle reposant sur une librairie de widgets.

Ainsi, dans la Figure 60, le graphe de scène est contrôlé par une librairie de widgets. L'espace est donc transformé en un graphe de widgets. Le gestionnaire de fenêtre gère ce graphe de widgets et le structure avec les notions de fenêtre, menu et d'icône. Il utilise pour cela un système d'automates qui peut être implémenté de façon *ad-hoc*, ou en réutilisant une librairie existante comme c'est le cas dans la figure. Ce gestionnaire expose ensuite ces fonctionnalités sous la forme d'une interface qui est exposée vers le CID (et qui hérite donc de l'interface ILangageInteraction). Cette interface décrit les opérations que l'on peut effectuer dans le langage d'interaction WIMP. Dans notre exemple, nous avons représenté cette API par l'API X11 qui est un standard ouvert. Cependant, le langage WIMP peut être implémenté en suivant différentes API.

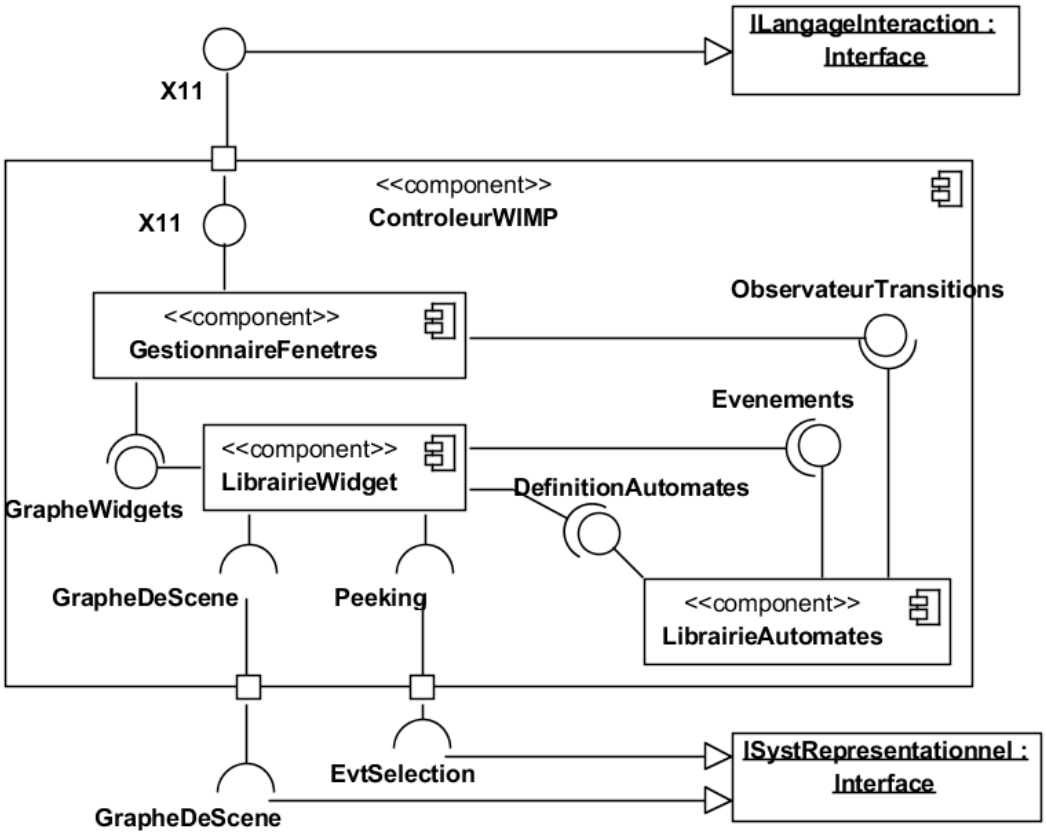


FIGURE 60 - EXEMPLE DE CONTROLEUR DE LANGAGE WIMP

VII.3.5 CID – LE COMPOSANT D'INTERACTION DEDIE

Le CID est le composant qui réalise un dialogue homme-machine spécifique à un besoin d'interaction entre le système et l'utilisateur. Ce composant s'appuie sur un (ou plusieurs) langage(s) d'interaction pour fournir un dialogue qui s'interface avec les fonctionnalités du noyau fonctionnel.

Définition

Un **composant d'interaction dédié** (**CID**) est une catégorie de composants logiciels développés dans le but de remplir un besoin utilisateur précis par l'implémentation d'un dialogue homme-machine qui s'appuie sur un langage d'interaction. Il interface les fonctionnalités définies dans le noyau fonctionnel. Les CID sont déclarés par le prédicat CID.

Par ailleurs, les CID occupent une certaine place dans la hiérarchie des composants du modèle architectural. Cette position est modélisée par le fait qu'il doit être connecté à un CL, à travers au moins une interface :

$$\forall c\ CID\ c\ \Rightarrow \exists i\ \ requiert\ c,i\ \vee \text{expose c,i}\ \ \wedge herite(i, ILangageInteraction)$$

Exemple

Les CID sont implémentés de façon complètement *ad-hoc* et offrent ainsi au concepteur toute liberté d'implémentation.

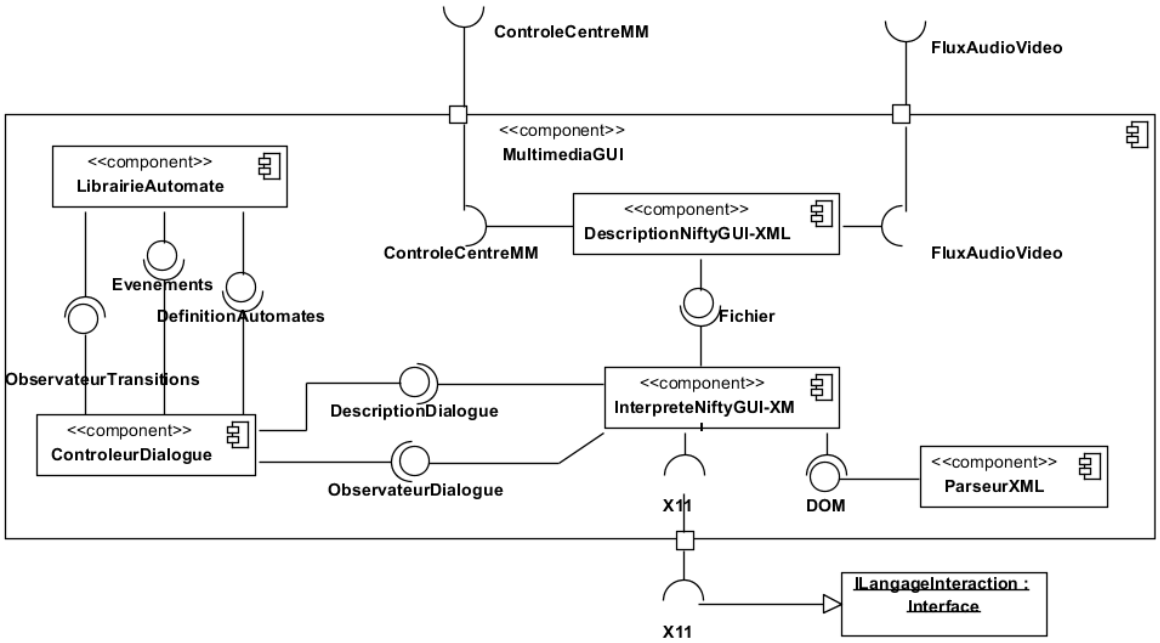


FIGURE 61 – EXEMPLE D'UN CID IMPLEMENTANT UNE INTERFACE GRAPHIQUE POUR UNE APPLICATION MULTIMEDIA EN S'APPUYANT SUR LE LANGAGE D'INTERACTION WIMP

La Figure 61 est un exemple d'interface graphique développée pour satisfaire le cas d'utilisation racine de l'application centre multimédia (c'est-à-dire le cas d'utilisation « manipuler des fichiers multimédias » dans la Figure 50, page 139). Dans cet exemple, l'interface a été développée grâce au langage NiftyGUI-XML[27] qui permet de décrire un ensemble de widgets graphiques dans un format XML. Cet exemple montre que les CID peuvent soit reposer sur une implémentation *ad-hoc,* soit sur un standard dont l'interprète est inclus dans le schéma de composition. Il démontre également que certaines fonctionnalités sont transverses à toute l'architecture et peuvent être réutilisées à différents niveaux. C'est le cas, dans cet exemple, du composant LibrairieAutomate.

VII.4 ANALYSE DE CAS – MODELISATION D'UNE INTERFACE GRAPHIQUE

Pour illustrer la décomposition que nous proposons, prenons l'exemple d'une interface conventionnelle utilisant un écran, une souris et un clavier et proposant une interface graphique pour un système multimédia. Voici une décomposition possible de cette situation.

[27] http://nifty-gui.lessvoid.com/

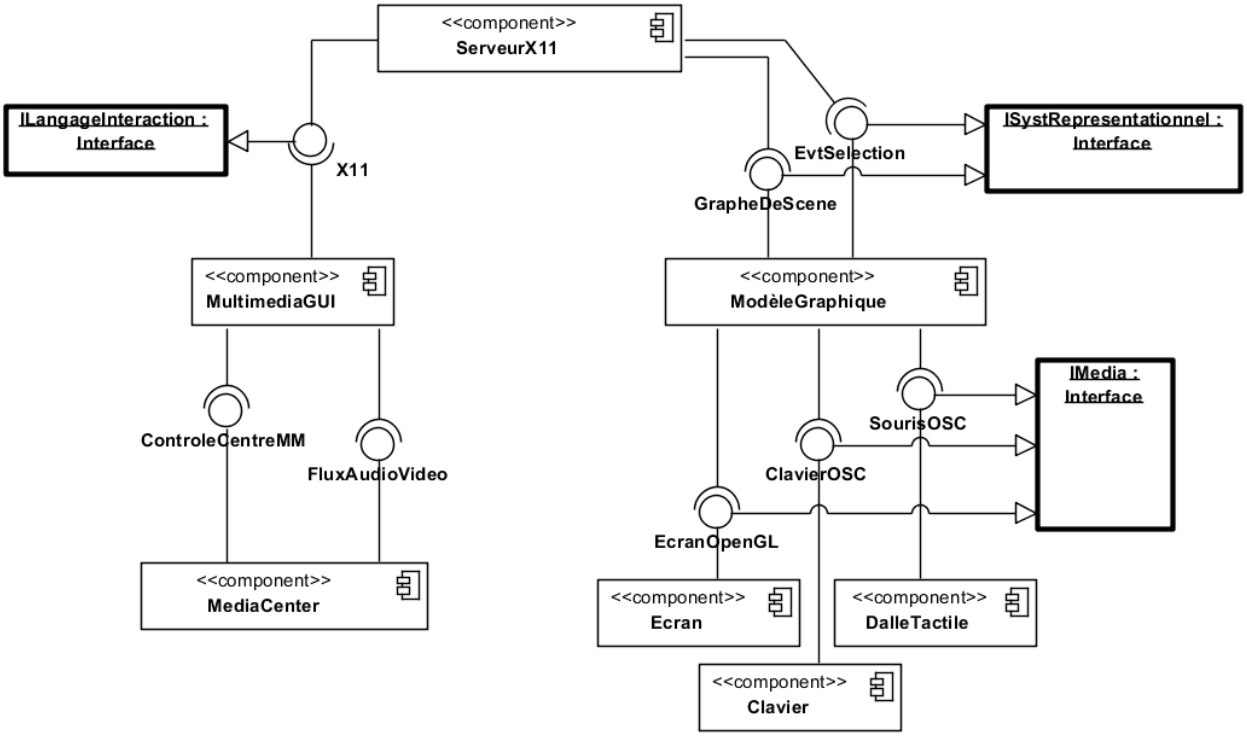


FIGURE 62 - MODELISATION D'UNE INTERFACE GRAPHIQUE TRADITIONNELLE POUR DU RENDU MULTIMEDIA

La Figure 62 réutilise le noyau fonctionnel de la Figure 57 pour illustrer l'implémentation d'une interface graphique quelconque. La réalisation d'un cas d'utilisation peut nécessiter plusieurs CID. Dans cet exemple, nous illustrons la réalisation du besoin « manipuler des fichiers multimédias » à travers un unique CID. Cependant, on peut imaginer tout un ensemble de CID qui couvrent uniquement certains aspects du cas d'utilisation racine. En associant ces différents CID, on pourra également satisfaire le cas d'utilisation racine.

Le choix d'un ensemble de CID qui couvrent tous les aspects d'un cas d'utilisation est discuté dans l'algorithme du modèle comportemental, section VIII.2.1. Nous verrons, au cours du chapitre IX, que ce choix est gouverné à la fois par la structure du modèle comportemental (et la satisfaction des contraintes en terme d'interfaces exposées et requises) et également par des recommandations produites par les concepteurs ou les ergonomes d'Ambiant. Ces recommandations sont formulées en utilisant les termes des modèles ergonomique et architectural. Il est donc primordial d'établir un lien entre ces deux modèles, ce que nous présentons dans la section suivante.

VII.5 LIENS ENTRE LES MODELES ERGONOMIQUE ET ARCHITECTURAL

Nous ajoutons au modèle architectural un ensemble de relations binaires qui permettent de faire le lien entre les composants d'une chaine d'interaction et leur description ergonomique. L'intérêt de cette description est de pouvoir identifier certaines caractéristiques communes des chaines d'interaction qui sont masquées par des implémentations différentes. Par exemple, le langage d'interaction WIMP peut être

exprimé à travers différentes interfaces. L'interface X11 proposée dans la Figure 60, page 153, n'est pas la seule implémentation à définir le langage d'interaction WIMP. En effet, chaque système d'exploitation se constitue sa propre API pour représenter ce langage. Pourtant, sur le plan ergonomique, il s'agit du même langage d'interaction car les modèles mentaux que s'en construit l'utilisateur sont très similaires. Cette factorisation de critères ergonomiques permet la rédaction de recommandations dans le modèle comportemental qui soient générales (i.e. qui s'appliquent à toutes instanciations d'un même langage). Il en va de même des notions de langages de média, ou de modalités.

Ces correspondances sont établies *a posteriori* par les concepteurs des composants qui, une fois un composant développé, décrivent ce dernier dans les termes des modèles ergonomique et architectural.

Définition
Nous définissons les relations binaires suivantes :
$realiseLI$ qui associe un controleur de langage aux langages d'interaction qu'il implémente.
$realiseModalite$ qui associe un contrôleur de médias aux modalités que son système représentationnel permet de réaliser.
$realiseCanal$ qui associe un média aux différents canaux qu'il implémente.
$realiseCU$ qui associe un CID aux cas d'utilisation qu'il réalise.

Ces relations binaires sont illustrées dans la Figure 63 sur l'exemple du centre multimédia. Nous verrons dans les chapitres VIII et IX comment ces relations sont exploitées pour décrire des recommandations ergonomiques et comment ces recommandations sont appliquées à l'exécution.

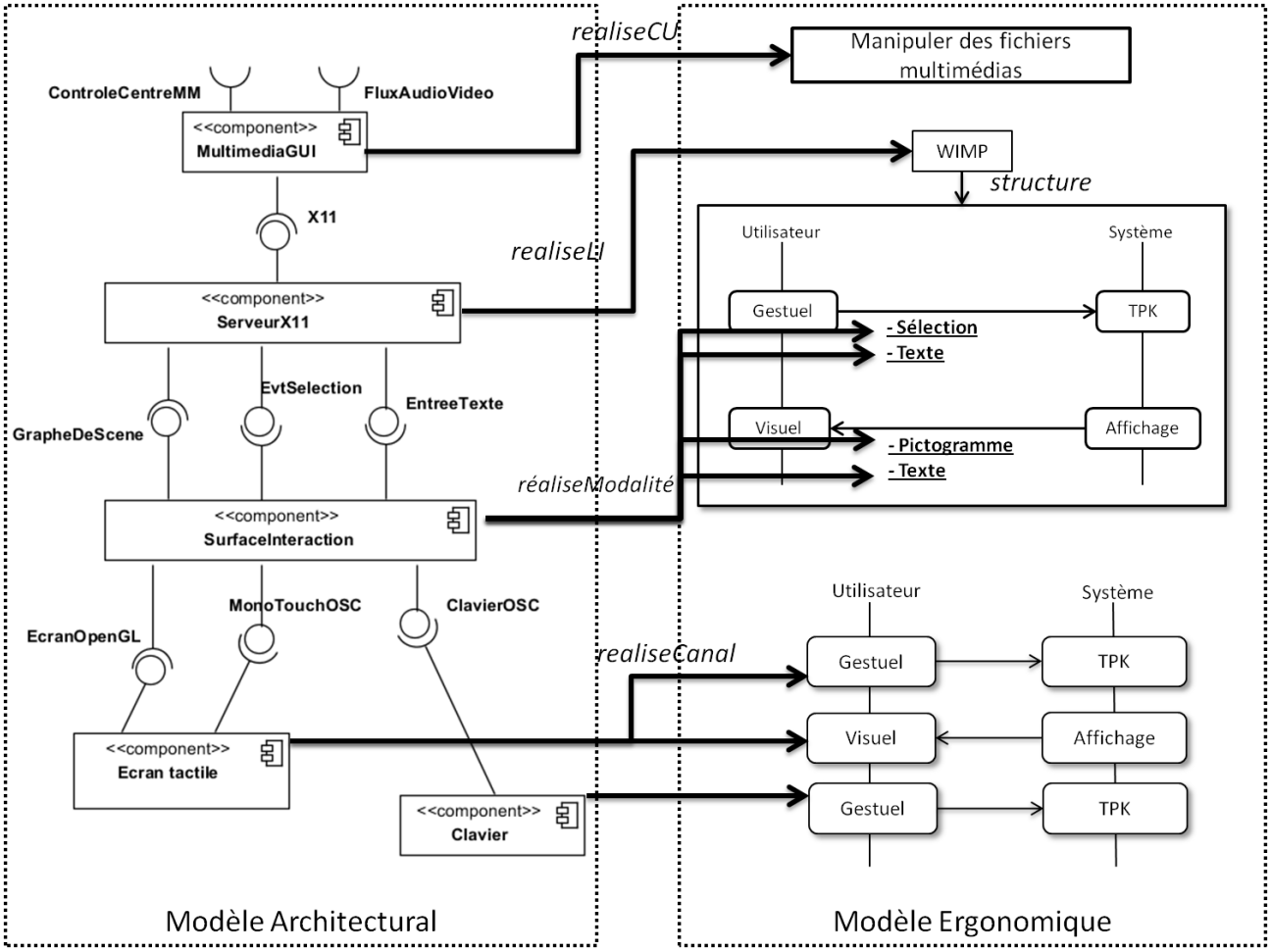


FIGURE 63 – LIENS ENTRE LES MODELES ERGONOMIQUE ET ARCHITECTURAL

VII.6 CONCLUSION

Le modèle architectural décrit une infrastructure logicielle d'IHM qui repose sur l'association de composants réutilisables. Le développement de tels composants vise à favoriser la flexibilité et la réutilisabilité des techniques d'interaction. Cette flexibilité est apportée par le modèle de composants qui découple l'implémentation et l'usage grâce à la notion d'interface (qui définit ici une abstraction d'un ensemble de fonctionnalités, à ne pas confondre avec l'IHM). Ce découplage permet une intégration flexible des composants et une factorisation du code.

Dans les approches multi-médias présentées dans le chapitre IV, les composants ne sont pas typés, ils peuvent être librement connectés entre eux. Cependant, ayant pour objectif final la composition automatique des composants, nous structurons la chaine de composants réalisant l'IHM. Cette chaine se décompose en 4 éléments : le CID, le contrôleur de langages, le contrôleur de médias et les médias. Cette décomposition guide l'implémentation pour les concepteurs en les aidant à identifier les trois niveaux d'abstraction qui permettent la réutilisabilité de la technique d'interaction dans divers

contextes, à savoir le langage de média, le système représentationnel et le langage d'interaction.

Ces 3 niveaux d'abstraction sont formalisés dans le modèle ergonomique. Leurs caractéristiques y sont décrites, ce qui permet d'écrire des recommandations ergonomiques générales et de les appliquer aux chaines d'interaction concernées en faisant le lien entre les modèles architectural et ergonomique.

Nous allons présenter, dans le chapitre suivant, un algorithme permettant la sélection des CID satisfaisant un besoin, la construction des chaines interactives qui permettent l'instanciation de ces CID et enfin, la sélection des chaines interactives les plus pertinentes au regard de recommandations fournies par le concepteur ou l'ergonome d'Ambiant.

Chapitre VIII
MODELE COMPORTEMENTAL ET ALGORITHME D'ADAPTATION DE L'INTERFACE

Nous présentons, dans ce chapitre, les différentes étapes qui permettent de passer d'un ensemble de besoins utilisateurs à une IHM adaptée aux médias disponibles. Nous présentons tout d'abord les grandes lignes de l'algorithme ainsi que la méthodologie suivie pour évaluer l'adéquation d'une chaine interactive au contexte. Nous étudierons ensuite dans le détail chacune des étapes de l'algorithme. Une description plus succinte de l'algorithme est présentée dans (Pruvost et al. 2011).

VIII.1 SELECTION, INSTANCIATION ET ADAPTATION DYNAMIQUE DE L'IHM

VIII.1.1 LES GRANDES LIGNES DE L'ALGORITHME

L'algorithme proposé s'appuie sur les informations contenues dans les instances des modèles précédemment présentés pour construire une IHM qui soit cohérente sur les plans architecturaux et ergonomiques.

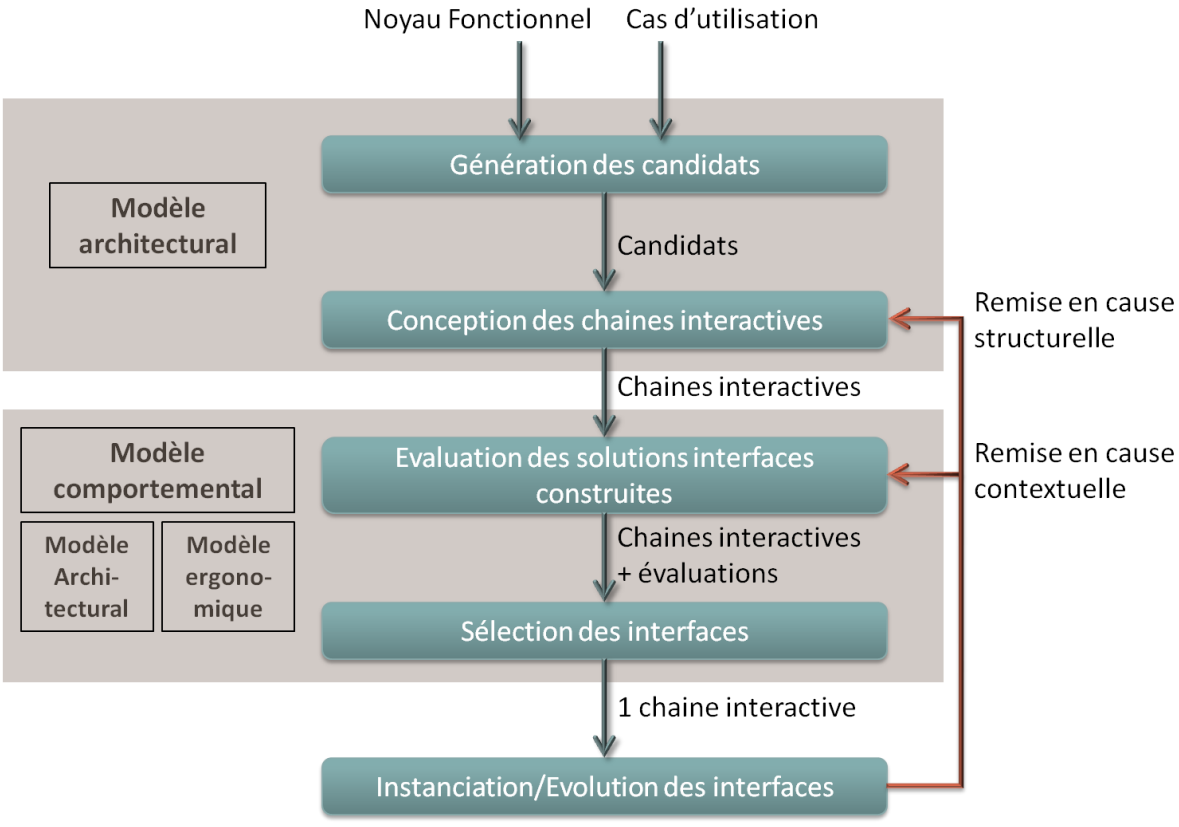


FIGURE 64 - ETAPES DE L'ALGORITHME D'APPLICATION DU MODELE COMPORTEMENTAL

Notre algorithme se décompose en 5 étapes clés qui sont illustrées dans la Figure 64. Nous détaillons chacune de ces étapes dans la section suivante. L'algorithme reçoit en entrée les données envoyées par le système, c'est-à-dire un noyau fonctionnel ainsi qu'un ensemble de cas d'utilisation à réaliser. La première étape consiste à sélectionner les candidats (c'est-à-dire les combinaisons de CID) qui pourraient remplir les cas d'utilisation tout en étant compatibles avec le noyau fonctionnel. Ensuite, pour chacun de ces candidats, il faut être en mesure de proposer une (ou plusieurs) chaines interactives qui permettent de les connecter aux médias existants. Ces deux premières étapes reposent uniquement sur l'usage du modèle architectural.

Une fois que cet ensemble de solutions structurellement valides est construit, les 2 étapes suivantes consistent à évaluer et sélectionner la solution qui fournit la « meilleure » interface. Cette notion de solution optimale dépend des recommandations qui sont définies par les concepteurs et ergonomes d'Ambiant. Cet ensemble de recommandations est appelé **MODELE COMPORTEMENTAL**. L'application du modèle comportemental consiste à évaluer les différentes chaines interactives selon des **CRITERES**. La définition et l'application de ces critères est détaillée dans la section VIII.2.3. A partir des évaluations fournies par le modèle comportemental, plusieurs politiques de sélection des solutions sont envisageables. Nous les présentons dans la section VIII.2.4.

Une fois les solutions optimales sélectionnées, elles sont instanciées et le système d'interaction ambiant a en charge de veiller à ce qu'elles restent optimales. Si un changement dans le contexte intervient, le système d'interaction ambiant relance la procédure vers :

- La 2ème étape lorsque les solutions ne sont plus applicables physiquement ou logiquement. C'est le cas lors de remises en cause de la structure de la solution. Par exemple, si l'un des périphériques employé par la solution venait à disparaitre, il faudrait recalculer toutes les chaines interactives qui reposent dessus.
- La 3ème étape lorsque les solutions restes applicables mais que les changements de contexte remettent en cause leur optimalité.

VIII.1.2 DEMARCHE DE FORMALISATION DE L'ALGORITHME

Nous avons pris soin de décrire les modèles d'architecture et d'ergonomie en logique du premier ordre. Cette formalisation nous fournit une structure déclarative solide pour laquelle de nombreux algorithmes de raisonnement automatique existent déjà. Parmi ceux-ci nous utiliserons à plusieurs reprises l'**unification**. L'unification consiste, pour une formule logique donnée, à associer à chacune des variables libres de la formule une constante qui fasse partie du modèle. Autrement dit, elle permet de chercher les substitutions (t-uples de constantes) qui satisfont la formule.

Cependant l'unification simple ne suffit pas. En effet, nous avons défini un ensemble de contraintes qui limitent les associations entre concepts. La programmation logique nous permet de déclarer des structures de données tandis que la programmation par satisfaction de contraintes (qui est une extension de la programmation logique) permet d'exprimer et de garantir que certaines contraintes soient vérifiées sur les données du modèle. Les contraintes que nous avons exprimées sont trop complexes pour être implémentées dans

des langages tels que Prolog (en particulier à cause des contraintes sur les cardinalités et de l'usage des quantificateurs universels). Cependant, il existe d'autres solutions qui nous permettent d'explorer l'espace problème et de trouver des solutions qui satisfont les contraintes. Nous verrons dans le chapitre X comment nous avons implémenté notre système à partir de telles solutions, notamment grâce aux systèmes experts.

La satisfaction de contraintes est un problème qui ne peut être résolu de façon générique en un temps raisonnable. Elle requiert donc l'emploi d'heuristiques qui guident la recherche de solutions. Nous considérons donc que nous disposons d'un modèle en logique du premier ordre sur lequel nous pouvons faire des requêtes simples (unification). Un autre module logiciel implémente les heuristiques qui nous permettent de résoudre le problème en organisant les requêtes et en traitant leur résultat dans un style impératif (cf. Figure 65). Les requêtes sont effectuées selon le vocabulaire défini par les modèles architectural et ergonomique.

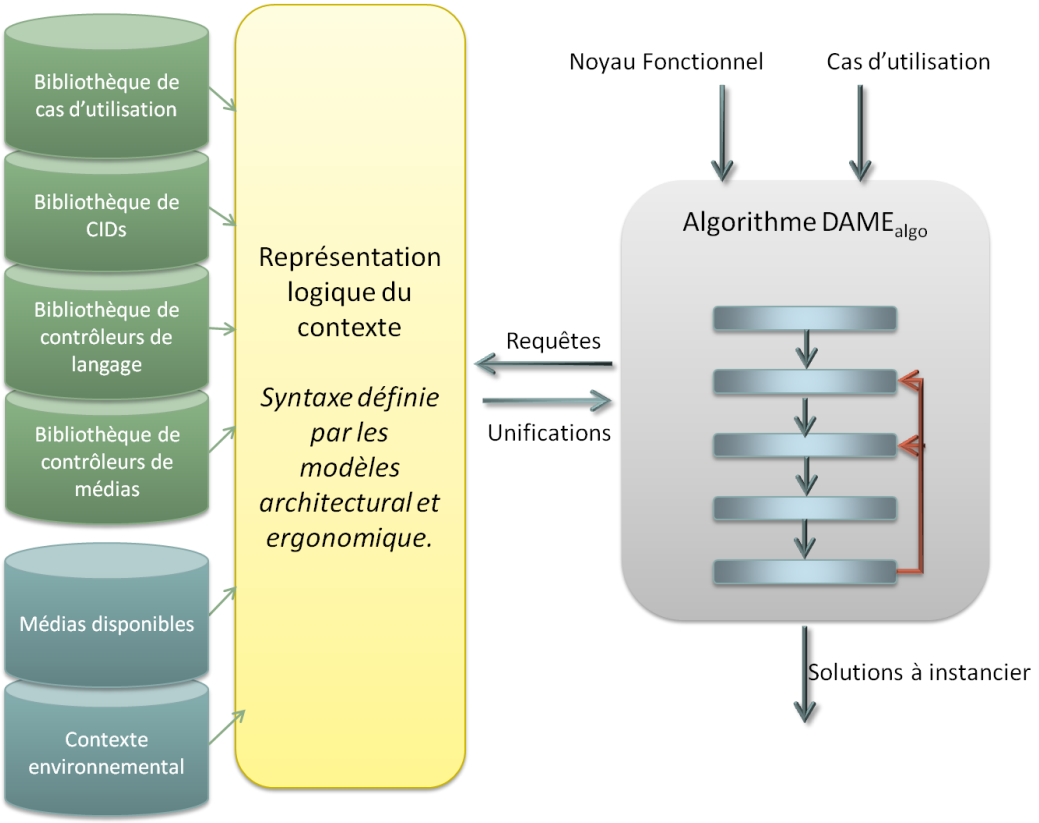


FIGURE 65 – ROLE DES MODELES ERGONOMIQUE ET ARCHITECTURAL DANS LA CONCEPTION AUTOMATIQUE D'UNE IHM

Nous fournissons dans la section suivante les descriptions de nos algorithmes qui mélangent la programmation impérative (en pseudo-code), la programmation logique ainsi que des opérations sur les ensembles. Nous définissons également quelques prédicats accessoires sur le modèle qui permettront de simplifier la syntaxe de nos algorithmes. Pour cela, nous définissons les opérations suivantes qui apparaitront dans le pseudo-code.

VIII.1.2.i UNIFICATION

Nous supposerons l'existence d'une fonction appelée unification qui est capable de trouver les substitutions valides (au regard des contraintes) des variables libres d'une formule.

Cette fonction prend en argument une formule qui peut contenir des variables libres et des variables liées aux données du programme. Elle prend en second argument, le nom de la variable libre dont on veut connaitre les substitutions valides. Elle renvoie une liste des substitutions valides.

requeteUnification(formule : String, variables : String) : Liste

Les variables liées aux données du programme dans les formules seront précédées par le symbole $. Par exemple, requeteUnification("Composant x ∧ requiert x, i ∧ herite(i, $z)", "x") renvoie un tableau contenant tous les composants du modèle dont l'une des interfaces requises hérite de l'interface « $z », « z » représentant une variable définie dans le programme. On notera que les unifications de la variable i sont ignorées dans cet exemple.

Nous limiterons l'usage de cette fonctionnalité à des requêtes simples.

VIII.1.2.ii Operations sur les ensembles

L'algorithme vise à créer des ensembles de candidats et de plans d'exécutions. Nous ferons donc appel au type Ensemble comme à un type de base possédant les opérations suivantes :

produitCardinal(e : Ensemble…) : Ensemble

Cette fonction prend en entrée un nombre variable d'arguments de type Ensemble et renvoie le produit cardinal de ces arguments sous la forme d'un nouvel ensemble.

union(e : Ensemble…) : Ensemble

Cette fonction prend en entrée un nombre variable d'arguments de type Ensemble et renvoie le produit cardinal de ces arguments sous la forme d'un nouvel ensemble.

ajout(e : Ensemble, x : Objet) : Ensemble

Cette fonction ajoute un élément x à un ensemble e et renvoie le nouvel ensemble.

aplanir(e : Ensemble) : Ensemble

Cette fonction reçoit en argument un ensemble dont les éléments sont eux même des ensembles. Elle renvoie l'union de ces ensembles fils. Par exemple, aplanir({{1,2}, {3,4}}) = {1,2,3,4}.

VIII.1.2.iii Operations sur les graphes

Notre algorithme va parcourir le graphe des cas d'utilisation. Pour se déplacer dans le graphe, nous utiliserons des requêtes d'unifications pour rechercher les fils d'un nœud. Le graphe sera par ailleurs étiqueté.

Pour étiqueter le graphe, nous utiliserons une structure de données complémentaire qui contiendra les étiquettes. Nous utiliserons pour cela une table de hachage dont le type sera noté TableHashage et dont les clés seront les cas d'utilisation et les valeurs seront des ensembles d'étiquettes.

Nous noterons [:] une table de hachage vide. L'accès à un élément d'une table nommée t se fait grâce à une clé selon la notation t[cle]. Cette notation est valable pour la lecture et l'écriture.

Par ailleurs nous définissons la fonction suivante :

fusion(t1 : TableHachage, t2 : TableHachage) : TableHachage

Cette fonction prend en argument deux tables de hachage dont les valeurs sont des ensembles d'étiquette et renvoie la table qui fusionne toutes les étiquettes. Pour chaque clé, cette table contient comme valeurs les unions des valeurs des tables t1 et t2.

VIII.2 LES ETAPES DE L'ALGORITHME

VIII.2.1 GENERATION DES CANDIDATS

La première étape de l'algorithme consiste à identifier un ensemble de CID qui pourrait convenir à la réalisation des cas d'utilisations requis en tenant compte du noyau fonctionnel disponible. L'ensemble des solutions possibles doit être énuméré pour qu'une sélection puisse ensuite avoir lieu, on ne se contentera donc pas de la première solution trouvée. Par ailleurs, on s'appuiera sur la structure récursive des cas d'utilisation pour envisager le cas de solutions sous-optimales. En effet, nous faisons l'hypothèse qu'un cas d'utilisation peut être réalisé si tous les cas qu'il inclut (selon la relation $include$) sont réalisés. Cette hypothèse permet d'apporter de la flexibilité dans la réalisation des cas d'utilisation. Chaque cas d'utilisation peut donc être décomposé en sous-cas, ce qui donne une structure d'arbre à la définition des cas d'utilisation (nous ignorons pour le moment la relation $extend$). En explorant cette structure, il est possible d'énumérer récursivement l'ensemble des « solutions valides ». Pour aborder ce problème plus rigoureusement, nous commençons par la définition de quelques concepts accessoires, nous proposons ensuite un algorithme de résolution.

VIII.2.1.i DEFINITIONS

On appelle **application** un couple $< nf, \mathcal{U} >$ où
- nf est un noyau fonctionnel
- $\mathcal{U}$ est un ensemble de cas d'utilisation. Lorsque cet ensemble est réduit à un unique élément, on notera $\{u\}$

On construit une interface pour l'application $< nf, \mathcal{U} >$ en créant des paires de CID et de cas d'utilisation associés qui couvre les besoins exprimés par l'ensemble $\mathcal{U}$ tout en étant instantiable dans le contexte représenté par nf.

De façon plus formelle, on appelle **interface de communication** tout ensemble de paires de CID et de cas d'utilisation $\mathcal{I} \subset \mathcal{E}_{CID} \times \mathcal{E}_{CU}$ bien formé, c'est-à-dire pour lequel :

$$\forall < u, c > \in I \;\; realiseCU(c, u)$$

On note $\mathcal{E}_{interfaces}$ l'ensemble des ensembles finis de paires du type $\mathcal{E}_{CID} \times \mathcal{E}_{CU}$ qui définissent une interface et on définit la **fonction suivante de composition des interfaces**, n étant le nombre d'interfaces à composer :

$$\phi_{\mathrm{n}} \quad \mathcal{E}_{interfaces}{}^{n} \to \mathcal{E}_{interfaces}$$
$$\mathbb{C}_0, \mathbb{C}_1, \dots, \mathbb{C}_{\mathrm{n}} \to \bigcup_{\mathrm{i}=0}^{\mathrm{n}} \mathbb{C}_{\mathrm{i}}$$

La composition de plusieurs interfaces est simplement l'union des paires qui composent chacune d'entre elles.

Les cas d'utilisation apparaissant dans $\mathcal{I}$ sont dits **couverts** par $\mathcal{I}$. La relation d'inclusion des cas d'utilisation créé une structure hierarchique en forme d'arbre. La définition de la couverture est donc récursive : un nœud d'un tel arbre est dit couvert par $\mathcal{I}$ ssi le cas d'utilisation associé est couvert par $\mathcal{I}$ ou si chacun de ses nœud fils est couvert par $\mathcal{I}$.

$couvert\ u, \mathcal{I}\quad ssi$
$\exists c\ tel\ que < u, c > \in I$ ***OU*** $\forall u' \in \mathcal{E}_{casUtilisation}\ include\ u, u' \Rightarrow couvert(u', \mathcal{I})$

Un interface $\mathcal{I}$ est **instanciable dans le contexte du noyau fonctionnel nf** ssi les interfaces requises par les CID qui ne sont pas du type $ILangageInteraction$ sont exposées par nf :

$instanciable\ \mathcal{I}, nf\quad ssi$
$\forall < u, c > \in I\ \forall i\ requiert\ c, i\ \wedge \neg inherits\ i, ILangageInteraction \Rightarrow expose\ nf, i$

On appelle **couverture interactionnelle de l'application** $< nf, \mathcal{U} >$ une interface $\mathcal{I}$ instanciable dans nf qui couvre tous les cas d'utilisation de $\mathcal{U}$.

$couverture\ \mathcal{I}, < nf, \mathcal{U} > \quad ssi\ instanciable\ \mathcal{I}, nf$ ***ET*** $\forall u \in \mathcal{U}\ couvert\ u, \mathcal{I}$

On remarque que l'ajout d'une paire quelconque de $\mathcal{E}_{CID} \times \mathcal{E}_{CU}$ à une couverture interactionnelle de l'application $< nf, \mathcal{U} >$ créé une interface qui est toujours une couverture interactionnelle de $< nf, \mathcal{U} >$. Nous éviterons donc d'énumérer des solutions inutiles en introduisant <u>la propriété de minimalité</u> suivante :

Une couverture interactionnelle d'une application $< nf, \mathcal{U} >$ sera dite **minimale** ssi le retrait d'une quelconque de ses paires la rend inapte à interfacer l'application $< nf, \mathcal{U} >$

$$\forall i \in \mathcal{I}\ \ \neg couverture\ \ \mathcal{I} \setminus i\ ,\quad < nf, \mathcal{U} >$$

Les couvertures interactionnelles minimales décrivent les interfaces « minimalistes » d'une application. Cependant, le graphe des cas d'utilisation inclut une relation qui permet d'enrichir les besoins satisfaits par des cas d'utilisation optionnels : la relation « extend ». Grâce à cette relation, on peut envisager des interfaces plus riches incorporant des CID qui ne seraient pas indispensables à l'interfaçage mais qui interfaceraient ces cas d'utilisation étendus. On appelle **couverture interactionnelle étendue** une couverture interactionnelle dont les paires vérifient l'une des deux conditions :

- La paire est indispensable à l'interfaçage de l'application
- La paire réalise un cas d'utilisation étendu de l'application

On appellera **candidat** à l'interfaçage de $< nf, \mathcal{U} >$ toute couverture interactionnelle étendue.

$candidat\ \mathcal{I}, < nf, \mathcal{U} >\ ssi$

$couverture\ \mathcal{I}, < nf, \mathcal{U} >$ ***ET***

$\forall i \in \mathcal{I}\ \neg couverture\ \ \mathcal{I} \setminus i\ ,\ \ \ \ < nf, \mathcal{U} >\ \vee (\exists u'\ extend\ u', u$
$\wedge realiseCU\ i, u'\ )$

VIII.2.1.ii ALGORITHME DE GENERATION DES CANDIDATS

La Figure 66 illustre les trois étapes de l'algorithme de génération des candidats.

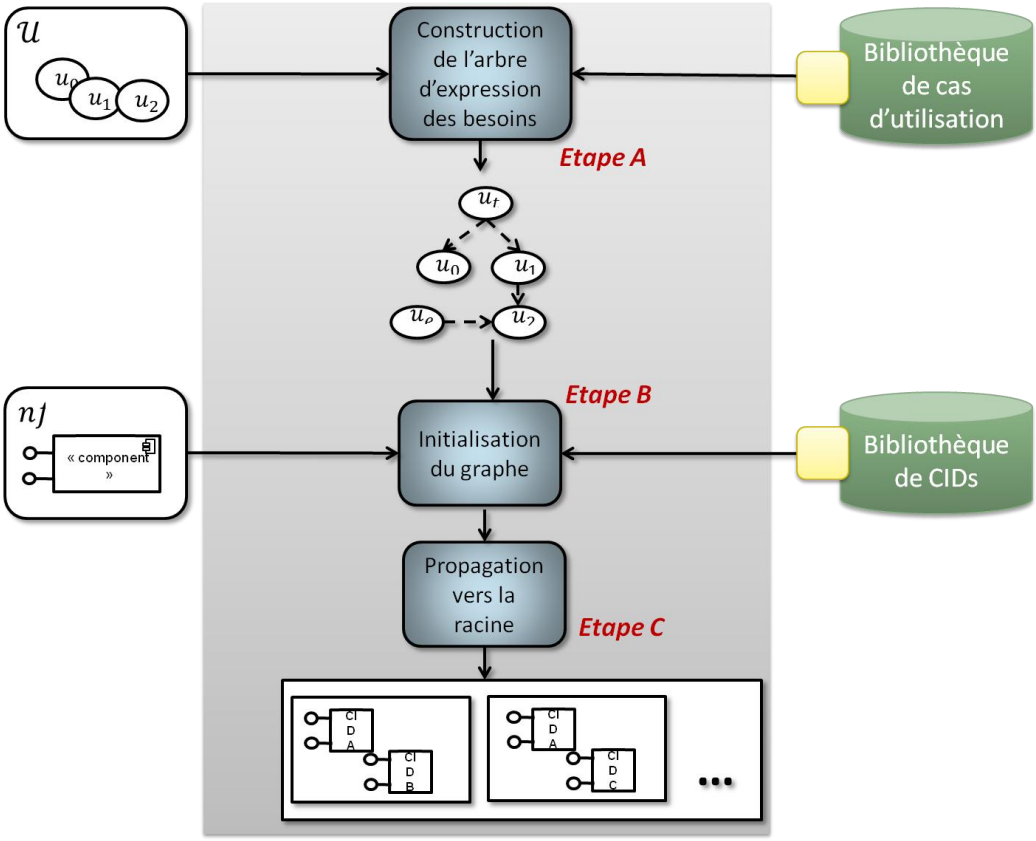


FIGURE 66 - ALGORITHME DE GENERATION DES CANDIDATS

La première étape (A) consiste à structurer les besoins de l'application $< nf, \mathcal{U} >$. Pour cela, on utilise les relations « include » et « extend » qui structurent le graphe des cas d'utilisation et on construit un graphe qui englobe tous les cas d'utilisations qui sont liés aux éléments de $\mathcal{U}$ par ces relations. La sémantique des relations « include » et « extend » garantit que le graphe obtenu sera acyclique, et par conséquent la construction de ce graphe s'obtient par simple énumération récursive.

On obtient ainsi un graphe dont les éléments de $\mathcal{U}$ sont les racines. Cependant, le calcul des candidats d'une application $< nf, \mathcal{U} >$ peut se réduire au calcul des candidats d'une application $< nf, \{u_t\} >$. Autrement dit, on peut toujours se ramener à un graphe possédant une unique racine. Il suffit pour cela de créer un cas d'utilisation temporaire appelé u_t auquel on rattachera tous les éléments de $\mathcal{U}$: $\forall u \in \mathcal{U}\ include\ u_t, u$. La récursivité des étapes (B) et (C) fera le reste. Nous utiliserons cette techniques de création

d'une racine « virtuelle » pour nous ramener à un graphe des besoins utilisateurs contenant tous les cas d'utilisation récursivement inclus par $\mathcal{U}$ que l'on pourra explorer récursivement à partir d'une racine unique.

La Figure 67 illustre l'obtention d'un tel graphe. On notera que le graphe ainsi obtenu est très proche d'un arbre enraciné (ou arborescence) dans la théorie des graphes. Il est en effet acyclique, orienté et dispose d'une unique racine. Cependant, certains nœuds peuvent avoir plusieurs parents (car certains cas d'utilisation sont inclus par plusieurs cas d'utilisation qui peuvent apparaitre dans $\mathcal{U}$), ce qui ne permet pas d'appeler ce graphe « arborescence ». Le fait que ce graphe soit orienté et acyclique nous permet cependant de le parcourir avec les méthodes usuelles de récursivité comme s'il était un arbre. Le fait que certains fils soient partagés a pour seule conséquence de factoriser le traitement réalisé dans les étapes (B) et (C) sur les nœuds du graphe.

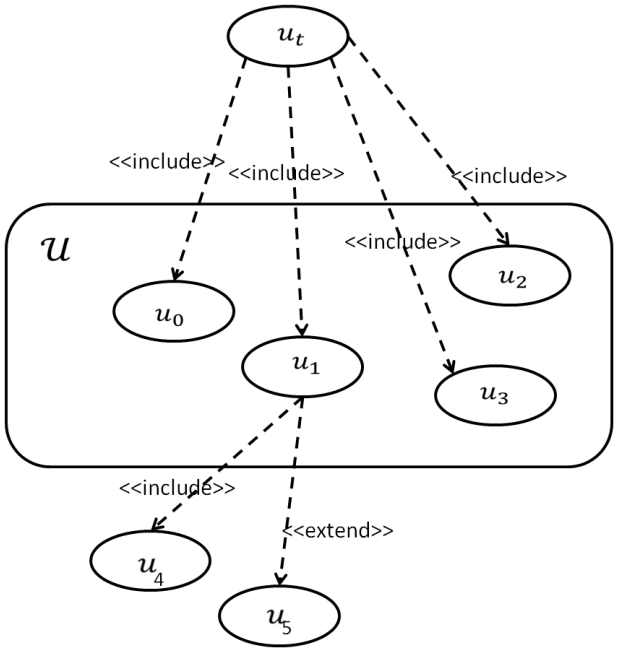


FIGURE 67 – STRUCTURE DES BESOINS D'UNE APPLICATION SOUS FORME DE GRAPHE ORIENTE, ACYCLIQUE ET A RACINE UNIQUE

La seconde étape consiste à trouver, dans le modèle, les CID qui pourraient permettre la réalisation des besoins. Pour cela, nous étiquetons les nœuds de l'arbre avec les interfaces candidates à la réalisation du besoin décrit par le nœud dans le noyau fonctionnel nf. Cette opération s'effectue en 2 étapes. Une étape d'initialisation (B) et une étape de propagation des solutions vers la racine (C).

L'étape d'initialisation consiste à parcourir le graphe pour étiqueter chaque nœud par la liste des CID qui couvrent le besoin tout en étant instanciables dans nf. Cette étape peut être réalisée par l'unification de la formule suivante dans laquelle u est une variable liée au nœud courant et nf au noyau fonctionnel de l'application cible $< nf, \mathcal{U} >$:

$$implementeCasUtilisation(c, u) \wedge instanciable \quad u \ , nf$$

Le Tableau 6 propose un algorithme réalisant l'initialisation de l'étiquetage.

Entrées

Un noyau fonctionnel nf
Un graphe des besoins $\mathcal{G}_\mathcal{U}$
La racine du graphe u_r
Une bibliothèque de CID disponibles

Sorties

Un graphe des besoins $\mathcal{G}_\mathcal{U}$ étiqueté par la liste des CID instanciables dans nf qui couvrent chaque nœud

Algorithme

```
fonction initialisation(G_U, u_r)
 //[:] est une table associative (table de hashage) vide
 retourne initialisationRec(G_U, u_r, [:])
fin fonction

fonction initialisationRec(G_U, u, etiquettes)
 //Les étiquettes ont déjà été initialisées par l'étape B

listeCID=requeteUnification("  implementeCasUtilisation(c, $u) ∧
instanciable  $u , $nf ", "x")
 pour c ∈ listeCID
  etiquettes[u] = ajout (etiquettes[u], <c,u>)
 fin pour

 //Enumère les nœuds fils
 fils = union( requeteUnification("x ∈ G_U  include($u, x)", "x") ,
        requeteUnification("x ∈ G_U  extend($u, x)", "x"))
 pour f ∈ fils
  etiquettes = fusion(etiquettes, initialisationRec(G_U, f))
 fin pour

 retourne etiquettes

fin fonction
```

TABLEAU 6 - ETAPE B : INITIALISATION DES ETIQUETTES

L'étape (C) de propagation des solutions vers la racine permet de construire récursivement la liste des candidats. Elle consiste en un parcours récursif du graphe qui calcule, pour chaque nœud, sa couverture interactionnelle étendue. Ce calcul s'effectue en deux étapes décrites en pseudo-code dans le Tableau 7 :

- Combiner les candidats partiels des **fils requis** (i.e. des nœuds liés par « include »)

  Un nœud peut être interfacé directement (par les CID énumérés à l'étape B) ou bien en interfaçant chacun des fils requis. Dans ce dernier cas, on construit un candidat partiel en associant un candidat partiel de chacun des fils requis. Ainsi, chaque élément du produit cartésien des candidats des fils requis décrit un candidat partiel pour le nœud courant.

Si $f_1, f_2, \ldots, f_n$ sont les fils requis du nœud u et $\mathbb{R}_1, \mathbb{R}_2, \ldots, \mathbb{R}_n$ sont leurs candidats respectifs. Alors on peut construire un ensemble de candidats partiels $\mathbb{R}$ pour u ainsi :

$$\mathbb{R} = \quad \phi_n \ \mathrm{x} \quad \text{pour } \mathrm{x} \in \prod_{i=1}^{n} \mathbb{C}_i$$

Où $\prod$ désigne ici l'opérateur n-aire de produit cartésien et ϕ_n la fonction de composition des interfaces définies dans la section précédente.

Si on note $\mathbb{I}$ l'ensemble des candidats triviaux identifiés par la phase d'initialisation des étiquettes (B), alors l'ensemble $\mathbb{I} \cup \mathbb{R}$ décrit les candidats primaires du nœud u.

- Combiner les candidats partiels des **fils optionnels** (i.e. des nœuds liés par « extend »)

  Les interfaces candidates des fils optionnels sont combinées avec les candidats de $\mathbb{I} \cup \mathbb{R}$ pour créer la couverture étendue. Les candidats de chacun des fils optionnels sont composés avec les candidats primaires du nœud courant grâce à l'opérateur suivant :

$$\Psi \begin{array}{c} \mathcal{E}_{interfaces} \times \mathcal{E}_{interfaces} \to \mathcal{E}_{interfaces} \\ < P, \mathbb{O} > \to \quad \phi_2 \ \mathrm{x} \quad \text{pour } \mathrm{x} \in \mathbb{P} \times \ \mathbb{O} \cup \ \emptyset \end{array}$$

  Si $f_1, f_2, \ldots, f_n$ sont les fils optionnels du nœud u et $\mathbb{O}_1, \mathbb{O}_2, \ldots, \mathbb{O}_n$ sont leurs candidats respectifs. Alors on peut construire l'ensemble des candidats pour u ainsi :

$$\mathbb{C} = \psi \ \ldots \psi \ \psi \ \mathbb{I} \cup \mathbb{R}, \mathbb{O}_1 \ , \mathbb{O}_2 \ \ldots, \mathbb{O}_n$$

  Autrement dit, chaque candidat primaire est associé à 0 ou 1 candidat du premier fils optionnel. Cela nous donne un nouvel ensemble de candidats et l'on réitère le processus avec le second fils optionnel et ainsi de suite. On obtient alors l'ensemble des couvertures interactionnelles étendues pour u.

A l'issu de cet algorithme, l'étiquette de la racine du graphe des besoins décrit la liste des candidats de l'application $< nf, \mathcal{U} >$. Le Tableau 7, page suivante, décrit en pseudo-code cet algorithme. La sous-section suivante (VIII.2.1.iii) donne un exemple d'application de l'algorithme.

Entrées

Un graphe des besoins $\mathcal{G}_{\mathcal{U}}$
Des étiquettes initiales associées aux nœud du graphe : *etiquettes*
La racine du graphe u_r

Sorties

L'ensemble des candidats à l'interfaçage de $< nf, \mathcal{U} >$:
$\{ \mathcal{I} \in \mathcal{P}(\mathcal{E}_{CID} \times \mathcal{E}_{casUtilisation})$ tels que $candidat\ \mathcal{I}, < nf, \mathcal{U} > \}$

Algorithme

```
fonction listerCandidats(G_U, u_r, etiquettes)
 candidats = listerCandidatsPartiels(G_U, u_r, etiquettes)
 retourne candidats[u_r]
fin fonction

fonction listerCandidatsPartiels(G_U, u, etiquettes)
 //On récupère 𝕀 (initialisé par l'étape B)
 cidsDirects = etiquette[u]
 //Enumère les nœuds fils
 filsRequis = requeteUnification("x ∈ G_U include($u, x)", "x")
 filsOptionnels = requeteUnification("x ∈ G_U extend($u, x)", "x")

 /////Lister les candidats partiels des fils
 pour f ∈ union(filsRequis, filsOptionnels)
  etiquettes = fusion(etiquettes, listerCandidatsPartiels(G_U, f, etiquettes))
 fin pour

 ///// Combiner les candidats partiels des fils requis
 couvReduite = {}
 //Fait le produit cardinal progressivement
 pour f ∈ filsRequis
  sousCandidatsPartiels = etiquette[f]
  //On applique la fonction de composition ϕ2 sur le produit cardinal intermédiaire
  couvReduite = aplanir(produitCardinal(couvReduite, sousCandidatsPartiels))
 fin pour
 //Candidats primaires = 𝕀 ∪ ℝ
 primaires = union(cidsDirects, couvReduite)

 ///// Combiner les candidats partiels des fils optionnels
 couvEtendue = primaires
 pour f ∈ filsOptionnels
  candidatsOptionnels = union(etiquette[f], ∅)
  //On applique la fonction de composition ϕ2 sur le produit cardinal
  couvEtendue = aplanir(produitCardinal(couvEtendue, candidatsOptionnels))
 fin pour

 //Candidats primaires = 𝕀 ∪ ℝ
 etiquettes[u] = couvEtendue
 retourne etiquettes

fin fonction
```

TABLEAU 7 – ETAPE C : PROPAGATION DES « CONTRAINTES » DANS LE GRAPHE POUR ENUMERER LES CANDIDATS VALIDES

VIII.2.1.iii APPLICATION SUR LE CAS DU CENTRE MULTIMEDIA

Nous appliquons l'algorithme de génération des candidats à l'exemple qui a été notre fil conducteur tout au long de cette étude : le centre multimédia.

L'application à interfacer est décrite par le noyau fonctionnel précédemment présenté (Figure 57, page 149) ainsi que par le cas d'utilisation « manipuler des fichiers multimédia » de la Figure 50, page 139. Dans la Figure 68, nous reprenons ce diagramme de cas d'utilisation et nous affichons l'étiquetage qui est calculé par notre algorithme. L'étiquetage est représenté par un cadre listant les t-uples de CID candidats. L'étiquetage associe normalement, à chaque CID, le CU qui lui a permis d'être sélectionné. Pour ne pas alourdir le schéma, nous ne représentons pas les candidats par les couples <CID,CU> mais simplement par une liste de CID.

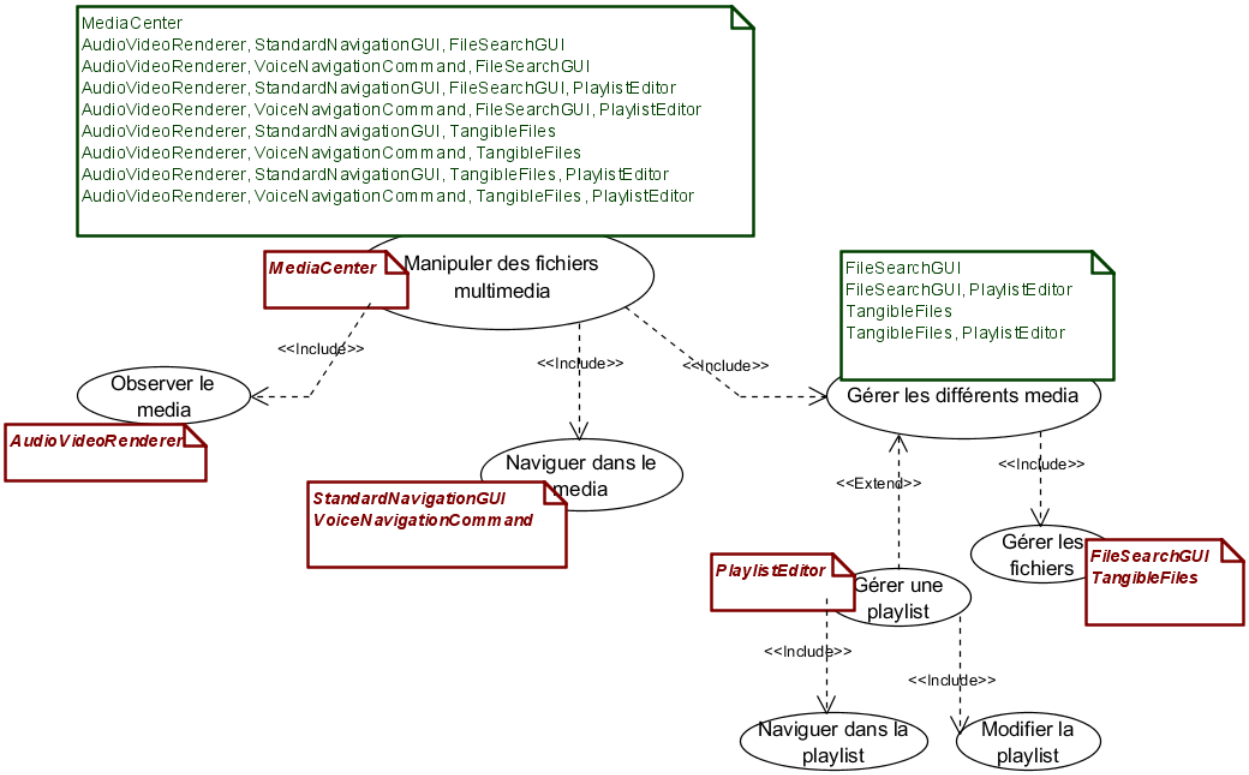


FIGURE 68 - EXEMPLE DU CENTRE MULTIMEDIA: ARBRE D'EXPRESSION DES BESOINS ET CANDIDATS SELECTIONNES

On imagine que les concepteurs ont implémenté plusieurs CID et que nous disposons donc d'un environnement riche. Les cadres écrits en gras italique (en rouge) représentent les CID qui sont disponibles et compatibles avec le noyau fonctionnel. Ils sont représentés à côté du CU qu'ils réalisent. Cet étiquetage initial correspond à la phase (B) de notre algorithme. Par exemple, le CID TangibleFiles est une interface tangible permettant de représenter des fichiers à l'aide de jetons du monde réel. Le même besoin de gestion des fichiers est rempli par une autre interface (graphique cette fois) appelée FileSearchGUI. De même, le CID MediaCenter est une interface graphique qui couvre tous les besoins d'un média center. Elle est plus complexe et plus complète et se suffit à elle-même. On observe sur cette figure que notre approche permet une réalisation des besoins de l'utilisateur à différents niveaux de granularité.

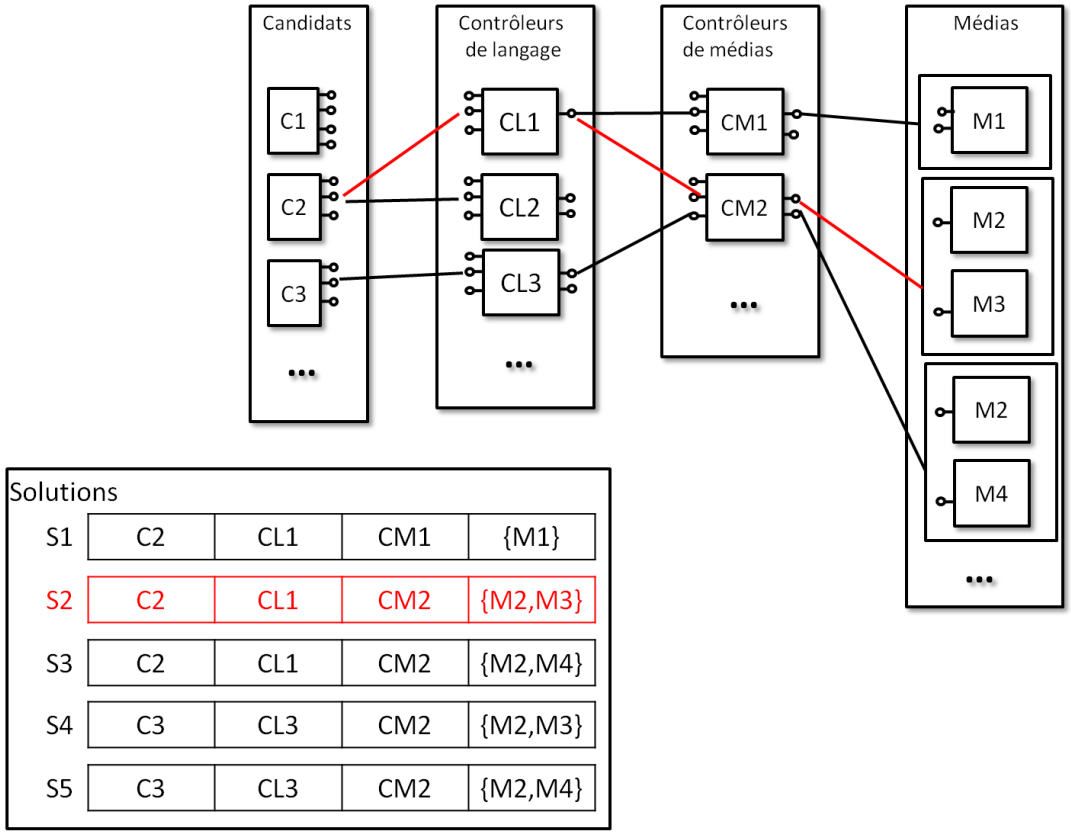
La phase (C) consiste à combiner ces étiquettes de façon à produire à la racine une liste des candidats. Cette opération se fait récursivement. En premier lieu sur le cas d'utilisation « gérer les différents médias » :

> La couverture interactionnelle minimale est déduite du seul CU requis (« Gérer les fichiers »). Elle permet de générer 2 candidats pour ce CU, chacun composé d'un unique CID. La couverture interactionnelle étendue est construite en partant de la couverture interactionnelle minimale et en ajoutant les éléments du produit cardinal de ces 2 candidats avec les candidats du CU optionnel « gérer une playlist ».

La même opération est réalisée sur le CU racine. On obtient alors l'ensemble des 9 candidats associant divers CID qui reposent sur des langages d'interaction parfois différents. Tous les candidats ne sont pas forcément instanciables et certains peuvent être instanciés de plusieurs façons. Nous présentons, dans la prochaine section, un algorithme permettant d'énumérer toutes les chaines d'interaction adaptées aux candidats générés.

VIII.2.2 CONCEPTION DES CHAINES D'INTERACTION

Une fois que la liste des candidats est connue, la phase de conception des chaînes d'interaction consiste à étudier, pour chaque candidat, comment il peut être instancié dans le système. C'est-à-dire par l'intermédiaire de quels composants et en s'appuyant sur quels périphériques d'interaction.



Solutions

S1	C2	CL1	CM1	{M1}
S2	C2	CL1	CM2	{M2,M3}
S3	C2	CL1	CM2	{M2,M4}
S4	C3	CL3	CM2	{M2,M3}
S5	C3	CL3	CM2	{M2,M4}

FIGURE 69 - ENUMERATION DES SOLUTIONS POSSIBLES DANS UN GRAPHE D'INSTANCIATION

Le modèle architectural a été conçu dans l'optique de simplifier cette étape. Nous avons identifié 3 familles de composants intermédiaires permettant d'instancier une interface : le contrôleur de langage, le contrôleur de médias et les médias eux-mêmes. Nous partons du principe que ces composants sont fournis par les concepteurs et que le système n'a pas besoin de les concevoir automatiquement[28]. On suppose de plus que la chaine interactive ne fait intervenir qu'un seul contrôleur de langage et un seul contrôleur de média. Ces contraintes définies dans le modèle architectural permettent de simplifier la recherche d'une chaine interactive en l'assimilant à la recherche d'un chemin connexe dans un graphe.

En effet, pour identifier ces chaines d'interaction, on construit un graphe des connexions possibles entre les candidats et les composants des 3 familles suscitées. Nous appellerons ce graphe le **graphe d'instanciation**. La Figure 70 et la Figure 71 illustrent les différentes étapes nécessaires au calcul de ce graphe. Une fois ces trois étapes réalisées, on obtient un graphe similaire à celui de la Figure 69. A partir de ce graphe, on peut énumérer les solutions en énumérant tous les chemins connexes liant un candidat à un ensemble de médias.

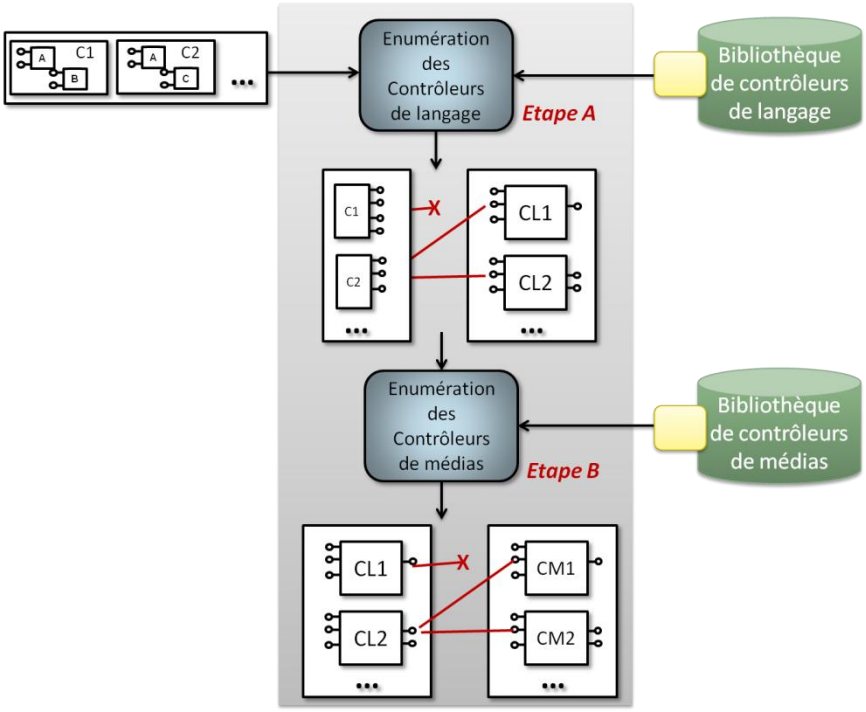


FIGURE 70 – ALGORITHME D'IDENTIFICATION DES CHAINES D'INTERACTION (ETAPES A ET B)

[28] La composition automatique de contrôleurs de langage ou de contrôleur de média est un domaine de recherche très interessant mais dont la complexité justifierait un travail de thèse à lui seul. Nous écartons donc cette question en première approche et laissons reposer sur les concepteurs la charge de définir ces entités. Cependant, on remarquera que le mécanisme de composition permet déjà une flexibilité à ce niveau par la réutilisation de sous-composants communs.

La conception du graphe de la Figure 69 se fait progressivement : chacune des 3 couches est construite en faisant appel à des requêtes d'unification. Pour les étapes A et B, ces requêtes décrivent trivialement les contraintes à remplir.

On utilisera ici le prédicat $heriteT$, cloture transitive du prédicat $herite$, qui permet de définir qu'une interface d'un composant dérive d'une autre interface.

Etape A :

Un contrôleur de langage peut être connecté à un candidat si et seulement s'il implémente toutes les interfaces requises par les CID du candidat qui sont de type $ILangageInteraction$.

$$ControleurLangage\ cl\ \wedge Candidat\ c$$
$$\wedge\ \forall < u, cid > \in c$$
$$requiert\ cid, i\ \wedge heriteT\ i, ILangageInteraction\ \wedge expose\ cl, i'\ \wedge heriteT(i', i)$$

Etape B :

Un contrôleur de média peut être connecté à un contrôleur de langage si et seulement s'il implémente toutes les interfaces requises par ce dernier qui sont de type IStructureInteraction.

$$ControleurMedia\ cm\ \wedge ControleurLangage\ cl$$
$$\wedge\ \forall i\ requiert\ cl, i\ \wedge heriteT\ i, IStructureInteraction\ \wedge expose\ cm, i'$$
$$\wedge heriteT(i', i)$$

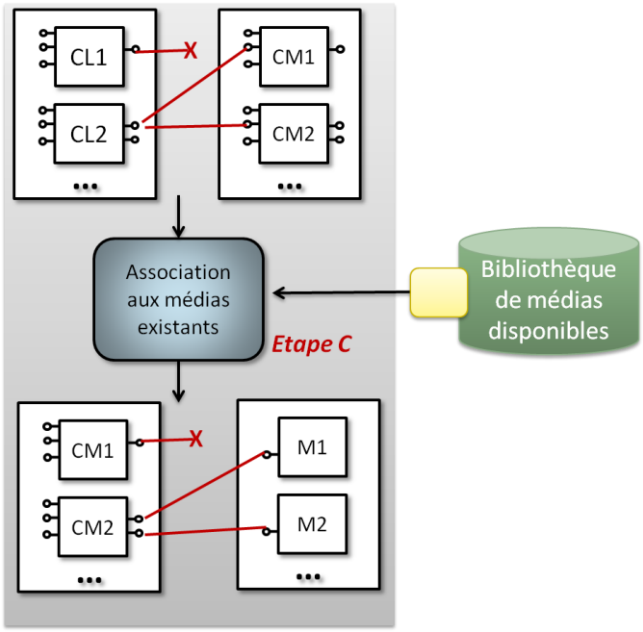


FIGURE 71 - ALGORITHME D'IDENTIFICATION DES CHAINES D'INTERACTION (ETAPE C)

L'étape C est un peu plus complexe car elle ne peut être réalisée en utilisant une unique requête d'unification. En effet, un même contrôleur de médias doit être connecté non pas à

un média mais à un ensemble de médias. Il faut donc construire ces ensembles selon la démarche suivante.

Etape C :

Soit cm un controleur de médias et $I_k \ _{0 \leq k \leq n}$ ses interfaces requises de type $IMedia$.
Pour chaque interface I_k, on définit l'ensemble des médias $\mathbb{M}_k$ qui peuvent l'instancier. Cet ensemble est calculé par unification sur la requête suivante :

$$Media\ m\ \wedge expose\ m, i'\ \wedge heriteT(i', i)$$

L'ensemble des t-uples de médias qui permettent d'instancier cm est alors le produit cardinal des ensembles $\mathbb{M}_k$:

$$fils\ cm\ = \prod_{k=0}^{n} \mathbb{M}_k$$

Avec $\prod$ le symbole du produit cardinal n-aire et $fils$ la relation décrivant les fils du composant cm dans l'arbre d'instanciation.

Cet algorithme fournit des solutions qui risquent de ne pas être ergonomiquement viables bien que logiquement acceptables. Par exemple, si un contrôleur de média requiert une interface « écran » et une interface « dalle tactile », on s'attend à ce que ces interfaces soient fournies par le même média. Il faudrait donc ignorer toutes les solutions qui feraient intervenir une composition de plusieurs médias pour ce contrôleur de médias. Ce type de considération est impossible à prendre en compte à ce niveau car elle dépend de connaissances ergonomiques qui n'entrent pas encore en jeu dans le système. On verra cependant dans la prochaine étape de l'algorithme que l'on peut définir un modèle comportemental permettant d'éliminer ce type de situations invalides.

VIII.2.3 EVALUATION DES INTERFACES CONÇUES

Le modèle comportemental est défini conjointement par :

- Le concepteur d'ambiant
- L'ergonome d'ambiant
- L'administrateur d'ambiant

Il s'agit d'un ensemble de métriques et de règles qui permettent d'évaluer la pertinence des solutions précédemment énumérées vis-à-vis du contexte. Ce modèle a pour objectif d'éliminer les solutions incohérentes et de choisir parmi celles qui restent, la solution la plus adaptée.

Pour ce faire, on introduit la notion de critère. Ces critères sont les facteurs que l'on souhaite évaluer. Ils regroupent un ensemble de recommandations qui ont un sens commun. **Les critères sont définis par les acteurs qui rédigent le modèle comportemental**. Ce dernier est modulaire et les critères sont les modules qui permettent de personnaliser le comportement du système.

VIII.2.3.i DEFINITIONS

Cette étape prend en entrée un ensemble de **solutions** structurellement valides $< candidat, chaine\ d'interaction >$ et applique à chacune d'elle un modèle

comportemental qui permet d'évaluer son adéquation au contexte. Le **modèle comportemental** est l'ensemble des lois gouvernant le mécanisme de sélection des solutions. Il mélange des considérations provenant de diverses catégories que nous appelons des **critères**. Un critère est un couple $< poids, metrique >$.

L'application de la métrique d'un critère du modèle comportemental aboutit en une évaluation qui est constituée d'un couple $< intention, note >$ tel que :

- $intention \in \mathcal{E}_{Intention} = Utiliser, Neutre, Interdire$
- $note \in \mathbb{R}$

La Figure 72 illustre l'application d'un ensemble de critères sur 3 des solutions obtenues par l'étape précédente de l'algorithme. On remarque que l'application de ces critères repose sur l'utilisation de la modélisation logique du contexte. Il s'agit ici de l'ensemble des informations disponibles à propos du contexte, et non seulement un sous-ensemble de bibliothèques tel que dans les précédents schémas. En particulier, ce modèle ne se limite pas au modèle architectural mais inclut également les concepts du modèle ergonomique ainsi que toute autre information que le concepteur d'ambiant aurait pu juger utile d'ajouter.

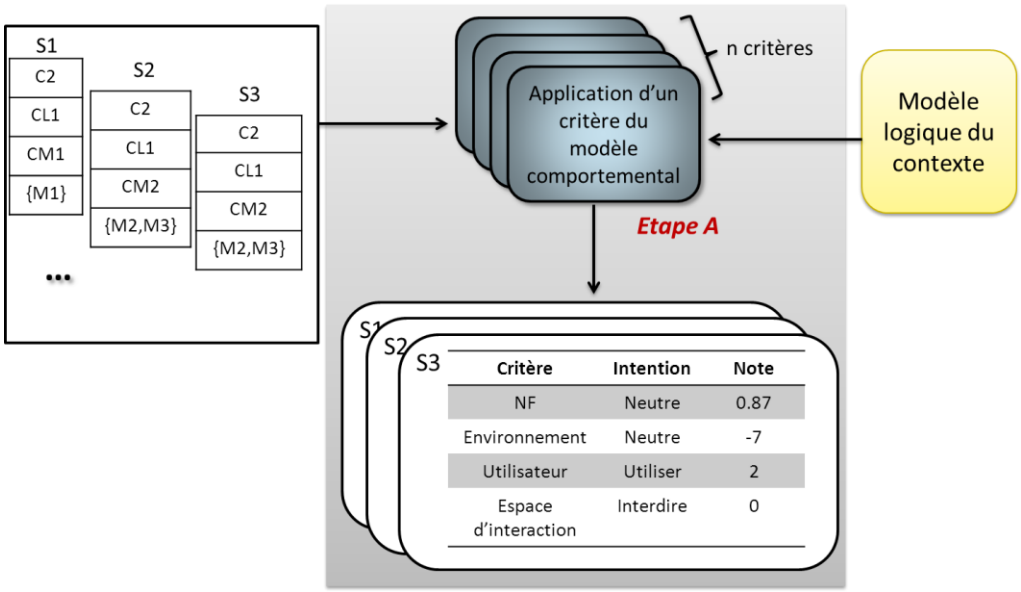


FIGURE 72 – APPLICATION DES CRITERES D'EVALUATION DES CANDIDATS

La note permet d'évaluer l'adéquation de la solution au contexte. Cette note permet de classer les solutions pour chaque critère et de déterminer la « meilleure » solution. A cette note est associée une intention qui indique un ordre du concepteur indépendamment de la note.

Cette intention permet d'interdire l'usage d'une solution, quelle que soit sa note par ailleurs. On peut par exemple souhaiter interdire l'usage du mode auditif pour des utilisateurs sourds. Une simple règle dévaluant les solutions reposant sur le mode auditif ne suffirait pas dans ce cas là puisque d'autres règles concernant d'autres aspects du contexte pourraient « relever » la note. A l'inverse, l'intention « Utiliser » est fréquemment employé

pour tester/évaluer une solution qui ne serait pas sélectionnée en temps normal (situations de tests, débuggage ou encore évaluation).

En résumé, l'intention permet de forcer l'usage de solutions non-optimales ou d'interdire l'usage de solutions optimales. Dans certains cas critiques, l'emploi d'une note (très mauvaise ou très bonne) ne permet pas de garantir que la solution sera choisie ou évincée de la sélection. L'intention est donc une coupure, un ordre absolu qui permet de gérer ces situations et se comporte comme les symboles $+\infty$ et $-\infty$ par rapport à la note.

La mise en place de métriques (on parle également d'heuristiques d'évaluation) est propre au critère considéré. Il incombe donc aux concepteurs et ergonomes d'ambiant de fournir les métriques adaptées aux critères qu'ils définissent. Ces métriques peuvent reposer sur toute famille d'algorithme issu de l'ingénierie des connaissances et de l'intelligence artificielle (systèmes experts, systèmes de classification non supervisée, réseaux de neurones…). Nous avons identifié deux familles de métriques:

- Les **métriques électives** : Il s'agit d'une évaluation basée sur un système d'élection similaire à celui de WWHT. Ce système d'élection est très proche des systèmes experts, il permet d'écrire des règles d'élection en utilisant la logique du premier ordre et est donc naturellement adapté à la structure de notre modèle.
- Les **métriques non-électives** : Il s'agit de calculs arbitraires réalisés sur le modèle. Un exemple de métrique non élective est fourni dans l'annexe B.

Les deux familles de métriques peuvent coexister dans le modèle comportemental. Chaque critère fournit une évaluation. Une fois les critères appliqués, les évaluations $< intention, note >$ obtenues sont fusionnées pour obtenir une évaluation globale de la solution au regard du contexte, ce qui permettra la sélection des solutions.

Le prochain paragraphe décrit comment les critères sont fusionnés pour obtenir une évaluation générale des solutions. Le paragraphe suivant décrira 3 critères électifs que nous proposons pour évaluer les solutions.

VIII.2.3.ii Fusion des notes et resolution des conflits d'intentions

L'application du modèle comportemental permet d'obtenir, pour chaque candidat, un ensemble de critères évalués sous la forme de couples $< intention, valeur >$.

La sélection des solutions qui seront instanciées repose sur une étape de prétraitement qui consiste à fusionner ces différents critères pour obtenir une évaluation générale (étape C de la Figure 74). Cette évaluation générale établit une relation d'ordre absolue sur l'ensemble des solutions, ce qui permettra leur sélection dans la prochaine phase de l'algorithme.

Nous distinguons la fusion des intentions de celle des notes (qui requiert une harmonisation préalable, étape B de la Figure 73).

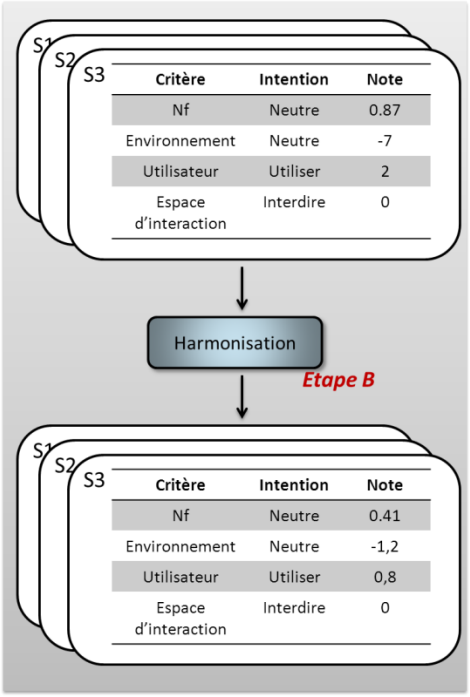


FIGURE 73 – HARMONISATION DES NOTES DES CRITERES

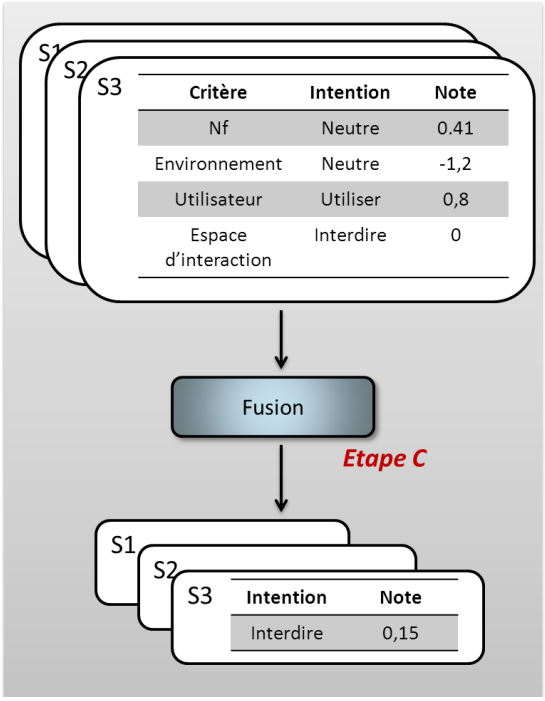


FIGURE 74 – FUSION DES INTENTIONS ET DES NOTES

Fusion des intentions :

Les intentions selon différents critères peuvent être contradictoires. Il est possible, en particulier, que l'intention selon un critère soit d'interdire l'instanciation tandis que selon un autre critère, elle soit de forcer l'instanciation. Pour résoudre de tels conflits, nous dotons l'ensemble $\mathcal{E}_{Intention}$ d'une loi de composition notée T.

($\mathcal{E}_{Intention}$,T) est une monade (au sens mathématique du terme) qui a les propriétés suivantes :

- $Neutre$ est un élément neutre pour T
- $Interdire$ est un élément absorbant pour T

L'opérateur de fusion des intentions est décrit par le Tableau 8. L'intention $Interdire$ indique que le concepteur a détecté une situation qui est conflictuelle et qui rend l'interaction impossible dans le contexte en utilisant cette solution. L'intention $Utiliser$ permet de forcer l'usage d'une solution même si celle-ci obtient une moins bonne note globale. Cette intention permet de forcer l'usage de solutions qui sont sous-optimales sur le plan ergonomique mais qui présentent quand même un intérêt dans le contexte donné. Par exemple, on peut se servir de cette intention pour instancier une solution sous-optimale sur le mode auditif en redondance de la solution principale sur le mode graphique.

La sémantique de ces intentions justifie que l'intention d'interdire soit absorbante et prédomine sur celle d'utiliser.

⊤	***Utiliser***	***Neutre***	***Interdire***
Utiliser	*Utiliser*	*Utiliser*	*Interdire*
Neutre	*Utiliser*	*Neutre*	*Interdire*
Interdire	*Interdire*	*Interdire*	*Interdire*

TABLEAU 8 – MONADE $(\mathcal{E}_{Intention}, \top)$

Fusion des valeurs :

Une approche naïve de fusion des valeurs consisterait à effectuer une moyenne des notes obtenues. Cependant l'évaluation des critères est libre : la moyenne, l'étendue et l'écart des notes peuvent donc grandement varier d'un critère à l'autre. Il serait donc « injuste » d'opérer à une simple moyenne, quand bien même pondérée. La fusion des valeurs s'effectue donc en 2 étapes : un prétraitement des données correspondant à une ***harmonisation des notes*** portées par chaque critère puis une moyenne pondérée de ces critères.

Le problème d'harmonisation des notes est bien connu des membres de jurys de passage ou d'admission. Il s'agit de corriger les biais statistiques produits par la subjectivité des correcteurs lors de l'évaluation des copies. Nous nous contenterons d'une méthode simple permettant de corriger la moyenne et l'écart-type, solution peu couteuse en temps de calcul (transformation affine) et suffisante pour notre problème.

Pour chaque série de notes correspondant à un critère donnée, nous transformerons la série pour lui faire atteindre une moyenne de 0 et un écart-type de 1 (on dit qu'elle est centrée et réduite). Cette transformation peut être obtenue par simple transformation affine des éléments de la série. En effet :

Soit c un critère et $n_0^c, n_1^c, \dots n_k^c$ les notes des k solutions obtenues lors des deux premières étapes de l'algorithme. Notons M l'estimation de l'espérance de la série et V l'estimateur de sa variance[29].

$$M\left(n_i^c\right)_{0\leq i\leq k} = \frac{1}{k}\sum_{i=0}^{k} n_i^c$$

$$V\left(n_i^c\right)_{0\leq i\leq k} = \frac{1}{k-1}\sum_{i=0}^{k}\left(n_i^c - M\left(n_j^c\right)_{0\leq j\leq k}\right)^2$$

Par ailleurs, si l'on note f une application affine $f_{x\to ax+b}^{\mathbb{R}\to\mathbb{R}}$, $m_c = M\left(n_i^c\right)_{0\leq i\leq k}$ et ${\sigma_c}^2 = V\left(n_i^c\right)_{0\leq i\leq k}$

[29] Nous prenons ici les estimateurs les plus couramment employés lors d'analyse de séries statistiques. On utilise en particulier l'opérateur S_{n-1} pour la variance qui permet de supprimer le biais lié à l'incertitude sur l'espérance mathématique (contrairement à l'opérateur S_n qui serait biaisé dans ce cas).

$$M\ \ f(n_i^c)_{\ 0\le i\le k}\ = am_c + b$$

$$V\ \ f(n_i^c)_{\ 0\le i\le k}\ = a^2{\sigma_c}^2$$

On en déduit que pour obtenir une espérance de 0 et une variance de 1, il faut appliquer la transformation affine f à chacune des notes avec pour valeurs de a et b les suivantes :

$$a = 1/\sigma_c$$
$$b = -m_c/\sigma_c$$

Ce qui donne la suite de notes harmonisées suivante :

$$\frac{n_0^c - m_c}{\sigma_c}, \frac{n_1^c - m_c}{\sigma_c}, \dots, \frac{n_k^c - m_c}{\sigma_c}$$

Une fois que les notes de chaque critère sont harmonisées, on peut les fusionner en effectuant la moyenne des notes obtenues. Bien sûr, certains critères sont plus importants que d'autres et l'administrateur d'ambiant peut même décider de changer ces poids d'une exécution à l'autre pour évaluer l'impact de chacun des critères, il nous semble donc indispensable de pouvoir pondérer l'impact de chaque critère dans la moyenne. C'est pour cette raison que la définition d'un critère ne se résume pas à sa métrique mais inclut un poids : $< poids, metrique >$

Pour un critère c donné, notons p_c le poids qui lui est associé et $n_c(s)$ la valeur de la note attribuée à s par la métrique du critère. Nous notons également m_c et σ_c la moyenne et l'écart-type de ce critère sur l'ensemble des solutions (tels que définis ci-dessus) et p la somme des poids de tous les critères :

$$p = \sum_{c\in\mathcal{E}_{Critere}} p_c$$

La note finale d'une solution S sera donc la moyenne pondérée des notes harmonisées :

$$\frac{1}{p}\sum_{c\in\mathcal{E}_{Critere}} p_c\ \frac{n_c\ s\ - m_c}{\sigma_c}$$

La Figure 74 illustre la structure des résultats obtenus après application et fusion des critères du modèle comportemental. Dans la sous-section suivante, nous présentons quelques critères que nous estimons intéressants. La section suivante éclairera le mécanisme de sélection que permet l'application du modèle comportemental.

VIII.2.3.iii CRITERES D'EVALUATION ET HEURISTIQUES PROPOSEES

Nous proposons la disjonction de critères suivante, qui peut être adaptée en fonction des besoins des concepteurs et ergonomes :

- Adaptation au contexte environnemental
- Adaptation à l'espace d'interaction logique
- Adaptation à l'utilisateur

Avant d'aborder plus en détail chacun de ces critères, nous présentons une méthode de représentation des critères sous la forme d'ensembles de règles : les critères électifs.

METRIQUES ELECTIVES : EVALUATION PAR DES REGLES

Le mécanisme d'élection permet aux concepteurs et ergonomes d'Ambiant d'écrire des règles reposant sur le vocabulaire décrit dans les modèles architectural et ergonomique pour évaluer la solution vis-à-vis d'un critère bien spécifique dans un contexte donné.

Sur le plan formel, une règle est traditionnellement décrite par une prémisse et une conclusion. La prémisse de la règle est la formule permettant de reconnaitre certains motifs dans la base de connaissance. Lorsque cette formule peut être unifiée, la conclusion s'applique pour chaque unification. Dans un système expert standard, la conclusion d'une règle consiste généralement en l'ajout de nouveaux faits à la base de connaissance. Dans un système électif, la conclusion d'une règle ne modifie pas la base de faits mais uniquement un ensemble de poids utilisés par le processus d'élection (cf. Figure 75).

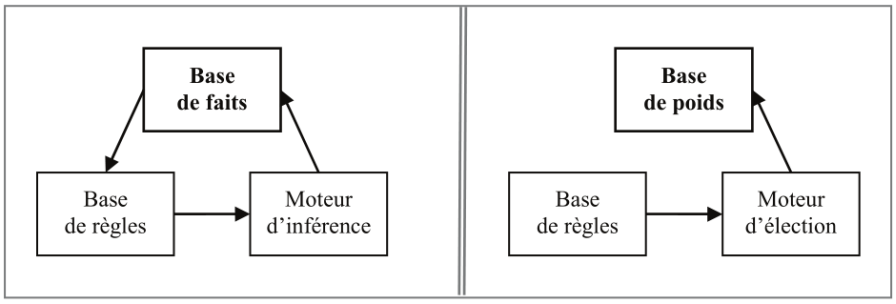


FIGURE 75 – DIFFERENCES ENTRE UN SYSTEME EXPERT ET UN SYSTEME ELECTIF

Dans notre cas, les prémisses des règles doivent porter sur les solutions que l'on souhaite évaluer. On distingue alors 4 conclusions possibles qui prennent en argument une solution :

- `interdire(s)`
  Applique l'intention « interdire » à la solution s pour le critère considéré
- `utiliser(s)`
  Applique l'intention « utiliser » à la solution s pour le critère considéré
- `favoriser(s, p)`
  Modifie la note de la solution s d'un poids +p pour le critère considéré
- `defavoriser(s,p)`
  Modifie la note de la solution s d'un poids –p pour le critère considéré

Par défaut, la solution à l'évaluation <Neutre, 0>. On utilise ensuite l'opérateur de composition T défini dans le Tableau 8 pour combiner les intentions dans le cas où les règles aboutiraient à des ordres contradictoires. La note s'obtient quant à elle en faisant la somme des contributions favorisant ou défavorisant la solution.

Le choix des poids dans un système de règles pose toujours un problème de maintenance. En effet, au fur et à mesure que la base de règles grandit, la gestion des poids (qui correspond à la priorité des règles) devient plus difficile car le nombre de facteurs à prendre en compte augmente et la valeur relative de chacune des règles devient plus floue.

Il n'existe aucune méthodologie pour aider à choisir des poids cohérents : la valeur des poids est donc soumise à l'arbitraire du concepteur. Cependant, cet arbitraire est compensé par la phase d'harmonisation des notes décrites dans la section précédente. De plus, la catégorisation des règles en différents critères qui sont chacun évalués indépendamment les uns des autres permet de séparer les préoccupations et de limiter le nombre de règles appliquées simultanément.

Ce système d'élection à base de règle est à la base des critères électifs, une famille de critères qui permet au concepteur et à l'ergonome d'ambiant de simplement décrire les situations qu'ils souhaitent favoriser ou éviter. En décrivant ces situations en logique du premier ordre, ils peuvent participer à l'élaboration du modèle comportemental sans avoir aucune connaissance relative à l'implémentation du système. Nous illustrons l'intérêt d'un tel système par les 3 critères suivants.

CRITERE D'ADEQUATION A L'ESPACE D'INTERACTION LOGIQUE

Nous définissons l'espace d'interaction logique (EIL) comme l'ensemble des composants logiciels qui sont déjà instanciés par le système. Il s'agit des solutions qui ont été précédemment instanciées. Cet EIL sera vide si aucune autre interaction n'a lieu dans la sphère d'activité courante, mais il peut également être dans un état plus complexe dont nous tiendrons compte lors de la sélection de la solution la plus appropriée.

On peut par exemple préférer utiliser les modes déjà instanciés pour d'autres interfaces de la même sphère d'activité pour instancier le nouveau cas d'utilisation. Pour cela, on écrirait :

$$sphere\ sph\ \wedge solutionInstanciee\ sph, s\ \wedge solution\ s' \\ \wedge\ \exists m\, utiliseModalite\ s, m\ \wedge utiliseModalite\ s', m\ => favoriser(s', 1)$$

On note que pour des interfaces multimodales, la même règle peut s'appliquer plusieurs fois et ainsi renforcer encore la préférence pour des interfaces partageant plusieurs modalités.

CRITERE DU CONTEXTE ENVIRONNEMENTAL

Nous appelons contexte environnemental toute valeur d'état qui décrit physiquement le contexte et qui peut ou non avoir un impact direct sur l'interaction. Il s'agit des critères tels que la température, la pression atmosphérique, le niveau sonore ambiant…

Un exemple d'usage serait d'interdire l'utilisation de la modalité « langage naturel » si le niveau sonore ambiant dépasse un certain seuil (30db). Une telle règle s'écrirait ainsi :

$$zone\ z\ \wedge utilisateur\ u\ \wedge dansZone\ u, z\ \wedge niveauSonore\ z, ns\ \wedge\ ns \geq 30 \\ \wedge solution\ s\ \wedge utiliseModalite(s, \text{"langage naturel"}) => interdire(s)$$

CRITERE UTILISATEUR

Le modèle ergonomique contient le concept d'utilisateur avec des liens vers la théorie des modalités. Cependant, ce concept est voué à être étendu au cas par cas, en fonction des besoins, pour apporter des informations complémentaires sur l'utilisateur (préférences,

capacités…). Le critère utilisateur vise à exploiter ces informations pour fournir une interface qui soit adaptée aux goûts et capacités de l'utilisateur.

Un exemple d'usage serait d'interdire l'usage d'un mode pour lequel l'utilisateur présente un handicap (par exemple le mode auditif pour une personne sourde). Une telle règle s'écrirait ainsi :

$$utilisateur\ u\ \wedge handicap\ u,m\ \wedge solution\ s\ \wedge utiliseMode\ s,m\ => interdire(s)$$

VIII.2.4 SELECTION DES INTERFACES OPTIMALES

Une fois le modèle comportemental appliqué et les évaluations fusionnées, on obtient pour chaque solution, un couple $< intention, note >$. Ces évaluations fournissent une relation d'ordre absolue sur l'ensemble des solutions. Il est donc possible de classer les solutions tel que le montre l'exemple du Tableau 9, en donnant la priorité à l'intention sur la note et en se servant de celle-ci pour classer les solutions d'intention égale.

Identifiant solution	Intention	Note
S6	**Utiliser**	**1,2**
S5	**Utiliser**	**0,72**
S2	**Neutre**	**1,3**
S7	Neutre	0,52
S4	Neutre	-0,12
S1	Neutre	-1,2
S3	Interdire	-0,1

TABLEAU 9 – EXEMPLE DE TRI DES SOLUTIONS

On appelle solution optimale la solution ayant obtenu la meilleur note sans être interdite (dans le Tableau 9, il s'agit de la solution S2). Les solutions qui seront sélectionnées pour instanciation sont :

- La solution optimale
- Les solutions dont l'intention est « utiliser » (ce qui peut inclure la solution optimale)

Dans notre exemple, les 3 premières lignes du tableau seraient ainsi sélectionnées pour instanciation. L'étape d'instanciation nécessite cependant une vérification préalable de la compatibilité des solutions sélectionnées. En effet, si chaque solutions est instanciable (par construction), certaines associations de solutions peuvent rentrer en conflit si elles cherchent à s'approprier la même ressource. Autrement dit, si deux solutions utilisent des contrôleurs de médias différents mais qu'au moins un des médias devrait être partagé par les deux contrôleurs de médias, alors l'instanciation échouera. Pour résoudre ce conflit, on décidera de supprimer la solution dont la note est la plus faible. Notre approche de résolution des conflits est simple : elle consiste à interdire l'usage d'un même média par plusieurs contrôleurs de médias. Cependant des variations plus complexes pourraient être envisagées à partir des travaux de (Barralon, Lachenal & Coutaz 2004).

Après avoir sélectionné les solutions à instancier, le système d'interaction ambiante réalise l'instanciation en connectant les composants sélectionnés et en mettant à jour la description de l'espace d'interaction logique (EIL) en conséquence. L'utilisateur peut alors interagir avec le noyau fonctionnel pour réaliser son activité. En fonction des évolutions du contexte cependant, l'interface ainsi instanciée peut être remise en cause.

VIII.2.5 INSTANCIATION ET EVOLUTION DES INTERFACES

Au cours du temps, le système évolue et les évaluations et choix faits par le modèle comportemental peuvent être remis en cause. Surveiller l'évolution des valeurs d'état du contexte est une tâche ardue. Si le concept de contexteur mis en avant par (Dey 2000) permet de modéliser ces informations changeantes et de mettre à jour le modèle en conséquences, il n'est pas évident de remettre en cause l'interface. Deux facteurs principaux s'opposent à une réévaluation systématique :

- Le coût computationnel

Les changements du contexte peuvent être donnés à des fréquences élevées (en fonction des capteurs). Or, la combinatoire du problème de composition est telle que même si nous l'avons simplifiée, il parait irréaliste de relancer l'algorithme à chaque modification du contexte. Par ailleurs, la plupart des changements du contexte n'auront pas d'impact sur les solutions possibles.

- Le coût ergonomique

Le remplacement d'une interface d'une application par une autre interface est appelée migration. Une migration trop fréquente de l'interface pourrait gêner l'utilisateur. De la même façon, certaines migrations qui changeraient complètement les modes de communication (du visuel à l'oral par exemple) pourraient être perturbantes pour l'utilisateur. On pourrait vouloir privilégier des migrations intra-mode ou bien reposant sur certaines modalités.

Pour adresser ces problèmes, plusieurs politiques peuvent être envisagées. Nous listons quelques-unes de ces politiques dans le Tableau 10.

	Facteur coûteux pour la remise en cause	Politique permettant de limiter l'impact de cette remise en cause
A	Fréquence de changement du contexte.	Filtrage des valeurs avec une limite temporelle ou un seuil de variation minimal.
B	La plupart des changements n'auront pas d'impact sur les solutions.	Identification des contexteurs dont les solutions instanciées dépendent.
C	Certains changements impliquent de recalculer l'ensemble des solutions tandis que d'autres ne nécessitent qu'une réévaluation des critères.	Choix de l'étape à remettre en cause en fonction de la nature du contexteur (par exemple, s'il appartient au modèle ergonomique, on n'a pas besoin de remettre en cause la structure des solutions).
D	Migration trop fréquente pour l'utilisateur.	On ne change pas l'interface dès qu'elle n'est plus optimale : on définit des seuils permettant de conserver une solution sous-optimale.
E	Migration perturbante pour l'utilisateur.	Pour éviter de perturber l'utilisateur, il faudrait un feedback qui avertisse de la migration. De plus, il faut veiller à ce que l'historique des états de l'espace logique d'interaction soit cohérent.

TABLEAU 10 – FACTEURS COUTEUX POUR L'EVOLUTION DES INTERFACES ET POLITIQUES ASSOCIEES

La politique A est simple à mettre en œuvre, elle permet d'éviter une fréquence trop élevé de remise en cause des solutions instanciées. La politique B nécessite qu'à chaque étape de l'algorithme, on identifie les faits du modèle logique sur lesquels reposent la structure ou l'évaluation de la solution. En surveillant ces faits uniquement, on peut ne remettre en cause la solution que lorsque l'on est sûr que ce calcul aboutira à un résultat différent. Pour ce qui est de la génération des candidats et des chaines d'interaction associées, cette identification est simple : les seuls faits ayant de l'importance et qui peuvent évoluer dans le temps sont les médias disponibles. Pour ce qui est de la phase d'évaluation cependant, les choses sont plus complexes. Il est tout à fait faisable de tracer les règles des critères électifs qui ont été appliquées et d'en déduire les faits à surveiller. Pour ce qui est des autres familles de critères (topologiques ou autres), cette identification ne peut être généralisée et ne peut être fournie qu'au cas par cas par celui qui les implémente. La politique C présente les mêmes limites que la politique B.

Les politiques D et E sont des politiques « après réévaluation » qui permettent de déterminer si le gain introduit par un changement d'interface est supérieur à son coût ergonomique pour l'utilisateur. La mise en place d'une telle politique est subtile et doit être adaptée à l'utilisateur. Les modèles de DAME fournissent les structures permettant ce type de raisonnement. En effet, la notion d'espace logique d'interaction permet d'identifier l'état des autres périphériques du système (sont-ils déjà utilisés ? par quel utilisateur ? pour quelle activité ?) et de conserver une trace de ceux-ci dans le temps. Cela permet lors de la phase d'évaluation, d'écrire des règles pour le critère « adéquation à l'EIL » décrit précédemment.

VIII.3 CONCLUSION

Nous avons défini dans cette section un algorithme reposant sur les modèles architectural et ergonomique permettant d'énumérer toutes les solutions possibles pour interfacer un noyau fonctionnel en fonction d'un ensemble de besoins utilisateurs.

Cet algorithme permet ensuite d'évaluer chacune de ces solutions pour estimer quelle est la meilleure. Nous avons défini pour cela la notion de critères qui sont des modules du modèle comportemental conçus par les différents acteurs d'un système ambiant pour établir des recommandations quant au comportement du système. Ces recommandations peuvent prendre une forme libre, mais nous proposons un cadre pour la rédaction de critères sous la forme d'un ensemble de règles attribuant des notes aux solutions. Les évaluations obtenues par les critères sont ensuite harmonisées et fusionnées pour obtenir une note finale. Cette note finale permet d'identifier les solutions optimales.

Enfin, nous avons abordé les politiques de sélection et d'évolution dans le temps des interfaces, qui permettent de maintenir une adaptation de l'interaction homme-machine tout au long de l'activité de l'utilisateur dans l'environnement ambiant.

Dans le chapitre suivant, nous proposons une analyse des contributions de la démarche DAME illustrée par quelques exemples.

Chapitre IX
EXEMPLES D'APPLICATIONS DE L'APPROCHE DAME

L'approche que nous avons proposée dans les chapitres précédents a été développée dans le but de couvrir les besoins spécifiques à l'interaction dans l'Ambiant identifiés dans la section V.2. Nous avons pour cela développé une méthode de conception modulaire dans laquelle chaque composant est associé *a posteriori* à une modélisation détaillée permettant de guider la composition automatique d'une IHM. Cette composition repose sur une description des caractéristiques structurelles (dans le modèle architectural) et ergonomique (dans le modèle du même nom) utilisée pour écrire des recommandations (dans le modèle comportemental).

La compatibilité de notre approche avec les IHM conventionnelles a été illustrée dans les chapitres précédents par l'exemple du centre multimédia, fil conducteur lors de la description des modèles. Cet exemple illustre l'efficacité de DAME pour une factorisation du code des IHM conventionnelles. Cependant, il n'illustre pas comment nos modèles s'appliquent dans des situations plus spécifiques à l'Ambiant.

Nous illustrons donc dans cette section l'adéquation des modèles de DAME à la description de situations courantes en IHM ambiante. Nous avons pour cela sélectionné deux situations que nous estimons représentatives de l'interaction dans l'Ambiant :

- la reconfiguration multi-médias qui permet de remplacer des médias par une association d'autres médias
- le développement de techniques d'interaction innovantes qui remettent en cause les architectures traditionnelles d'IHM (par exemple, l'interaction tangible).

Dans les 2 premières sections de ce chapitre, nous proposons des exemples illustrant ces 2 situations et nous modélisons ces exemples selon l'approche DAME. Nous concluons ensuite ce chapitre par une analyse de la satisfaction des besoins exprimés dans la section V.2.

IX.1 RECONFIGURATIONS MULTI-MEDIAS ET REDISTRIBUTION

Les approches multi-médias telles que celles décrites en IV.2 proposent un modèle de composition des périphériques basé sur un flux de donnés ou d'évènements. Ces diagrammes de flux permettent d'associer plusieurs médias pour créer un nouveau composant plus complexe ou pour remplacer un média manquant. Il est par exemple possible d'instancier une interface graphique prévue pour fonctionner sur une plateforme disposant d'un clavier et d'une souris sur une tablette tactile. Ce mécanisme, appelé redistribution (Coutaz 2007), permet une certaine adaptation au contexte tout en dégradant parfois la qualité de l'interaction.

Nous étudions dans cette section dans quelle mesure nos modèles permettent d'appliquer des mécanismes similaires de redistribution des périphériques d'interaction.

Reprenons l'exemple d'une interface graphique traditionnelle telle que celle du centre multimédia pour laquelle nous aurions élaboré la chaine interactive de la Figure 76, c'est-à-dire que l'on instancie cette interface graphique sur la composition suivante de médias : Tablette tactile + clavier. Cette figure a déjà été analysée page 157. La Figure 76 illustre à la fois la chaine d'interaction issue du modèle d'architecture et également la description ergonomique que l'on peut en tirer à partir du modèle ergonomique.

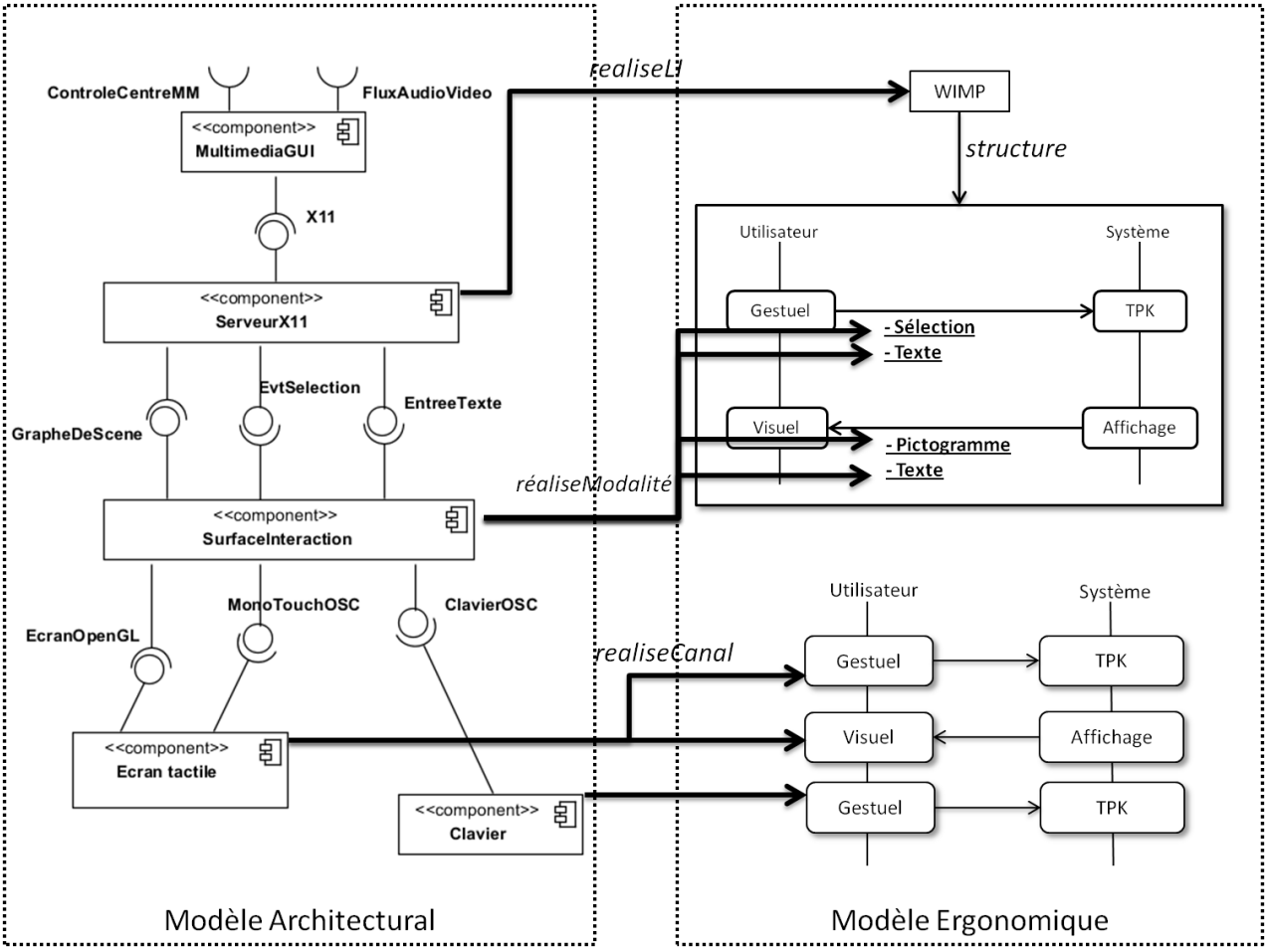


FIGURE 76 – CHAINE D'INTERACTION ET CONNAISSANCES ERGONOMIQUES DEDUITES DANS L'EXEMPLE ECRAN TACTILE+CLAVIER

En considérant que nous sommes dans un univers riche en périphériques d'interaction (i.e. en médias), nous pouvons imaginer plusieurs scénarios d'instanciation du même CID sur d'autres médias :

1. L'écran tactile pourrait être composé d'un ensemble d'écrans tactiles couplés dans l'espace (cf. techniques de couplage, section III.2.4).
2. Le clavier n'est pas nécessaire et pourrait être simulé par une interface graphique adéquate.

3. Le rendu de l'interface pourrait être effectué sur une tablette braille pour un utilisateur aveugle (c'est une fonction qui existe déjà au travers de logiciels appelés « lecteur d'écran »).

Analysons, pour chacun de ces scénarios, l'impact qu'il a sur la constitution de la chaine d'interaction.

IX.1.1 SCENARIO 1 : COUPLAGE DE MEDIAS

Le couplage de médias peut être obtenu par la création d'un composant de type Média qui exposera l'interface du nouveau média.

Ainsi, dans l'exemple précédent, l'écran tactile peut être remplacé par un écran composite qui réalisera la redirection des primitives graphiques (évènements de sortie) et des d'évènements d'entrée sur deux écrans tactiles couplés. Le coupleur d'écran tactile est un composant offrant des primitives graphiques et tactiles sur une surface virtuelle (cf. Figure 77). Cette surface couvre l'ensemble des deux surfaces indépendantes réalisées par les écrans physiques. Cette technique de redistribution des évènements est présentée dans (Barralon, Lachenal & Coutaz 2004).

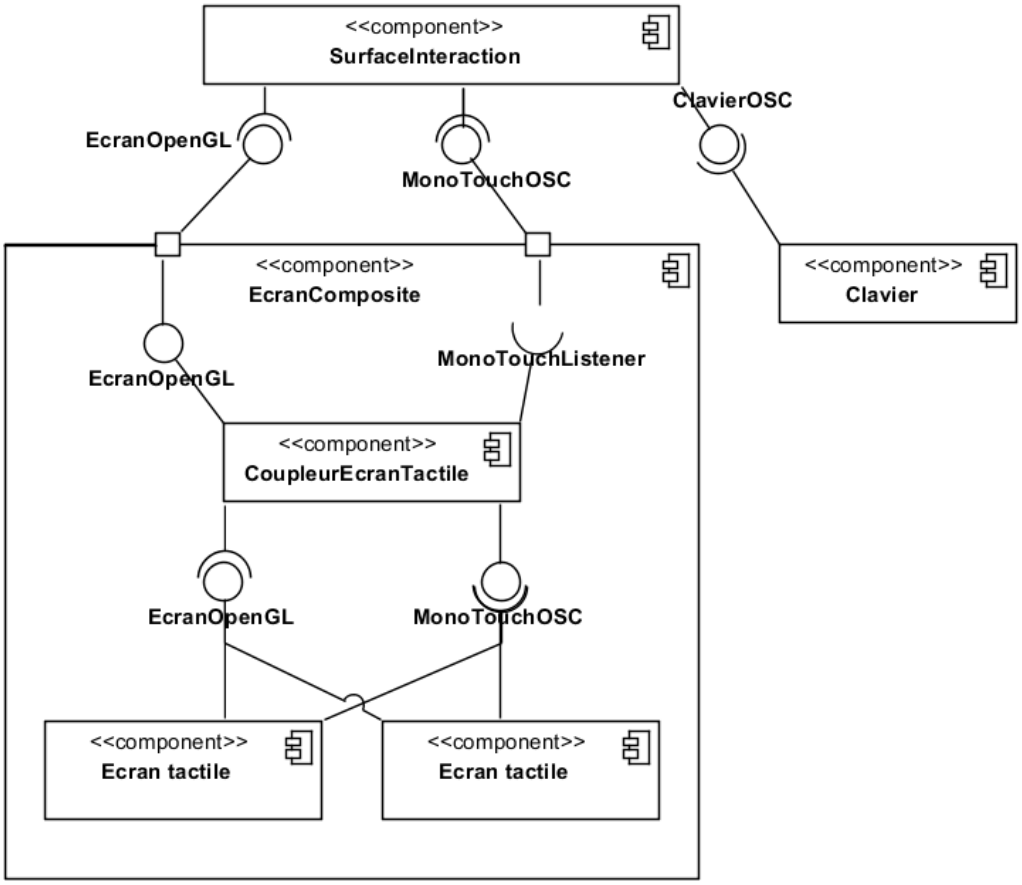


FIGURE 77 – COUPLAGE DE DEUX ECRANS TACTILES

Son implémentation requiert une expertise technique et notamment, une bonne compréhension et interprétation des évènements de bas niveau proposés par les périphériques. Il incombe donc aux concepteurs de produire ces composants au cas par cas.

De plus, l'algorithme présenté dans le chapitre précédent n'autorise pas d'instancier automatiquement de telles solutions. Lors de la phase de recherche des différentes chaines interactives, il serait possible d'inclure, en plus des médias immédiatement disponibles, tout couplage de média autorisé par les composants disponibles dans la librairie. Cependant, le couplage est une technique d'interaction qui requiert la participation active de l'utilisateur. Nous n'avons dons pas implémenté de recherche automatique des couplages possibles. Les couplages peuvent être activés par le système, soit en configurant certains composants de couplage pour qu'ils s'activent automatiquement au démarrage du système (dans ce cas le couplage est permanent), soit en fournissant des techniques d'interaction permettant à l'utilisateur de manipuler de façon consciente l'activation ou la désactivation de ces composants.

IX.1.2 Scenario 2 : Remplacement d'un media manquant

Le contrôleur de média transforme les messages reçus des médias en structures de données qui permettent l'instanciation des modalités. Cependant, même lorsqu'un média est absent, une modalité peut parfois être accessible via l'utilisation de structures intermédiaires. La Figure 78 illustre ce mécanisme d'adaptation par le remplacement du contrôleur de média basique par un contrôleur de média plus évolué.

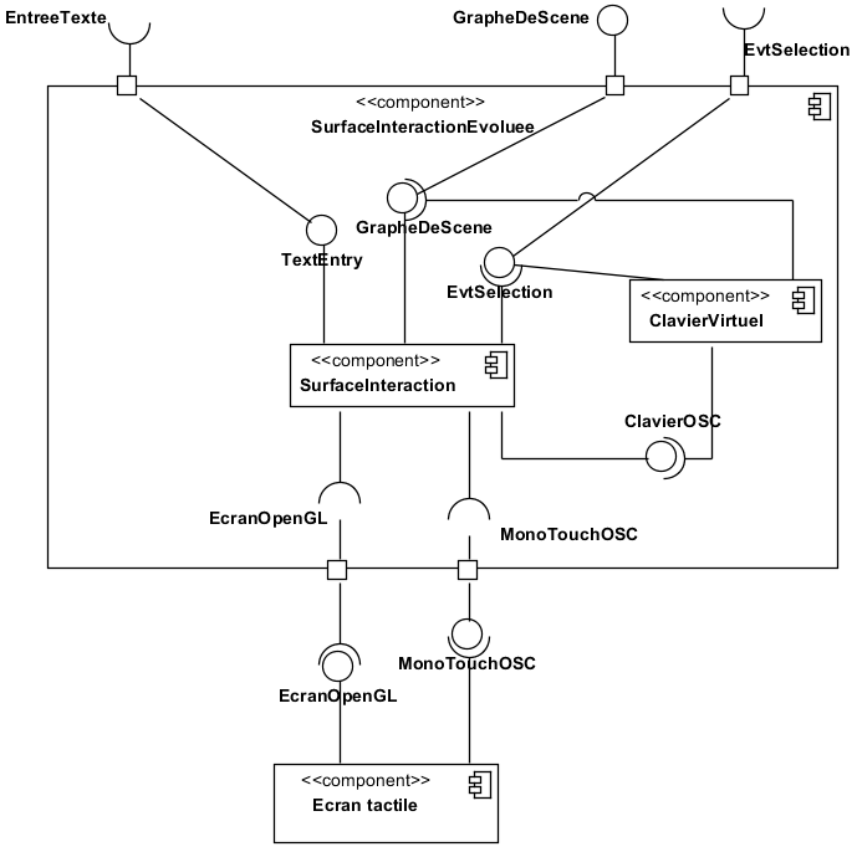


Figure 78 – Remplacement d'un clavier par un clavier virtuel

Les IHM conventionnelles nous fournissent un exemple de ce mécanisme d'adaptation au travers des claviers virtuels. Sur certaines plateformes conventionnelles (en particulier les tablettes tactiles et les *tablets PC*), l'absence de clavier physique peut être compensée par l'apparition, à l'écran, d'un clavier virtuel qui permet de réaliser la modalité d'entrée de

texte. Dans la Figure 78, le contrôleur de média SurfaceInteraction est remplacé par un contrôleur de média SurfaceInteractionEvoluee. Ce dernier réutilise le contrôleur de média basique et l'agrémente de nouvelles fonctionnalités en agissant directement sur ses entrées et ses sorties pour produire un clavier virtuel. Il peut alors se passer de requérir l'interface ClavierOSC.

Lors de l'application de l'algorithme de composition de l'interface, les deux chaines d'interaction seront générées par le système. Elles seront départagées en fonction des critères qui sont fournies par l'ergonome d'Ambiant. Ainsi, l'ergonome peut attribuer un handicap à la solution utilisant un clavier virtuel, estimant que cela est moins confortable pour l'utilisateur. Il peut également attribuer un handicap à la solution utilisant le clavier si ce dernier ne se trouve pas à proximité de l'écran tactile. Le compromis entre ces deux handicaps, exprimés dans les critères du modèle comportemental guidera le choix de l'une ou l'autre des solutions.

IX.1.3 SCENARIO 3 : REMPLACEMENT D'UN MEDIA EXISTANT

Le dernier scénario, enfin, envisage l'adaptation d'un certain langage d'interaction à un jeu de média et de modalités non prévues à cet effet. Cette technique est couramment mise en pratique pour faciliter l'accès de personnes handicapées à une interface classique. Pour compenser leur handicap, l'usage de nouvelles modalités est nécessaire et des modules sont chargés de faire la traduction d'une modalité à une autre. Par exemple, il existe des lecteurs d'écrans permettant d'adapter une interface graphique à la lecture sur des terminaux braille pour les non-voyants. Cette transformation de la modalité graphique et textuelle à la modalité braille se fait au détriment de la qualité de l'interaction (surtout lors de la navigation entre les contrôles graphiques) mais cette dégradation de l'interface est justifiée par le profil utilisateur.

Dans l'architecture DAME, les modalités sont associées au contrôleur de médias. C'est lui qui interprète les données manipulées par le contrôleur de langage d'interaction en un rendu (ou un échange de messages) avec les médias. Le remplacement de ces modalités peut donc se faire grâce à l'implémentation de contrôleurs de médias spécifiques aux nouvelles modalités et aux nouveaux médias que l'on souhaite interfacer avec le langage d'interaction.

La Figure 79 illustre un échange de contrôleur de médias qui permet de réaliser un lecteur d'écran. Ce dernier adapte la présentation graphique issue de l'interface X11 (et donc dans le langage WIMP) pour les périphériques clavierBraille et Haut-Parleur. De même que dans l'exemple précédent, à condition que les périphériques requis soient disponibles, l'application du modèle comportemental génèrera à la fois la solution de la Figure 76 et celle de la Figure 79. Cependant, en analysant le modèle ergonomique de la Figure 79 (partie droite de la figure), on peut déduire qu'il y a une incompatibilité entre le langage d'interaction WIMP et le rendu sur un clavier braille. Le modèle architectural permet donc de réaliser des constructions qui sont valides sur le plan logiciel alors qu'elles présentent apparemment des contradictions sur le plan ergonomique.

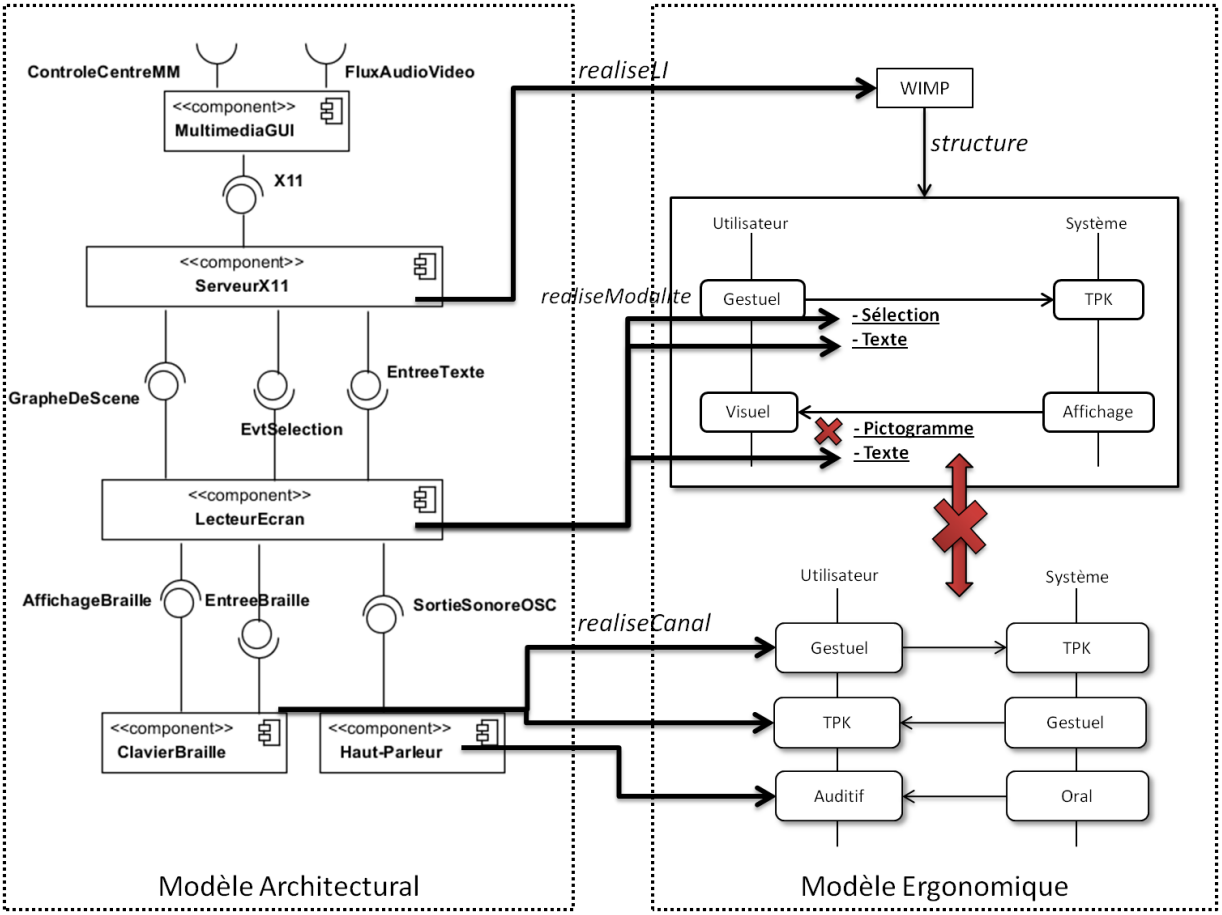


FIGURE 79 – REALISATION D'UN LECTEUR D'ECRAN POUR PERSONNES MALVOYANTES

Ce sont les règles du modèle comportemental qui permettront de trancher entre ces deux solutions. En fonction des règles écrites par l'ergonome, cette chaine d'interaction pourrait donc se voir attribuer une mauvaise évaluation dans le cas d'un utilisateur voyant. Cependant, dans le cas d'un utilisateur non-voyant, il serait judicieux d'ajouter une règle interdisant l'usage du mode visuel. Par inférence, cette règle aurait alors pour conséquence d'invalider la solution classique au profit de la solution de la Figure 79, bien que celle-ci dégrade la qualité de l'interaction.

IX.1.4 CONCLUSION SUR LA REDISTRIBUTION

Au travers des exemples ci-dessus, nous avons étudié les capacités de redistribution dans l'architecture DAME. Cette redistribution est réalisée par la satisfaction des contraintes issues du modèle architectural (i.e. les contraintes sur les interfaces des composants). Lors de la génération des chaines d'interaction possibles, les liens établis avec le modèle ergonomique permettent d'évaluer les solutions sur le plan ergonomique et de rejeter les solutions qui sont viables sur le plan logiciel mais qui ne respectent pas les recommandations encodées dans le modèle comportemental. En fonction des caractéristiques de l'utilisateur et de la disposition des périphériques, et plus généralement, de toute information du contexte que le concepteur et l'ergonome d'Ambiant jugeront pertinente, le choix de la distribution de l'interface peut varier.

Il existe une forme de redistribution qui n'est cependant pas prise en charge par le système. Il s'agit de la distribution des CID sur plusieurs périphériques spécialisés. On retrouve cette forme de redistribution dans la « métaphore du peintre » (Rekimoto 1997). Cette métaphore consiste à utiliser deux écrans simultanément, chacun dans un rôle spécifique : un petit écran appelé « palette » fournit une boîte à outils tandis qu'un écran plus grand appelé « tableau » fournit la surface de travail sur laquelle appliquer l'outil. Dans cette métaphore, les deux écrans n'ont pas le même rôle, ils sont différenciés sur le plan sémantique. Par conséquent, il s'agit d'un langage d'interaction différent de celui des interfaces graphiques traditionnelles. Ce langage suppose que le CID spécifie quelle partie de l'interface est distribuée sur la palette et quelle partie est distribuée sur le plateau, cette information ne pouvant être inférée automatiquement. Par conséquent, ils supposent l'usage d'un ensemble de CID spécifiques à cette métaphore associés à des contrôleurs de langages également spécifiques à la métaphore. Ces composants peuvent coexister dans le système avec des composants implémentant le langage d'interaction graphique traditionnel, ils partageront d'ailleurs bon nombre de sous-composants (c'est encore un exemple de factorisation de code permise par l'approche orientée composants) mais génèreront des chaines d'interaction qui n'ont rien en commun.

IX.2 MODELISATION DE TECHNIQUES D'INTERACTION POST-WIMP

L'implémentation de techniques d'interaction post-WIMP dans DAME est similaire à celle de méthodes d'interaction plus traditionnelles. Elle consiste à identifier 3 niveaux d'abstraction :

- Le langage d'interaction associé à la technique/métaphore d'interaction. Ce langage d'interaction, représenté dans le modèle ergonomique par les modalités et les canaux sur lesquels il repose doit être représenté également dans le modèle architectural par des interfaces décrivant les fonctions qui permettent de construire un dialogue. Ce jeu d'interfaces permettra de découpler le contrôleur de langage des différents CID qui pourraient utiliser cette technique d'interaction.
- Les structures de données permettant la réification des modalités sur lesquelles s'appuie le contrôleur de langage. Ces structures de données seront exposées par le contrôleur de médias qui réalise la passerelle avec le niveau d'abstraction suivant.
- Les langages de médias utilisés pour implémenter la technique/métaphore d'interaction. Ces langages sont exposés par les composants de type média.

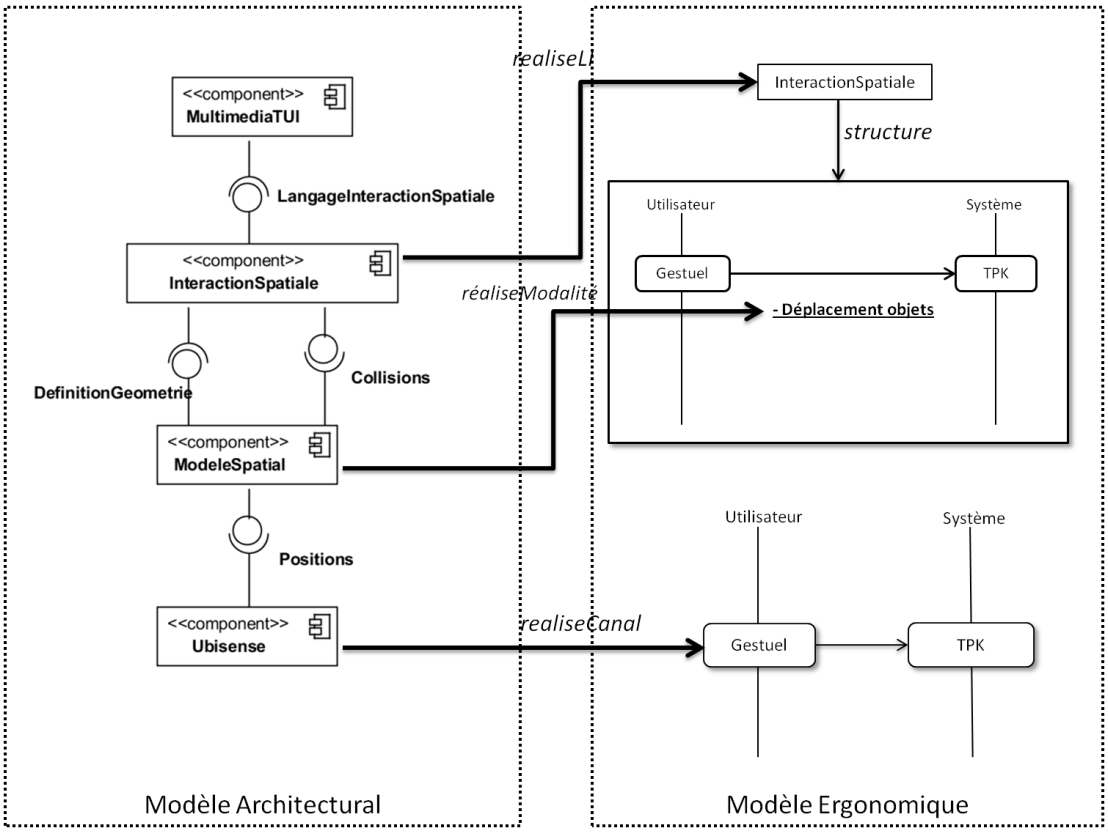


FIGURE 80 – REALISATION D'UNE TECHNIQUE D'INTERACTION TANGIBLE DANS DAME

La Figure 80 illustre la réalisation d'une interface tangible permettant de contrôler la lecture de fichiers multimédias par une interface tangible. Le langage associé consiste à attribuer un rôle à certains objets du monde physique appelés phicons (Ishii & Ullmer 1997). Un phicon désigne la fonction de lecture, et d'autres représentent les fichiers multimédia à jouer. Lorsqu'un phicon représentant un fichier multimédia est posé près de celui représentant la fonction de lecture, le fichier est joué par le système. Cette technique d'interaction tangible est décrite dans (Ishii & Ullmer 1997). Elle peut être implémentée grâce à un média de détection de la position. Dans la Figure 80, il s'agit d'Ubisense, un système donnant la position de tags présents dans une pièce sous la forme d'un triplet (x,y,z). Le contrôleur de média réalise un modèle spatial à partir des évènements de bas niveau fournis par ubisense[30]. Ce modèle permet de détecter les mouvements des objets réalisés par l'utilisateur, et notamment de déterminer les collisions entre phicons. Enfin, le contrôleur de langage interprète les informations de haut niveau (collisions entre phicons) pour en déduire les actions à effectuer, actions qui sont définies par le CID MultimediaTUI.

3 aspects de DAME concourent à la réutilisabilité des techniques d'interaction ainsi modélisées :

- Le découplage entre le dialogue et le langage d'interaction.
- Le découplage entre le langage d'interaction et les médias qui le réalisent.

[30] http://www.ubisense.net/

- L'établissement d'une liste de recommandations concernant ce langage dans le modèle comportemental. Ces recommandations permettent de limiter ou de favoriser l'utilisation de la technique d'interaction en fonction du contexte.

En conclusion, la démarche DAME permet la combinaison de techniques d'interactions variées. Elle repose sur l'identification des interfaces permettant de découpler les différents modules réutilisables de code. Ce découplage permet une redistribution d'un même langage d'interaction sur plusieurs périphériques et une cohabitation des différentes techniques d'interaction. La section suivante explore la représentation de la multimodalité dans DAME.

IX.3 SATISFACTION DES BESOINS DE L'IHM AMBIANTE ET CONCLUSION

Dans ce chapitre, nous avons étudié quelques exemples de modélisation d'IHM ambiantes à travers les modèles ergonomique et architectural de DAME. Nous avons également illustré le rôle du modèle comportemental dans l'évaluation ergonomique d'une chaine d'interaction. Nous pouvons en conclure la satisfaction des besoins que nous avions identifiés dans la section V.2 à propos de la réalisation d'IHM ambiantes.

Les exemples de la section IX.1 démontrent la capacité du système à produire, pour un même candidat, des chaines d'interaction multi-médias reposant sur diverses distributions, remplissant ainsi le besoins (a).

Les exemples de la section IX.2 illustrent la capacité du système à accueillir de nouvelles techniques d'interaction (besoin (b)). La modularité de la programmation orientée composants permet de garantir l'extensibilité du système par de nouveaux modules logiciels. Le modèle architectural de DAME permet de guider le concepteur dans l'identification des rôles des composants de l'IHM et notamment dans l'identification des interfaces permettant le découplage entre les différentes couches logicielles.

Enfin, le besoin (c) est réalisé par l'évaluation des solutions proposées par l'algorithme présenté dans le chapitre VIII. Cette évaluation est réalisée par l'application de recommandations ergonomiques écrites par le concepteur ou l'ergonome d'Ambiant. Dans les sections IX.1 et IX.2, nous avons illustré ce rôle du modèle comportemental dans l'application de solutions adaptées aux personnes handicapées.

CONCLUSION DE LA PARTIE B

A partir de l'analyse de l'état de l'art, nous avons extrait les limites des systèmes d'interaction ambiants actuels. Pour adresser ces limites, nous avons adopté une démarche orientée composants dans laquelle les éléments d'une IHM sont représentés par des composants appelés CID qui sont conçus dans le but de réaliser des cas d'utilisation. Le système ambiant peut ainsi réaliser une tâche en identifiant les cas d'utilisation et les services qui la réalisent. Il demande ensuite au système d'interaction de construire l'IHM correspondante.

Nous nous sommes concentrés dans cette partie, sur la réalisation d'un tel système d'interaction. Nous proposons un algorithme de génération d'IHM qui tienne compte de contraintes structurelles et de recommandations ergonomiques. Cette génération d'IHM repose sur une démarche de modélisation intitulée DAME qui divise la modélisation en 3 composantes :

- Le modèle ergonomique
- Le modèle architectural
- Le modèle comportemental

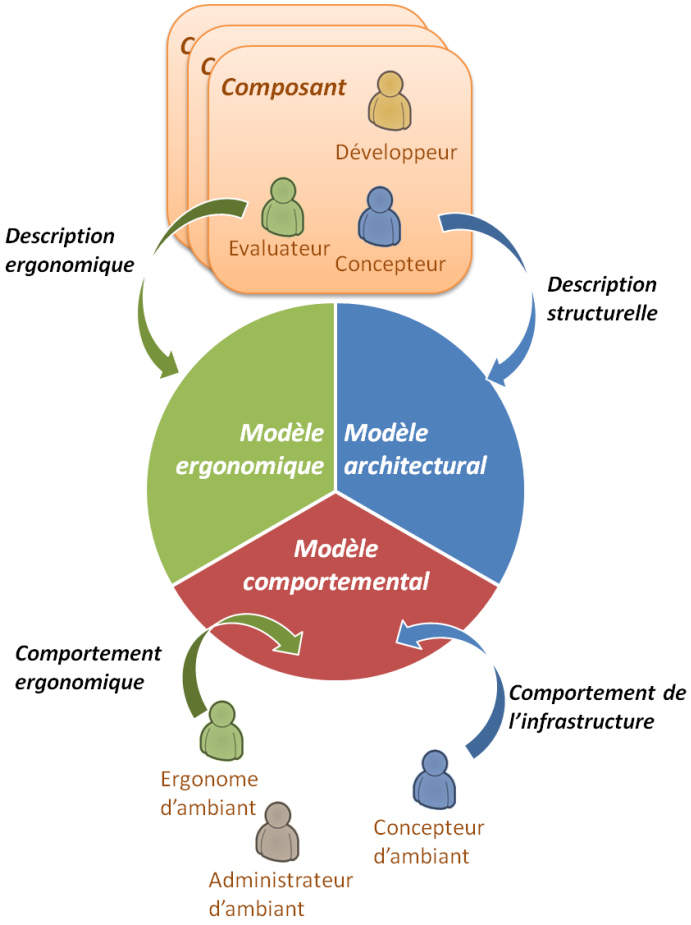


FIGURE 81 - RESUME DE LA DEMARCHE DE MODELISATION DE DAME

La démarche de conception DAME peut être résumée par la figure 81, initialement présentée page 118. Nous avons identifié, dans le processus de conception de composants logiciels pour l'Ambiant, différents acteurs qui collaborent autour de la même plateforme. Les 3 modèles proposés fournissent un langage commun permettant à ces acteurs de partager de l'information. Ainsi, les composants développés pour un environnement sont décrits dans les modèles ergonomique et architectural. Le comportement que doit avoir le système vis-à-vis de ces composants est décrit par le modèle comportemental. Cette démarche est décrite et justifiée dans le chapitre V.

Les chapitres VI, VII et VIII détaillent la structure des modèles proposés. Le modèle ergonomique que nous proposons s'inspire d'une terminologie orientée utilisateurs de la multimodalité. Il est structuré par les concepts de mode, modalité, et média auxquels nous ajoutons les définitions de langage de média, langage d'interaction et de canal de communication. Le modèle ergonomique décrit une décomposition stéréotypée des composants de l'IHM. Il propose une architecture en couche qui définit 3 niveaux d'abstraction : le média, le contrôleur de médias et le contrôleur de langages. Ces 3 types de composants permettent de décrire de façon réutilisable comment on instancie un langage d'interaction donné. L'instanciation de cette chaine d'interaction permet de faire le lien entre les CID et les médias.

Le modèle comportemental, enfin, repose sur un algorithme qui construit une interface sous la forme d'une liste de CID qui, une fois réunis, réalise tous les besoins de l'utilisateur requis par le système ambiant. Cette liste de CID est appelée un candidat. Les chaines d'interaction permettant de réaliser chaque candidat sont ensuite énumérées. Chacune d'entre elles est composée d'un contrôleur de langages, d'un contrôleur de médias et d'un ensemble de médias. Les candidats et leurs chaines d'interaction représentent les différentes solutions possibles. Ces solutions sont évaluées en appliquant les critères du modèle comportemental. Ces critères sont des heuristiques d'évaluation implémentées de façon arbitraires par les concepteurs. Elles permettent d'établir des recommandations sur différents critères. Nous proposons un mécanisme standard pour la définition de ces critères : le système électif. Ce dernier consiste à rédiger les recommandations sous la forme de règles qui contribuent à modifier (augmenter ou diminuer) la note finale d'une solution. Les règles sont rédigées en logique du premier ordre, en utilisant les prédicats et relations binaires définis par les modèles ergonomique et architectural pour produire un système de règle réutilisable.

Nous avons par la suite, appliqué notre démarche de modélisation à divers exemples caractéristiques de l'Ambiant (cf. Chapitre IX) : couplage, remplacement de médias et redistribution de l'IHM. Nous en avons déduit que l'approche DAME répondait aux besoins exprimés dans le chapitre V. C'est-à-dire qu'elle permet :

- De gérer plusieurs configurations multi-médias en entrée **et** en sortie
- D'implémenter et d'intégrer au système de nouvelles techniques d'interaction innovantes.
- D'automatiser la création d'une interface adaptée à la tâche, aux contraintes ergonomiques et d'une manière plus générale, à toute information du contexte pertinente pour son adaptation.

La partie suivante décrit une implémentation de la démarche DAME reposant sur les ontologies et un système expert. Nous présenterons également notre expérience de ce système dans un appartement intelligent.

Partie C
IMPLEMENTATION ET VALIDATION DE LA DEMARCHE DAME

Dans cette partie, nous étudions l'application de la démarche DAME dans des situations réelles. Nous présentons, dans un premier temps, une implémentation du système reposant sur un système expert, l'utilisation d'ontologies et d'un conteneur de composants. Dans un second temps, nous présentons une application de DAME dans le contexte de « l'Ambiant à la maison ». Il s'agit d'un prototype conçu et évalué dans le cadre du projet ATRACO.

Chapitre X
IMPLEMENTATION D'UN SYSTEME D'INTERACTION AMBIANT

Les 3 modèles (architectural, ergonomique et comportemental) présentés dans les sections précédentes ont été implémentés en un système capable de proposer des interfaces dynamiquement adaptées au contexte d'interaction dans un environnement ambiant.

Cette implémentation se compose de deux parties : un noyau décisionnel et un système de composants (cf. Figure 82). Le noyau décisionnel est le composant du système d'interaction qui modélise l'environnement et qui, à partir de cette modélisation, prend des décisions sur les composants de la chaine d'interaction à instancier. Le conteneur de composants a pour rôle de répertorier les composants disponibles et d'instancier la chaine d'interaction en établissant les connexions entre les différents composants (CID, CL et CM) et les médias disponibles dans l'environnement ambiant.

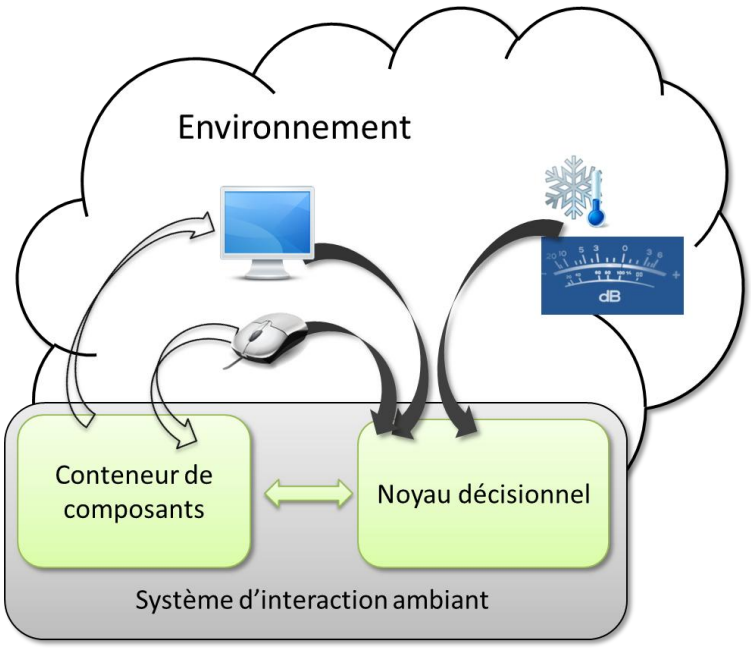


FIGURE 82 - ARCHITECTURE GENERALE DU SYSTEME D'INTERACTION AMBIANT

Ce chapitre se divise en trois sections. Les deux premières décrivent chacun des deux sous-composants et leurs interactions avec l'environnement, la troisième décrit les interactions entre le noyau décisionnel et le conteneur de composants.

X.1 LE NOYAU DECISIONNEL

Le noyau décisionnel a deux rôles : il doit permettre de modéliser l'environnement dans lequel le système d'interaction est immergé mais également de raisonner sur ce modèle. La

modélisation de l'environnement doit se faire au moyen de structures logiques compatibles avec les modèles ergonomiques et d'architecture que nous avons proposés, mais surtout ouvertes à modification et extension par le monde extérieur. Nous présentons dans un premier temps notre approche basée sur les ontologies (Pruvost et al. 2009). Nous aborderons dans un second temps la problématique du raisonnement sur les instances de cette ontologie via un système expert.

X.1.1 L'ONTOLOGIE D'INTERACTION

X.1.1.i PRESENTATION DES ONTOLOGIES ET DE OWL

En philosophie, l'ontologie est l'étude des propriétés générales de ce qui existe. Il s'agit d'une branche de la métaphysique qui s'applique à extraire le sens des concepts à travers les liens qui les unissent. Cette vision d'une sémantique qui est dérivé des liens existant entre concepts a donné lieu, en informatique, à une forme de logique descriptive appelée également ontologie. Dans (Gruber 1995), Gruber définit les ontologies ainsi[31]:

« Une ontologie est la spécification d'une conceptualisation. […] Une conceptualisation est une vue abstraite et simplifiée du monde que l'on veut représenter. »

Une ontologie est un ensemble structuré de termes et concepts qui représentent un domaine de connaissances. Lors de la conception d'une ontologie, il faut conserver à l'esprit qu'une ontologie est la conceptualisation d'un domaine, c'est-à-dire qu'il s'agit d'un choix quant à la manière de décrire ce domaine. Ces choix favorisent une vision particulière du domaine. Comme pour tout modèle, l'ontologie permet de spécifier certains aspects d'un domaine dans un but donné.

Sur un plan plus concret, l'ontologie est un réseau sémantique de concepts, c'est-à-dire un graphe orienté étiqueté dont les nœuds représentent des concepts ou des instances de ces concepts et dont les arcs définissent soit une notion de hiérarchie entre concept, soit un lien sémantique entre deux concepts. Plusieurs formalismes existent pour décrire ces graphes, nous emploierons le formalisme OWL (Ontology Web Language) qui est un formalisme très répandu pour les ontologies dans les services webs.

Le langage OWL définit 3 sous-langages inclus les uns dans les autres et d'expressivité croissante : OWL-Lite, OWL-DL et OWL-Full. En fonction des caractéristiques du langage employé (classes nommées, classes anonymes, cardinalités simples ou multiples…), l'ontologie pourra se restreindre à l'un de ces trois sous-langages. L'inférence est entièrement décidable pour OWL-Lite et OWL-DL, elle est indécidable pour OWL-Full (Motik, Sattler & Studer 2005). Les modèles ergonomique et architectural ont été spécifiés en utilisant une logique descriptive qui permet de les traduire dans OWL-DL et d'assurer que les raisonnements que nous exprimerons dans ce langage sont décidables.

[31] *An ontology is an explicit specification of a conceptualization. [...] A conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose*

X.1.1.ii INTERET DES ONTOLOGIES POUR LA FORMALISATION DES MODELES ERGONOMIQUE ET ARCHITECTURAL

Il n'existe pas une unique méthodologie pour concevoir une ontologie. Cependant, de nombreux guides de bonnes pratiques ont été discutés dans la littérature. Nous retenons les 5 critères de Gruber (Gruber 1995) sur lesquels juger une ontologie :

- La clarté

  La définition d'un concept doit faire passer le sens voulu du terme de façon objective et complète, c'est-à-dire par l'ajout de conditions nécessaires et suffisantes, mais également par leur documentation en langage naturel. En effet, à chaque concept est associée une étiquette qui permet aux personnes réutilisant l'ontologie de mieux comprendre le concept au-delà de sa définition mathématique. Ce mécanisme s'apparente aux commentaires d'un code informatique.

- La cohérence

  Aucune information inférée de l'ontologie ne doit entrer en contradiction avec le reste de l'ontologie.

- L'extensibilité

  Les extensions qui pourront être ajoutées à l'ontologie doivent être anticipées. Il doit être possible d'y ajouter de nouveaux concepts sans avoir à toucher aux fondations de l'ontologie.

- Une déformation d'encodage minimale

  La spécification ne doit pas influencer la conceptualisation. Pour un langage d'ontologie donné, les structures du langage peuvent pousser à définir un concept d'une certaine façon (car elle est plus simple) alors qu'elle dénature le sens initial du concept. Ce type de déformations est à éviter.

- Un engagement ontologique minimal

  On n'attend pas d'une ontologie qu'elle soit en mesure de fournir systématiquement une réponse à une question arbitraire sur le domaine qu'elle décrit (contrairement aux bases de faits d'un système expert par exemple). Elle doit se contenter de définir les termes nécessaires pour partager les connaissances consistantes d'un domaine. Elle décrit les éléments d'une théorie et offre un vocabulaire du domaine permettant ensuite de décrire plusieurs visions parfois contradictoires. Cette considération est à rapprocher des critères de cohérences et d'extensibilité.

Les préoccupations d'extensibilité et d'engagement ontologique minimal différencient l'approche ontologique des systèmes experts (et de leurs bases de faits). L'ontologie représente un vocabulaire minimaliste pour décrire un domaine dans un monde ouvert tandis que la base de faits vise à décrire une structure permettant de répondre à un ensemble de questions bien définies dans un monde fermé. L'extension des ontologies est plus généralement discutée à travers la technique dite d'alignement d'ontologie (Noy & Musen 2000) qui vise à établir, de façon automatique, des liens entre les concepts de deux

ontologies distinctes. Ces méthodes d'alignement (et de fusion) ont pour objectif de traduire un modèle en un autre modèle, ou du moins d'identifier les concepts sur lesquels ces modèles se recouvrent et d'établir des ponts entre les deux vocabulaires.

L'hypothèse de monde ouvert/fermé a déjà été abordée en introduction du chapitre V. Elle est l'un des facteurs spécifiques de l'ambiant. Nous envisageons un environnement dont les constituants auront été développés par des équipes séparées, selon des normes distinctes (cf. section V.3.2). Chaque constituant (service et/ou périphérique) pourra contribuer au système pourvu qu'il soit décrit dans un certain vocabulaire, cette description pouvant être fournie par le service lui-même ou par un service tierce. Les différentes entités du système pourront alors se comprendre sur la base de ce vocabulaire descriptif commun. L'ontologie qui est présentée dans la section ci-dessous définit un vocabulaire qui permet de décrire les informations qui concernent le système interactif.

De nombreux facteurs inciteront les futurs concepteurs à ne pas se contenter de ce vocabulaire de base : évolution des usages, disparité des besoins, apports d'autres études académiques, conflits de modèles économiques avec des entreprises définissant leurs propres standards… Cependant, l'approche ontologique nous permet, grâce aux outils de traduction (manuels, semi-automatiques, voire automatiques) qui l'accompagnent, de fournir un vocabulaire extensible, traduisible, et donc évolutif. Par ailleurs, chaque concept d'une ontologie OWL est identifié par un identifiant unique (selon la norme IRI – Internationalized Resource Identifier) qui permet notamment de retrouver sur le web le fichier dans lequel il est défini. L'approche ontologique est donc parfaitement adaptée à la spécification des modèles descriptifs de l'approche DAME (modèle architectural et modèle ergonomique), plus encore que la description dans une base de faits.

X.1.1.iii Specification du modele architectural et du modele ergonomique

Notre méthodologie pour concevoir l'ontologie d'interaction s'est appuyée sur les 5 critères développés dans la section précédente. Elle a de plus été facilitée par la similitude entre le formalisme de logique descriptive employé et le formalisme d'OWL. En effet, le langage OWL est structuré autour de trois concepts fondamentaux : la classe, l'individu et la propriété. Il existe une correspondance immédiate entre ces trois concepts et ceux (respectivement) de classe, d'instance et de relation que nous utilisons pour formaliser nos modèles. Les classes définissent des ensembles « d'objets », des catégories qui ont un sens particulier et qui peuvent hériter les unes des autres. Les propriétés lient les individus entre eux (ou à des types de données de base), ils correspondent aux relations binaires dont les ensembles de départ et d'arrivée sont définis grâce aux classes.

X.1.1.iv Raisonneurs et systemes de regles : Le modele comportemental

Le sujet de la représentation des contraintes des modèles d'ergonomie et d'architecture ou des règles de conception contenues dans le modèle comportemental n'a pas encore été abordé. Certaines des contraintes de nos modèles peuvent être naturellement traduites dans OWL-DL. Par exemple, ce dernier fournit des attributs permettant de décrire qu'une propriété est transitive, symétrique, réflexive… Ces attributs ne couvrent cependant pas toutes les contraintes arbitraires dont on peut avoir besoin dans la définition d'un modèle

(notamment les contraintes de cardinalité, ou bien des contraintes faisant intervenir plusieurs propriétés).

Un langage de règles dédié aux ontologies existe, il s'appelle SWRL (Semantic Web Rule Language). Il repose sur une syntaxe qui est compatible avec OWL et permet de définir des règles de logique du prédicat (à l'aide de clauses de Horn). Ce type de règles est utile pour déduire un certain nombre de faits simples à partir des faits existants, il permet notamment de peupler automatiquement une partie de l'ontologie à partir de quelques éléments de base, limitant ainsi la saisie fastidieuse des individus.

Cependant, SWRL est limité au calcul propositionnel et ne permet donc pas de définir des contraintes reposant sur la logique du 1er ordre (faisant intervenir les quantificateurs universels ∀ et ∃). La raison en est simple : raisonner sur ces quantificateurs requiert de faire l'hypothèse du monde fermé (*Closed-World Assumption*). Cette hypothèse, lorsqu'elle est appliquée à la logique, correspond à la négation par l'échec (*Negation as failure*): tant qu'un fait n'appartient pas à la base des connaissances, il est supposé comme faux. Or, les ontologies reposent sur l'hypothèse du monde ouvert, pour laquelle les faits n'apparaissant pas dans la base des faits ont une véracité indéterminée.

Or, la plupart des contraintes que nous avons construites utilisent ces quantificateurs. Il en va de même pour une partie des règles du modèle comportemental. L'implémentation de ceux-ci requière donc des langages de règles plus avancées en logique du premier ordre, qui sont disponibles au travers des systèmes experts.

X.1.1.v LE GESTIONNAIRE D'ONTOLOGIES

Le langage OWL permet d'importer d'autres ontologies et donc de disperser la base de connaissance dans une hiérarchie de fichiers. Les concepts des modèles architectural et ergonomique sont ainsi séparés dans des fichiers distincts. Un troisième fichier importe les deux et permet de faire la jonction entre eux. C'est ce fichier qui est chargé à l'initialisation du noyau décisionnel et qui définit le vocabulaire d'échange d'informations avec l'environnement. Une fois chargé, les différents concepts sont fusionnés dans le même espace mémoire que l'on appelle **ontologie d'interaction**. Toute extension des modèles d'architecture et/ou ergonomique, requiert de modifier le chargement de ce fichier pour modifier l'ontologie d'interaction.

Le gestionnaire d'ontologies est l'entité logicielle chargée d'importer/exporter les ontologies à partir des fichiers. Nous utiliserons le format OWL/XML pour l'import/export de fichiers et la librairie OWLAPI pour réaliser ces opérations. Cette librairie fournit une structure de données permettant de représenter les connaissances d'une ontologie, elle permet également d'effectuer des opérations courantes sur les ontologies telles que la résolution des imports, la fusion de plusieurs ontologies, etc. Elle fournit également une interface permettant d'interfacer ce gestionnaire d'ontologies avec des systèmes de raisonnement.

X.1.2 UN SYSTEME EXPERT POUR LE RAISONNEMENT

X.1.2.i PRESENTATION DES SYSTEMES EXPERTS

Un système expert se compose de 3 parties : une base de faits, une base de règles et un moteur d'inférence. Le moteur d'inférence est chargé de l'application des règles de manière optimale. Cette application peut se faire selon deux principales stratégies qui peuvent être combinées :

- Le chainage avant

  Il consiste à analyser les prémisses des règles pour détecter les règles qui peuvent être appliquées. L'algorithme de Rete (Forgy 1982) optimise cette méthode en détectant les motifs similaires dans les prémisses des règles et en utilisant un réseau de Pétri organisant ses motifs de façon optimale. Les jetons du réseau portent les faits qui satisfont le motif. Les états finaux du réseau sont les règles à appliquer. Ainsi, on obtient après construction et application du réseau sur la base de faits, la liste des faits qui satisfont les prémisses. Un agenda a ensuite pour rôle d'appliquer une politique d'ordonnancement des règles à appliquer. Chaque règle peut alors entrainer la création d'un ou plusieurs faits qui sont ajoutés à la base, ce qui amène à la réapplication des règles (on parle de règles réentrantes). Ce mécanisme présente un risque de boucle infinie qui est abordé différemment en fonction des systèmes. L'exécution du moteur s'arrête lorsqu'il n'y a plus de règles à appliquer.

- Le chainage arrière

  Il consiste à partir à la recherche d'un but bien défini (sous la forme d'une formule dont certaines variables sont inconnues) en observant les conclusions des règles qui pourraient permettre d'établir le fait recherché. A partir des prémisses de ces règles, on identifie les faits qui pourraient conduire au but recherché, et l'on fait une recherche sur ces faits. On construit ainsi par backtracking, un arbre de recherche dans la base de fait qui peut établir la suite de règles à appliquer (si elle existe) pour atteindre le but recherché. L'optimisation se fait au moyen de coupures dans l'arbre de recherche. Cette approche a été rendue célèbre par le langage Prolog reposant sur l'algorithme de SLD-résolution.

La plupart des systèmes experts actuels à base de règles permettent une résolution combinant les méthodes de chainage avant et arrière.

X.1.2.ii CHOIX D'UN SYSTEME EXPERT

Nous avons choisi Jess[32] comme système expert pour notre implémentation. Jess est un système expert développé en Java au sein du laboratoire national Sandia[33]. Ce système offre une compatibilité et une interaction avancée avec le code Java. Il est par ailleurs léger

[32] http://herzberg.ca.sandia.gov/

[33] Les laboratoires Sandia sont rattachés au département de l'énergie des Etats-Unis d'Amérique.

(par rapport à son seul concurrent sérieux en Java : Drools[34]) et offre une syntaxe simple inspirée de Lisp (ce qui sera un avantage pour générer du code Jess automatiquement, comme nous l'évoquerons dans la section suivante).

Jess est très largement inspiré du système expert Clips, développé par la NASA en 1985 en C, dont il reprend la syntaxe. Cette dernière est présentée dans l'annexe C.

X.1.2.iii De l'ontologie a une base de faits

Les structures de données employées pour représenter une ontologie et une base de faits dans un système expert diffèrent. Par conséquent, nous avons dû interfacer le système des ontologies avec le système expert Jess. Notre noyau décisionnel se compose donc de 3 entités (cf. Figure 83) : le gestionnaire d'ontologies, le traducteur et le système expert.

Le gestionnaire d'ontologies gère les entrées/sorties du noyau décisionnel, il permet d'organiser les connaissances importées de fichiers au format OWL, de résoudre les importations et d'interagir avec la base de connaissances. Une fois chargées, les ontologies sont représentées sous forme d'un graphe sémantique, qui peut être sérialisé en une liste de triplets (on parle de *triple store*) représentant les prédicats du modèle.

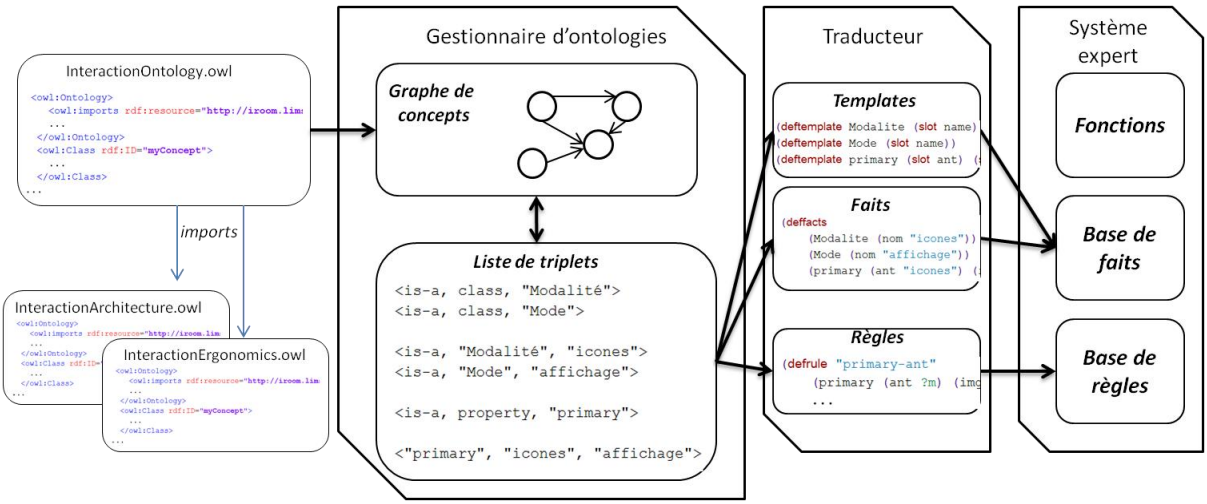


Figure 83 - Traduction d'une ontologie en base de faits

Le traducteur initialise la base de faits avec des templates décrivant la structure du modèle OWL, c'est-à-dire le méta-modèle de notre ontologie de l'interaction. Ces templates décrivent par exemple la notion de classe (`Class`) et de propriété (`Property`).

Le traducteur récupère ensuite les données issues du gestionnaire d'ontologies sous la forme d'une liste de triplets. Il génère le code Jess/Clips correspondant en fonction de la nature des prédicats représentés. Le Tableau 11 illustre les conventions utilisées pour faire cette mise en correspondance. Le traducteur s'assure également de réordonner les morceaux de code ainsi générés pour garantir leur cohérence et leur intégrité. On souhaite éviter, par exemple, qu'un fait soit déclaré avant son template, ce qui générerait une erreur.

[34] http://www.jboss.org/drools/

Nature du triplet	Structure Jess correspondante
Définition d'une classe A	(Class (name A)) (deftemplate A (slot name))
Définition d'une propriété p1 allant de A vers B	(Property (name p1)) (deftemplate p1 (slot ant) (slot img))
Instance x d'une classe X	(assert (X (name x))
Lien entre deux instances x et y par une propriété p1	(p1 (ant x) (img y))

TABLEAU 11 - CONVENTIONS UTILISEES POUR LA MISE EN CORRESPONDANCE DES STRUCTURES DE L'ONTOLOGIE ET DE LA BASE DE FAITS

Les faits ainsi générés dans la base de faits incluent des données du modèle et des données du méta-modèle. Ainsi par exemple, la définition d'une classe engendre à la fois la définition d'un fait du type `Class` et la définition d'un template portant le nom de cette classe. L'inclusion de faits décrivant le métamodèle permet d'écrire des règles garantissant sa structure, implémentant ainsi les contraintes permises par le langage OWL (contraintes de cardinalité, ou bien par exemple les contraintes définissant une propriété). L'ensemble de ces règles est exposé dans l'annexe D.

X.1.2.iv LE CŒUR DU NOYAU DECISIONNEL

Le cœur du noyau décisionnel est écrit en Jess/Clips dans des fichiers sont chargés à l'initialisation du module qui définissent un ensemble de templates, de règles et de fonctions nécessaires à la définition du comportement spécifique du système expert dans notre application.

Ces fonctions décrivent, en particulier, les mécanismes d'initialisation de la base de faits par traduction des ontologies. Elles décrivent également les mécanismes de résolution des imports de fichiers externes. Enfin, ce code contient une implémentation de l'algorithme présenté dans le chapitre VIII. Ce dernier y est implémenté dans une approche mariant programmation impérative et programmation logique (i.e. similaire à prolog). La programmation logique à l'aide de requêtes sur la base de faits permet d'implémenter les requêtes d'unification décrites dans le chapitre VIII. L'emploi de règles permet de construire à chaque étape de l'algorithme des faits qui aboutissent à la construction des chaines d'interaction. Les opérations sur les ensembles, enfin, sont implémentées via les structures Java correspondantes (Jess ne permet pas de construire des structures complexes simplement mais permet un interfaçage direct avec les objets Java).

X.1.3 EXTENSIONS DU NOYAU DECISIONNEL

Pour résumer, le noyau décisionnel est conçu comme la réunion de 3 entités :

- Un gestionnaire d'ontologie qui charge les informations pertinentes pour le système d'interaction dans un langage ouvert et extensible.
- Un système expert dont le fonctionnement, personnalisé par un ensemble de scripts, implémente la prise de décision au sein du système d'interaction.
- Un traducteur qui se charge d'interfacer les deux.

Certaines contraintes des modèles ne peuvent être exprimées en OWL ; elles sont alors immédiatement traduites par le traducteur. Cependant, comme nous l'avions précisé dans la section précédente, le modèle OWL ne suffit pas à décrire toutes les règles qui décrivent nos modèles. De plus, le modèle comportemental doit être personnalisé par le concepteur et/ou l'évaluateur d'ambiant. Nous avons donc ajouté un mécanisme d'extension à ce noyau décisionnel, qui permet d'exécuter des scripts lors de son initialisation. Ces scripts permettront de définir de nouveaux critères du modèle comportemental (des critères électifs ou non), ou plus simplement des règles permettant de vérifier que certaines contraintes sont bien vérifiées dans le modèle.

X.1.3.i MECANISME D'EXTENSION

Idéalement, ces extensions que nous souhaitons incorporer devraient être incluses dans l'ontologie pour maintenir une unité sémantique (c'est d'ailleurs la raison d'être de SWRL). Puisque c'est impossible, nous avons choisi d'importer ces règles directement sous la forme de fichiers Jess/Clips externes.

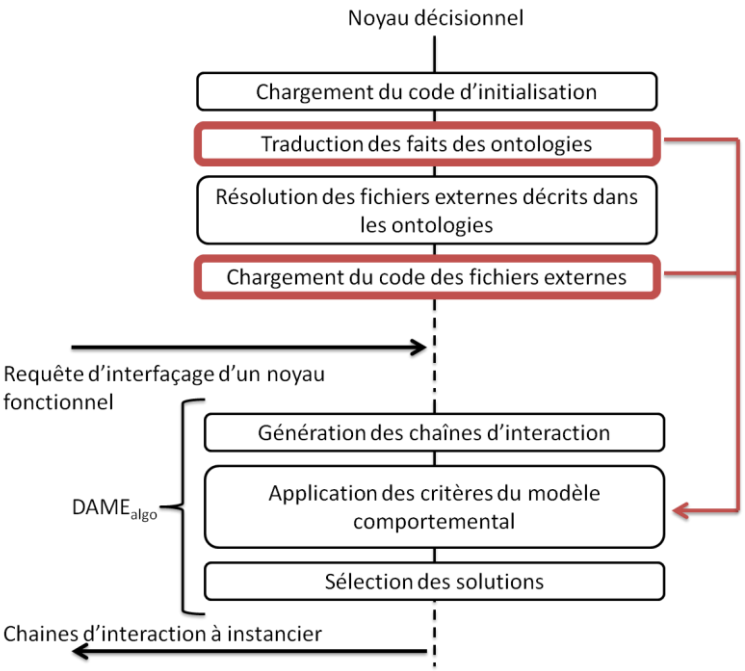


FIGURE 84 – ACTIVITES DU NOYAU DECISIONNEL

Pour conserver l'unité sémantique et faciliter la gestion de ces imports, nous avons implémenté un mécanisme d'extension de l'ontologie par des sources externes. Ce mécanisme consiste en une ontologie des métadonnées qui permet de décrire la nature, la

localisation et les dépendances de fichiers externes. On peut ainsi augmenter les connaissances d'une ontologie en important cette ontologie de métadonnées et en instanciant ses concepts avec les valeurs adéquates.

La Figure 84 liste les activités du noyau décisionnel et illustre l'impact des phases d'initialisation sur la phase d'application du modèle comportemental. Le système expert est initialisé avec le code permettant de résoudre ces imports de fichiers externes lorsqu'ils sont de type Jess/Clips. La résolution d'une dépendance implique son téléchargement et son application sur la base de faits. Ces fichiers externes peuvent contenir tout type de scripts Jess/Clips, bien qu'ils se limitent généralement à un ensemble de règles. Puisque ces scripts s'appliquent sur les données de l'ontologie d'interaction, ils sont appliqués à la fin de la phase de traduction des ontologies.

Ainsi, par exemple, l'ontologie modélisant le modèle architectural contient la description d'un fichier clips externe ajoutant des règles permettant de valider l'état du modèle. Il en va de même pour l'ontologie du modèle ergonomique.

Ce sont les concepteurs/évaluateurs du système ambiant qui définissent l'ontologie d'interaction chargée à l'initialisation du système. Ils peuvent choisir d'y inclure de nouveaux concepts pour étendre les modèles d'architecture et d'ergonomie. Ils peuvent également (et doivent) définir les critères qui font le modèle comportemental, au moyen des métadonnées ajoutées dans l'ontologie du système d'interaction.

X.1.3.ii EXTENSION DES CRITERES

Ce mécanisme d'extension est particulièrement utile pour la définition de critères non-électifs. Puisque ceux-ci sont définis au cas par cas, le mécanisme d'extension fournit le moyen d'inclure le code qui permet de calculer la note d'une chaine d'interaction, à la manière d'un plugin.

Dans le cas des critères électifs, ce mécanisme d'extension peut également être utilisé pour créer des syntaxes raccourcies.

X.2 LE CONTENEUR DE COMPOSANTS

Le conteneur de composants est la plateforme d'exécution du système d'interaction ambiant. Il s'agit d'une plateforme logicielle permettant de gérer dynamiquement des modules logiciels et offrant des mécanismes d'introspection et de contrôle sur les modules qui sont actifs et leurs interactions. Le conteneur que nous avons employé pour notre implémentation est OSGi. Il s'agit d'un conteneur Java ayant quelques années d'ancienneté garantissant sa fiabilité et son utilisation dans certains secteurs industriels. Il est par exemple utilisé pour certains logiciels open-source (Eclipse et Protégé pour ne citer qu'eux) pour sa capacité à fournir une architecture facilement extensible par des plugins.

Nous commençons par une présentation rapide des concepts relatifs à OSGi et à iPojo qui définissent l'architecture du conteneur utilisé dans notre implémentation de DAME. Nous présentons ensuite par quels mécanismes implémenter les composants décrits dans le

modèle architectural. Enfin, nous concluons sur la compatibilité d'une telle approche avec les systèmes largement distribués.

X.2.1 PRESENTATION D'OSGI

Le conteneur OSGi[35] permet une gestion dynamique des dépendances entre fichiers de code. Ce conteneur permet de charger, dans un même espace mémoire (i.e. au sein d'une même application), plusieurs fichiers de code appelés bundle. Chaque bundle est en fait un fichier de librairie Java – jar – qui est augmenté de métadonnées. La Figure 85 présente l'architecture de ce conteneur. On y voit que chaque bundle est décomposé en unités appelés packages qui peuvent être privé ou publics.

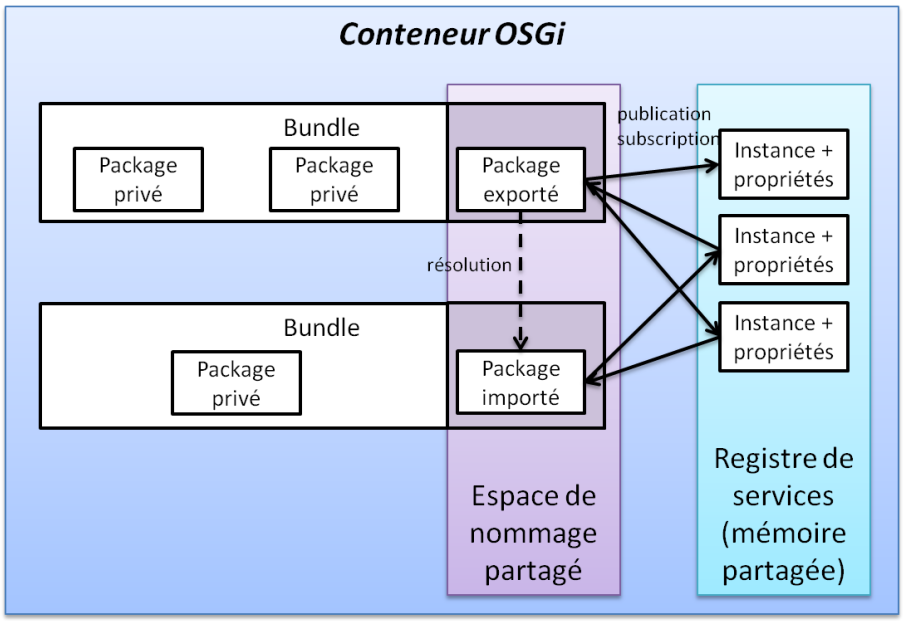


FIGURE 85 - ARCHITECTURE DU CONTENEUR OSGI

Le conteneur OSGi permet d'isoler les espaces de nom des différents bundles, ce qui permet à plusieurs versions du même code (de la même classe) de coexister en mémoire sans qu'il y ait de collisions entre ces différentes versions. Cela permet d'éviter les conflits entre dépendances dans le code. Les bundles peuvent communiquer grâce à deux mécanismes.

Le premier est la capacité des bundles à importer/exporter des espaces de noms entre eux (i.e. des packages publics). A travers la résolution de ces imports/exports, les bundles partagent une définition commune des classes appartenant aux packages publics. Ces packages contiennent la plupart du temps de simples interfaces permettant aux bundles d'exposer des services (i.e. des implémentations d'une interface donnée). Cependant, l'import/export d'un package ne définit qu'un espace de nommage partagé, il n'implique pas le partage automatique des instances des classes de ces packages.

[35] Open Service Gateway initiative

Le second mécanisme est le partage d'instances réalisant une interface (i.e. un service). Il est réalisé grâce à un système de publication/souscription dans un registre de services. Ce registre est commun à l'ensemble des bundles du conteneur OSGi et réalise la mémoire partagée du système. Chaque service y est exporté avec une éventuelle liste de propriétés. Les bundles qui souhaitent utiliser ces services peuvent se servir des propriétés pour filtrer les services utilisés.

Le conteneur OSGi propose une forme de programmation orientée services. Cette forme est limitée, de prime abord, à un même espace mémoire. Nous verrons cependant à la fin de cette section comment dépasser cette limite. On peut y voir une forme primitive de programmation orientée composants, à travers l'abstraction des implémentations des services dans un registre. Cependant, cette architecture est insuffisante pour concevoir simplement des composants pour deux raisons :

- La gestion du cycle de vie des connexions entre services est laissée à la charge du développeur. Celui-ci doit surveiller le registre pour invalider son composant dès lors que le service n'est plus disponible et il doit implémenter manuellement les politiques de reconnexion lorsque cette dernière est possible.
- La notion de composant n'est pas explicite. Chaque composant peut exposer plusieurs services mais il n'est pas clair, pour le bundle qui importe les services, de savoir quels services sont issus d'un même composant. Le mécanisme de propriétés et de filtres peut être utilisé pour réaliser cela mais il n'existe aucune convention à ce propos et cela reste à la charge du développeur d'implémenter de tels mécanismes.

X.2.2 PRESENTATION DE IPOJO

iPOJO est un ensemble de bundles OSGi qui étendent les fonctionnalités de ce conteneur pour lui permettre de gérer des architectures à base de composants.

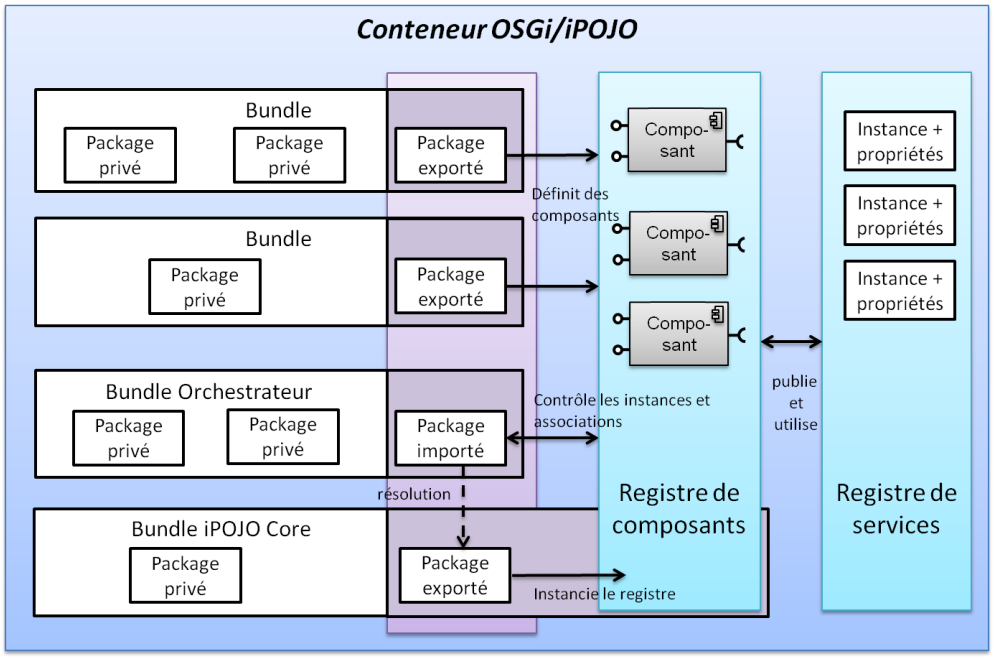


FIGURE 86 - APPORT DE IPOJO A L'ARCHITECTURE D'OSGI

Les bundles iPOJO implémentent un registre de composants (cf. Figure 86) qui abstraient le registre de services natif d'OSGi. Cette abstraction permet de décharger le développeur de la création d'une structure de composants et de la gestion de leurs cycles de vie. Celui-ci peut alors se concentrer sur le développement de composants, laissant à l'architecture iPOJO le soin de connecter les composants en fonction des besoins. Il est cependant possible de contrôler certains mécanismes de la gestion des composants grâce à une API qui est exposée par le bundle du noyau d'OSGi. iPOJO propose de plus un modèle de composants évolué permettant de définir récursivement des associations de composants qui sont compatibles avec notre approche.

Il est souvent reproché à OSGi une certaine lourdeur qui est due à deux facteurs :

- La définition des métadonnées du bundle est parfois complexe et sujette à erreurs.
- L'API d'OSGi qui permet de gérer le cycle de vie des bundles et des services prend une place importante dans le design du code.

iPOJO est un outil qui brise ces barrières technologiques en offrant un modèle de composant plus simple à utiliser et des outils permettant d'intégrer ce modèle naturellement aux phases d'édition et de compilation de code. *POJO* signifie *Plain Old Java Object* et représente le principe de base de la conception sous iPOJO : les composants sont de simples classes. Des méthodes avancées d'introspection et de modification du bytecode permettent à la librairie d'injecter dynamiquement les services découverts dans les attributs de la classe (d'autres méthodes d'injection existent également). Ainsi, le développeur code un composant en créant une classe qu'il « décore » grâce au système d'annotations fourni (le même résultat peut être obtenu sans modification du code, par l'ajout de fichiers XML au bundle). Cette simplicité d'usage est présentée dans l'annexe E.

X.2.3 ETIQUETAGE DES COMPOSANTS DE L'ESPACE LOGIQUE D'INTERACTION

Les composants du système d'interaction ambiant se divisent en deux catégories : les composants de l'espace logique d'interaction (CID, CL, CM, Média) et les composants qui orchestrent leur fonctionnement et permettent au système d'interaction ambiant de s'interfacer avec le reste du système, de raisonner et ainsi de s'adapter au contexte.

Les composants de l'EIL nécessitent d'être découverts automatiquement à l'exécution pour être décrits dans la base de faits. Cette description pourrait venir d'ontologies importées par le système mais il est plus flexible d'incorporer la description des composants dans les composants mêmes. Pour cela, nous employons le mécanisme d'étiquetage fourni par OSGi/iPOJO, qui permet d'attacher des métadonnées aux composants/services exposés.

Par convention, les étiquettes dont le nom commence par « DAME- » indiquent une information concernant le moteur d'inférence. En particulier, le type de composant sera étiqueté par un étiquette « DAME-cType » qui prendra une valeur parmi : « CID », « ControleurLangages », « ControleurMedias », « Media ». Ainsi, il sera aisé pour le système de lister tous les composants disponibles pour un type précis à partir de leurs étiquette.

Ce système permet également d'étendre les concepts ou les faits présents dans le moteur d'inférence. L'étiquette « DAME-ontology » permet d'associer à un composant, l'URL d'une ontologie décrivant des informations complémentaires le concernant. Le fichier

correspondant sera chargé à l'exécution, lors de la découverte du composant, par le mécanisme d'introspection présenté dans la section X.3.1 suivante.

X.2.4 OSGi/iPOJO vs programmation distribuee

Le conteneur OSGi/iPOJO réalise une version locale de la programmation orientée services/composants. Les systèmes ambiants sont cependant par nature distribués. Il est donc impératif que des composants puissent se partager sur plusieurs machines.

Différents protocoles existent pour ce faire. Certains sont orientés services (RMI, Web Services, SOAP) et d'autres composants (UPnP, Corba). Les composants d'un conteneur peuvent être importés/exportés selon ces protocoles de distribution en utilisant un bundle qui réalise la passerelle (on parle de *proxy*). Le conteneur OSGi dispose dans ses spécifications de bundles réalisant cet interfaçage automatique pour le protocole UPnP et RMI (cf. Distributed-OSGi).

On notera cependant que ce conteneur ne permet pas la migration de code et constitue donc une version minimaliste des préceptes de la programmation distribuée. En particulier, il n'existe aucun mécanisme d'équilibrage de la charge (indispensable pour une gestion de la qualité de services – QoS). Ce type de mécanismes n'a donc pu être étudié dans notre implémentation.

X.3 Architecture du systeme d'interaction ambiant

Nous avons étudié, dans les deux sections précédentes, les deux composants principaux de notre système. Nous proposons dans cette section d'étudier l'architecture globale du système, en analysant les différents modules nécessaires à son implémentation. Nous définissons, dans un premier temps, le rôle et les interactions entre ces modules. Nous verrons ensuite comment cette architecture s'adapte aux environnements distribués.

X.3.1 Interactions entre noyau decisionnel et composants de DAME

La Figure 87 représente l'architecture globale du système d'interaction ambiant que nous avons implémenté selon les principes de l'approche DAME. Les boites blanches y représentent les bundles OSGi développés. Chacun de ces bundles implémente un ou plusieurs composants qui interagissent tous entre eux. Nous avons clairement fait apparaitre les composants de l'EIL définis dans DAME. Ceux-ci sont représentés comme appartenant au même bundle car c'est ainsi que nous les avons implémenté mais il est tout à fait possible d'étendre la bibliothèque de composants en injectant de nouveaux bundles décrivant des composants supplémentaires, à condition que ceux-ci soient correctement étiquetés (cf. section précédente).

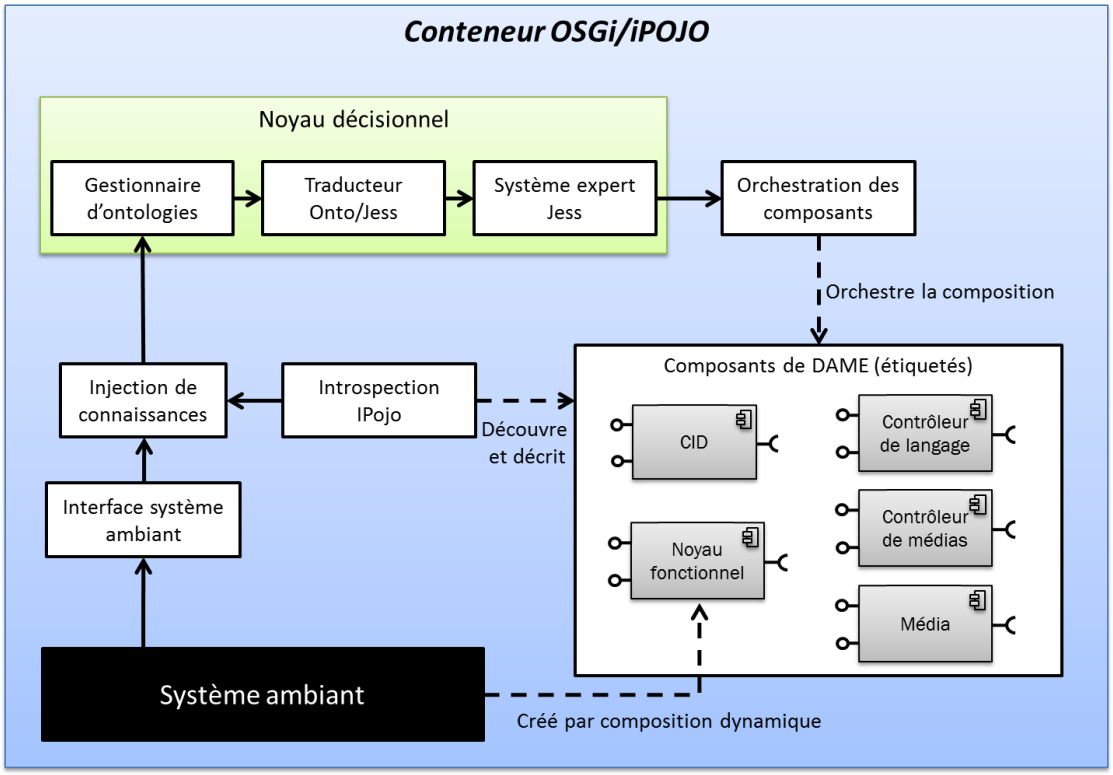


FIGURE 87 – ARCHITECTURE IMPLEMENTANT L'APPROCHE DAME

Le système ambiant représenté en noir représente l'ensemble des bundles qui ne font pas partie du système d'interaction. Il s'agit par exemple, dans la vision d'ATRACO, du contrôleur de sphère qui commande le système interactif. Ce système ambiant est représenté par une « boite noire » pour illustrer le fait que le système interactif ne sait rien à son propos, sur sa structure ou son fonctionnement, et ne peut interagir aveclui qu'au moyen d'une interface prédéfinie.

Cette interface sert à injecter des informations dans le noyau décisionnel (par exemple, l'information « il faut interfacer le noyau fonctionnel x »). Ce module d'injection de connaissance est également employé par le module d'introspection qui surveille la bibliothèque de composants de l'EIL. Lors de l'ajout ou du retrait de l'un de ces composants, ce module est chargé de mettre à jour la base de connaissances.

A la sortie du noyau décisionnel, le module d'orchestration des composants de l'EIL interprète les résultats de la phase d'élection en instructions de composition/destruction des composants iPOJO correspondants. Une fois que les composants sont connectés, l'interaction entre l'utilisateur et le noyau fonctionnel peut se réaliser. Le système ambiant informe en continu le noyau décisionnel des changements du contexte (position des objets, contexte environnemental…). De même, le module d'introspection iPOJO informe le noyau décisionnel des modifications de la bibliothèque de composants (apparition/disparition d'un média par exemple). A partir de ces informations, le noyau décisionnel peut mettre à jour sa sélection de chaines d'interaction qui est ainsi continuellement adaptée aux évolutions du contexte.

X.3.2 DISTRIBUTION DU SYSTEME D'INTERACTION AMBIANT

La Figure 87 a présenté de façon simplifiée l'architecture du système d'interaction ambiant. Il s'agit de la situation idéale dans laquelle tous les composants sont exécutés sur la même plateforme. Dans la plupart des cas cependant, un certain nombre de composants seront exécutés sur des plateformes distantes, reliés par un réseau Ethernet. La Figure 88 illustre la situation la plus commune dans laquelle le système ambiant est situé sur un hôte distant et dans lequel les médias ne sont pas directement connectés à la plateforme et où les noyaux fonctionnels sont construits à partir d'un ensemble de services répartis dans le réseau.

Notre implémentation tient compte de cette vision en implémentant les médias et les noyaux fonctionnels comme des composants distribués grâce au protocole UPnP. UPnP est un protocole pour lequel de nombreux périphériques ont été standardisés (lumières, air conditionné, multimédia…). Cela nous a permis de tester l'approche sur des systèmes industriels existants. Par ailleurs, concernant les médias, il offre l'intérêt d'être indépendant du langage (contrairement à RMI par exemple qui impose le langage Java). Il est ainsi facile, bien que l'ensemble du système soit codé en Java, d'incorporer des médias pour lesquels les pilotes n'existent qu'en C/C++.

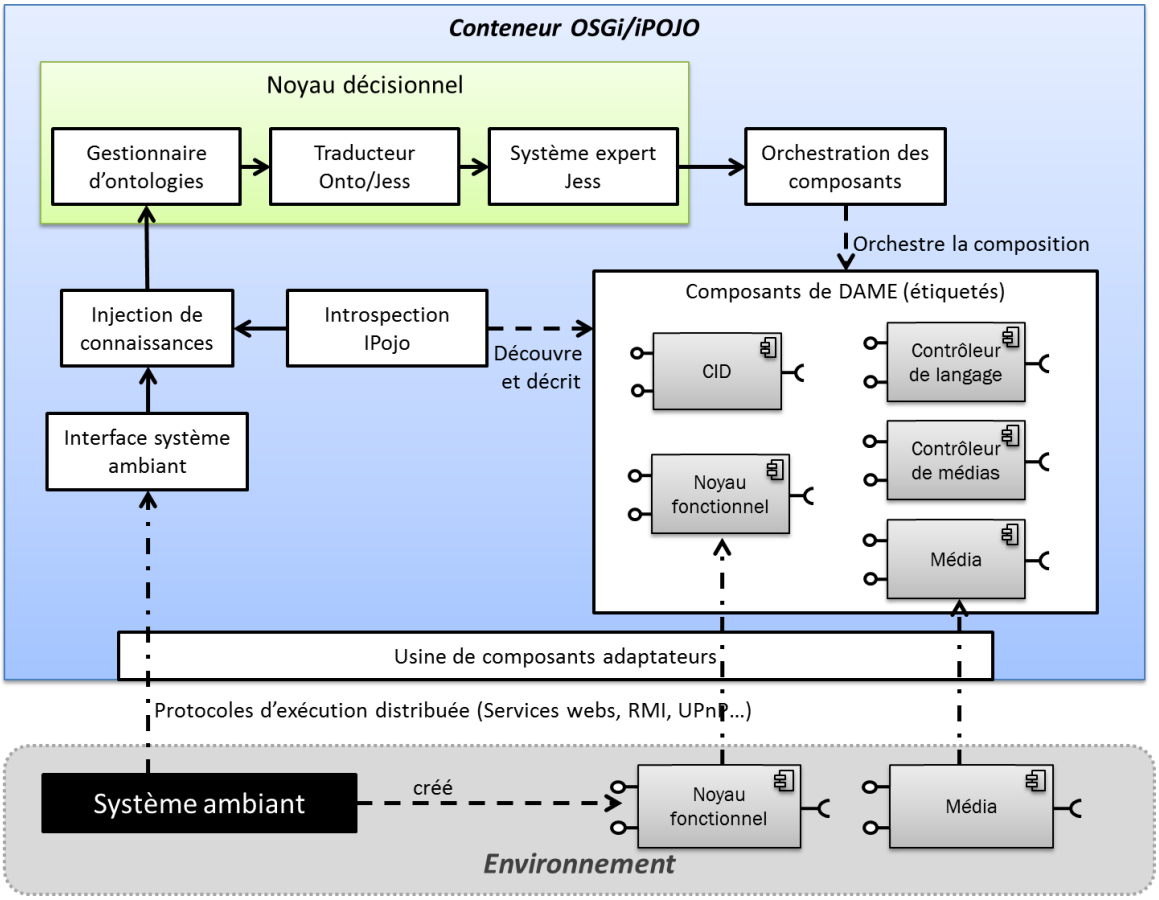


FIGURE 88 - ADAPTATION DE NOTRE IMPLEMENTATION A LA PROGRAMMATION DISTRIBUEE

Nous avons présenté l'architecture globale du système dans cette section en définissant, dans un premier temps, le rôle de chacun des modules et leurs interactions relatives. Nous avons présenté, dans un second temps, comment le caractère distribué du système impactait l'architecture par l'introduction d'un composant « proxy » qui établit la passerelle avec un intergiciel de communication (Service web, RMI, UPnP…). Cette implémentation a fait l'objet d'une évaluation que nous présentant dans le chapitre suivant. Ce dernier illustre une application de notre implémentation de DAME dans le cadre d'une évaluation de l'acceptation des systèmes ambiants auprès d'utilisateurs.

Chapitre XI
ETUDE DE L'ACCEPTATION UTILISATEUR ET DE L'IMPACT SOCIAL DANS LE CADRE DU PROJET EUROPEEN ATRACO

Nous présentons dans cette section le prototype et l'évaluation réalisés dans le cadre du projet de recherche européen ATRACO. Au cours des trois années du projet, nous avons progressivement défini les contours et les spécificités de l'approche DAME pour la conception de système ambiants interactifs. Nous avons pu, au cours du projet, développer trois prototypes dont le dernier a donné lieu à une évaluation qualitative face à des utilisateurs réels. Le but de cette évaluation est double, il s'agit à la fois de fournir une preuve de faisabilité de l'approche tout en s'intéressant à son degré d'acceptation par les utilisateurs finaux.

Nous avons présenté le cadre de référence d'ATRACO dans la section V.1.2. Nous fournissons cependant quelques éléments de rappels dans la première sous-section. Nous donnons ensuite quelques détails sur l'implémentation de l'agent d'interaction et de son interfaçage avec les autres composants d'ATRACO. A ce module logiciel actif, nous avons ajouté des outils d'édition et de simulation permettant aux concepteurs et aux ergonomes d'Ambiant d'explorer des scénarios avant de les concrétiser. Enfin, la dernière sous-section décrit les conditions et les résultats de l'expérience réalisée auprès d'utilisateurs finaux.

XI.1 VISION & ARCHITECTURE DU SYSTEME

Le cadre de réflexion d'ATRACO a été présenté dans la section V.1.2, page 103. Il s'agit d'organiser un écosystème de périphériques, de services et d'interfaces autour d'une activité de l'utilisateur. Au fur et à mesure des actions de l'utilisateur, des bulles de collaboration se forment entre les différentes entités de l'écosystème pour supporter l'activité de ce dernier. Ces collaborations éphémères, appelées « sphères d'activité », sont gérées par un ensemble d'agents logiciels qui fournissent les briques de bases nécessaires à l'orchestration des tâches proposées à l'utilisateur.

Le rôle et la nature des agents logiciels composant une sphère d'activité ont été décrits page 104. La Figure 89 illustre l'action conjointe de ces différents agents pour réaliser une sphère d'activité dans la plateforme expérimentale nommée ISpace (cf. Figure 90). Cette plateforme, constituée d'un appartement augmenté par des capteurs et des effecteurs, est située dans l'université d'Essex, au Royaume-Uni.

Dans le cadre du projet ATRACO, nous avons travaillé à la définition du concept de sphère d'activité et au rôle attribué à l'agent d'interaction dans celle-ci. Nous avons également contribué à l'implémentation d'un prototype et à la mise en place d'un protocole expérimental permettant d'évaluer de façon qualitative l'impact d'un système ambiant sur

la vie courante des utilisateurs. Cette évaluation est donc ancrée dans le contexte des tâches de la vie quotidienne et plus particulièrement centrée sur l'habitat intelligent (économies d'énergies, systèmes multimédias, cuisines augmentées, systèmes de sécurité…). Ce chapitre offre une description du prototype et des outils développés dans ce cadre ainsi qu'un retour d'expérience sur l'évaluation d'un système interactif ambiant.

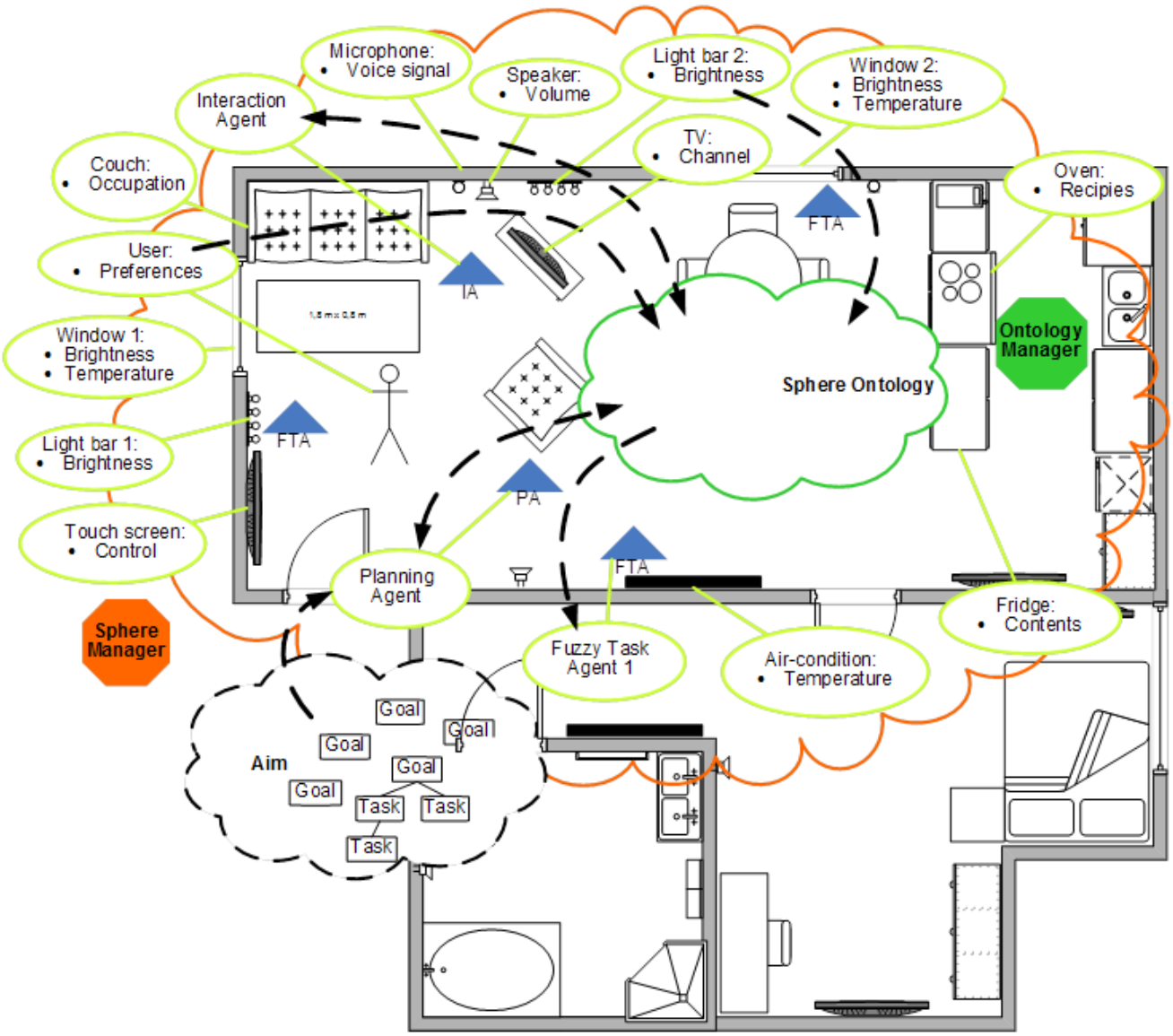


FIGURE 89 – MISE EN ŒUVRE D'UNE SPHERE D'ACTIVITE DANS L'ISPACE

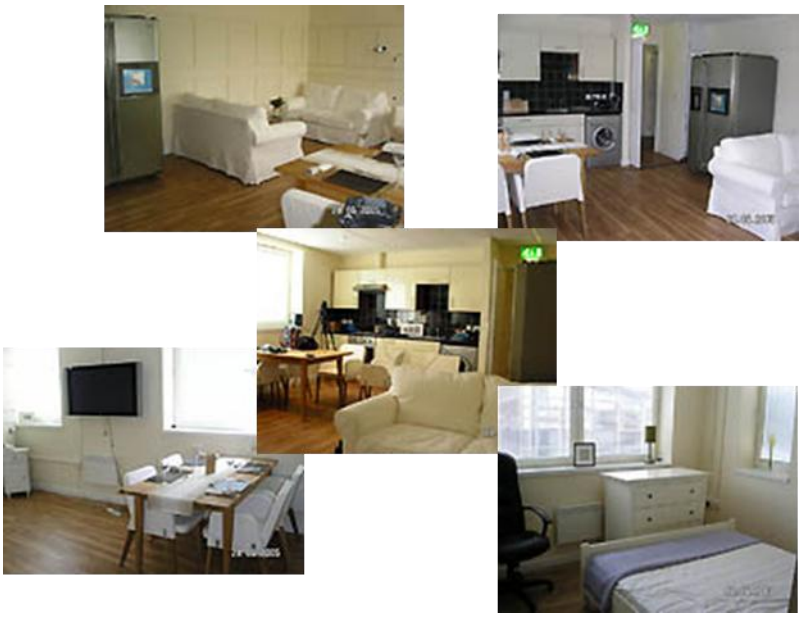


FIGURE 90 - PHOTOS DE L'ISPACE

XI.2 SPECIFICITES DE L'AGENT D'INTERACTION DANS ATRACO

L'agent d'interaction développé dans le cadre d'ATRACO est une version préliminaire du prototype que nous avons précédemment présenté. Il repose sur les mêmes fondations théoriques et est implémenté en suivant la même architecture associant 3 modules : une ontologie pour l'interaction, un système de raisonnement (implémenté grâce au système expert Jess) et un module d'instanciation de composants qui permet d'instancier les CID et autres composants de la chaîne d'interaction et de les interfacer avec le noyau fonctionnel.

A cause de son antériorité, mais également de son intégration au sein d'un projet de plus grande envergure, il existe quelques différences entre le prototype mis en place dans ATRACO et celui que nous avons présenté au cours des précédentes sections. Ces différences sont décrites dans cette section. Il s'agit principalement de variations dans les interactions entre le système interactif et le reste du système ambiant, mais également de simplifications sur le plan des modèles ergonomiques et d'architecture, qui ont permis d'obtenir un premier retour d'expérience de l'usage de ces concepts en conditions réelles.

L'implémentation de ce premier prototype a permis de prouver la faisabilité de l'approche DAME. Elle a également démontré l'intérêt des notions issues des approches à base de composants et de la théorie des modalités pour construire des interfaces et écrire des règles concernant leur adéquation au contexte. Grâce à cette implémentation, nous avons pu enrichir les modèles ergonomiques et d'architecture.

XI.2.1 DIFFERENCES ONTOLOGIQUES

L'ontologie de l'interaction utilisée dans ATRACO est disponible à l'adresse suivante :

http://iroom.limsi.fr/atraco/InteractionOntology.owl

Cette ontologie décrit les concepts clés des modèles ergonomique et architectural. Ces concepts sont décrits en anglais, langue d'origine du projet de recherche. Par ailleurs, nous avons opéré à une **simplification de certaines notions** pour permettre une intégration plus simple dans le reste du projet ATRACO. Ainsi par exemple, la chaine d'interaction décomposée en un CM et un CL est représentée par un unique composant appelé Mediator qui fait le lien entre les CID (Off-the-shelf Interaction Object en anglais) et les périphériques d'interaction. Il s'agit du médiateur apparaissant dans la Figure 54.

De même, le lien de dépendance entre les différents composants (CID, CL, CM et Media) n'est pas détaillé par l'ensemble des interfaces qui décrivent cette dépendance mais par une simple relation binaire.

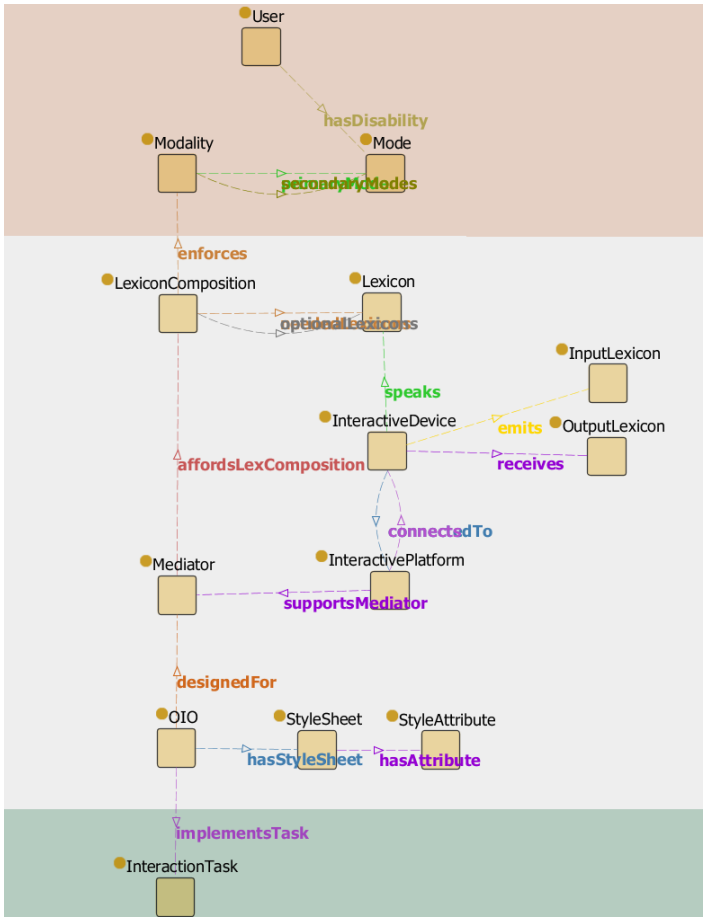


FIGURE 91 – CONCEPTS ET RELATIONS CLES DE L'ONTOLOGIE D'INTERACTION DANS ATRACO

La Figure 91 illustre les concepts clés utilisés dans l'ontologie d'interaction ATRACO. On peut séparer ces concepts en deux catégories (de haut en bas) : les concepts ergonomiques liés à la théorie des modalités et les concepts d'architecture. Le mécanisme de composition est simplifié par l'emploi d'un seul intermédiaire entre l'OIO (i.e. le CID) et les périphériques d'interaction.

L'ontologie plus complète comporte d'autres concepts qui permettent de décrire le contexte interactif et notamment les critères d'allocations (allocation dimensions, environmental conditions, user…), ce que l'on peut observer sur la Figure 92.

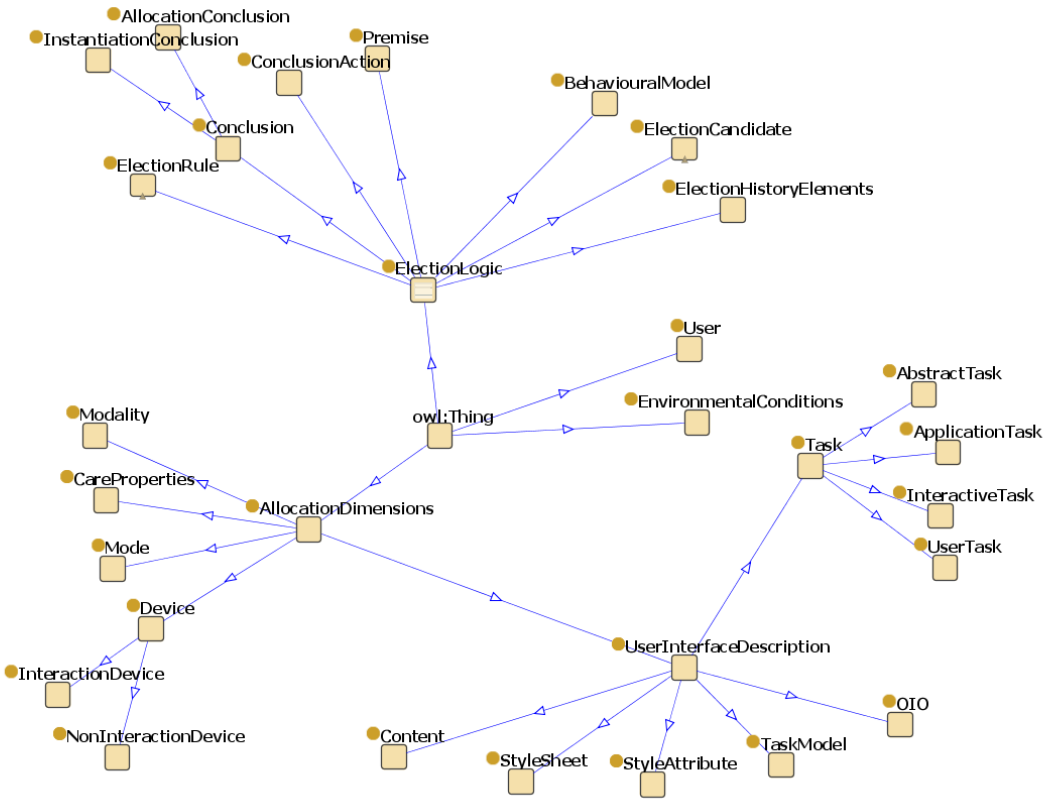


FIGURE 92 – CONCEPTS DE L'ONTOLOGIE D'INTERACTION DANS ATRACO

XI.2.2 MODALITES IMPLEMENTEES

Les interfaces proposées dans ATRACO permettent de contrôler l'ambiance d'une pièce (luminosité, ambiance musicale, gestion des cadres photos numériques) et de contrôler les effecteurs qui s'y trouvent (air conditionné, alarme, ouverture de la porte d'entrée…).

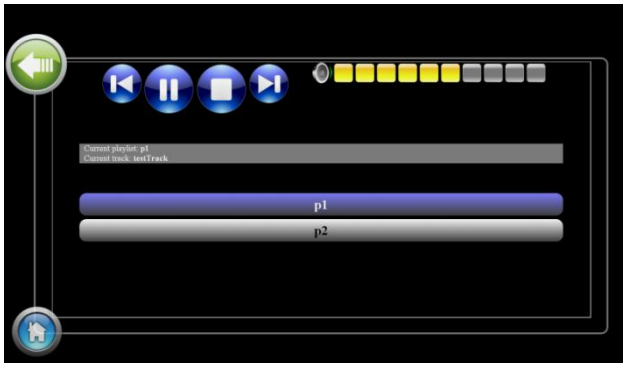


FIGURE 93 – INTERFACE DE CONTROLE GRAPHIQUE DE L'AMBIANCE MUSICALE

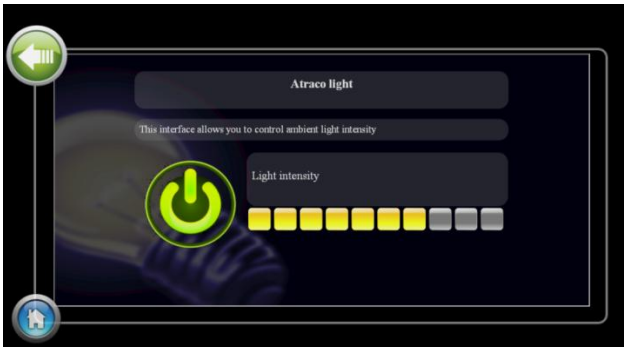


FIGURE 94 – INTERFACE DE CONTROLE GRAPHIQUE DE LA LUMINOSITE AMBIANTE

Chacune des interfaces a été implémentée selon deux langages d'interaction : le langage d'interaction WIMP reposant sur les modalités de sortie graphique et texte et le langage de commande vocale. Le détail des interfaces et des dialogues supportés est présenté dans

(Van Helvert, Hagras & Kameas 2010). Enfin, le critère principal utilisé pour sélectionner l'interface est la position des médias par rapport à l'utilisateur (Dooley et al. 2011).

XI.3 OUTILS DE SIMULATION

Nous avons développé trois outils qui permettent aux ergonomes/concepteurs d'ambiants de concevoir et tester un modèle comportemental sans avoir à en implémenter les composants. L'un de ces outils, appelé *Describe*, permet de décrire les instances des modèles ergonomiques et d'architecture. Le second outil, appelé *Behave*, permet d'éditer le modèle comportemental, c'est-à-dire les règles utilisées dans les critères électifs. Enfin, le troisième outil, appelé *Simulate*, est un simulateur permettant d'appliquer l'algorithme de composition automatique d'interfaces dans des conditions contrôlées pas le concepteur.

Nous présentons, dans cette section, ces outils qui nous ont permis de concevoir le prototype expérimental. Le lecteur intéressé pourra trouver plus de détails sur leur implémentation dans (Bellik & Pruvost 2010).

XI.3.1 DESCRIBE

Cet outil repose sur l'API de Protégé[36] pour fournir un éditeur simple des individus peuplant l'ontologie d'interaction. Autrement dit, cet outil permet de définir les instances des modèles d'architecture et ergonomiques.

En utilisant un éditeur générique (tel que Protégé), les concepteurs ont accès à tout le modèle originel et prennent donc le risque de l'altérer par erreur. L'outil Describe (cf. Figure 95) leur offre un cadre simplifié et sécurisé pour l'édition des instances décrivant un environnement. Cet outil garantit la conformité des instances ainsi créées.

XI.3.2 BEHAVE

Behave est un simple éditeur de règles (cf. Figure 96). Il permet l'édition de règles en mode texte, la vérification de la syntaxe de ces règles, ainsi que l'import/export de fichiers de règles dans le format de Clips (cf. annexe C). Les règles peuvent être changées dans le modèle comportemental en cours d'exécution dans le simulateur, l'éditeur est donc disponible comme une application standalone mais également à travers le menu *Simulation -> Behavioural Model -> Edit Ruleset* du simulateur.

[36] Protégé est un projet open-source d'editeur d'ontologies en Java. Voir sur http://protege.stanford.edu pour plus d'informations

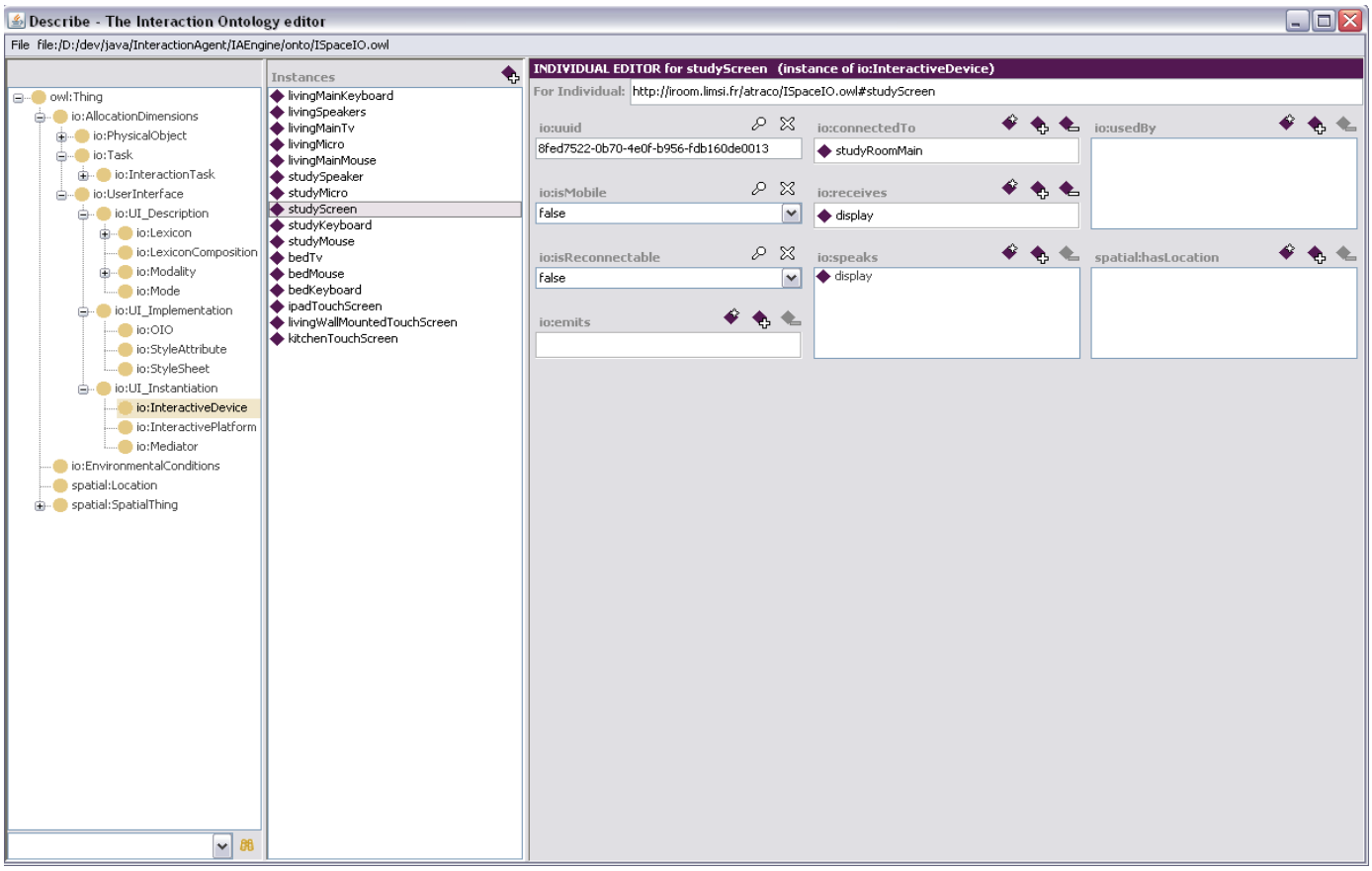


FIGURE 95 - CAPTURE D'ECRAN DE L'EDITEUR DE MODELE *DESCRIBE*

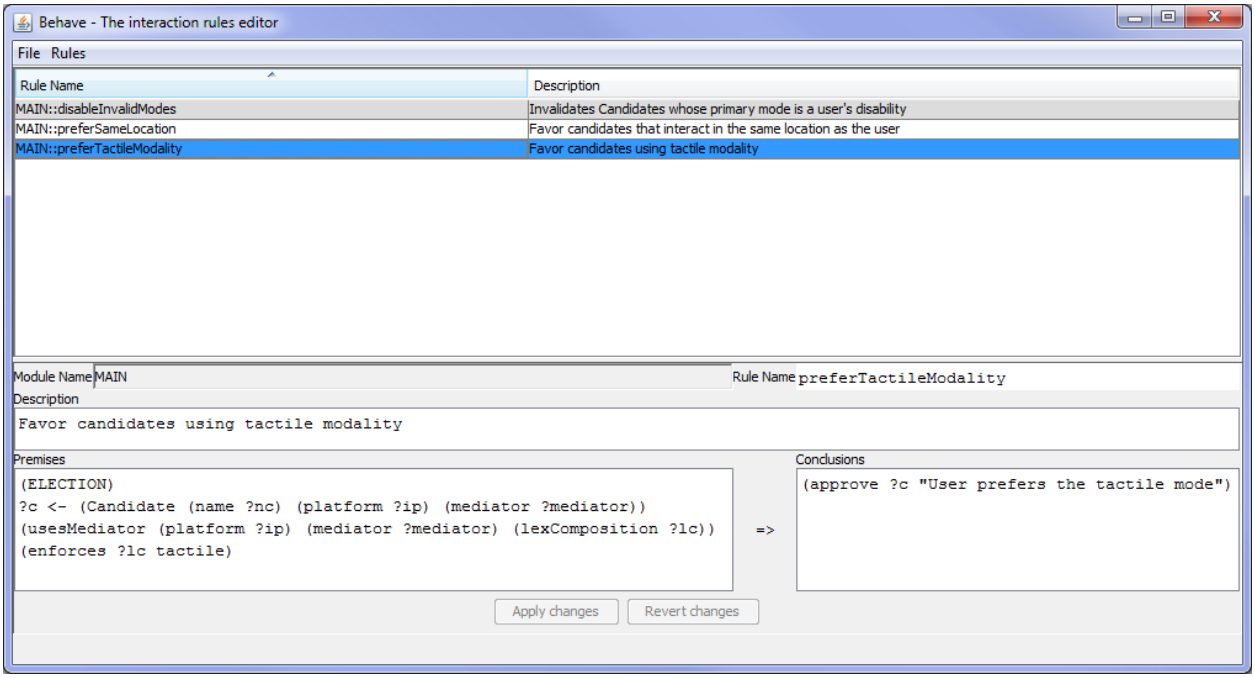


FIGURE 96 - CAPTURE D'ECRAN DE L'EDITEUR DE REGLES *BEHAVE*

XI.3.3 SIMULATE

Simulate est l'outil de simulation qui nous a permis de tester notre modèle comportemental dans différentes configurations. Nous analysons, dans un premier temps, les apports et inconvénients d'une étape de simulation dans la conception d'un système interactif ambiant. Puis nous présenterons l'outil développé.

XI.3.3.i INTERETS ET LIMITES DE LA SIMULATION DE MODELES COMPORTEMENTAUX

Ce simulateur instancie l'agent d'interaction et lui connecte une base de faits fictive. Au lieu de recevoir des informations produites à propos de l'environnement à travers le contrôleur d'ontologie, le système reçoit ainsi des informations contrôlées par le concepteur au moyen de l'interface graphique de *Simulate*. Cette approche permet de simuler des situations inédites, d'évaluer le modèle comportemental défini et de juger de l'utilité de certains composants avant même d'implémenter ces derniers. Le rôle de la simulation dans l'approche DAME est triple :

- Tester et déboguer le modèle comportemental

  L'environnement Ambiant étant ouvert, il est par nature non fiable et non reproductible. Nous entendons par là que certains composants peuvent tout simplement disparaitre et qu'un bug apparaissant à un moment peut être provoqué par des conditions et une suite d'évènements impossibles à reproduire. Cela rend la conception et le débogage du modèle comportemental complexe. Dans ce cas, la simulation offre un univers clos, contrôlé et reproductible, permettant au concepteur d'identifier les règles du modèle comportemental qui produisent un effet indésirable.

- Minimiser les coûts d'implémentation

  La pile d'interaction des systèmes d'exploitation actuels est monolithique. Notre approche, qui vise à la modulariser, nécessite la ré-implémentation de certains pilotes logiciels et une intervention dans les couches profondes des systèmes d'exploitation. Ces développement sont lourds et ont un coût non négligeable. La simulation permet dans ces cas d'envisager le comportement du système avec un composant non implémenté. Elle permet donc d'évaluer l'impact du manque de tel ou tel périphérique dans le système, et donc de définir les développements prioritaires.

- Explorer les usages de nouvelles techniques d'interaction sans les implémenter

  De façon similaire, l'implémentation de nouvelles techniques d'interaction peut parfois être difficile pour des raisons de coût, ou pour des raisons technologiques (matériel nécessaire mais non existant ou imprécis). Par exemple, il est à l'heure actuelle difficile d'obtenir un système de reconnaissance vocale fiable dans un environnement ouvert sans équiper l'utilisateur d'un micro. Dans ces situations, la simulation permet d'étudier l'usage qui pourrait être fait d'une technique d'interaction irréalisable en l'état actuel des connaissances.

Cependant, la simulation a également ses limites. Elle ne permet pas de couvrir toutes les situations possibles qui sont en nombre infini. Elles ne garantissent donc pas la validité du modèle comportemental de façon formelle mais seulement sur un ensemble de scénarios

bien définis. Par ailleurs, la reproductibilité des simulations est elle-même un biais. En effet, elle ne tient pas compte d'un ensemble de facteurs réels tels que le bruit des capteurs ou les situations de compétitions. Un scénario simulé pourra donc dériver du même scénario testé en conditions réelles. Enfin, la simulation ne permet pas de tester l'extraordinaire créativité des utilisateurs lorsqu'il s'agit de prendre le système en défaut. Les utilisateurs, que ce soit pour s'approprier la technologie ou par incompréhension, vont tester le système hors des limites prévues de son usage. Les usages non prévus qui en découlent ne peuvent pas être anticipés lors de la phase de simulation.

Par conséquent, nous restreignons l'utilisation de la simulation à une étape préliminaire de la phase d'analyse. Cette étape est incluse dans la phase d'analyse du cycle de développement. Elle permet de définir les composants qui doivent être implémentés, dans quelle priorité, ainsi que les règles de base du modèle comportemental. La simulation peut également servir lors de la phase d'évaluation du modèle comportemental, mais en complément d'une validation par des tests en conditions réelles.

XI.3.3.ii OUTIL DEVELOPPE

Le simulateur présenté dans la Figure 97 permet aux concepteurs de :

- Définir la géométrie du système (i.e. les différentes zones que le système de localisation peut reconnaitre).
- Définir la localisation de l'utilisateur
- Définir la localisation des Médias
- Modifier des valeurs des capteurs environnementaux (température, luminosité, niveau sonore…)
- Démarrer/Arrêter des tâches d'interaction
- Voir les solutions candidates construites par le système interactif
- Pour chaque candidat, observer le contenu de son évaluation (quelles règles ont été appliquées, quelles contributions elles apportent).
- Changer en continu la position des utilisateurs/médias et observer les évolutions de la liste des interfaces sélectionnées pour chaque tâche interactive.

Le cadre de gauche fournit une vision sous forme de liste des instances du modèle, il permet notamment de sélectionner une solution candidate et de voir l'association de médias correspondante surlignée dans la partie droite de l'écran. Cette partie est une visualisation graphique des positions des différents objets/zones géographiques de l'environnement. La partie de droite est constituée d'un second onglet qui permet d'afficher les détails de chaque candidat (cf. Figure 98). Cet onglet liste les règles qui ont été appliquées durant la phase d'élection des solutions candidates. (Bellik & Pruvost 2010) montre sur un exemple issu du prototype expérimental, comment le simulateur peut être utilisé pour analyser l'impact des règles sur le comportement du système interactif.

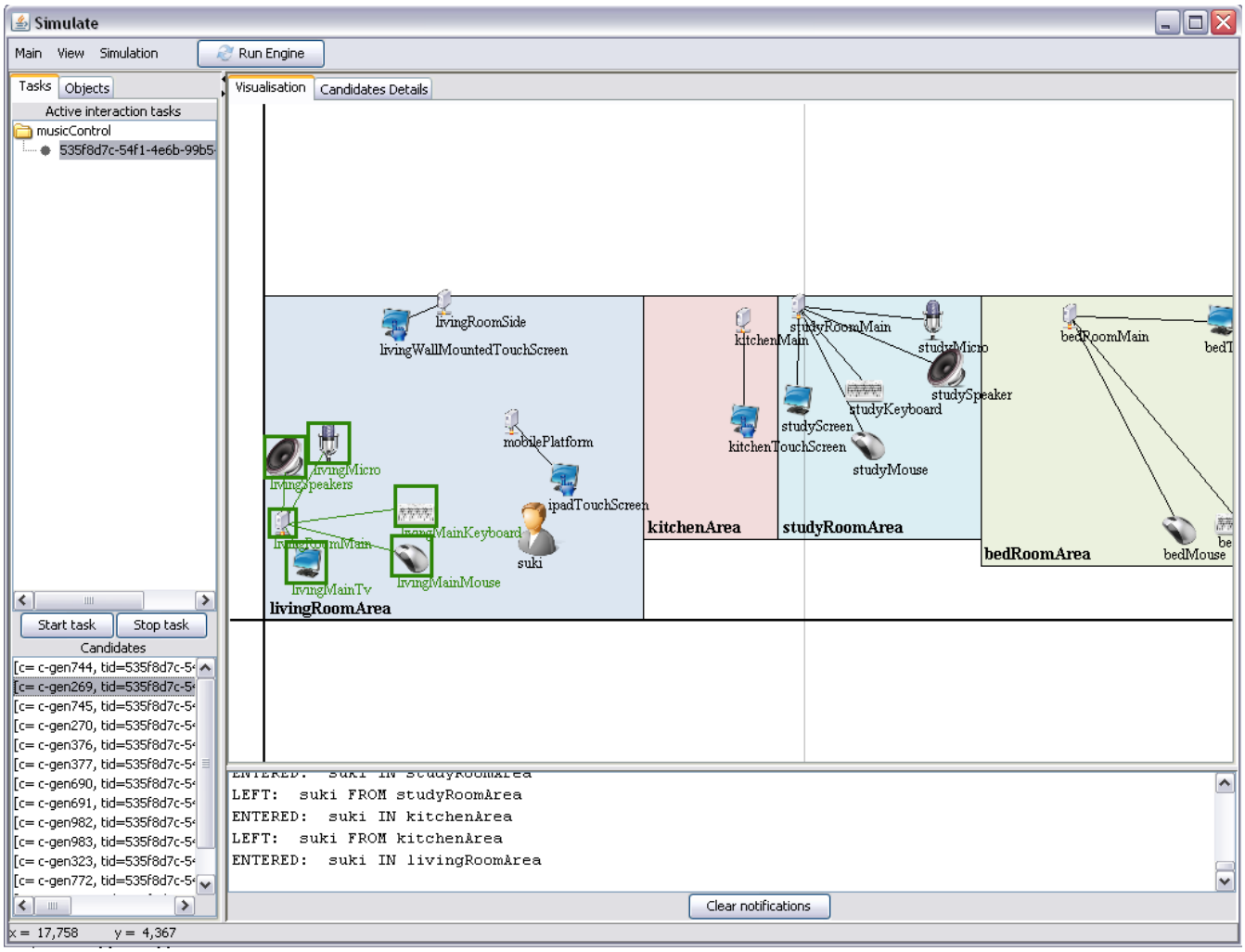


FIGURE 97 - CAPTURE D'ECRAN DU SIMULATEUR DE MODELE COMPORTEMENTAL *SIMULATE*

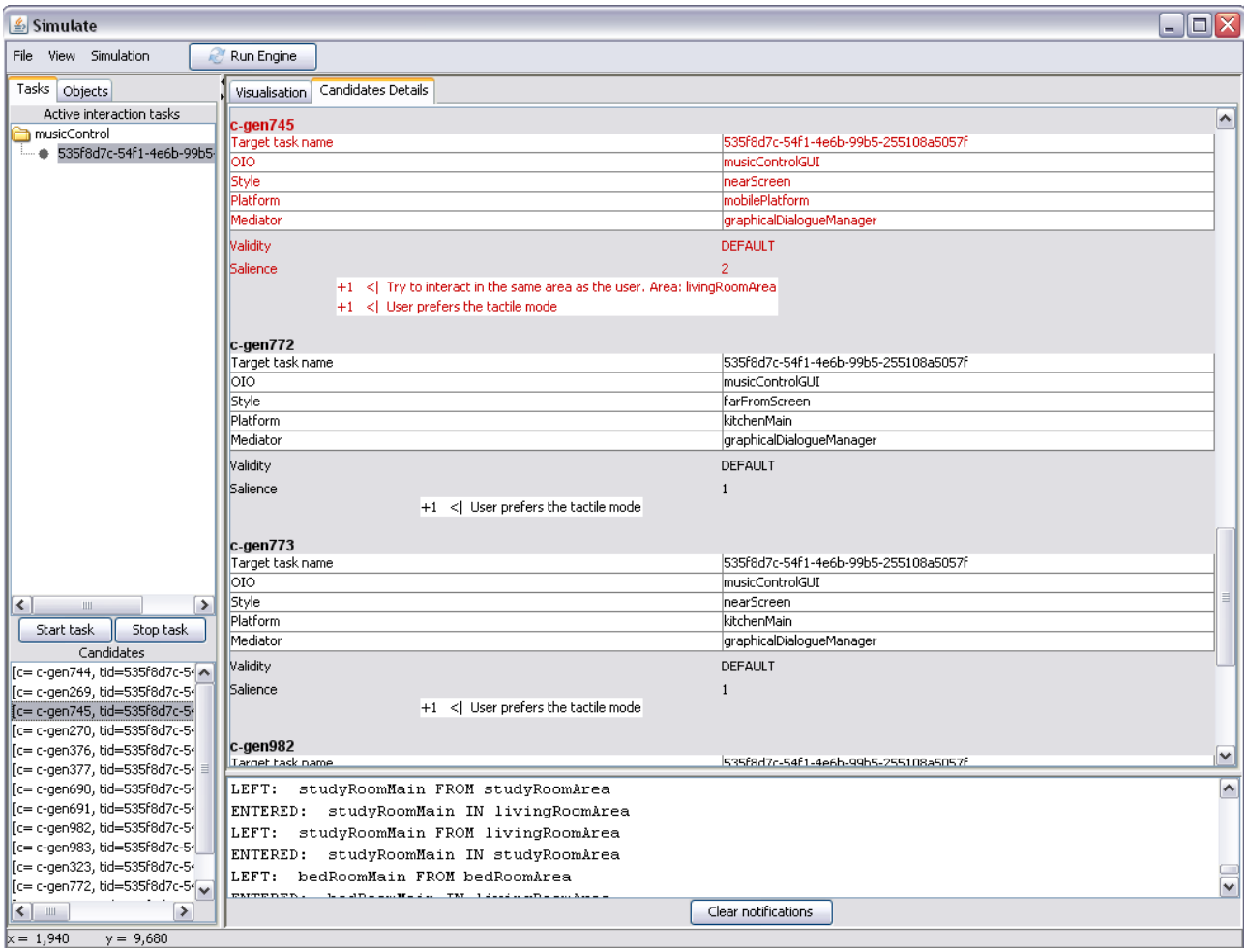


FIGURE 98 - CAPTURE D'ECRAN DE LA VISION DES SOLUTIONS CANDIDATES DANS *SIMULATE*

Grâce à ce simulateur, nous avons pu élaborer un modèle comportemental cohérent pour l'évaluation utilisateur décrite dans la section suivante.

XI.4 FORMAT DE L'EVALUATION UTILISATEUR

L'évaluation du projet ATRACO par des utilisateurs a été au centre du développement continu des prototypes. Cette section décrit l'évaluation du prototype final et les résultats obtenus concernant le système interactif ambiant.

Cette évaluation a été menée par l'équipe de l'université d'Essex en collaboration avec les autres membres du projet. Nous avons en particulier contribué à la définition du protocole expérimental ainsi qu'à l'implémentation du prototype utilisé. Nous développons dans cette section les points sur lesquels nous avons contribué, notamment dans la définition du scénario et le développement de l'ISpace. Les détails de cette étude (composition de l'échantillon, résultats concernant d'autres composantes du projet ATRACO) sont disponibles dans (Van Helvert, Hagras & Kameas 2010). Ce document fournit notamment la liste précise des questionnaires employés ainsi qu'une analyse sociologique des participants à l'expérience (âge, sexe, connaissances en informatique…).

XI.4.1 ISPACE : LA PLATEFORME EXPERIMENTALE

L'ISpace est un appartement équipé de capteurs et d'effecteurs permettant de surveiller l'évolution du contexte et d'agir sur ce dernier pour répondre aux besoins de l'utilisateur. Ces capteurs sont pour la plupart dissimulés dans les murs ou dans des faux plafonds (cf. Figure 99 et Figure 100).

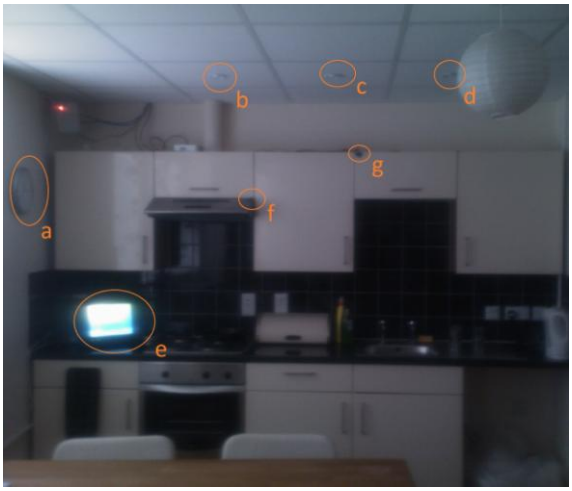


FIGURE 99 - CAPTEURS ET EFFECTEURS MIS EN EVIDENCE DANS LA CUISINE DE L'ISPACE (EXTRAIT DE (WAGNER, ET AL. 2010 – D14))

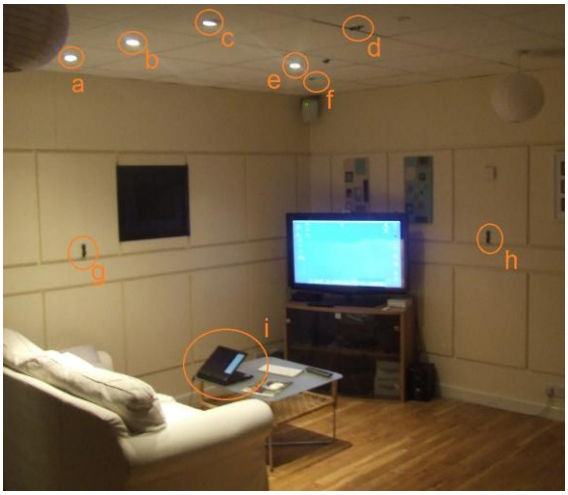


FIGURE 100 – CAPTEURS ET EFFECTEURS MIS EN EVIDENCE DANS LE SALON DE L'ISPACE (EXTRAIT DE (WAGNER, ET AL. 2010 – D15))

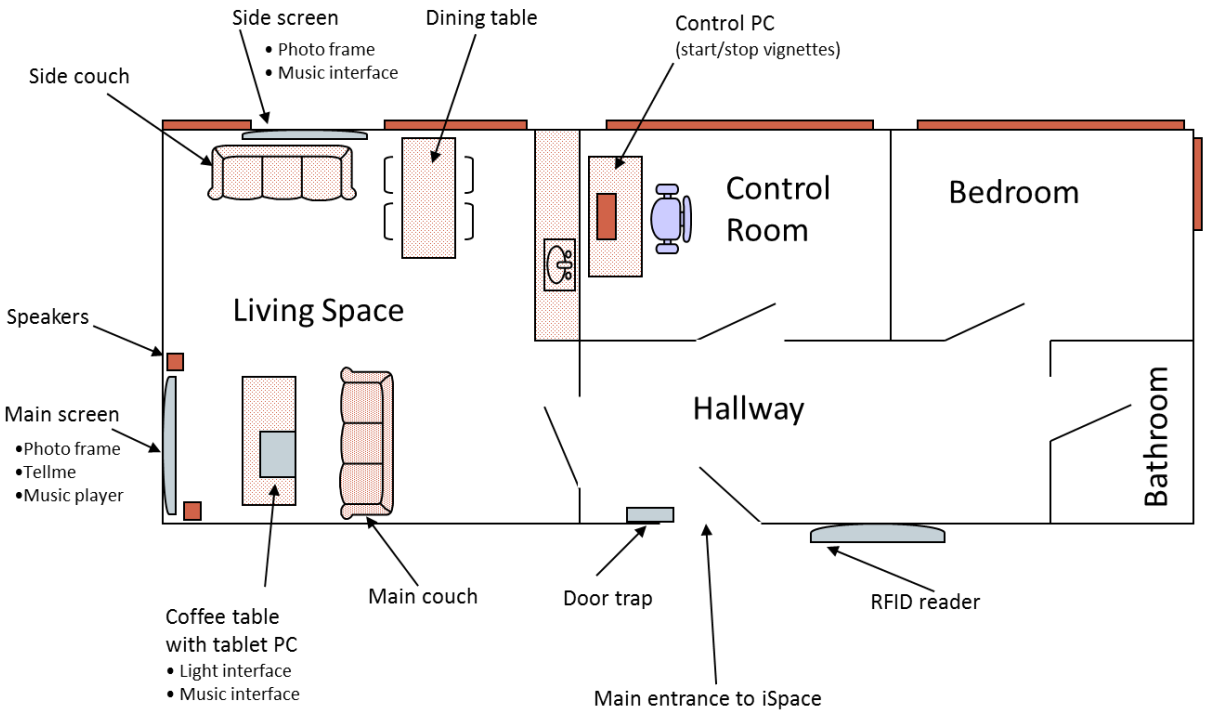


FIGURE 101 - PLAN DE L'ISPACE

La Figure 101 présente le plan de l'ISpace ainsi que la position des meubles et des périphériques (capteurs, effecteurs et périphériques d'interaction) les plus importants. Etant donné le nombre important de périphériques (un peu moins d'une centaine), nous ne les avons pas tous fait apparaitre sur le schéma.

Les périphériques et services les plus utilisés lors du scénario expérimental sont :

- Le système de sécurité de l'appartement : lecteur RFID, verrou magnétique de la porte, capteur d'ouverture des fenêtres, alarme.
- Le système de contrôle de la température : capteurs de température et modules d'air conditionné pilotables à distance.
- Le système de contrôle de l'ambiance lumineuse : capteurs de luminosité, lumières
- Le système multimédia : des cadres photos pilotables à distance, des services de rendu multimédia (sonore en particulier).
- Le système interactif : une tablette tactile, un ordinateur portable, un grand écran tactile, un grand écran non tactile, une souris et un clavier sans-fil.

XI.4.2 FORMAT DE L'ETUDE AUPRES D'UTILISATEURS FINAUX

L'évaluation d'un système ambiant peut difficilement se faire sur le plan quantitatif. En effet, une étude quantitative implique d'identifier une ou deux variables expérimentales à évaluer et à s'assurer que les autres variables soient bien contrôlées pour fournir des conditions expérimentales reproductibles. Ces contraintes sont irréalisables dans l'Ambiant pour plusieurs raisons :

- Les développements dans ce domaine sont récents et il est difficile de formuler des questions précises relatives à l'ergonomie, aux performances ou à l'acceptabilité d'une solution ambiante par rapport à une autre.
- Les conditions expérimentales d'un système ambiant font intervenir un nombre important de paramètres. Il est donc illusoire de tous les contrôler ou même de pouvoir les reproduire.
- Il existe trop peu d'éléments de comparaison (i.e. d'autres système ambiants facilement instanciables) permettant de donner un sens aux chiffres obtenus par une étude quantitative.
- Pour obtenir des résultats statistiquement significatifs, une analyse numérique requiert le passage d'un nombre conséquent de sujets dans chaque condition expérimentale. Or puisque l'Ambiant est un monde ouvert, le nombre important de conditions expérimentales différentes peut s'avérer une barrière à la faisabilité d'une telle étude. Par ailleurs, l'Ambiant étant un contexte d'usage nouveau pour les utilisateurs, il faut tenir compte des effets d'apprentissage, ce qui augmente le temps de passage de chaque sujet.

L'étude quantitative des utilisateurs dans les systèmes ambiants est pour ces raisons, non reproductible à l'heure actuelle, et donc vaine. L'émergence de nouvelles méthodologies est nécessaire. Il existe d'autres domaines dans lesquelles le contexte est difficilement

contrôlable (études épidémiologiques en médecine, études en psychologie…) et qui pourraient donner des pistes vers une méthodologie de l'évaluation des systèmes ambiants. Cependant, ce travail ne sera pas abordé dans cette thèse ni dans le projet européen ATRACO.

Nous avons donc fait le choix d'une étude qualitative de l'acceptation du système par les utilisateurs. En nous basant sur les méthodes de conception participative et orientée utilisateur, nous avons établi un protocole mélangeant des phases d'interaction avec le système et des phases d'entretien semi-guidés et/ou libres. Le protocole se divise en 2 phases. Le déroulement de chacune de ces phases est relativement long pour chaque sujet (notamment à cause des entretiens qui suivent les phases d'interaction), ce qui justifie un échantillon expérimental réduit. Dans notre cas, 9 sujets ont pris part à l'étude. Pour plus de détails sur la constitution de l'échantillon, le lecteur pourra se référer à (Van Helvert, Hagras & Kameas 2010).

XI.4.2.i Phase 1 : Sessions d'interaction dans l'ISpace

Au début de chaque session, le participant est amené à remplir un questionnaire permettant de le caractériser vis-à-vis du reste de l'échantillon. Ce questionnaire permet de rassembler des données démographiques (âge, sexe, profession…) et de situer l'utilisateur vis-à-vis des systèmes informatique (expérience/aisance en informatique, comportements vis-à-vis des nouvelles technologies…).

A l'issue de ce questionnaire, chaque participant a été invité à interagir avec le système à 5 reprises. Chacune de ces interactions (appelées vignettes) vise à illustrer le fonctionnement du système et son impact sur les activités de la vie courante. Le participant est mis en situation par l'évaluateur : la situation initiale lui est précisée et il lui est demandé d'imaginer que l'ISpace est son propre domicile et que l'activité illustrée par la vignette est une activité qu'il réalise quotidiennement. Une fois la vignette activée, le participant peut interagir à sa guise, sans intervention de l'évaluateur.

A l'issue de chaque vignette, un bref entretien semi-guidé est mené pour connaitre le ressenti de l'utilisateur sur ce qu'il a perçu du système (en tant que fonctionnalités) et sur ce qu'il a apprécié ou sur ce qui l'a perturbé (sur le plan cognitif ou émotionnel). Pour chaque interview, l'utilisateur peut s'exprimer librement, l'évaluateur s'assurant simplement qu'un certain nombre de points prédéterminés sont bien abordés. Le détail de chaque vignette, des questionnaires et des guides pour les entretiens sont disponibles dans (Van Helvert, Hagras & Kameas 2010).

Les thématiques abordées sur le plan de l'interaction homme-machine dans chaque vignette sont détaillées dans le Tableau 12.

Vignette	Description	Thématique abordée sur le plan interactif
Arrivée à la maison	L'utilisateur accède à l'ISpace via la porte magnétique. A son entrée, le système l'accueille et lui propose des interfaces permettant de contrôler l'ambiance de l'appartement (luminosité, musique, photos défilant dans les cadres).	Identification de l'utilisateur Action à l'initiative du système Contrôle continu de l'environnement grâce à la composition automatique des interfaces.
Invités et vie privée	L'utilisateur regarde des photos dont certaines sont à caractère privé. L'évaluateur joue le rôle d'un visiteur et se présente à la porte. Après accord par l'utilisateur, le système ouvre la porte et cache les photos privées.	Politique de protection de la vie privée.
Gestion des pannes	L'utilisateur utilise l'un des services multimédias de la pièce (musique, cadre photo). Au bout d'un certain temps, l'évaluateur, caché dans une autre pièce (technique de magicien d'Oz) simule la défaillance du service en question grâce à son interface de contrôle.	Résilience à l'échec/disparition d'un service par reconnexion dynamique. => Adaptation aux évolutions du noyau fonctionnel.
Migration des services	La position de l'utilisateur est obtenue grâce à un système de localisation. L'utilisateur est invité à interagir avec son environnement en se déplaçant dans l'appartement. Les interfaces migrent pour suivre ses déplacements.	Contrôle continu de l'environnement grâce à la composition automatique des interfaces Migration des interfaces pour suivre l'utilisateur => Adaptation aux évolutions du contexte environnemental (positions relatives de l'utilisateur et des media).
Adaptation aux préférences utilisateurs	L'utilisateur vaque à ses occupations lorsque le système simule un panne. Sur le signal d'un compère, l'une des lumières tombe en panne et s'éteind. Le système tente de rétablir la luminosité en augmentant l'intensité des autres lampes. L'utilisateur peut à tout moment contrôler cette adaptation en utilisant l'IHM.	Adaptation du système Sentiment de contrôle de l'utilisateur en cas de tentative d'adaptation automatique du système.

TABLEAU 12 – DESCRIPTION DES VIGNETTES DE L'EVALUATION PAR LES UTILISATEURS

XI.4.2.ii PHASE 2 : SESSION DE REFLEXION LIBRE ENTRE LES PARTICIPANTS

Plusieurs jours après les sessions d'interaction, les participants de l'étude sont invités à prendre part à une discussion d'une heure environ autour des impacts sociaux de l'ambiant. Ils sont invités à réagir sur un scénario futuriste inspiré des technologies testées dans l'ISpace. Cette phase était destinée à laisser aux utilisateurs la possibilité d'exprimer, après réflexion, des propositions d'évolution du système ainsi qu'à formuler leurs craintes vis-à-vis de la place que peut prendre un tel système dans leur vie de tous les jours. L'entretien est organisé sous la forme d'un brainstorming dans lequel les participants sont libres d'intervenir.

XI.4.3 RESULTATS DE L'EVALUATION

L'analyse des entretiens issus des phases 1 et 2 a permis de dégager un certain nombre d'attentes et de craintes de la part des utilisateurs vis-à-vis de l'ambiant. L'ensemble de ces conclusions est disponible dans (Van Helvert, Hagras & Kameas 2010). Nous nous focaliserons dans cette section sur les conclusions apportées qui concerne le système interactif ambiant. En plus de ces résultats sur les facteurs humains, cette étude nous a permis d'obtenir un retour d'expérience sur l'usage, en conditions réelles, de l'approche DAME et de l'algorithme d'adaptation automatique d'interfaces associé.

XI.4.3.i RETOUR D'EXPERIENCE EN SCIENCES HUMAINES

La conclusion principale de cette étude sur le plan des facteurs humains est que l'acceptation d'un tel système dépend fortement du sentiment de contrôle et de présence que fournit le système interactif.

Le sentiment de contrôle est la capacité ressentie par l'utilisateur à pouvoir décider du comportement de son environnement. Dans les situations où l'utilisateur s'est senti dépossédé de ce contrôle sur le système, les réactions furent fortement négatives ; les utilisateurs se sentant rapidement sous l'influence d'un « big brother ». La frontière entre le sentiment d'avoir le système sous contrôle ou non est cependant subtile et dépendante du contexte. Dans certaines situations, il apparait normal que le système initie l'interaction (par exemple pour avertir de l'arrivée d'un invité dans l'appartement) alors que pour d'autre, cela est unanimement considéré comme intrusif (par exemple lorsque le système accueille l'utilisateur à son arrivée, dans la vignette 1, en lui proposant d'écouter un peu de musique). Il semble que le canal d'interaction joue un rôle également important dans ce sens, les interactions initiées par la voix étant perçues comme plus intrusives que les interactions graphiques.

Le sentiment de présence sociale est la capacité du système à adapter son comportement au point de donner l'illusion d'une autonomie. Il est montré dans (Garrison, Cleveland-Innes & Fung 2010) que de tels comportements ont un impact positif sur la charge cognitive de l'utilisateur. D'après les réactions des participants lors des interviews, il semble que l'interaction vocale ait joué un rôle plus important que l'interaction graphique dans le sentiment de présence du système. Ce résultat est à tempérer en tenant compte du fait

que seule la modalité iconique était implémentée comme forme d'interaction graphique. L'usage d'un avatar, par exemple, a été écarté de cette étude. Nous avons approfondi cette question du sentiment de présence éveillé par un avatar dans l'ambiant lors d'une étude menée au LIMSI (Tan et al. 2011). Cette étude a notamment démontré que la capacité du système à percevoir et à réagir à son environnement, notamment sur le plan spatial, augmentait ce sentiment de présence.

Enfin, il a été suggéré à l'occasion de la phase 2, de tenir compte de l'état affectif de l'utilisateur pour adapter l'IHM. Les utilisateurs ont exprimé le besoin d'avoir un système qui s'adapte à leur humeur et dont les préférences ne soient pas figées et identiques de jour en jour. Ils proposent l'ajout d'une interface permettant de contrôler entre différents stéréotypes comportementaux du système. Ce type de systèmes interactifs relève de l'informatique affective (Affective computing).

XI.4.3.ii RETOUR D'EXPERIENCE TECHNIQUE

Sur le plan technique, l'implémentation du prototype de cette étude a tout d'abord constitué une preuve de faisabilité. Elle a démontré qu'il était possible de mélanger des techniques issues de l'ingénierie de la connaissance[37] avec une conception modulaire de l'IHM dans un système qui adapte la structure logicielle de l'interface au contexte. D'un point de vue plus conceptuel, cette expérience a permis de mettre à l'épreuve notre vision d'un système interactif composé de 3 modèles (un modèle ergonomique, un modèle architectural et un modèle comportemental) qui gouvernent l'association de composants créant une interface.

Nous avons atteint un double objectif en réalisant ce prototype :

- Permettre la conception d'un système interactif adaptatif qui ne restreigne pas la liberté du concepteur. Par opposition aux systèmes de génération automatique de l'ingénierie des modèles, qui reposent sur une description prédéfinie des éléments d'interface, notre approche a permis de concevoir des CID reposant sur une interface librement conçue (à partir de boites à outils prédéfinies ou bien « faites-main »). Bien que ces interfaces soient librement conçues, leur découpage en CID de granularité, de style et de complexité différents permet une reconfiguration qui permet un certain degré d'adaptabilité.
- Exprimer des modèles comportementaux basiques à partir de règles s'appuyant sur les concepts issus des modèles ergonomiques et d'architecture. Les règles ont permis de décrire un ensemble varié de situations et le comportement à suivre en conséquence.

Cette expérience a également montré les limites techniques de notre approche. La principale de ces limites concerne la flexibilité de la recomposition des interfaces. Celle-ci dépend fortement du nombre et de la variété des CID qui sont disponibles pour remplir un besoin utilisateur donné. Cette nécessité d'atteindre un seuil de CID minimum pour permettre l'adaptation au contexte est la contrepartie de la liberté offerte aux concepteurs.

[37] Certains parleraient d'intelligence artificielle, terme plus souvent employé bien que souvent galvaudé.

Dans la conclusion qui suit, nous comparons plus en détails nos résultats avec les autres approches pour l'IHM dans l'Ambiant et plus particulièrement vis-à-vis de l'ingénierie des modèles. Nous en déduirons également des perspectives d'évolution qui sortent du cadre de cette thèse.

CONCLUSION DE LA PARTIE C

Dans cette partie, nous avons proposé une implémentation de l'approche DAME. Cette implémentation repose sur l'usage conjoint de trois technologies :

- Les ontologies
- Les systèmes experts
- Les conteneurs de composants

Nous avons illustré, dans le chapitre X, le rôle de chacune de ces technologies dans la réalisation de l'approche que nous avons présentée dans la partie B. Nous avons également montré comment ces technologies pouvaient être combinées (avec les mécanismes de traduction que cela comporte). Ce chapitre nous démontre la faisabilité de l'approche.

Dans le chapitre XI, nous avons intégré cette implémentation à une architecture existante. Nous avons développé, sur la base de cette implémentation, des outils d'aide à la spécification des modèles ergonomique, architectural et comportemental. Nous avons de plus, conçu un simulateur qui permet de tester les modèles comportementaux dans des conditions parfaitement contrôlées. Cet outil de simulation permet d'explorer la richesse de l'approche DAME. Il a servi à concevoir une expérience avec des utilisateurs dans le cadre du projet ATRACO. Cet expérience a conclu qualitativement sur l'acceptation par l'utilisateur, d'un système d'interaction adaptatif tel que DAME dans l'Ambiant. Les retours des utilisateurs sont majoritairement positifs et soulignent le besoin qu'ont les utilisateurs de se sentir en « contrôle » de la situation et d'anticiper le comportement du système. Il s'agit là d'une recommandation importante qui aurait pu être remise en cause par la nature automatisée de l'adaptation. Il n'en est cependant rien et nous encourageons donc les futurs concepteurs à développer des modèles comportementaux qui suivent une logique accessible à l'utilisateur.

CONCLUSION ET PERSPECTIVES

Dans cette thèse, nous avons présenté notre contribution aux recherches sur l'interaction homme-machine dans l'informatique ambiante. Après une étude de l'état de l'art qui nous a permis de mettre en évidence certains problèmes importants propres aux interfaces multimodales ambiantes, nous avons conçu une démarche de conception d'IHM ambiantes. Cette démarche vise à laisser aux concepteurs et aux développeurs la liberté d'implémenter des techniques d'interaction qui sortent d'un cadre prédéfini, tout en permettant la réutilisabilité de ces techniques.

Nous avons pour cela conçu et développé trois modèles complémentaires qui permettent de décrire différentes facettes de ces techniques d'interaction (à savoir, leurs caractéristiques logicielles, ergonomiques et leur comportement au regard du contexte). Ces modèles ont donné lieu à l'implémentation d'une plateforme d'interaction ambiante ainsi qu'à des outils de spécification et de simulation. Nous avons pu appliquer ces développements à la réalisation d'un prototype expérimental opérationnel dans lequel le système compose et sélectionne les interfaces pertinentes en fonction du contexte d'interaction.

Ces travaux constituent un premier pas vers une interaction homme-machine riche, flexible et adaptative dans les environnements ambiants. Nous en résumons les apports, dans un premier temps, en comparant notre approche aux systèmes existants. Puis nous explorons les principaux axes futurs vers lesquels nous pensons que les recherches sur l'IHM ambiante devraient être orientées.

CONTRIBUTIONS

Nous avons proposé, à l'issue de la partie A, une analyse comparative des travaux existants pour la génération d'IHM ambiantes (cf., page IV.5). Les tableaux 13, 14 et 15 (cf. page suivante) proposent de confronter notre approche DAME par rapport aux axes de cette analyse.

Cette description nous permet d'identifier les trois contributions principales de notre approche :

- Une représentation explicite des médias d'entrée **et** des médias de sortie

  Les approches de génération automatique par IDM décrivent la plateforme cible en termes de langages d'interaction qu'elle peut implémenter, sans rentrer dans le détail des médias qui la composent. A contrario, les approches multi-médias modélisent les médias en entrée mais les médias de sortie sont cachés et fortement couplés aux représentations du noyau fonctionnel.

  Dans notre approche, les médias d'entrée et de sortie sont explicitement représentés. Ils sont découplés du langage d'interaction qu'ils permettent d'instancier par un composant intermédiaire (le contrôleur de médias).

  Cette approche permet de générer des IHM adaptées à des plateformes variées obtenues par couplage opportuniste des médias disponibles.

- Une démarche adaptative **personnalisable**

  L'architecture logicielle proposée dans DAME est tout d'abord **adaptable**. Son modèle de composants lui confère en effet flexibilité et modularité, ce qui permet de concevoir automatiquement plusieurs solutions pour un même besoin d'IHM. Le modèle comportemental rend de plus le système **adaptatif** en automatisant l'application de recommandations qui couvrent les domaines de l'architecture logicielle et de l'ergonomie. Enfin, ces recommandations elles-mêmes sont **personnalisables**. Elles sont définies par les acteurs du système ambiant et peuvent donc varier d'un environnement ambiant à un autre.

- Une démarche **extensible**

  Le découplage entre l'implémentation d'un composant et son interface permet de favoriser la réutilisation des composants. De plus, l'architecture que nous proposons divise l'implémentation d'un dialogue homme-machine en quatre couches qui représentent chacune un des aspects de l'IHM. **Cette architecture guide le concepteur dans l'identification de structures communes et réutilisables d'une technique d'interaction à l'autre.**

Abstraction du noyau fonctionnel	Ensemble d'interfaces fournies/requises
Assemblage de l'interface	Agrégation de CID
Gestion des Entrées	Explicite
Gestion des Sorties	Abstraction des structures de données des entrées/sorties au sein du même composant : le contrôleur de médias

TABLEAU 13 – STRUCTURE DES IHM DANS DAME

Dynamicité	Adaptatif intra-session
Cible	Noyau fonctionnel + Plateforme + Environnement + Utilisateur
Stratégie	Règles
Identification des situations à adapter	Pas d'observateurs : Chaque évolution du système est susceptible de remettre en cause l'interface générée

TABLEAU 14 – CAPACITES D'ADAPTATION DE DAME

Réutilisabilité des composants	Totale
Extension de la plateforme	Ajout de composants et modèles qui les décrivent Ajout de critères Modèles du contexte extensibles (ontologies OWL)
Profilage du comportement automatisé	Critères

TABLEAU 15 – CAPACITES D'EXTENSION DE DAME

PERSPECTIVES

La démarche DAME apporte un support et une factorisation des techniques d'interaction innovantes qui la démarque des approches génératrices issues de l'IDM. Elle se distingue de celles-ci par l'utilisation d'IHM préconçues réalisant un dialogue interactif spécifique (les CID). La conception manuelle de ces dialogues est à la fois son point fort et sa principale limite. Nous proposons, dans un premier temps, deux perspectives de recherche qui permettront de réduire l'impact de cette limite. Nous explorons ensuite trois perspectives plus générales concernant les IHM ambiantes.

DECOUPLAGE IHM – NOYAU FONCTIONNEL

La réalisation automatique d'un lien entre les évènements d'une IHM et les fonctionnalités du NF (Noyau Fonctionnel) est un problème encore ouvert. Dans DAME, nous échappons à ce problème par l'utilisation des interfaces entre composants, qui garantit l'adéquation entre IHM et NF.

Bien que ces interfaces permettent un découplage entre IHM et NF, une certaine dépendance subsiste. En effet, pour réaliser des interfaces réutilisables, notre approche suppose que les interfaces du NF puissent être standardisées. Cette approche est cohérente avec les efforts de standardisation faits par les consortiums qui développent les services réseaux (l'UPnP Forum[38] en est un bon exemple).

Cependant, cette vision est ébranlée par la volonté des différents fabricants de se démarquer les uns des autres en proposant chacun des fonctionnalités différentes. Il existe donc un réel besoin pour la génération automatique d'alignements (ou mappings) entre les fonctionnalités exprimées par l'IHM et celles exprimées par le NF. Un premier pas vers la réalisation de ces alignements est fait dans les approches d'IDM qui intègrent un modèle du domaine de la tâche. Les travaux explorant cet axe de recherche devraient permettre de rendre notre approche plus flexible.

COMBINAISON DE PLUSIEURS CID

Dans notre approche, nous générons des IHM par la composition de plusieurs CID. Par composition, nous entendons ici que les fonctionnalités offertes par les CID se complètent, mais leurs représentations ne sont pas assemblées en une IHM cohérente. En effet, les CID qui composent un candidat sont instanciés côte-à-côte, sans assemblage de leurs manifestations respectives.

L'assemblage de plusieurs composants graphiques a été étudié dans (Gabillon, Calvary & Fiorino 2008) et (Gabillon et al. 2011). Cependant, l'automatisation de ces assemblages dans le cadre des autres modalités demeure un problème ouvert. Il devient particulièrement complexe lorsque certaines des modalités sont partagées entre deux composants qui reposent sur des langages d'interaction différents.

Nous pensons que les résultats issus de ce domaine de recherche permettraient, dans l'approche DAME, de renforcer la cohérence des rendus (graphiques, audio ou autres) générés par le système adaptatif.

DISTRIBUTION DE LA PRISE DE DECISION

L'algorithme que nous avons présenté repose sur une méthode de raisonnement centralisée : les informations provenant de divers périphériques distribués dans

[38] L'UPnP Forum est un consortium rassemblant plusieurs constructeurs autour de la définition d'interfaces standardisées pour les matériels utilisant le protocole de communication UPnP.

http://www.upnp.org/

l'environnement sont centralisées dans une base de faits unique sur laquelle sont réalisés les raisonnements qui permettent de générer une IHM.

Or, les limites physiques d'un environnement ambiant sont difficiles à définir car elles varient en fonction des utilisateurs et des usages. La représentation de ce système par un acteur unique est donc trop simplificatrice : les environnements ambiants sont par nature distribués. Par conséquent, ils requièrent des mécanismes de contrôle qui suivent la même distribution.

Le choix réalisé de façon centralisée par l'algorithme que nous avons proposé devrait donc être réalisé de façon distribuée par une communauté d'agents établie dynamiquement. L'application de recherches en collaboration multi-agents et en raisonnement distribué, telles que celles de (Scafes & Badica 2009), pourraient permettre une approche innovante de la prise de décision automatique dans l'Ambiant.

CONTROLE DU COUPLAGE PAR L'UTILISATEUR

A l'issue de l'expérience utilisateur présentée dans le chapitre XI, nous concluons sur la nécessité de laisser à l'utilisateur le sentiment de contrôle sur le système. De nouvelles études devraient être produites pour délimiter les contours de ce qui est automatisable par le système et des variables sur lesquelles l'utilisateur à besoin de conserver le contrôle.

Nous proposons pour cela de rendre l'utilisateur maître de la composition de médias qui est instanciée. Cette maîtrise par l'utilisateur requiert une IHM dédiée au couplage des médias telle que celles proposées dans (Hinckley 2003). Cette IHM dédiée au contrôle de l'IHM est parfois appelée meta-IHM (Balme et al. 2004). Un contrôle du couplage semi-automatique (i.e. réalisé par le système mais modifiable par l'utilisateur) est également une piste intéressante pour permettre au système adaptatif de ne pas donner à l'utilisateur le sentiment d'avoir perdu le contrôle sur le système.

EVALUATION AVEC DES CONCEPTEURS

Enfin, ces travaux concernent l'ensemble du cycle de développement, depuis le concepteur jusqu'à l'utilisateur final. Lors de nos recherches, nous avons pu constater que l'essentiel des travaux d'évaluation se portent sur les utilisateurs finaux. Or, lorsqu'il s'agit de concevoir une architecture pour un système interactif, le système interactif produit en dit bien peu sur l'architecture sous-jacente. Par conséquent, il nous semble que l'évaluation par les utilisateurs finaux devraient s'accompagner d'évaluations auprès des utilisateurs réels d'une plateforme telle que DAME : les concepteurs de systèmes interactifs.

De telles études sont complexes à mener, elles requierent une certaine maturité technologique de la plateforme à évaluer et un engagement sur le long terme des concepteurs qui participent à l'expérience. De plus, le métier de concepteur de systèmes ambiants n'existe pas encore et les règles de conceptions, dans cette discipline naissante, ne font pas consensus. A l'issue de ces travaux de thèse, nous estimons avoir franchi le premier de ces obstacles, celui de la maturation technologique. Nous espérons donc à plus long terme, pouvoir partager notre approche avec des concepteurs de systèmes ambiants pour accumuler un retour d'expérience permettant de mieux adapter notre approche aux besoins du domaine.

CONCLUSION

Après la révolution de l'informatique mobile, l'informatique ambiante se profile comme un nouveau changement de paradigme. Sa nature distribuée remet en cause la nature même des logiciels, et plus particulièrement, de leur interface homme-machine.

Dans cette thèse, nous avons présenté notre apport aux recherches actuelles dans le domaine de l'interaction ambiante et nous venons de présenter quelques axes futurs qu'il nous paraît important de suivre. En réalisant un prototype opérationnel, nous avons pu confronter théorie et pratique et valider la démarche proposée.

Nous espérons que les travaux présentés dans cette thèse apporteront une pierre supplémentaire dans la construction de nos environnements informatiques de demain.

ANNEXES

ANNEXE A – FORMALISATION DES DIAGRAMMES OMT EN LOGIQUE DU PREMIER ORDRE

Notre approche de modélisation repose sur la définition de prédicats et de contraintes sous la forme de logique du premier ordre. Cependant, la notation OMT (Object Modelling Technique), qui décrit la structure des diagrammes de classe, est plus dense et plus lisible qu'une définition mathématique dès lors qu'il s'agit d'une définition par intension. Nous l'utilisons donc, dans le chapitre VII, pour simplifier la lecture du modèle. Le langage OCL (Object Constraint Language), souvent associé à OMT, ne sera pas employé car il nous semble moins lisible (et moins expressif) que les formules de logique du premier ordre. Nous utilisons donc ces dernières pour exprimer les contraintes sur les relations entre concepts d'un ensemble de prédicats et de relations binaires du modèle.

Convention

Pour conserver une cohérence du modèle, nous considérons qu'un diagramme OMT constitue une définition mathématique implicite :

- des prédicats unaires permettant de typer les concepts
- des relations binaires permettant de lier les concepts entre eux ou de leur associer des attributs.

Nous nous restreindrons donc volontairement à un sous-ensemble du langage OMT dont la traduction mathématique, triviale, est illustrée dans le Tableau 16. Nous éviterons en particulier d'utiliser la représentation d'attributs, de méthodes ou de relations bidirectionnelles qui compliqueraient inutilement la représentation mathématique associée.

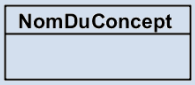
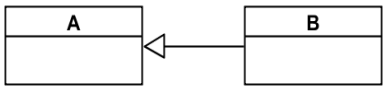
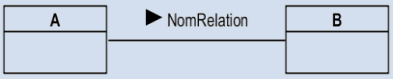
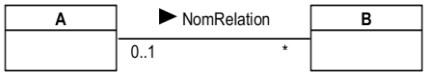
Objet du diagramme OMT	Objet mathématique implicitement associé
Classe 	On notera $NomDuConcept$ le prédicat qui définit les instances dudit concept et $\mathcal{E}_{NomDuConcept}$ son extension. Si x est une instance de ce concept, on notera donc $NomDuConcept(x)$ ou $x \in \mathcal{E}_{NomDuConcept}$
Héritage entre classes 	Si le concept B hérite du concept A, alors on a : $\forall x \ B \ x \ \Rightarrow A \ x$ $\mathcal{E}_B \subset \mathcal{E}_A$
Relation monodirectionnelle entre classes 	On notera $NomRelation$ la relation binaire qui lie les ensembles $\mathcal{E}_A$ et $\mathcal{E}_B$. Soient $a \in \mathcal{E}_A$ et $b \in \mathcal{E}_b$. On utilisera la notation préfixée pour indiquer que a et b sont en relation : $NomRelation(a, b)$ On note définit l'image d'un élément par la fonction $Img_{NomRelation} \quad a \longrightarrow \quad b \in \mathcal{E}_B \ tq \ NomRelation \ a, b \quad \mathcal{E}_A \ \rightarrow \mathcal{P}(\mathcal{E}_B)$ On définit l'antécédent d'un élément est par la fonction $Ant_{NomRelation} \quad b \longrightarrow \quad a \in \mathcal{E}_A \ tq \ NomRelation \ a, b \quad \mathcal{E}_B \ \rightarrow \mathcal{P}(\mathcal{E}_A)$
Cardinalité aux extrémités d'une relation 	Les cardinalités se traduisent en formules reposant sur la théorie des ensembles. $\forall a \in \mathcal{E}_A \ Card \ Img_{NomRelation}(a) \ \geq 0$ $\forall b \in \mathcal{E}_B \ Card \ Ant_{NomRelation}(b) \ \in \{0,1\}$

TABLEAU 16 – RELATIONS ENTRE MODELISATION GRAPHIQUE ET MATHEMATIQUE

Exemple

La Figure 102 équivaut aux définitions suivantes :

On définit les prédicats $Client, Compte$ et $Banque$ correspondant aux concepts susnommés et leurs extensions respectives $\mathcal{E}_{Client}$, $\mathcal{E}_{Compte}$ et $\mathcal{E}_{Banque}$.

Nous définissons de plus les relations binaires suivantes :

$$aCompte : \mathcal{E}_{Client} \times \mathcal{E}_{Compte}$$
$$estClient : \mathcal{E}_{Client} \times \mathcal{E}_{Banque}$$
$$domicilieDans : \mathcal{E}_{Compte} \times \mathcal{E}_{Banque}$$

Les cardinalités des relations binaires précités vérifient par ailleurs les contraintes suivantes :

$$\forall cl \in \mathcal{E}_{Client} \ Card \ Img_{aCompte}(cl) \ \geq 0$$
$$\forall co \in \mathcal{E}_{Compte} \ Card \ Ant_{aCompte}(co) \ = 1$$

$$\forall cl \in \mathcal{E}_{Client}\ Card\ \ Img_{estClient}(cl) \quad \geq 0$$
$$\forall cl \in \mathcal{E}_{Banque}\ Card\ \ Ant_{estClient}(b) \quad \geq 0$$
$$\forall co \in \mathcal{E}_{Compte}\ Card\ \ Img_{domicilieDans}(co) \quad \geq 0$$
$$\forall b \in \mathcal{E}_{Banque}\ Card\ \ Ant_{domicilieDans}(b) \quad \geq 0$$

Même sur un cas simple, une description purement formelle est assez peu pratique à analyser. Le diagramme OMT permettra d'éviter au lecteur la lourdeur de telles définitions tout en conservant un formalisme suffisamment puissant pour pouvoir exprimer par la suite des contraintes de la forme :
« Tout client détenteur d'un compte dans une banque doit être client de la banque »
Ce qui se peut se traduire formellement par :
$$\forall cl, co\ aCompte\ \ cl, co\ \ \Rightarrow\ \exists !\, b \in \mathcal{E}_{Banque}\ \ estClient\ \ cl, b\ \ \wedge domicilieDans(co, b)$$

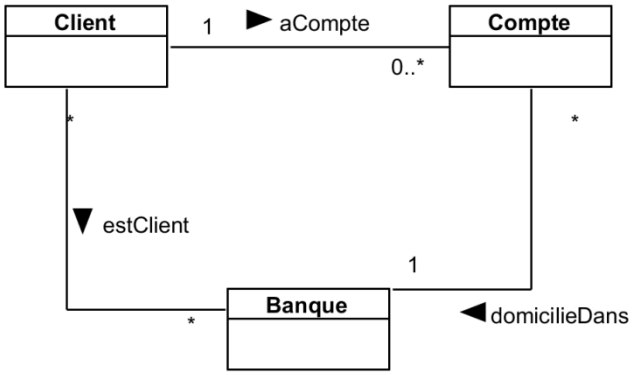


FIGURE 102 – MODELE DU DOMAINE D'APPLICATION BANCAIRE

En nous appuyant sur cette double notation (graphique et logique), nous pouvons définir formellement nos modèles conceptuels ergonomiques et d'architecture. Nous commencerons par définir les termes communs à ces deux modèles.

ANNEXE B – EXEMPLE DE CRITERE NON-ELECTIF

Les critères sont définis dans le chapitre VIII comme des heuristiques fournissant une note aux différentes solutions générées par le système. Parmi ces critères, nous avons présenté les critères électifs qui sont particulièrement adapté à la rédaction de recommandations car ils s'écrivent en logique du premier ordre (ce qui est à la fois la langue des modèles ergonomique et architectural et un langage suffisamment simple pour être rapidement assimilé).

Cependant, toute heuristique est acceptable et nous ne restreignons pas notre approche aux critères électifs. Nous fournissons dans cette annexe, la description d'un critère non-électif, à titre d'exemple. Nous attirons l'attention du lecteur sur le fait qu'il s'agit là d'une heuristique d'évaluation, c'est-à-dire une méthode ***arbitraire*** de notation des solutions.

CRITERE DE COUVERTURE DES CAS D'UTILISATION

L'algorithme identifiant les candidats à partir du graphe des cas d'utilisation fournit des solutions de qualité variable. En effet, un CID décrivant la racine de l'arbre des besoins est, dans le cas général, plus adapté que la réunion des candidats des cas d'utilisation requis. Ainsi, l'usage de solutions construites à partir des relations « include » du graphe a un coût pour l'ergonomie de l'interface. De plus, les candidats qui implémentent un cas d'utilisation et ses extensions permettront une interaction plus riche et doivent donc être favorisés.

Nous adoptons donc la démarche d'évaluation suivante qui permet de ***classer les candidats en fonction de la hauteur des cas d'utilisation qu'ils réalisent dans l'arbre des besoins*** :

- Un CID implémentant la racine a une valeur de 1 (valeur maximale)
- Tout CID implémentant un cas d'utilisation subalterne (requis) se verra attribuer une note strictement inférieure
- Tout CID implémentant un cas d'utilisation optionnel se verra attribuer un bonus.

Rappelons que la description d'un candidat n'est pas une simple liste de CID mais une liste de paires $< cas\ d'utilisation, CID >$, ce qui nous permet de savoir, pour chaque CID intervenant dans une solution, à quelle profondeur du graphe des besoins il s'applique.
Nous proposons donc la métrique d'évaluation suivante, qui consiste à appliquer une note à chaque cas d'utilisation qui traduit sa qualité ergonomique :

- Soit $\alpha \in]0,1[$
- On associe à chaque nœud du graphe des besoins un poids lié à sa profondeur :

  Le poids de la racine est 1.

Soit un nœud n de poids p. Pour traduire la dégradation de l'interface lors du passage du nœud n à ses fils requis, on distribue le poids αp équitablement entre les fils requis. Les nœuds optionnels se voient partager les $1 - \alpha\ p$ restants.

- L'évaluation de la solution se fait par la somme des poids associés :

  A chaque solution correspond un candidat, c'est-à-dire un ensemble de paires $< cas\ d'utilisation, CID >$. On obtient la note du candidat (et donc de la solution) en réalisant la somme des poids des différents cas d'utilisations apparaissant dans ces paires.

Cette méthode permet de garantir que la note des candidats se trouve dans l'intervalle [0,1]. La Figure 103 reprend le diagramme de cas d'utilisation de l'exemple de l'application multimédia, présenté page 139. Elle illustre la distribution des poids sur chacun des cas d'utilisation du graphe des besoins de l'application multimédia pour $\alpha = 0{,}9$.

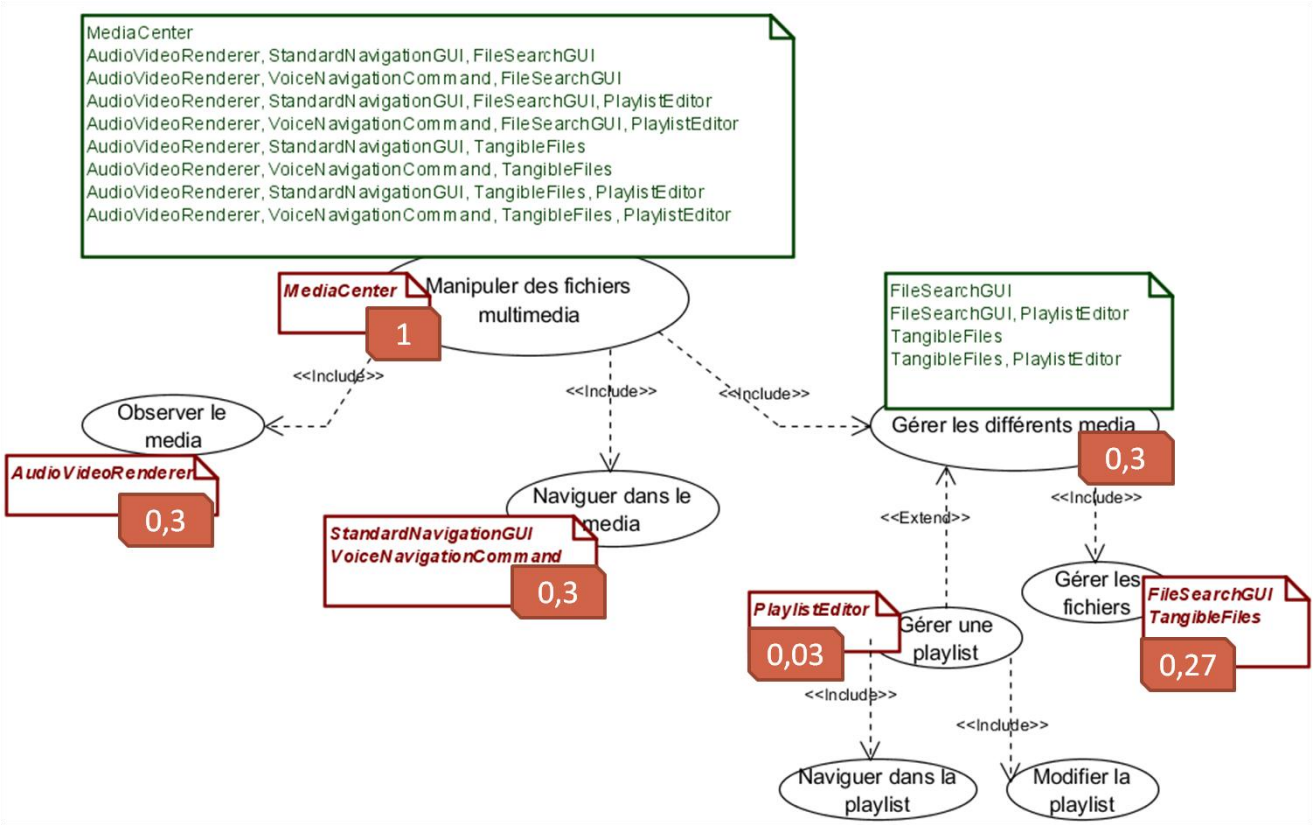


FIGURE 103 – DISTRIBUTION DES POIDS DES CAS D'UTILISATION DANS LE GRAPHE DES BESOIN DE L'APPLICATION MULTIMEDIA

A partir de cet exemple, on peut déduire les notes suivantes :

Candidat	Note
MediaCenter	1
AudioVideoRenderer, StandardNavigationGUI, FileSearchGUI	0,3+0,3+0,27 = 0,87
AudioVideoRenderer, StandardNavigationGUI, FileSearchGUI, playlistEditor	0,3+0,3+0,27+0,03= 0,9
…	…

Le facteur alpha traduit la vitesse de dégradation de la note des candidats en fonction de la profondeur des cas d'utilisation dans l'arbre. Une valeur de α proche de 1 traduit une faible dégradation en réduisant les écarts et en autorisant une plus grande profondeur dans le graphe des besoins utilisateur. Une valeur proche de 0 traduit une très grande dégradation et aurait tendance à rapidement discréditer un candidat éloigné de la racine. La valeur exacte de ce paramètre n'a aucun impact sur l'ordre des solutions ainsi évaluées. Elle a par contre un impact sur la distribution des notes dans l'intervalle [0,1].

Cet exemple illustre l'usage d'heuristiques arbitraires pour évaluer des critères qui ne peuvent s'exprimer sous forme de règles. Les notes attribuées par les critères sont harmonisées avant d'être fusionnées et ce, quelle que soit la nature du critère (électif ou autre).

ANNEXE C – SYNTAXE DE JESS

Nous présentons ici les éléments de langages nécessaires à la compréhension du fonctionnement de Jess et du reste du système. Une documentation plus complète de la syntaxe peut être trouvée sur le site web du projet. Cette syntaxe est dérivée de Lisp, nous supposerons donc que le lecteur est familier avec ce style de programmation.

La construction de la base de faits dans Clips/Jess impose de structurer les faits par des *templates*. Les *templates* définissent des types de faits qui sont définis par une liste de *slots* nommés. Deux faits de même type dont les contenus des *slots* sont identiques sont considérés comme étant le même fait. Jess supporte des slots pouvant accepter une seule valeur ou une liste ordonnée de valeurs (*multislots*). La définition d'un template s'effectue grâce à l'instruction *deftemplate*.

```
(deftemplate personne (slot nom))
(deftemplate frere (slot n1) (slot n2))
(deftemplate fils (slot nPere) (slot nFils))
(deftemplate oncle (slot nOncle) (slot nNeveu))
```

Les faits peuvent ensuite être déclarés grâce aux instructions *assert* (pour un fait unique) et *deffacts* (pour un groupe de faits).

```
(deffacts liste-des-personnes
  « Définit la liste des personnes utilisées dans l'exemple »
  (personne (nom « Pierre »))
  (personne (nom « Paul »))
  (personne (nom « Jacques »))
  (frere (n1 « Pierre ») (n2 « Paul »))
)
(assert (fils (nPere « Pierre ») (nFils « Jacques »)))
```

Il est également possible de définir des fonctions grâce à l'instruction *deffunction*. Ces fonctions nous permettront de réaliser des opérations sur les structures de données manipulées lors de l'application de l'algorithme du modèle comportemental.

```
(deffunction affiche-relation-oncle ( ?n1 ?n2)
  « Affiche un message indiquant que n1 est l'oncle de n2 »
  (printout t ?n1 « est l'oncle de « ?n2 crlf)
  )
```

La définition d'une règle s'effectue grâce à l'instruction *defrule*. La règle se décompose en deux parties : la prémisse et la conclusion. La prémisse indique le motif que l'on recherche dans la base de fait. Les variables libres y sont précédées d'un point d'interrogation « ? ». La conclusion est constituée d'un ensemble d'actions à effectuer lorsque la prémisse est satisfaite. Il s'agit d'une suite d'appels de fonctions qui peut réutiliser les variables des prémisses. Dans ce cas, les variables sont liées aux faits qui satisfont la prémisse, et la conclusion est appelée autant de fois qu'il y a de combinaisons valides des variables

définies dans la prémisse. On peut utiliser, dans une conclusion, l'instruction *assert* pour créer de nouveaux faits, ce qui permet de créer des règles réentrantes.

```
(defrule symetrie-fratrie
  (frere (n1 ?x) (n2 ?y))
  =>
  (assert (frere (n1 ?y) (n2 ?x)))
  )


(defrule detecte-relation-oncle
  (frere (n1 ?n1) (n2 ?n2))
  (fils (nPere ?n1) (nFils ?x))
  =>
  (affiche-relation-oncle ?n2 ?x)
  (assert (oncle (nOncle ?n2) (nNeveu ?x)))
  )
```

La règle *detecte-relation-oncle* fait appel à un motif qui lie deux faits de type *frere* et *fils*. Lorsque cela n'est pas précisé, la relation implicite qui satisfait le motif est un « et » logique (opérateur *and* dans Jess). Jess offre cependant la possibilité de définir d'autres relations entre les faits au moyen des opérateurs suivant : *and*, *or*, *not*, *exists*, *forall*. Ces opérateurs sont implémentés en utilisant le principe de négation par l'échec (*negation as failure*), ils ne représentent donc pas toujours une vérité tautologique mais une vérité en l'état actuel des connaissances de la base de faits. Jess propose également d'autres opérateurs permettant d'enrichir les formules d'une prémisse :

- *test* permet d'appeler une fonction booléenne pour réaliser un test (utilisé pour effectuer une comparaison numérique d'une valeur par exemple).
- *accumulate* permet d'accumuler dans une liste tous les faits qui satisfont un motif, ce qui permet d'utiliser des fonctions traitant un ensemble de faits (et non les faits un-à-un) dans la conclusion de la règle.

On obtient ainsi une syntaxe dont l'expressivité, bien qu'imparfaite, s'approche de celle de la logique du premier ordre.

Contrairement à certains systèmes experts, le moteur d'inférence ne tourne pas perpétuellement en tâche de fond. Il faut l'appeler lorsque les bases de fait et de règles sont prêtes grâce à la fonction *run*. La fonction *facts* permet d'afficher à l'écran la base des faits.

```
(run)
(facts)
```

Le fait que le programmeur ait le contrôle sur l'exécution du moteur d'inférence permet au système de réaliser plusieurs phases d'applications de règles. Dans notre implémentation du noyau décisionnel, on se sert de ce mécanisme pour séparer l'application de certains groupes de règles entre lesquels des opérations sont réalisées sur les faits dans un style impératif.

L'exécution du programme donné en exemple[39] dans cette section donne le rendu suivant :

```
Paul est l'oncle de Jacques
f-0  (MAIN ::initial-fact)
f-1  (MAIN ::personne (nom « Pierre »))
f-2  (MAIN ::personne (nom « Paul »))
f-3  (MAIN ::personne (nom « Jacques »))
f-4  (MAIN ::frere (n1 « Pierre ») (n2 « Paul »))
f-5  (MAIN ::fils (nPere « Pierre ») (nFils « Jacques »))
f-6  (MAIN ::oncle (nOncle « Paul ») (nNeveu « Jacques »))
f-7  (MAIN ::frere (n1 « Paul ») (n2 « Pierre »))
```

On notera également qu'il est possible de faire une requête sur la base de faits lors des phases impératives (i.e. en dehors d'une règle), grâce à l'instruction *defquery*. Cette instruction possède un fonctionnement légèrement plus complexe qui ne sera pas illustré dans cette courte introduction, elle permet cependant de faire une recherche sur des motifs de la base de faits et de récupérer un itérateur sur l'ensemble des variables qui satisfont ce motif.

[39] L'execution de ce programme nécessite en fait l'insertion d'une instruction supplémentaire *(reset)* qui ne sera pas détaillée dans cette simple introduction à Jess.

ANNEXE D – REPRESENTATION DE OWL DANS JESS

Le lien entre les informations contenues dans les ontologies et la base de faits Jess est décrit par le code Jess suivant :

```
;Defines the structure of OWL
(deftemplate owl:Class)
(deftemplate owl:NamedClass extends owl:Class (slot name))
(deftemplate owl:Instance (slot name))
(deftemplate owl:subClassOf (slot parent) (slot child))
(deftemplate owl:Property (slot name))
(deftemplate owl:PropertyInstance (declare (ordered TRUE)))
(deftemplate owl:subPropertyOf (slot parent) (slot child))
(deftemplate owl:ObjectProperty extends owl:Property)
(deftemplate owl:DatatypeProperty extends owl:Property)
(deftemplate owl:sameAs (declare (ordered TRUE)))
(deftemplate owl:differentFrom (declare (ordered TRUE)))

;Define the basic root object
(deftemplate owl:Thing extends owl:Instance)
(assert (owl:NamedClass (name owl:Thing)))

;Defines the intented behavior of OWL
(deffunction owl:createClass (?className)
        "Creates a new owl NamedClass (derives from owl:Thing by default)"
        (if (is-named-instance ?className owl:NamedClass)
                then (return FALSE)
                else
                        (eval (str-cat "(deftemplate " ?className " extends owl:Instance)"))
                        (eval (str-cat "(assert (owl:NamedClass (name " ?className ")))"))
                        (eval (str-cat "(assert (owl:subClassOf (child " ?className ") (parent
owl:Thing)))"))
                        (return TRUE)))

(deffunction owl:createDatatypeProperty (?propName)
```

```
        "Creates a new owl datatype property"
        (if (is-named-instance ?propName owl:Property)
                then (return FALSE)
                else
                        (eval (str-cat "(deftemplate " ?propName " extends
owl:PropertyInstance)"))
                        (eval (str-cat "(assert (owl:DatatypeProperty (name " ?propName ")))"))
                        (assert-binary-property ?propName)
                        (return TRUE)))

(deffunction owl:createObjectProperty (?propName)
        "Creates a new owl datatype property"
        (if (is-named-instance ?propName owl:Property)
                then (return FALSE)
                else
                        (eval (str-cat "(deftemplate " ?propName " extends
owl:PropertyInstance)"))
                        (eval (str-cat "(assert (owl:ObjectProperty (name " ?propName ")))"))
                        (assert-binary-property ?propName)
                        (return TRUE)))

(defrule owl:subClassOf-closure
        "Create rules for computing transitive closures of owl:subClassOf"
        (owl:subClassOf (child ?c1) (parent ?c2))
        =>
        (eval (str-cat "(defrule owl:" ?c1 "-" ?c2 "-closure
                (logical (" ?c1 " (name ?n1)))
                 =>
                (assert (" ?c2 " (name ?n1))))")))

(defrule owl:subPropertyOf-closure
        "Create rules for computing transitive closures of owl:subPropertyOf"
        (owl:subPropertyOf (child ?p1) (parent ?p2))
        =>
        (eval (str-cat "(defrule owl:" ?p1 "-" ?p2 "-closure
                (logical (" ?p1 " ?subject ?object))
                 =>
                 (assert (" ?p2 " ?subject ?object)))")))
```

ANNEXE E – NOTATION SIMPLIFIEE AVEC IPOJO

Le Tableau 17 illustre la simplicité de mise en œuvre de composants iPOJO à travers le système d'annotations qu'il propose.

Dans cet exemple, l'annotation `@Component` indique que la classe décrit un composant qui doit apparaitre dans le registre. Les services fournis sont représentés par les interfaces qu'implémente la classe et l'annotation `@Provides`. Le service requis par ce composant y est représenté par un attribut de classe décoré par `@Requires`. Le développeur ne doit pas se préoccuper de son initialisation qui sera réalisée à l'exécution par le système (une fois le service requis découvert). Enfin, le cycle de vie du composant est géré par de simples méthodes appelées par l'infrastructure pour signaler au composant qu'il est valide (i.e. ses interfaces requises sont associées toutes associées) ou non. La librairie repère ces méthodes grâce aux annotations `@Validate` et `@Invalidate`.

```
@Component
@Provides
public class IPojoExample implements ProvidedInterface {

  @Requires
  private RequiredInterface myAttribute;

  @Validate
  public void start() {
        System.out.println("Le composant est valide");
  }

  @Invalidate
  public void stop() {
        System.out.println("Le composant est invalide");
  }
}
```

TABLEAU 17 – EXEMPLE DE CREATION D'UN COMPOSANT IPOJO

iPOJO offre bien sûr d'autres mécanismes plus avancés, tels que la gestion des instances créés et de leur connexion, mais qui sortent du cadre de cette introduction.

BIBLIOGRAPHIE

Aarts, E & Ruyter, BD 2009, 'New research perspectives on Ambient Intelligence', *Journal of Ambient Intelligence and Smart Environments*, vol 1, no. 1, pp. 5-14.

Abrams, M, Phanouriou, C, Batongbacal, AL, Williams, SM & Shuster, JE 1999, 'UIML: an appliance-independent XML user interface language', *Computer Networks*, vol 31, no. 11, pp. 1695-1708.

Agarwal, Y, Balaji, B, Gupta, R, Lyles, J, Wei, M & Weng, T 2010, 'Occupancy-driven energy management for smart building automation', *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*.

Ali, MF, Perez-Quinones, MA, Abrams, M & Shell, E 2002, 'Building multi-platform user interfaces with UIML', *Computer-Aided Design of User Interfaces*.

Almendros-Jiménez, JM & Iribarne, L 2008, 'An extension of UML for the modeling of WIMP user interfaces', *Journal of Visual Languages \& Computing*, vol 19, no. 6, pp. 695-720.

Appert, C & Beaudouin-Lafon, M 2008, 'SwingStates: adding state machines to Java and the Swing toolkit', *Software: Practice and Experience*, vol 38, pp. 1149-1182.

Appert, C, Huot, S, Dragivevic, P & Beaudouin-Lafon, M 2009, 'FlowStates : Prototypage d'applications interactives avec des flots de données et des machines à états', *21ème conférence francophone sur l'Interaction Homme-Machine, IHM 2009*, ACM Press.

Athanasopoulos, D, Zarras, AV, Issarny, V, Pitoura, E & Vassiliadis, P 2008, 'CoWSAMI: Interface-aware context gathering in ambient intelligence environments', *Pervasive and Mobile Computing*, vol 4, no. 3, pp. 360-389.

Atzori, L, Iera, A & Morabito, G 2010, 'The internet of things: A survey', *Computer Networks*, vol 54, no. 15, pp. 2787-2805.

Augusto, JC 2007, *Ambient intelligence: the confluence of ubiquitous/pervasive computing and artificial intelligence*, London, UK: Springer-Verlag.

Augusto, J & Nugent, C 2006, 'Smart Homes Can Be Smarter', *Lecture Notes in Computer Science*, vol 4008, pp. 1-15.

Balme, L, Demeure, A, Barralon, N, Coutaz, J, Calvary, G & others 2004, 'Cameleon-rt: A software architecture reference model for distributed, migratable, and plastic user interfaces', *Lecture Notes in Computer Science*, pp. 291-302.

Baron, M, Lucquiaud, V, Autard, D & Scapin, D 2006, 'K-MADe: un environnement pour le noyau du modèle de description de l'activité', *Proceedings of the 18th International Conference of the Association Francophone d'Interaction Homme-Machine*.

Barralon, N & Coutaz, J 2008, 'Coupling interaction resources in ambient spaces: There is more than meets the eye!', *Engineering Interactive Systems*, pp. 537-554.

Barralon, N, Coutaz, J & Lachenal, C 2007, 'Coupling interaction resources and technical support', *Universal Access in Human-Computer Interaction. Ambient Interaction*, pp. 13-22.

Barralon, N, Lachenal, C & Coutaz, J 2004, 'Couplage de ressources d'interaction', *Proceedings of the 16th conference on Association Francophone d'Interaction Homme-Machine*, ACM.

Bass, L, Faneuf, R, Little, R, Mayer, N, Pellegrino, B, Reed, S, Seacord, R, Sheppard, S & Szczur, MR 1992, 'A Metamodel for the Runtime Architecture of an Interactive System', *ACM SIGCHI Bulletin*, vol 24, no. 1, pp. 32-37.

Becker, E, Metsis, V, Arora, R, Vinjumur, J, Xu, Y & Makedon, F 2009, 'SmartDrawer: RFID-based smart medicine drawer for assistive environments', *Proceedings of the 2nd International Conference on PErvasive Technologies Related to Assistive Environments*, ACM.

Bellik, Y 1995, 'Interfaces multimodales : concepts, modèles et architectures', Ph.D. dissertation, Université d'Orsay Paris-Sud.

Bellik, Y, Kameas, A, Goumopoulos, C, Hagras, H, Heinroth, T, Pruvost, G, Meliones, A, Economou, D, Minker, W & Gardner, M 2009, 'Multidimensional Pervasive Adaptation into Ambient Intelligent Environments', *DASC '09: Proceedings of the 2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing*, IEEE Computer Society, Washington, DC, USA, <http://www.computer.org/portal/web/csdl/doi/10.1109/DASC.2009.120>.

Bellik, Y & Pruvost, G 2010, 'Simulation tools - D20', Tech. rep., ATRACO.

Bellik, Y, Rebai, I, Machrouch, E, Barzaj, Y, Jacquet, C, Pruvost, G & Sansonnet, JP 2009, 'Multimodal Interaction within Ambient Environments: an Exploratory Study', *Proc. of 12th IFIP TC13 Conference on Human-Computer Interaction (INTERACT 2009)*, <http://www.springerlink.com/content/n513550460537rgu/>.

Bernsen, NO 1994, 'Modality Theory in support of multimodal interface design', *Proceedings of the AAAI Spring Symposium on Intelligent Multi-Media Multi-Modal Systems*.

Berti, S, Correani, F, Paterno, F & Santoro, C 2004, 'The TERESA XML language for the description of interactive systems at multiple abstraction levels', *Proceedings Workshop on Developing User Interfaces with XML: Advances on User Interface Description Languages*.

Blackwell, AF 2006, 'The reification of metaphor as a design tool', *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol 13, no. 4, pp. 490-530.

Blattner, MM & Dannenberg, RB 1990, 'CHI'90 Workshop on multimedia and multimodal interface design', *ACM SIGCHI Bulletin*, vol 22, no. 2, pp. 54-58.

Bouchet, J & Nigay, L 2004, 'ICARE: a component-based approach for the design and development of multimodal interfaces', *CHI'04 extended abstracts on Human factors in computing systems*.

Bradbury, R 1953, *Fahrenheit 451*, Ballantine Books.

Brel, C, Dery-Pinna, AM, Renevier-Gonin, P & Riveill, M 2011, 'OntoCompo: a tool to enhance application composition', *Human-Computer Interaction--INTERACT 2011*, pp. 588-591.

Brel, C & Renevier-Gonin, P 2011, 'Composing applications with OntoCompo', *23rd French Speaking Conference on Human-Computer Interaction*.

Brusilovsky, P, Kobsa, A & Nejdl, W 2007, *The adaptive web: methods and strategies of web personalization*, Springer.

Calvary, G & Coutaz, J 2007, 'Métamorphose des IHM et Plasticité', *Revue d'Interaction Homme-Machine Vol*, vol 8, no. 1.

Calvary, G, Coutaz, J, Thevenin, D, Limbourg, Q, Bouillon, L & Vanderdonckt, J 2003, 'A unifying reference framework for multi-target user interfaces', *Interacting with Computers*, vol 15, no. 3, pp. 289-308.

Carbonell, N 2006, 'Ambient multimodality: towards advancing computer accessibility and assisted living', *Universal Access in the Information Society*, vol 5, no. 1, pp. 96-104.

Carroll, J 2000, 'Five reasons for scenario-based design', *Interacting with Computers*, vol 13, no. 1, pp. 43-60.

Carroll, JM & Thomas, JC 1982, 'Metaphor and the Cognitive Representation of Computing Systems', *Systems, Man and Cybernetics, IEEE Transactions on*, vol 12, no. 2, pp. 107-116.

Cervantes, H & Hall, RS 2003, 'Automating service dependency management in a service-oriented component model', *ICSE CBSE Workshop*.

Chalmers, M & MacColl, I 2003, 'Seamful and Seamless Design in Ubiquitous Computing', *Workshop At the Crossroads: The Interaction of HCI and Systems Issues in UbiComp*.

Charfi, S 2009, 'Conception et évaluation des systèmes interactifs mixtes selon une approche centrée utilisateurs', Ph.D. dissertation, Université Toulouse I.

Coutaz, J 1987, 'PAC, an Object Oriented Model for Dialog Design', *Human-Computer Interaction-INTERACT*, vol 87, pp. 431-436.

Coutaz, J 2007, 'Meta-user interfaces for ambient spaces', *Task Models and Diagrams for Users Interface Design*, pp. 1-15.

Coutaz, J & Nigay, L 1994, 'Les propriétés CARE dans les interfaces multimodales', *IHM*.

Dahley, A, Wisneski, C & Ishii, H 1998, 'Water lamp and pinwheels: ambient projection of digital information into architectural space', ACM.

De Hert, P, Gutwirth, S, Moscibroda, A, Wright, D & Fuster, GG 2009, 'Legal safeguards for privacy and data protection in ambient intelligence', *Personal and ubiquitous computing*, vol 13, no. 6, pp. 435-444.

Dey, AK 2000, 'Providing Architectural Support for Building Context-Aware Applications', Ph.D. dissertation, Georgia Institute of Technology.

Dey, AK, Abowd, GD & Salber, D 2001, 'A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications', *Human-Computer Interaction*, vol 16, no. 2-4, pp. 97-166.

Dohr, A, Modre-Opsrian, R, Drobics, M, Hayn, D & Schreier, G 2010, 'The internet of things for ambient assisted living', *Ieee*.

Dooley, J, Wagner, C, Hagras, H & Pruvost, G 2011, 'FollowMe : The Persistent GUI', *to appear in the 6th International Symposium on Parallel Computing in Electrical Engineering, Luton, UK, April 2011*.

Dragicevic, P & Fekete, JD 2001, 'Input device selection and interaction configuration with ICON', *People and Computers*, pp. 543-558.

Duarte, C & Carriço, L 2006, 'A conceptual framework for developing adaptive multimodal applications', *Proceedings of the 11th international conference on Intelligent user interfaces*.

Dubois, E 2001, 'Chirurgie augmentée : un cas de réalité augmentée ; Conception et réalisation centrées sur l'utilisateur', Ph.D. dissertation, Université de Grenoble 1.

Ducatel, K, Bogdanowicz, M, Scapolo, F, Leijten, J & Burgelman, J 2001, *Istag: Scenarios for ambient intelligence in 2010*, Office for Official Publications of the European Communities.

Dumas, B, Lalanne, D & Oviatt, S 2009, 'Multimodal interfaces: A survey of principles, models and frameworks', *Human Machine Interaction*, pp. 3-26.

Elrod, S, Bruce, R, Gold, R, Goldberg, D, Halasz, F, Janssen, W, Lee, D, McCall, K, Pedersen, E, Pier, K & others 1992, 'Liveboard: a large interactive display supporting group meetings, presentations, and remote collaboration', *Proceedings of the SIGCHI conference on Human factors in computing systems*.

Estrin, D, Culler, D, Pister, K & Sukhatme, G 2002, 'Connecting the Physical World with Pervasive Networks', *IEEE PERVASIVE COMPUTING*, pp. 59-69.

Feiner, S, Macintyre, B & Seligmann, D 1993, 'Knowledge-based augmented reality', *Communications of the ACM*, vol 36, no. 7, pp. 53-62.

Fishkin, KP 2004, 'A taxonomy for and analysis of tangible interfaces', *Personal and Ubiquitous Computing*, vol 8, no. 5, pp. 347-358.

Fitzmaurice, GW, Ishii, H & Buxton, WAS 1995, 'Bricks: laying the foundations for graspable user interfaces', *Proceedings of the SIGCHI conference on Human factors in computing systems* , ACM Press/Addison-Wesley Publishing Co. New York, NY, USA.

Forgy, CL 1982, 'Rete: A fast algorithm for the many pattern/many object pattern match problem', *Artificial intelligence*, vol 19, no. 1, pp. 17-37.

Gabillon, Y, Calvary, G & Fiorino, H 2008, 'Composing interactive systems by planning', *Proceedings of the 4th French-speaking conference on Mobility and ubiquity computing*.

Gabillon, Y, Calvary, G, Fiorino, H & others 2011, 'Composition d'Interfaces Homme-Machine en contexte: approche par planification automatique', *Technique et Science Informatiques (TSI)*, vol 30, no. 10, pp. 1143-1166.

Gajos, K & Weld, DS 2004, 'SUPPLE: automatically generating user interfaces', *Proceedings of the 9th international conference on Intelligent user interfaces*.

Gallissot, M 2012, 'Modéliser le concept de confort dans un habitat intelligent: du multisensoriel au comportement', Ph.D. dissertation, Université de Grenoble.

Garrison, D, Cleveland-Innes, M & Fung, TS 2010, 'Exploring causal relationships among teaching, cognitive and social presence: Student perceptions of the community of inquiry framework', *The Internet and Higher Education*, vol 13, no. 1, pp. 31-36.

Gibson, JJ 1977, 'The theory of affordances', *Perceiving, acting and knowing: toward an ecological psychology*, pp. 67-82.

González, G, Angulo, C, Lopez, B & la, JLD 2005, 'Smart user models for ambient recommender systems', *Ambient Intelligence and (Everyday) Life*, pp. 11-20.

Goumopoulos, C, Calemis, I, Togias, K & Kameas, A 2010, 'Integrated component platform for prototype testing and updated specification and design report - Deliverable D7', ATRACO.

Goumopoulos, C, Calemis, I, Togias, K, Seremeti, L & Kameas, A 2011, 'Ecology configuration adaptation component – Deliverable D17', Tech. rep., Adaptive and TRusted Ambient eCOlogies (ATRACO European Project), DOI: ATRACO / 216837.

Goumopoulos, C & Kameas, A 2008, 'Ambient Ecologies in Smart Homes', *The Computer Journal*, vol 52, no. 8, pp. 922-937.

Gruber, TR 1995, 'Toward principles for the design of ontologies used for knowledge sharing', *International journal of human computer studies*, vol 43, no. 5, pp. 907-928.

Hagras, H & Wagner, C 2011, 'Artefact Adaptation in Ambient Intelligent Environments', in *Next Generation Intelligent Environments: Ambient Adaptive Systems*, Springer Verlag.

Hagras, H, Wagner, C, Kameas, A, Goumopoulos, C, Meliones, A, Seremeti, L, Bellik, Y, Pruvost, G, Heinroth, T & Minker, W 2012, 'Symbiotic Ecologies in Next Generation Ambient Intelligent Environments', *International Journal of Next-Generation Computing*, vol 3, no. 1, <http://perpetualinnovation.net/ojs/index.php/ijngc/article/view/138>.

Haux, R 2006, 'Health information systems: past, present, future', *International Journal of Medical Informatics*, vol 75, no. 3-4, pp. 268-281.

Heinroth, T, Kameas, A, Pruvost, G, Seremeti, L, Bellik, Y & Minker, W 2011, 'Human-Computer Interaction in Next Generation Ambient Intelligent Environments', *Intelligent Decision Technologies - Special Issue on Knowledge-based environments and services in HCI*, vol 5, no. 1, pp. 31-46.

Heinroth, T, Minker, W (eds.) 2011, *Next generation intelligent environments*, Springer.

Henricksen, K, Indulska, J & Rakotonirainy, A 2002, 'Modeling Context Information in Pervasive Computing Systems', *Lecture Notes in Computer Science*, pp. 167-180.

Hinckley, K 2003, 'Synchronous gestures for multiple persons and computers', *Proceedings of the 16th annual ACM symposium on User interface software and technology*.

Holman, D & Vertegaal, R 2008, 'Organic user interfaces: designing computers in any way, shape, or form', *COMMUNICATIONS OF THE ACM*, vol 51, no. 6, pp. 48-55.

Holman, D, Vertegaal, R, Altosaar, M, Troje, N & Johns, D 2005, 'Paper windows: interaction techniques for digital paper', *ACM*.

Hornecker, E & Buur, J 2006, 'Getting a grip on tangible interaction: a framework on physical space and social interaction', *Proceedings of the SIGCHI conference on Human Factors in computing systems*, ACM.

Ishii, H 2004, 'Bottles: A Transparent Interface as a Tribute to Mark Weiser', *IEICE Transactions on information and systems*, vol 87, no. 6, pp. 1299-1311.

Ishii, H & Ullmer, B 1997, 'Tangible bits: towards seamless interfaces between people, bits and atoms', *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM.

Ito, M, Iwaya, A, Saito, M, Nakanishi, K, Matsumiya, K, Nakazawa, J, Nishio, N, Takashio, K & Tokuda, H 2003, 'Smart furniture: improvising ubiquitous hot-spot environment', *Proceedings of the 23rd International Conference on Distributed Computing Systems Workshops*.

Jacquet, C 2006, 'Présentation opportuniste et multimodale d'informations dans le cadre de l'intelligence ambiante', Ph.D. dissertation, Université d'Orsay Paris-Sud.

Jacquet, C, Bellik, Y & Bourda, Y 2007, 'KUP, un modèle pour la présentation multimodale et opportuniste d'informations en situation de mobilité', *Conférence de l'Association Française d'interaction homme-machine (IHM 2007)*, ACM.

Jara, AJ, Zamora, MA & Skarmeta, AF 2011, 'An internet of things-based personal device for diabetes therapy management in ambient assisted living (AAL)', *Personal and Ubiquitous Computing*, vol 15, no. 4, pp. 431-440.

Joffroy, C 2011, 'Composition d'applications et de leurs Interfaces homme-machine dirigée par la composition fonctionnelle', Ph.D. dissertation, Université de Nice Sophia-Antipolis.

John, BE & Kieras, DE 1996, 'The GOMS family of user interface analysis techniques: Comparison and contrast', *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol 3, no. 4, pp. 320-351.

Kalawsky, R 1993, *The science of virtual reality and virtual environments*, Addison-Wesley Longman Publishing Co., Inc.

Kameas, A, Goumopoulos, C, Hagras, H, Gardner, M, Heinroth, T, Minker, W, Meliones, A, Economou, D, Bellik, Y & Pruvost, G 2009, 'A Pervasive System Architecture that supports Adaptation using Agents and Ontologies', *The 10th International Symposium on Pervasive Systems, Algorithms and Networks (I-SPAN)*, IEEE Computer Society, Los Alamitos, CA, USA.

Keegan, S, O'Hare, GMP & O'Grady, MJ 2008, 'Easishop: Ambient intelligence assists everyday shopping', *Information Sciences*, vol 178, no. 3, pp. 588-611.

Kindberg, T & Fox, A 2002, 'System software for ubiquitous computing', *Pervasive Computing, IEEE*, vol 1, no. 1, pp. 70-81.

Kindberg, T & Zhang, K 2003, 'Secure spontaneous device association', *UbiComp 2003: Ubiquitous Computing*, Springer.

Kolski, C 1995, 'Méthodes et modèles de conception et d'évaluation des interfaces homme-machine (HDR)', Ph.D. dissertation, Université de Valenciennes.

Kolski, C, Ezzedine, H & Abed, M 2001, 'Développement du logiciel: des cycles classiques aux cycles enrichis sous l'angle des IHM', *Analyse et conception de l'IHM, Interaction Homme-Machine pour les SI, Hermes, Paris*, pp. 145-174.

Kolski, C, Forbrig, P, David, B, Girard, P, Tran, C & Ezzedine, H 2009, 'Agent-Based Architecture for Interactive System Design: Current Approaches, Perspectives and Evaluation', *Human-Computer Interaction. New Trends*, pp. 624-633.

Krasner, GE & Pope, ST 1988, 'A cookbook for using the model-view controller user interface paradigm in Smalltalk-80', *Journal of Object-Oriented Programming*, vol 1, no. 3, pp. 26-49.

Kuflik, T, Busetta, P, Penserini, L, Bresciani, P & Zancanaro, M 2004, 'Personalized information delivery in dynamic museum environment by implicit organizations of agents', *Proceedings of the Workshop on Environments for Personalized Information Access (in conjunction with AVI2004)*.

Langheinrich, M 2002, 'A Privacy Awareness System for Ubiquitous Computing Environments', *LECTURE NOTES IN COMPUTER SCIENCE*, pp. 237-245.

Lawson, JYL, Al-Akkad, AA, Vanderdonckt, J & Macq, B 2009, 'An open source workbench for prototyping multimodal interactions based on off-the-shelf heterogeneous components', *Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*.

Lopez-Velasco, C, Villanova-Oliver, M, Gensel, J & Martin, H 2005, 'Services Web adaptés aux utilisateurs nomades', *Proceedings of the 2nd French-speaking conference on Mobility and ubiquity computing*, ACM.

Macedonia, MR & Zyda, MJ 1997, 'A taxonomy for networked virtual environments', *Multimedia*, vol 4, no. 1, pp. 48-56.

Malinowski, U, Kühme, T, Dieterich, H & Schneider-Hufschmidt, M 1992, 'A taxonomy of adaptive user interfaces', *PEOPLE AND COMPUTERS*, pp. 391-414.

Mann, S 1997, 'Smart clothing: The wearable computer and wearcam', *Personal and Ubiquitous Computing*, vol 1, no. 1, pp. 21-27.

Mokhtar, S, Georgantas, N & Issarny, V 2005, 'Ad hoc composition of user tasks in pervasive computing environments', *Software Composition*.

Mori, G, Paterno, F & Santoro, C 2002, 'CTTE: support for developing and analyzing task models for interactive system design', *Software Engineering, IEEE Transactions on*, vol 28, no. 8, pp. 797-813.

Motik, B, Sattler, U & Studer, R 2005, 'Query answering for OWL-DL with rules', *Web Semantics: Science, Services and Agents on the World Wide Web*, vol 3, no. 1, pp. 41-60.

Myers, B 1994, 'Challenges of HCI design and implementation', *interactions*, vol 1, no. 1, pp. 73-83.

Nguyen, T 2005, 'Context-aware access control in pervasive computing environments', Master thesis, Washington state university.

Nichols, J 2006, 'Automatically generating high-quality user interfaces for appliances', Ph.D. dissertation, Carnegie Mellon University.

Nichols, J, Myers, B, Litwack, K, Higgins, M, Hughes, J & Harris, T 2004, 'Describing appliance user interfaces abstractly with xml', *Institute for Software Research*, <http://repository.cmu.edu/isr/773>.

Nichols, J, Rothrock, B, Chau, DH & Myers, BA 2006, 'Huddle: automatically generating interfaces for systems of multiple connected appliances', *Proceedings of the 19th annual ACM symposium on User interface software and technology*.

Nigay, L 1994, 'Conception et modélisation logicielles des systèmes interactifs: application aux interfaces multimodales', Ph.D. dissertation, Université Joseph-Fourier-Grenoble I.

Nigay, L & Coutaz, J 1993, 'A design space for multimodal systems: concurrent processing and data fusion', *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*.

Nigay, L & Coutaz, J 1995, 'A generic platform for addressing the multimodal challenge', *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM Press/Addison-Wesley Publishing Co.

Norman, DA 1986, 'Cognitive engineering', *User centered system design*, pp. 31-61.

Noy, NF & Musen, MA 2000, 'Algorithm and tool for automated ontology merging and alignment', *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-00). Available as SMI technical report SMI-2000-0831*.

OMG 2011, 'OMG Unified Modeling Language (OMG UML) Infrastructure'.

Orwell, G 1949, *1984*, Secker and Warburg.

Padovitz, A, Loke, SW & Zaslavsky, A 2007, 'The ECORA framework: A hybrid architecture for context-oriented pervasive computing', *Pervasive and Mobile Computing*, vol 4, no. 2, pp. 182-215.

Parkes, A, I., P & I., I 2008, 'Designing kinetic interactions for organic user interfaces', *COMMUNICATIONS OF THE ACM*, vol 51, no. 6, pp. 58-65.

Paterno, F, Santoro, C & Spano, LD 2011, 'Engineering the authoring of usable service front ends', *Journal of Systems and Software*, vol 84, no. 10, pp. 1806-1822.

Patten, J & Ishii, H 2007, 'Mechanical constraints as computational constraints in tabletop tangible interfaces', *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM.

Petersen, SA & Kofod-Petersen, A 2006, 'Learning in the city: Context for communities and collaborative learning', *IET - Intelligent Environments, 2006*.

Pfaff, GE 1985, *User interface management systems*, Springer-Verlag New York, Inc.

Piper, B, Ratti, C & Ishii, H 2002, 'Illuminating clay: a 3-D tangible interface for landscape analysis', *Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves*, ACM.

Pruvost, G, Heinroth, T, Bellik, Y & Minker, W 2011, 'User Interaction Adaptation within Ambient Environments', in Heinroth, Minker (eds.), *Next Generation Intelligent Environments: Ambient Adaptive Systems*, Springer, Boston (USA).

Pruvost, G, Kameas, A, Heinroth, T, Seremeti, L & Minker, W 2009, 'Combining agents and ontologies to support task-centred interoperability in Ambient Intelligent Environments', *Proceedings of the 2009 Ninth International Conference on Intelligent Systems Design and Applications (ISDA)*, IEEE Computer Society, Washington, DC, USA.

Pyla, PS, Tungare, M & Pérez-Quinones, M 2006, 'Multiple user interfaces: Why consistency is not everything, and seamless task migration is key', *Proceedings of the CHI 2006 workshop on the many faces of consistency in cross-platform design*.

Raffle, HS, Parkes, AJ & Ishii, H 2004, 'Topobo: a constructive assembly system with kinetic memory', *Proceedings of the SIGCHI conference on Human factors in computing systems*.

Rekimoto, J 1997, 'Pick-and-drop: a direct manipulation technique for multiple computer environments', *Proceedings of the 10th annual ACM symposium on User interface software and technology*, ACM.

Rigole, P, Clerckx, T, Berbers, Y & Coninx, K 2007, 'Task-driven automated component deployment for ambient intelligence environments', *Pervasive and Mobile Computing*, vol 3, no. 3, pp. 276-299.

Riva, G, Vatalaro, F, Davide, F & Alcañiz, M 2005, *Ambient Intelligence*, IOS Press.

Rogers, Y 2006, 'Moving on from Weiser's Vision of Calm Computing: Engaging UbiComp Experiences', *LECTURE NOTES IN COMPUTER SCIENCE*, vol 4206, p. 404.

Rouillard, J 2004, *VoiceXML: le langage d'accès à Internet par téléphone*, Vuibert.

Rouillard, J 2008, 'Adaptation en contexte : Contribution aux interfaces multimodales et multicanal', Habilitation à Diriger des Recherches (HDR), Université des Sciences et Technologies de Lille1.

Rouillard, J, Vantroys, T & Chevrin, V 2007, *Architectures orientées services: une approche pragmatique des SOA*, Vuibert.

Rousseau, C 2006, 'Présentation multimodale et contextuelle de l'information', Ph.D. dissertation, Université d'Orsay Paris-Sud.

Rousseau, C, Bellik, Y, Vernier, F & Bazalgette, D 2006, 'A framework for the intelligent multimodal presentation of information', *Signal Processing*, vol 86, no. 12, pp. 3696-3713.

Salber, D & Coutaz, J 1993, 'Applying the wizard of oz technique to the study of multimodal systems', *Human-Computer Interaction*, pp. 219-230.

Salvador, T, Anderson, K, Dey, A & Schmidt, A,MJ 2003, 'Practical Considerations of Context for Context Based Systems: An Example from an Ethnographic Case Study of a Man Diagnosed with Early Onset Alzheimer's Disease', *Lecture Notes in Computer Science*, vol 2864, pp. 243-255.

Scafes, M & Badica, C 2009, 'Distributed Goal-Oriented Reasoning Engine for Multi-agent Systems: Initial Implementation', *Intelligent Distributed Computing III*, pp. 305-311.

Schmidt, A 2005, 'Ambient Intelligence', IOS Press.

Serrano, M, Nigay, L, Lawson, JY, Ramsay, A, Murray-Smith, R & Denef, S 2008, 'The OpenInterface framework: a tool for multimodal interaction.', *CHI'08 Extended abstracts on human factors in computing systems*, ACM, New York, NY, USA.

Shaer, O, Leland, N, Calvillo-Gamez, EH & Jacob, RJK 2004, 'The TAC paradigm: specifying tangible user interfaces', *Personal and Ubiquitous Computing*, vol 8, no. 5, pp. 359-369.

Sinnig, D 2008, 'Use case and task models: formal unification and integrated development methodology', Ph.D. dissertation, Concordia University.

Snoonian, D 2003, 'Smart buildings', *Spectrum*, vol 40, no. 8, pp. 18-23.

Sommer, R 1959, 'Studies in personal space', *Sociometry*, vol 22, no. 3, pp. 247-260.

Sousa, JP & Garlan, D 2002, 'Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments', Kluwer, BV.

Stanciulescu, A 2008, 'A Methodology for Developing Multimodal User Interfaces of Information Systems', Ph.D. dissertation, Université catholique de Louvain.

Stanciulescu, A, Limbourg, Q, Vanderdonckt, J, Michotte, B & Montero, F 2005, 'A transformational approach for multimodal web user interfaces based on UsiXML', *Proceedings of the 7th international conference on Multimodal interfaces*, ACM New York, NY, USA.

Stephanidis, C & Savidis, A 2001, 'Universal Access in the Information Society: Methods, Tools, and Interaction Technologies', *UAIS Journal*, vol 1, no. 1, pp. 40-55.

Stock, O, Zancanaro, M, Busetta, P, Callaway, C, Krüger, A, Kruppa, M, Kuflik, T, Not, E & Rocchi, C 2007, 'Adaptive, intelligent presentation of information for the museum visitor in PEACH', *User Modeling and User-Adapted Interaction*, vol 17, no. 3, pp. 257-304.

Svanaes, D & Verplank, W 2000, 'In search of metaphors for tangible user intefaces', *Proceedings of DARE 2000 on Designing augmented reality environments*, ACM.

Tan, N, Pruvost, G, Courgeon, M, Clavel, C, Bellik, Y & Martin, JC 2011, 'A location-aware virtual character in a smart room: effects on performance, presence and adaptivity', *Procedings of the 16th international conference on intelligent user interfaces*, ACM.

Tarpin-Bernard, F 2006, 'Interaction Homme-Machine Adaptative (HDR)', Ph.D. dissertation, INSA de Lyon / Université Claude Bernard Lyon I.

Tennenhouse, D 2000, 'Proactive computing', *Communications of the ACM*, vol 43, no. 5, pp. 43-50.

Thevenin, D 2001, 'Adaptation en Interaction Homme-Machine: le cas de la Plasticité', Ph.D. dissertation, Université Joseph-Fourier-Grenoble I.

Tigli, JY, Lavirotte, S, Rey, G, Hourdin, V, Cheung-Foo-Wo, D, Callegari, E & Riveill, M 2009, 'WComp middleware for ubiquitous computing: Aspects and composite event-based Web services', *Annals of Telecommunications*, vol 64, no. 3, pp. 197-214.

Traum, D & Larsson, S 2000, 'Information state and dialogue management in the TRINDI dialogue move engine toolkit', *Natural Language Engineering*, vol 6, pp. 323-340.

Truillet, P 1999, 'Modélisation de coopérations intermodales: application à l'interaction non visuelle', Ph.D. dissertation, Université de Toulouse 3.

UIMS 1983, 'User Interface Management Systems', *Workshop on User Interface*, Seeheim, Germany.

Ullmer, B & Ishii, H 2000, 'Emerging frameworks for tangible user interfaces', *IBM Systems Journal*, vol 39, no. 3, pp. 915-931.

Ullmer, B, Ishii, H & Glas, D 1998, 'mediaBlocks: physical containers, transports, and controls for online media', *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM.

Urbieta, A, Barrutieta, G, Parra, J & Uribarren, A 2008, 'A survey of dynamic service composition approaches for ambient systems', *ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering)*.

Van Helvert, J, Hagras, H & Kameas, A 2010, 'Prototype testing and validation - Deliverable D27', Tech. rep., The ATRACO Project (FP7/2007-2013 grant agreement n° 216837).

Van Kranenburg, R 2007, 'The Internet of Things, A critique of ambient technology and the all-seeing network of RFID', *Network Notebooks*, vol 2.

Vanderdonckt, J, Grolaux, D, Van Roy, P, Limbourg, Q, Macq, B & Michel, B 2005, 'A design space for context-sensitive user interfaces', *Proceedings of IASSE*.

Vertegaal, R & Poupyrev, I 2008, 'Organic user interfaces', *COMMUNICATIONS OF THE ACM*, vol 51, no. 6, pp. 26-30.

Vredenburg, K, Mao, JY, Smith, PW & Carey, T 2002, 'A survey of user-centered design practice', *Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves*, ACM New York, NY, USA.

Wagner, C & Hagras, H 2010, 'An approach for the generation and adaptation of zSlices based general type-2 fuzzy sets from interval type-2 fuzzy sets to model agreement with application to Intelligent Environments', *IEEE International Conference on Fuzzy Systems (FUZZ)*, IEEE.

Wagner, C, Hagras, H & Bilgin, A 2010, 'Artefact Operation Adaptation Component - Deliverable D14', Project deliverable, The ATRACO Project (FP7/2007-2013 grant agreement n° 216837).

Wagner, C, Hagras, H & Bilgin, A 2010, 'User Adaptation Behaviour Component - Deliverable D15', Project deliverable, The ATRACO Project (FP7/2007-2013 grant agreement n° 216837).

Waldner, JB 2007, *Nano-informatique et intelligence ambiante: inventer l'ordinateur du XXI siècle*, Lavoisier - Hermes Science.

Want, R, Pering, T, Danneels, G, Kumar, M, Sundar, M & Light, J 2002, 'The Personal Server: Changing the Way We Think About Ubiquitous Computing', *LECTURE NOTES IN COMPUTER SCIENCE*, pp. 194-209.

Weiser, M 1991, 'The Computer for the Twenty-First Century', *Scientific American*, vol 265, no. 3, pp. 94-104.

Weiser, M 1993, 'Some computer science issues in ubiquitous computing', *Communications of the ACM*, vol 36, no. 7, pp. 75-84.

Weiser, M 1994, 'Creating the invisible interface', *Proceedings of the 7th annual ACM symposium on User interface software and technology*, ACM.

Weiser, M & Brown, JS 1996, 'Designing Calm Technology', *PowerGrid Journal*, vol 1, no. 1, pp. 75-85.

Wilson, A & Shafer, S 2003, 'XWand: UI for intelligent spaces', *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM Press New York, NY, USA.

Wright, D, Gutwirth, S, Friedewald, M, Vildjiounaite, E & Punie, Y 2008, *Safeguards in a world of ambient intelligence*, Springer-Verlag New York Inc.

Yan, B & Huang, G 2009, 'Supply chain information transmission based on RFID and internet of things', *IEEE*.

Yu, L, Wang, N & Meng, X 2005, 'Real-time forest fire detection with wireless sensor networks', *Wireless Communications, Networking and Mobile Computing, 2005. Proceedings. 2005 International Conference on*, IEEE.