Numéro d'ordre : 8511

Université Paris XI
Faculté des sciences d'Orsay

THÈSE

présentée par

Christophe Jacquet

pour obtenir le grade de DOCTEUR EN SCIENCES
de l'Université de Paris-Sud XI Orsay

Spécialité : INFORMATIQUE

Présentation opportuniste et multimodale d'informations dans le cadre de l'intelligence ambiante

Soutenue le 14 décembre 2006 devant la commission d'examen suivante :

M. Yacine Bellik, encadrant
Mme Yolaine Bourda, encadrante
M. Joseph-Jean Mariani, examinateur
M. Philippe Palanque, rapporteur
M. Jean-Paul Sansonnet, directeur de thèse
M. Jean Vanderdonckt, rapporteur

Ce document a été mis en page avec le logiciel LaTeX 2_ε et la classe `memoir`.

Numéro de révision du document : 228.

Fichier PDF généré le 26 janvier 2007.

Remerciements

Je tiens à remercier :

Yacine Bellik et Yolaine Bourda, mes deux encadrants de thèse, qui m'ont guidé et conseillé au quotidien. Yacine m'a notamment fait découvrir le domaine de la communication homme-machine multimodale, et Yolaine m'a apporté son éclairage sur les aspects liés à la représentation des connaissances.

Jean-Paul Sansonnet, mon directeur de thèse, responsable du groupe Architecture et Modèles pour l'Interaction (AMI) du LIMSI, pour ses conseils, son inventivité, sa grande culture scientifique, et pour ses apports toujours originaux à nos discussions.

Joseph Mariani, directeur de recherche CNRS, ancien directeur du LIMSI, ancien directeur du département Technologies de l'Information et de la Communication du ministère de la recherche, pour s'être intéressé à mes travaux et avoir accepté de faire partie du jury de thèse.

Philippe Palanque, professeur en informatique à l'université de Toulouse 3 (Paul Sabatier), chercheur à l'IRIT (Institut de Recherche en Informatique de Toulouse) où il est responsable de l'équipe LIIHS (Logiciels Interactifs et Interaction Homme-Système), pour avoir accepté d'être rapporteur de ce travail.

Jean Vanderdonckt, professeur en informatique à l'université catholique de Louvain, directeur du Laboratoire belge d'Interaction Homme-Machine (BCHI), pour avoir accepté d'être rapporteur de ce travail.

Tous les membres du Département d'Informatique et du Centre de Ressources Informatiques de Supélec, pour leur bonne humeur, leur soutien et les discussions fructueuses que nous avons eues sur des sujets divers. En particulier, l'aide du technicien, Gilbert Dahan, m'a été précieuse pour la réalisation des circuits imprimés du système de badges présenté en annexe C.

Les personnes qui m'ont accueilli au LIMSI, et tout spécialement les membres du groupe AMI, qui m'ont permis d'évoluer dans un environnement scientifique très riche et diversifié.

À Séverine, à ma famille, à mes amis.

Table des matières

Chapitre 1

Introduction

Nos travaux s'inscrivent dans le domaine de l'informatique contextuelle et de la multimodalité. Nous nous intéressons à la présentation d'informations pertinentes à des utilisateurs mobiles, selon des modalités adaptées à leurs préférences, capacités et éventuels handicaps. Nous cherchons à doter les systèmes informatiques de comportements opportunistes, de façon à leur permettre d'interagir avec les utilisateurs humains de façon discrète et « intelligente ». En ce sens, nous suivons les paradigmes de l'intelligence ambiante.

1.1 Motivations

Les utilisateurs des lieux publics éprouvent souvent des difficultés à obtenir les informations dont ils ont besoin. Par exemple, lorsqu'un passager arrive dans un aéroport, il ne sait pas a priori où il doit se rendre pour prendre son avion. Ces difficultés sont d'autant plus grandes que l'environnement est inconnu. Cependant, elles sont toujours présentes : dans un aéroport, une personne, même habituée, peut difficilement deviner à l'avance où aura lieu son embarquement.

Afin de fournir aux utilisateurs les informations dont ils ont besoin, on installe donc généralement des *points d'information* de place en place. Ce peut être des écrans, des haut-parleurs, des kiosques interactifs, ou tout simplement des panneaux d'affichage. Par exemple, des moniteurs affichent des informations sur les vols dans un aéroport, des plans indiquent la position des boutiques dans un centre commercial, etc.

Cependant, ces points d'information fournissent des renseignements non ciblés, à destination de tout le monde. En conséquence, ils sont le plus souvent surchargés d'informations, ce qui rend leur lecture difficile. Or une personne donnée est généralement intéressée par un nombre très restreint d'informations (le plus souvent une seule) : la rechercher parmi une multitude d'informations inintéressantes est une tâche longue, voire fastidieuse.

Pour améliorer cette situation, et partant du principe qu'il ne sert à rien de présenter une information qui n'intéresse personne, nous nous proposons de concevoir un système

d'informatique ubiquitaire qui serait capable de fournir des informations personnalisées à des utilisateurs mobiles. Il ne s'agit pas pour autant de fournir des informations *personnelles*, mais plutôt d'opérer une *sélection* parmi toutes les informations disponibles, de sorte à ne fournir que les informations *pertinentes* pour les utilisateurs présents.

En ce sens, les informations pourraient être fournies par les dispositifs *publics* mentionnés ci-dessus. Ainsi, des moniteurs disposés au hasard dans un aéroport pourraient fournir des informations sur leurs vols aux passagers situés à proximité. Seules les informations à destination des passagers situés devant un écran donné seraient affichées, de façon à améliorer la lisibilité de ce dernier et à réduire la surcharge cognitive imposée aux utilisateurs.

Nous venons de le voir, tous les utilisateurs, quels qu'ils soient, sont confrontés à des difficultés lorsqu'il s'agit d'obtenir des informations et de se diriger dans un environnement inconnu. Cependant, il existe une catégorie de personnes pour lesquelles ces tâches sont particulièrement délicates : les handicapés. En effet, les dispositifs d'information ne leur sont pas nécessairement adaptés. Ainsi, un écran d'information ne sera d'aucune utilité à un non-voyant ; de même, un malentendant ne percevra pas les informations diffusées par haut-parleur.

Il faudrait donc que les dispositifs soient capables non seulement de fournir des informations pertinentes pour les utilisateurs concernés, mais également de n'utiliser que des modalités de sortie qui soient compatibles avec les modalités d'entrée de ces utilisateurs. De cette façon, on éviterait par exemple que l'information destinée à un non-voyant ne soit véhiculée par un moniteur vidéo.

1.2 Objectifs

Cette thèse vise donc à concevoir un modèle théorique et une plate-forme qui permettent la conception et l'implémentation d'applications d'assistance aux utilisateurs mobiles. Ces applications auront pour but de fournir des informations utiles aux personnes qui se déplacent dans un lieu donné, selon des modalités qui leur seront adaptées.

Nous nous proposons donc de placer la multimodalité au cœur du système : à la fois les utilisateurs et les dispositifs de présentation seront décrits par un *profil multimodal*, ce qui permettra lors de chaque interaction de choisir une instance de modalité adaptée.

Nous souhaitons prendre en compte à la fois des dispositifs d'interaction *privés* (i.e. portés par les utilisateurs), aussi bien que des dispositifs *publics* (i.e. présents dans l'environnement et accessibles à tous). Nous nous baserons donc sur la rencontre aléatoire[1] entre des utilisateurs et des dispositifs pour présenter de façon *opportuniste* des informations.

Dans ce travail, nous traitons de la transmission d'information par des dispositifs à des utilisateurs. Nous travaillons donc sur les interactions *en sortie*, et en particulier sur la

[1] Par contre, nous ne nous préoccuperons pas de savoir si cette rencontre aléatoire est *fortuite*, ou bien *provoquée* par un utilisateur qui se serait délibérément dirigé vers un dispositif interactif.

multimodalité *en sortie*. Nous ne donnons pas de précisions sur d'éventuelles interactions en entrée, qui ne font pas partie de notre sujet de recherche. Par contre, la prise en compte d'interactions en entrée pourrait très certainement constituer une perspective intéressante à notre travail.

1.3 Structure du mémoire

Ce mémoire est organisé en trois parties :

- la première partie présente un état de l'art dans les domaines des interactions homme-machine et de l'informatique contextuelle, de façon à aborder des notions qui seront utiles par la suite. Le chapitre 2 introduit des concepts de base en interfaces homme-machine : modèles d'architecture logicielle classiques, multimodalité, nouveaux concepts de l'intelligence ambiante, et particularités des interfaces pour les handicapés sensoriels. Le chapitre 3 s'intéresse à la notion de contexte et aux applications contextuelles. Il décrit en détail les problématiques liées à une composante particulièrement importante du contexte, la *localisation*. Enfin, il introduit les plates-formes classiques qui permettent l'accès au contexte ;

- la seconde partie présente notre travail théorique. À partir d'un cahier des charges (chapitre 4), nous introduisons notre modèle conceptuel, nommé KUP (chapitre 5). Le chapitre 6 expose deux algorithmes qui utilisent ce modèle KUP. L'un d'entre eux permet de choisir une instanciation de modalité pertinente pour l'interaction entre un utilisateur donné et un dispositif donné. L'autre permet de choisir un dispositif d'interaction lorsque plusieurs dispositifs sont disponibles ;

- enfin, la troisième partie présente notre implémentation et des résultats expérimentaux. Le chapitre 7 introduit une plate-forme qui implémente le modèle KUP et les algorithmes associés. Nous appelons cette plate-forme PRIAM, comme PRésentation d'Informations dans l'AMbiant. Cette plate-forme est conçue pour la réalisation d'applications réelles, conformes aux paradigmes exposés dans les chapitres 4, 5 et 6. Nous avons réalisé quelques-unes de ces applications, que ce soit sur un simulateur intégré à la plate-forme, ou grandeur nature. Le chapitre 8 présente ces réalisations, et donne les résultats expérimentaux obtenus.

Finalement, nous donnons une conclusion de ce travail, et ouvrons des perspectives pour les travaux à venir.

1.4 Collaboration

Ce travail de thèse a été réalisé en collaboration entre Supélec et le LIMSI-CNRS[2]. À Supélec, les travaux ont été menés au Département d'Informatique ; au LIMSI, ils ont pris place au sein du groupe AMI (Architecture et Modèles pour l'Interaction).

[2]Laboratoire d'Informatique pour la Mécanique et les Sciences de l'Ingénieur, unité propre de recherche du Centre National de la Recherche Scientifique.

Première partie

État de l'art

Chapitre 2

Techniques des interactions homme-machine

Le but de l'étude des interfaces homme-machine (IHM) est d'analyser la façon dont les utilisateurs humains interagissent avec les systèmes informatiques, ainsi que de proposer de nouvelles solutions pour améliorer ces interactions. Du point de vue des concepteurs d'applications, il s'agit également de proposer des outils qui permettent une réalisation plus facile d'applications mieux adaptées aux utilisateurs.

Dans ce chapitre, nous étudierons deux aspects des IHM qui nous seront utiles pour la suite : tout d'abord les différents modèles d'architecture logicielle qui ont été proposés pour les systèmes interactifs, ensuite les IHM multimodales. Ensuite, nous verrons en quoi les IHM doivent être spécialement adaptées pour les déficients sensoriels, et en quoi des IHM spécialement adaptées peuvent aider ces déficients dans leur vie quotidienne. Enfin, nous introduirons de nouveaux concepts liés à l'*intelligence ambiante*.

2.1 Modèles d'architecture logicielle

Les modèles d'architecture logicielle formalisent la façon dont l'utilisateur peut interagir avec le cœur procédural des applications (souvent appelé *noyau fonctionnel*) via une interface. Depuis les années 1970, différents modèles de ce type ont été proposés. Cette section se donne comme objectif d'en donner une liste concise.

MVC Apparu dans les années 1970, le modèle MVC (Modèle-Vue-Contrôleur) [Krasner 1988] a proposé en premier de formaliser les relations entre une interface homme-machine et les données et algorithmes sous-jacents. Développé par Trygve Reenskaug au PARC[1] de Xerox pour Smalltalk-80, MVC est devenu depuis un concept général, indépendant du langage utilisé. MVC introduit une séparation entre trois entités :

[1] Palo Alto Research Center, centre de recherche de Xerox situé à Palo Alto en Californie.

Le modèle : il correspond au noyau fonctionnel d'une application et à ses données métier. Par exemple, dans un programme de visualisation des données d'un recensement, le modèle contiendra toutes les données démographiques ;

La vue : il s'agit de la présentation qui est faite des données, à destination de l'utilisateur. Une vue consiste généralement en une interface (ou une partie d'interface) graphique ou textuelle. Dans l'exemple ci-dessus, une vue possible serait une carte sur laquelle des dégradés de couleurs représenteraient la densité de population ;

Le contrôleur : il est chargé de la gestion des actions en entrée effectuées par l'utilisateur, par exemple via une souris ou un clavier. Le contrôleur interprète ces actions, et peut, en réaction :

- soit modifier la vue associée : par exemple, l'utilisateur peut demander à zoomer sur une portion d'une carte ;
- soit modifier le modèle : par exemple, si l'utilisateur effectue une modification sur les données démographique d'une ville, le contrôleur la transmet au modèle.

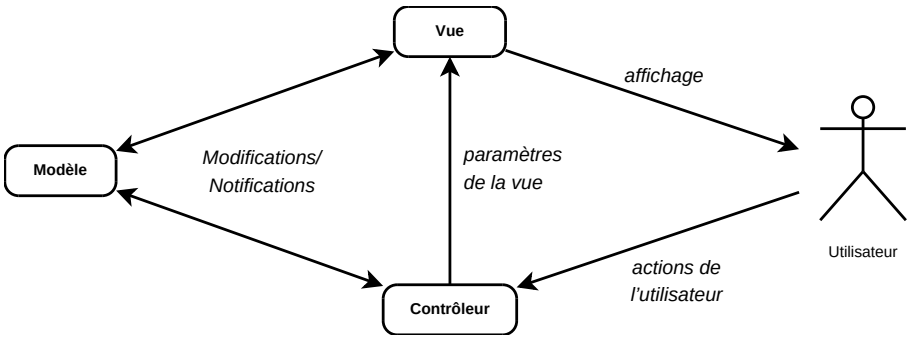


Figure 2.1 : *Architecture du modèle MVC.*

De ce fait, vue et contrôleur sont intimement liés, et ne sont généralement pas interchangeables. Ils peuvent même, dans certaines réalisations pratiques, être regroupés en une seule et même entité. Ainsi, si la bibliothèque Swing de Java distingue bien le modèle d'une part, elle regroupe d'autre part la vue et le contrôleur.

Il est possible qu'un modèle possède plusieurs de ces couples vue-contrôleur. Par exemple, à des données démographiques, il est possible d'associer une vue sous forme de carte de densité (et le contrôleur associé) comme évoqué précédemment, mais aussi une vue de l'évolution de la population sous forme d'histogrammes pour chaque ville, ou encore une carte des flux migratoires.

Ainsi, il est possible que le modèle se trouve modifié, soit par l'un quelconque des contrôleurs qui lui sont associés, soit par le noyau fonctionnel lui-même. Le modèle doit donc alors informer chacune de ses vues que ses données ont changé, afin qu'elles puissent le cas échéant mettre à jour leurs informations, par exemple en se redessinant. Pour cette raison, il existe dans MVC un mécanisme de notification qui permet au modèle de transmettre des notifications de modifications à l'ensemble de ses vues et contrôleurs.

Le schéma de la figure 2.1 récapitule le fonctionnement du modèle MVC.

Seeheim et ARCH Le modèle Seeheim [Pfaff 1985] part d'une analogie avec les langues, dans lesquelles il existe des niveaux sémantique (sens d'un discours), syntaxique (règles d'agencement des mots en phrases, des phrases en paragraphes, etc.) et lexical (mots pris individuellement). Par analogie, le niveau sémantique correspond au noyau fonctionnel ; le niveau syntaxique est chargé de l'ordonnancement des interactions ; le niveau lexical correspond à la présentation physique des informations (voir fig 2.2).

Cette architecture prévoit éventuellement une rétroaction directe du noyau fonctionnel vers la présentation physique, sans passer par le contrôleur de dialogue (bloc noté « court-circuit » sur la figure 2.2).

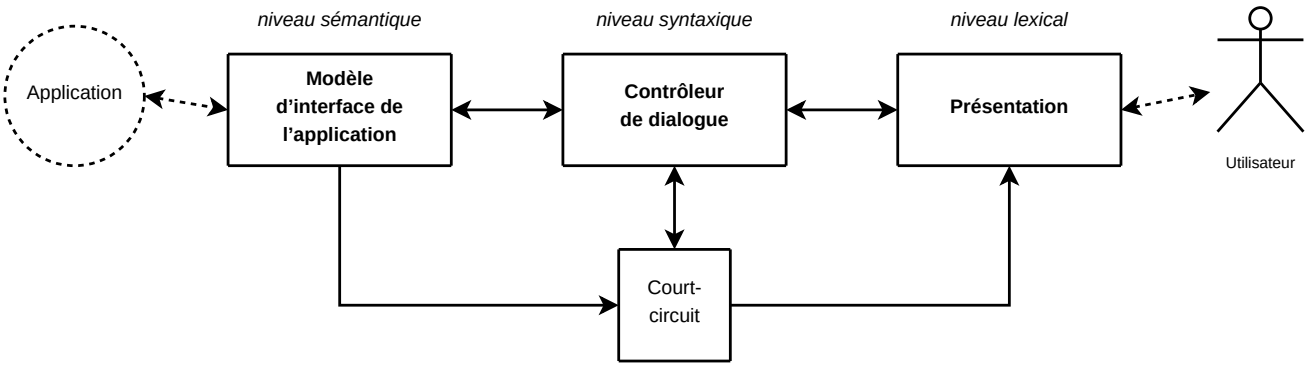


Figure 2.2 : *Architecture du modèle Seeheim.*

Le modèle ARCH [Bass 1992] raffine le modèle Seeheim. En effet, ARCH reprend les mêmes composants de base (noyau fonctionnel, contrôleur de dialogue, et interaction physique), mais ajoute deux composants intermédiaires (voir fig. 2.3) :

- l'adaptateur du noyau fonctionnel fait le lien entre le noyau fonctionnel et le contrôleur de dialogue ;
- un composant situé entre le contrôleur de dialogue et le composant d'interaction concrète se charge de mettre en œuvre des techniques de présentation.

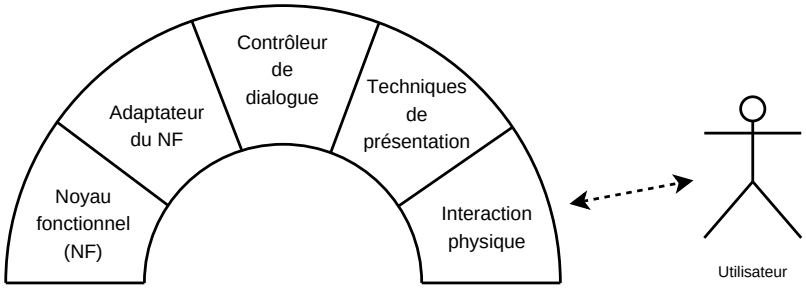


Figure 2.3 : *Architecture du modèle ARCH.*

PAC Le modèle PAC [Coutaz 1987, Nigay 1995] (Présentation, Abstraction, Contrôle) dérive du modèle MVC. Dans PAC, une hiérarchie d'agents communiquent entre eux, de façon à se déléguer les différentes sous-tâches d'un système interactif. Chaque agent PAC possède trois facettes :

Présentation : elle correspond aux vues et aux contrôleurs de l'architecture MVC, dont nous avons vu qu'ils étaient souvent liés, voire indissociables ;

Abstraction : elle correspond à la notion de modèle telle que définie pour MVC ;

Contrôle : elle fait le lien entre présentation et abstraction, et permet d'exprimer des dépendances entre ces deux facettes. De plus, la facette contrôle est chargée de la communication avec les autres agents PAC au sein de leur hiérarchie.

L'utilisateur interagit avec les facettes de présentation des différents agents PAC qui constituent un système (voir fig. 2.4).

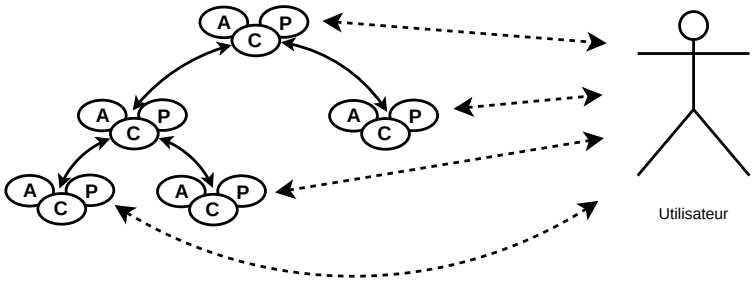


Figure 2.4 : *Architecture du modèle PAC.*

Résumé Dans les architectures que nous venons de présenter, deux composantes sont toujours présentes :

- le modèle des informations à présenter. Il représente l'abstraction du problème à traiter, sous forme de données métier. Il est directement lié au noyau fonctionnel d'une application donnée ;
- une implémentation concrète, voire physique, de l'interface avec l'utilisateur et des interactions qui lui sont associées.

De là, les différents modèles se distinguent par la présence éventuelle de couches intermédiaires, par exemple le contrôleur de dialogue du modèle ARCH, qui, aidé de deux couches d'adaptation, permet à l'utilisateur d'accéder aux fonctionnalités du noyau de l'application. La figure 2.5 indique la structure générale de tous les modèles d'architecture logicielle pour les interfaces évoqués plus haut.

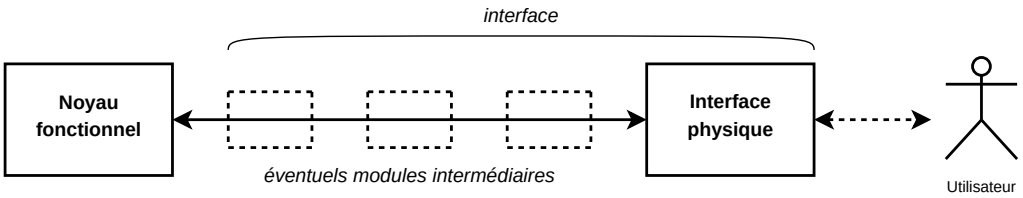


Figure 2.5 : *Structure générale des modèles d'architecture logicielle présentés ci-dessus. Un module d'interface physique présente à un utilisateur des informations issues d'un noyau fonctionnel, et transmet en retour les modifications effectuées par l'utilisateur à ce noyau fonctionnel.*

2.2 Multimodalité

2.2.1 Définitions

Nous posons tout d'abord quelques définitions de base en ce qui concerne les interactions multimodales [Bellik 1992] :

Mode en sortie. Par référence aux cinq sens des êtres humains, un mode en sortie correspond à la nature d'un moyen de communication, dans le sens de la machine vers l'homme : mode visuel, mode auditif, mode tactile, etc. Il existe également des *modes en entrée* (sens de l'homme vers la machine) : ces modes correspondent donc aux moyens d'expression des êtres humains (parole, gestes, etc.). Dans ce travail, nous nous intéressons aux interactions de la machine vers l'homme, donc seuls les modes *en sortie* sont pris en compte.

Modalité. Une modalité est une forme concrète particulière d'un mode. Par exemple, en ce qui concerne le mode auditif, on peut citer les modalités suivantes : parole, musique, bruit, chant, etc.

Média. Un média est un dispositif informatique qui permet l'interfaçage de la machine avec des utilisateurs, en entrée ou en sortie, selon une ou des modalités données. Un média sert donc à *véhiculer une modalité*. Par exemple, un écran permet de véhiculer des modalités visuelles, un système de synthèse vocale (haut-parleurs compris) la modalité parole synthétique, etc.

Multimodalité. On dit qu'un système est *multimodal* s'il utilise plusieurs modalités pour ses interactions.

On considère généralement que les êtres humains possèdent cinq sens : vue, ouïe, toucher, goût et odorat. Il peut donc théoriquement y avoir cinq modes en sortie. Cependant, le goût n'est jamais exploité en IHM, et les quelques études menées sur l'odorat n'ont jamais dépassé le stade de l'expérimentation. Nous considérerons donc qu'il existe trois modes en sortie : visuel, auditif et *tactilo-proprio-kinesthésique* (TPK). On préfère généralement ce dernier

terme à *tactile* car il englobe aussi bien le sens du toucher que l'intuition de la position de ses membres (*proprioception*).

2.2.2 Taxonomie des modalités

Dans la section 2.2.1, nous avons donné quelques exemples de modalités. Cependant, nous aurons besoin dans la suite d'une classification « complète » des modalités. Bien entendu, il ne s'agit pas de donner une description *exhaustive* des modalités, car il est toujours possible d'en imaginer de nouvelles, mais plutôt de définir clairement un ensemble de modalités que nous prendrons en compte tout au long de ce document.

Plutôt que de réinventer notre propre taxonomie, nous nous basons sur la liste de modalités établie par N. Berrami [Berrami 2001]. De cette classification, nous ne conservons que les modalités effectivement utilisables dans une interaction homme-machine.

La figure 2.6 présente, sous forme d'un diagramme UML[2] (diagramme de classes), les modalités que nous prenons en compte. Seules les feuilles de l'arbre correspondent effectivement à des modalités ; les nœuds internes représentent tous des classes abstraites de modalités.

De façon classique, les modalités sont réparties en trois modes utilisables en sortie : visuel, auditif et TPK. Le premier niveau de modalités abstraites au sein de la hiérarchie correspond donc à ces trois modes. Nous considérons que chaque mode peut être exploité selon différentes modalités, qui sont présentes aux niveaux hiérarchiques inférieurs. Cependant, nous nous interdisons de rattacher une modalité à plusieurs modes.

En effet, Rousseau [Rousseau 2005], s'il permet à une modalité d'être rattachée à plusieurs modes, définit néanmoins une notion de *relation principale* entre une modalité et un mode donné. Par exemple, les vibrations d'un téléphone portable sont *principalement* liées au mode TPK, même si on peut les percevoir via les modes auditif (grondement sourd) et visuel (mouvements du téléphone s'il est posé sur une table). De même, l'écriture Braille est principalement destinée à être perçue par le toucher (par les non-voyants), mais rien n'empêche les personnes voyantes de lire visuellement des textes en Braille.

Nous ne prenons donc en considération que le *mode principal* d'une modalité . De cette façon, lorsque nous choisirons une modalité pour la présentation d'une information, nous pourrons nous assurer que cette dernière sera bien perçue par l'utilisateur. Si éventuellement la modalité choisie « déborde » sur un autre mode, cela ajoutera éventuellement une perception *périphérique* dont nous ne nous préoccupons pas.

Éventuellement, s'il avait existé une modalité qui aurait pu être rattachée à égalité à plusieurs modes (pour elle, plusieurs modes pourraient être considérés comme étant *aussi principaux les uns que les autres*), nous aurions de toute façon pu la *dédoubler*, et donc la faire figurer dans plusieurs branches à la fois.

[2] Unified Modeling Language.

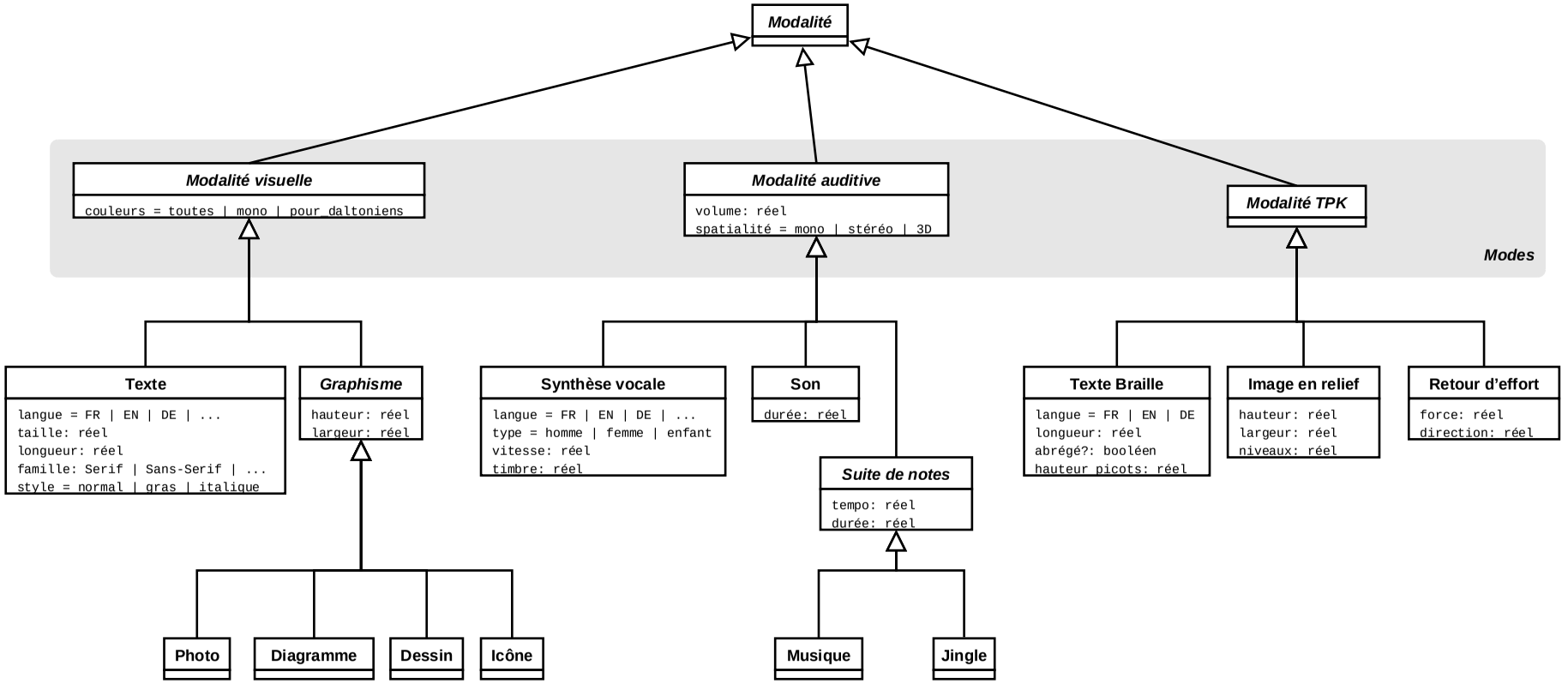


Figure 2.6 : *Diagramme UML présentant la taxonomie des modalités. Les trois modalités abstraites situés au premier niveau après la racine de l'arbre (ici sur fond grisé) correspondent aux trois* modes *pris en compte.*

Comme nous l'avons vu ci-dessus, une modalité détermine la nature et la forme d'une interaction. Par exemple, une information peut être présentée visuellement (*nature* de la présentation, qui correspond au *mode* utilisé), sous *forme* d'un texte (*modalité* en elle-même). Cependant, en indiquant une modalité, on est loin de déterminer de façon précise la forme de la présentation. Si l'on reprend l'exemple précédent, il manque (au moins) les informations suivantes :

- la taille des caractères ;
- leur couleur ;
- la fonte de caractères utilisée ;
- le texte en lui-même ;

Dans cet exemple, les trois premiers points sont ce qu'on appelle des *attributs*. Pour une modalité donnée, les attributs permettent de définir de façon plus précise certains *paramètres* de la présentation. Le dernier point correspond quant à lui au *contenu concret* de la présentation : il porte une bonne partie de la sémantique[3].

Les modalités sont donc munies d'attributs, qui peuvent être de deux sortes :

- *discrets* : l'attribut prendra une valeur parmi un ensemble fini de valeurs ;
- *continus* : l'attribut prendra une valeur sur un intervalle de $\mathbb{R}$.

Pour les modalités dépendantes de la langue (texte, synthèse vocale), la langue est spécifiée par un attribut discret de la modalité. Nous considérons donc qu'un changement de langue n'implique pas un changement de modalité, mais seulement un changement de la valeur de cet attribut.

Ce qui nous intéresse dans notre travail, ce ne sont pas les attributs des modalités en termes de description (exemple : *« le texte est écrit en rouge, avec une fonte Helvética de corps 32 »*), mais plutôt en termes de contraintes sur la génération de la présentation (exemple : *« la taille doit être au moins 10 cm pour cet utilisateur situé à quatre mètres, et il faut utiliser un jeu de couleurs adaptées au daltonisme »*, si l'on veut générer un contenu destiné à un daltonien myope).

Ce sont ces considérations qui ont présidé à notre choix des attributs des modalités. Ceux-ci sont détaillés ci-dessous, et résumés par la figure 2.6.

[3]Cependant, il ne porte pas la totalité de la sémantique. Imaginons par exemple un écriteau où « STOP » est inscrit en rouge, à l'aide de lettres grasses. Dans ce cas, une partie de la sémantique est également portée par la couleur et le choix de la fonte. Pour s'en convaincre, il suffit d'imaginer l'effet que ferait un écriteau « STOP » écrit en lettres vertes, fines et italiques...

Modalités visuelles Il est possible de dénombrer un certain nombre d'attributs qui sont communs à toutes les modalités de type visuel. Ces attributs sont donc caractéristiques de la perception visuelle. Dans le cadre de notre taxonomie, nous n'avons pris en considération qu'un seul de ces attributs communs : l'attribut *couleur*.

Cet attribut indique la façon dont sont utilisées les couleurs pour réaliser une présentation. De façon classique, cet attribut peut donc prendre les valeurs « monochrome » ou « en couleur », avec éventuellement une précision de la profondeur de couleur (16 bits, 24 bits, etc.). Cependant, si nous voulons effectuer une présentation à destination d'une personne atteinte de daltonisme, nous sommes dans ce cas confrontés au dilemme suivant :

- soit nous présentons les informations en noir et blanc, ce qui nous conduit malheureusement à nous passer des bénéfices de la couleur en termes de clarté des informations ;
- soit nous présentons les informations en couleurs, mais dans ce cas, nous courons le risque de les rendre illisibles par la personne daltonienne.

Afin de résoudre ce dilemme, nous introduisons une troisième valeur possible pour cet attribut, « couleurs adaptées au daltonisme » : de cette façon, nous pouvons utiliser plus de deux couleurs, sans pour autant risquer de produire une présentation illisible pour les daltoniens.

Le tableau 2.1 détaille les modalités visuelles et leurs attributs.

Modalité	**Attribut**	**Type**	**Description**
Texte	langue	français, anglais, allemand, etc.	langue du texte
	taille	nombre (cm)	taille des caractères
	longueur	nombre (caractères)	longueur du texte
	famille	nom de fonte ou de famille	fonte ou famille des caractères (avec ou sans empattements, fantaisie, etc.)
	style	normal, gras, italique, etc.	style des caractères
Graphisme	hauteur	nombre (cm)	hauteur du graphisme
	largeur	nombre (cm)	largeur du graphisme
+ Photo	*sous-modalité de* Graphisme		
+ Diagramme	*sous-modalité de* Graphisme		
+ Icône	*sous-modalité de* Graphisme		
+ Dessin	*sous-modalité de* Graphisme		

Tableau 2.1 : *Modalités visuelles. Dans ce tableau, les modalités indentées sont considérées comme des sous-modalité de la modalité de niveau supérieur.*

Modalités auditives De même que précédemment, nous avons identifié deux attributs associables à toutes les présentations de type auditif :

- niveau sonore : niveau de pression acoustique à 1 m, en dB ;
- spatialité : monophonique, stéréophonique, 3D.

Les modalités auditives sont présentées dans le tableau 2.2.

Modalité	**Attribut**	**Type**	**Description**
Synthèse de parole	langue	français, anglais, allemand, etc.	langue du texte synthétisé
	type	homme, femme, enfant	type de voix
	vitesse	nombre (syllabes par seconde)	vitesse d'élocution
	timbre	nombre	timbre de la voix
Suite de notes	tempo	nombre	tempo
	durée	nombre (secondes)	durée
+ Musique	*sous-modalité de* Suite de notes		
+ Jingle	*sous-modalité de* Suite de notes		
Bruit du monde réel	durée	nombre (secondes)	durée

Tableau 2.2 : *Modalités auditives.*

Modalités TPK Nous n'avons pas identifié d'attribut associé intrinsèquement au mode TPK. Les modalités de ce mode sont présentées dans le tableau 2.3.

2.2.3 Relations entre modalités

Un système multimodal peut utiliser plusieurs modalités différentes en entrée et/ou en sortie. On peut alors se demander quels rapports ces modalités entretiennent les unes aux autres, et la façon dont leur combinaison peut être mise à profit par le système.

2.2.3.1 Propriétés CARE

Les propriétés CARE (Complémentarité, Assignation, Redondance, Équivalence) [Coutaz 1995, Nigay 1997] permettent de décrire la façon dont les modalités sont utilisées dans un système multimodal. Elles introduisent une caractérisation des relations qu'entretiennent entre elles les modalités. Notons que dans les propriétés CARE, une modalité est définie comme un couple ⟨ langage d'interaction, dispositif ⟩.

Modalité	**Attribut**	**Type**	**Description**
Texte Braille	langue	français, anglais, allemand, etc.	langue du texte affiché en Braille
	longueur	nombre (caractères)	longueur du texte
	abrégé?	booléen	précise l'utilisation on non du Braille abrégé
	hauteur des picots	nombre (mm)	hauteur des picots
Image en relief	hauteur	nombre (cm)	hauteur de l'image
	largeur	nombre (cm)	largeur de l'image
	niveaux	nombre	nombre de niveaux de relief possibles
Retour d'effort	force	nombre (N)	force de résistance
	direction	nombre (angle)	direction selon laquelle est appliquée la force

Tableau 2.3 : *Modalités de type TPK.*

La liste suivante donne les propriétés CARE pour les modalités.

Équivalence. Deux modalités sont dites équivalentes s'il est possible d'effectuer une tâche donnée indifféremment au moyen de l'une quelconque de ces deux modalités. L'équivalence est dite *totale* si les deux modalités sont équivalentes vis-à-vis de *toutes* les tâches d'un système donné. Dans le cas contraire, elle est dite *partielle*.

Assignation. On dit qu'une modalité est assignée à une tâche donnée si cette dernière ne peut être effectuée qu'à l'aide de cette modalité. Autrement dit, il n'existe pas de modalité équivalente pour la tâche considérée.

Complémentarité. Des modalités sont dites complémentaires si elles doivent être utilisées en même temps pour la réalisation d'une tâche. Plus formellement, l'expression de la tâche se décompose en un ensemble de sous-expressions dont chacune utilise une et une seule modalité. Ce sont ces modalités qui sont déclarées *complémentaires*.

Redondance. La redondance est un sous-cas de l'équivalence. Deux modalités sont dites redondantes si elles sont équivalentes, et si elles peuvent être utilisées simultanément ou successivement pour l'expression d'une tâche donnée.

2.2.3.2 Multimodalité exclusive

Nous venons de voir que les propriétés CARE permettent (entre autres) de caractériser les systèmes qui font usage de *plusieurs modalités à la fois*. Dans ce cas, ces systèmes sont dits multimodaux car ils sont capables de combiner plusieurs modalités pour effectuer une interaction donnée.

Or, on peut également qualifier de *multimodaux* des systèmes conçus pour utiliser *une* modalité parmi un certain nombre de modalités possibles. Par exemple, en sortie, une présentation donnée peut n'être réalisée que selon une seule modalité bien définie. Bien que les présentations effectuées par ce genre de systèmes ne comportent qu'une seule modalité, ces systèmes sont bel et bien multimodaux, car la modalité peut changer lors de chaque présentation pour une information donnée (selon le contexte). On parle alors de *multimodalité exclusive* [Teil 2000, Coutaz 1991].

2.2.4 Élaboration d'une présentation multimodale

2.2.4.1 Sélection des modalités

Dans un système multimodal, lorsqu'on doit présenter une information, il est nécessaire de déterminer la ou les modalités utilisées. Généralement, l'information est en premier lieu décomposée en un ensemble d'unités d'information élémentaires : c'est ce qu'on appelle la *fission* [Rousseau 2003].

Une fois cette décomposition effectuée, il est alors possible d'*élire* une modalité pour chacune des unités d'information élémentaires. L'ensemble des modalités choisies pour chacune des unités d'information permet de spécifier (partiellement, voir ci-dessous) la présentation multimodale. Cependant, se pose la question de savoir *comment* sont choisies les modalités pour chaque unité d'information. Élisabeth André [André 2000] identifie un certain nombre de facteurs qui influent sur ce choix :

- les caractéristiques de l'information à présenter ;
- les caractéristiques des modalités ;
- les buts de l'entité présentatrice ;
- les caractéristiques de l'utilisateur ;
- la tâche assignée à l'utilisateur ;
- les limitations portant sur les ressources.

En pratique, il est possible d'établir des règles heuristiques pour effectuer ce choix, par exemple :

- préférer les graphismes par rapport au texte pour les informations spatiales, sauf si le besoin de précision l'emporte sur la vitesse de lecture, auquel cas le texte est préférable ;
- utiliser le texte pour les informations quantitatives ;
- etc.

2.2.4.2 Instanciation des modalités

Dans notre taxonomie des modalités, nous avons introduit la notion d'*attribut*. Ainsi, sélectionner une modalité ne suffit pas à déterminer complètement la forme d'une présentation : il faut également fixer des valeurs aux attributs.

Prenons en exemple un écran d'information situé dans un lieu public. Supposons que les utilisateurs se situent à une distance fixée de l'écran. Si l'on considère que la taille des caractères affichés est fixe, les concepteurs de l'écran sont confrontés au dilemme suivant :

1. si la taille des caractères est trop petite, alors les utilisateurs malvoyants, ou tout simplement myopes, ne pourront pas lire les informations. Il faudrait donc utiliser de gros caractères ;

2. mais si la taille des caractères est trop grande, il ne sera pas possible d'afficher beaucoup d'informations. Il faudra peut-être même faire défiler le texte, ce qui permet certes de faire passer des messages relativement longs, mais se révèle assez contraignant pour les utilisateurs. Du point de vue des utilisateurs ayant une bonne vue, il faudrait donc de préférence utiliser d'assez petits caractères.

On le voit, un bon compromis est dans ce cas difficile à trouver, et laissera très certainement une catégorie d'utilisateurs lésés. En revanche, s'il était possible d'*adapter* la taille des caractères en fonction des utilisateurs, le dilemme évoqué ci-dessus ne se poserait plus : l'écran utiliserait de grands caractères pour les malvoyants (car c'est la seule solution possible), et de petits caractères pour les autres (de façon à pouvoir afficher toutes les informations sans devoir recourir au défilement).

Nous appelons *instanciation* le choix des attributs de la modalité [André 2000]. Cette opération est appelée *media design* dans [Rist 2005].

L'instanciation est une étape a priori complexe, car le nombre de combinaisons possibles des attributs peut rapidement être grand. Reprenons ainsi l'exemple précédent, en supposant que la taille des caractères peut prendre 20 valeurs différentes, qu'il existe 16 couleurs différentes dans la palette disponible, et qu'il est possible de choisir entre 4 fontes différentes. Cela fait $20 \times 16 \times 4 = 1280$ instanciations possibles, même pour cet exemple simple. La situation serait encore pire si on décidait de plus d'admettre des variations continues pour les attributs.

On le voit, réaliser l'instanciation correspond à résoudre un problème à forte combinatoire. Cependant, les bénéfices sont non négligeables, en terme d'adéquation des présentations avec les capacités physiques des utilisateurs (acuité visuelle et auditive, daltonisme, etc.). En pratique, même s'il est nécessaire de parcourir un grand nombre de combinaisons d'attributs à la recherche d'une bonne solution, le problème n'en est pas pour autant insoluble, en raison des capacités de traitement des ordinateurs actuels. Il est donc envisageable d'effectuer de façon automatique l'instanciation d'une modalité.

Cependant, il en est tout autre du choix du *contenu concret* d'une modalité. Des travaux ont certes été menés dans les domaines de la génération de textes [Varile 1996, Zock 2002] ou d'images, mais ils ne permettent pas encore de générer à coup sûr un contenu concret qui convienne à des utilisateurs réels dans le cas général. Dans nos travaux, nous nous sommes donc limités à l'instanciation des modalités, et à la sélection d'un contenu concret parmi une liste de contenus préétablis (voir section 5.3.2).

2.2.4.3 Sélection des dispositifs

Une fois une instance de modalité sélectionnée, il est nécessaire de déterminer par quel dispositif elle doit être présentée. Ainsi, il est nécessaire de choisir une *disposition*, un *agencement* des modalités sur les différents dispositifs disponibles, ou éventuellement au sein d'un même dispositif. Par exemple, si l'on dispose de deux informations visuelles et de deux écrans, il est nécessaire soit d'affecter chacune des instanciations de modalité à chaque écran, soit de déterminer comment les deux instanciations peuvent être affichées sur le même écran, en réglant notamment les problèmes de partage d'espace.

Notons qu'il est nécessaire d'effectuer une *coordination spatio-temporelle* des sorties des différents dispositifs [Bordegoni 1997]. Cette étape est très importante, mais il n'existe pas de mécanismes universels capables de l'accomplir dans tous les cas.

En général, dans les systèmes multimodaux, un module est spécialement dédié à la sélection des dispositifs et à l'agencement des modalités. Ainsi, dans le modèle WWHT [Rousseau 2005], cette tâche est réalisée par un *module de gestion des présentations multimodales*.

2.3 Prise en compte des handicaps sensoriels

Dans le cadre de la conception d'un système adapté à tous types d'utilisateurs, nous devons notamment prendre en compte les handicaps sensoriels dont ils peuvent souffrir. Cette section présente les problèmes majeurs générés par les deux principaux types de handicaps sensoriels (handicap visuel et handicap auditif), les solutions mises en œuvre pour y remédier, ainsi que les particularités liées aux IHM à destination des utilisateurs handicapés.

2.3.1 Importance du handicap

D'après l'Organisation Mondiale de la Santé, il y a 45 millions de personnes aveugles dans le monde, ce qui représente, selon les estimations, 1 ‰ à 2 ‰ de la population des pays industrialisés. La surdité est encore plus répandue : en France, on dénombre 500.000 personnes sourdes, et 4 millions de personnes malentendantes (soit plus de 6 % de la population).

Globalement, on estime que 10 % des Européens, soit 38 millions de personnes, souffrent d'un handicap. On ne peut donc pas négliger cette population lors de la conception d'espaces

publics, d'autant que ces populations rencontrent des difficultés particulières lors de leurs déplacements et de l'utilisation des lieux de vie.

2.3.2 Handicap visuel

Adaptation des IHM Les IHM les plus classiques mettent en œuvre des écrans, des claviers et des souris. Ces interfaces posent particulièrement problème pour les personnes non-voyantes ou malvoyantes. Ainsi, un malvoyant aura du mal à lire les informations affichées à l'écran ; un non-voyant ne pourra pas du tout utiliser l'écran ou la souris, et aura des difficultés à trouver les touches sur le clavier.

Pour pallier ces difficultés, il est possible de communiquer avec les aveugles par l'intermédiaire de la modalité auditive, ou bien de la modalité tactile. Par exemple, on peut penser à :

- des systèmes de synthèse et de reconnaissance vocale ;
- des claviers où les touches comportent des caractères Braille en relief ;
- des lignes de texte dynamiques en caractères Braille ;
- des imprimantes en relief ;
- des souris à pavés tactiles[4].

Ainsi, ces deux modalités se prêtent bien aux informations de type textuel. Cependant, toutes les informations ne sont pas de nature purement textuelle. Par exemple, la présentation des formules mathématiques a donné lieu à des travaux bien spécifiques [Podevin 2002].

Difficultés de déplacement Dans leur vie de tous les jours, la principale difficulté rencontrée par les non-voyants est liée à leurs déplacements. En ville, les rues, les transports publics et les centres commerciaux représentent des environnements hostiles en perpétuel changement. En conséquence, les personnes aveugles peuvent se sentir en situation de danger lorsqu'elles se déplacent seules, ce qui limite leur autonomie.

En effet, si les non-voyants connaissent bien en général le parcours à suivre pour se rendre dans quelques endroits connus, ils ne peuvent en revanche pas prévoir à l'avance les obstacles inopinés qui pourraient se présenter. Même s'ils utilisent une canne blanche, ils ne peuvent percevoir les obstacles silencieux[5] qu'au bout de leur canne, donc leur capacité d'anticipation s'en trouve très limitée [Jansson 1991]. Au final, la peur de l'inconnu conduit souvent ces aveugles à restreindre leur univers à un petit nombre d'endroits familiers. Ils n'osent pas s'aventurer ailleurs, ce qui limite fortement leur liberté de déplacement.

[4] Par exemple, la souris VTPlayer de Virtouch, voir `http://www.virtouch2.com/Features_and_Specs.htm`.

[5] i.e. qui n'émettent pas de bruit permettant de les déceler.

Les chiens-guides constituent une aide précieuse pour éviter les obstacles et trouver son chemin en environnement inconnu. En effet, les chiens perçoivent les obstacles à distance, et peuvent ainsi anticiper les manœuvres d'évitement pour eux-mêmes et donc également pour la personne non-voyante qu'ils accompagnent. Ainsi, les chiens peuvent améliorer les performances d'anticipation de leurs maîtres, ce qui conduit directement à des trajectoires plus fluides, à des déplacement plus aisés, et à une bien plus grande confiance en soi [Blash 1991].

Cependant, les chiens-guides sont très onéreux : le coût de formation pour un binôme personne-chien est compris entre 15.000 et 30.000 euros. En dépit du soutien financier de certaines associations comme le Lions Club, rares sont les aveugles qui peuvent disposer d'un chien-guide en pratique, alors que les demandes sont nombreuses.

Pour cette raison, de nombreux systèmes d'aide au déplacement des non-voyants ont été proposés. En général, ce sont des *détecteurs d'obstacles* qui avertissent les utilisateurs *à l'avance*, de façon à leur donner la capacité à *anticiper* la présence d'obstacles et à adapter leur comportement en conséquence. Ces systèmes sont beaucoup moins onéreux que les chiens-guides, et ils fournissent des informations très utiles, sans toutefois égaler les performances des chiens. Les informations peuvent être fournies sous forme auditive (notes de musique dans une oreillette pour indiquer la distance), ou bien sous forme tactile grâce à des vibreurs fixés aux doigts [Jacquet 2004c].

Outre des détecteurs de distance, il est également possible de mettre en place des systèmes de repérage, qui aident à *localiser* des lieux précis. Ainsi, dans le projet RAMPE [Baudoin 2005], les arrêts d'autobus émettent des sons lorsqu'ils détectent des aveugles à proximité, de façon à guider ces derniers.

2.3.3 Handicap auditif

Une personne jeune perçoit des sons à des niveaux de pression acoustique[6] variant entre 0 dB (seuil de l'audition) et 120 dB (seuil de la douleur). Cependant, le seuil de l'audition se détériore avec l'âge, et ceci d'autant plus que la fréquence est élevée : on parle de *presbyacousie* [Pireaux 1964].

Kavanagh [Kavanagh 2001] reprend les résultats de Spoor [Spoor 1967] et Robinson & Sutton [Sutton 1979] (ces dernières sont à la base de la norme ISO[7] 1999) : on constate qu'à 60 ans, la sensibilité auditive est notablement diminuée (voir tableau 2.4).

À première vue, les handicaps auditifs peuvent sembler moins cruciaux dans les déplacements des personnes que les handicaps visuels. En effet, pour les déplacements élémentaires, la vue est le sens le plus sollicité. De plus, les directions sont le plus souvent indiquées par des panneaux, donc par des modalités visuelles.

Cependant, comme nous venons de le voir, les handicaps auditifs touchent toute la population à partir d'un certain âge, sans qu'ils soient forcément corrigés par des appareils adaptés

[6] Pour les définitions concernant l'acoustique, voir page 137.
[7] International Organization for Standardization.

Fréquence (en Hz)	**Robinson** [Sutton 1979]		**Spoor** [Spoor 1967]	
	Hommes	*Femmes*	*Hommes*	*Femmes*
250	5,3	5,3	6,9	7,3
500	6,2	6,2	7,7	8,1
1000	7,1	7,1	7,8	8,4
2000	13,4	10,6	14,9	12,3
3000	20,3	13,2	22,2	15,4
4000	28,2	15,9	28,4	18,8
6000	31,8	21,2	33,3	24,7
8000	38,8	26,5	35,2	25,3

Tableau 2.4 : *Seuil de l'audition à 60 ans, exprimé en dB, selon deux études. Ces valeurs sont à comparer au seuil fixé à 0 dB pour des sujets jeunes.* Tableau tiré de [Kavanagh 2001].

(mais onéreux). Même si elles peuvent effectuer leurs déplacements habituels, les personnes qui souffrent de ces handicaps peuvent être gênées de ne pas percevoir les informations transmises par voie auditive. Par exemple, dans une gare ou un aéroport, des annonces importantes peuvent passer par haut-parleurs, comme des informations sur des retards ou des modifications d'heures et de lieux de départ.

2.3.4 Conclusion : importance des handicaps sensoriels

De ce qui précède, il est possible de tirer les deux conclusions suivantes :

1. les handicapés visuels peuvent éventuellement disposer de moyens d'éviter et d'anticiper les obstacles, mais pour obtenir des informations de plus haut niveau (trouver leur chemin, obtenir des horaires d'ouverture, les heures de départ des trains), ils en sont toujours réduits à compter sur la bonne volonté des passants non-handicapés ;

2. le handicap auditif concerne *tout le monde* à partir d'un certain âge, et même s'il est moins gênant pour se déplacer que le handicap visuel, il peut tout de même fortement perturber la vie des personnes qui en souffrent, et les priver de certaines informations.

Pour ces raisons, il nous semble donc important de prendre en compte les handicaps sensoriels, aussi bien visuel qu'auditif, dans la conception d'un système d'aide au déplacement.

2.4 Vers l'intelligence ambiante

Longtemps, les ordinateurs ont été des appareils que l'on utilisait exclusivement au bureau, à l'aide d'une interface relativement figée : un clavier, un écran, et plus récemment une souris. Cependant, on imagine depuis de nombreuses années que les ordinateurs pourraient à

l'avenir accompagner les utilisateurs dans leurs déplacements (informatique mobile), et même être miniaturisés et embarqués dans les objets du quotidien de façon à les doter de capacités supplémentaires.

Augmentation Ainsi, on introduit la notion d'*objets augmentés* : ce sont des objets en apparence classiques, qui conservent leurs fonctions habituelles, mais disposent de fonctionnalités supplémentaires apportées par l'informatique. Par exemple, un four qui calcule automatiquement la durée de cuisson des aliments peut être qualifié d'*augmenté* : il garde sa fonction originale, à laquelle il ajoute une nouvelle fonctionnalité.

Nous distinguons deux types d'augmentation :

- lorsque le dispositif technique à l'origine de l'augmentation est inclus dans l'objet augmenté, nous parlons d'*augmentation intrinsèque*. Ainsi, le four cité ci-dessus est augmenté de façon intrinsèque ;
- lorsque par contre le dispositif à l'origine de l'augmentation est extérieur à l'objet augmenté, nous parlons d'*augmentation extrinsèque*. Par exemple, des lunettes à affichage intégré peuvent servir à fournir des informations en superposition lorsque les visiteurs d'un musée regardent les œuvres exposées. Dans ce cas, nous disons que les œuvres sont augmentées de façon extrinsèque.

Lorsque l'augmentation est intrinsèque, tous les utilisateurs perçoivent *conjointement* l'objet et les informations responsables de son augmentation. Par contre, lorsque l'augmentation est extrinsèque, chaque utilisateur perçoit séparément l'objet et les informations à l'origine de l'augmentation (voir fig. 2.7).

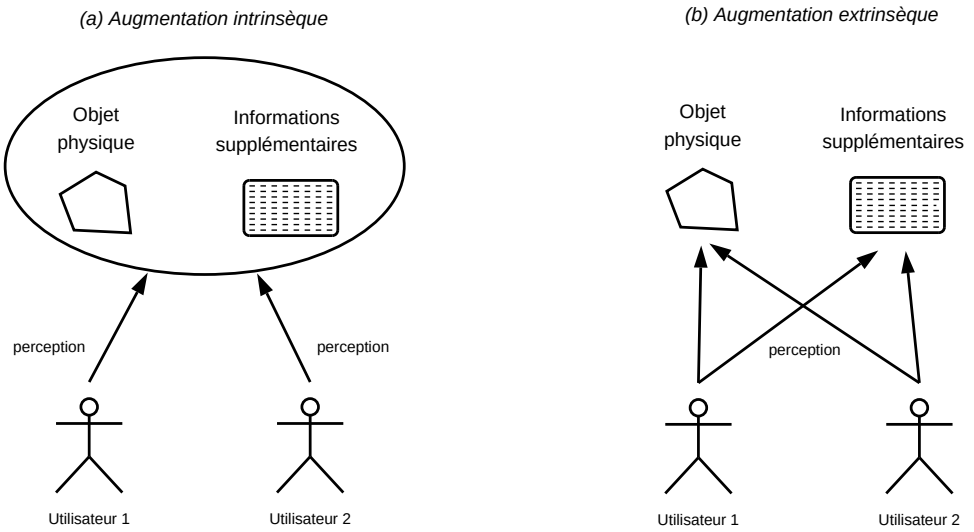


Figure 2.7 : *Différences de perception selon que l'augmentation d'un objet est intrinsèque (a) ou extrinsèque (b).*

Notons bien qu'un objet augmenté conserve ses fonctions de base si on lui retire ses caractéristiques d'augmentation. Par exemple, le four augmenté reste un four même si sa fonction

de calcul du temps de cuisson tombe en panne. C'est pourquoi certains disent que ce type d'objets est *numérique-indépendant* [Roudaut 2006].

Ambiant Les techniques déployées dans les objets augmentés ont conduit certains à rêver d'un monde où tout notre environnement serait augmenté, où nous serions entourés d'une multitude d'objets intelligents et discrets, chargés de nous faciliter l'existence. Cette vision a été nommée *informatique ubiquitaire* par Mark Weiser en 1993 [Weiser 1993].

Jusqu'à il y a peu, ces objectifs ont paru irréalisables. Les prototypes développés dans les années 1990 n'ont pas franchi les portes des laboratoires, car leur déploiement était souvent complexe (voir chapitre 3). Cependant, les progrès technologiques récents en matière de miniaturisation, de réseaux sans fils et de capteurs laissent penser qu'il sera bientôt possible de déployer à grande échelle des systèmes enfouis et réagissant à leur environnement (contexte).

De plus, dans le même temps, des progrès considérables ont été accomplis dans le domaine des interfaces homme-machine, notamment au niveau des interactions multimodales. Depuis quelques années, il est possible de recourir à de nouvelles formes d'interaction (par exemple, reconnaissance vocale), ce qui met peu à peu fin à la suprématie du trio clavier-écran-souris.

La convergence [Friedewald 2005] de toutes ces techniques donne naissance à ce qui est appelé *intelligence ambiante* [Ducatel 2001]. De façon générale, il s'agit de passer d'un monde où les utilisateurs se préoccupent d'*appareils* (ordinateurs, lecteurs DVD, chaînes HiFi, téléphones, etc.) à une vision où ils se préoccupent uniquement de *services* (regarder un film, dialoguer avec leurs amis, obtenir des informations, etc.) [Weber 2003]. Là où l'augmentation ne concerne que des objets isolés, l'intelligence ambiante vise à créer des *environnements* complètement nouveaux [Aarts 2004]. Ces environnements électroniques devront être sensibles à la présence des utilisateurs, et répondre de façon « intelligente » à leurs besoins [van Loenen 2003].

La plupart des applications ambiantes envisagées jusqu'ici prennent place au domicile des utilisateurs : habitat intelligent ou communicant [Ricquebourg 2006], chambre d'étudiant intelligente [Hagras 2004], etc.

Les caractéristiques clés de l'intelligence ambiante sont les suivantes :

- l'électronique doit être cachée, diffuse, disséminée dans l'environnement. Sa présence doit être *transparente* pour les utilisateurs ;
- l'environnement doit être sensible au contexte, conscient de la présence des utilisateurs ;
- le système doit agir de façon autonome, tout en restant contrôlable (Mark Weiser parle d'informatique *calme*). Il est donc nécessaire de réaliser un apprentissage de ce qu'il est souhaitable de faire ou non [Shadbolt 2003] ;
- lors des interactions avec les utilisateurs, il faut utiliser des interfaces naturelles (objets tangibles, interfaces multimodales [den Os 2003]).

De façon générale, les systèmes d'intelligence ambiante interagissent avec leurs utilisateurs de façon non classique. Non seulement les systèmes ambiants se placent-ils dans la mouvance des interfaces *post-WIMP*[8], mais ils vont même jusqu'à supprimer les interfaces tangibles (boutons, télécommandes, etc.) et les médias classiques (photographies sur papier, disques, cassetes, etc.).

En effet, comme l'électronique est dissimulée dans l'environnement, l'utilisateur n'a plus directement accès aux appareils et à leurs panneaux de contrôle. De plus, les données sont dématérialisées de façon à éviter une manipulation d'appareils : la musique n'est plus stockée sur des disques (vinyle ou CD), mais sur un serveur caché ; de même pour les photographies qui n'ont plus d'existence physique.

On aboutirait ainsi à un monde d'interactions complètement dématérialisées, ce qui n'est pas forcément pratique pour les utilisateurs. Certains proposent donc d'utiliser des objets augmentés pour servir de support aux interactions [van Loenen 2003]. Ainsi, des souvenirs pourraient servir de *raccourcis physiques* pour accéder à des données numériques associées. Par exemple, un coquillage rapporté de ses vacances à la mer[9] pourrait servir de point d'accès aux photographies prises pendant les vacances.

Les techniques de base suivantes sont envisagées pour la réalisation d'applications dans l'intelligence ambiante :

Informatique ubiquitaire : il faut pouvoir embarquer des microprocesseurs à l'intérieur d'objets qui ne sont pas habituellement électroniques [Kindberg 2002] : meubles, vêtements, peinture, revêtements muraux ou de sol, etc. Il s'agit donc de *miniaturiser* les systèmes électroniques et de leur donner une autonomie énergétique correspondant à leur utilisation[10]. On parle d'*intégration physique*, qui peut aller jusqu'à la création de « poussière intelligente » (*« smart dust »*). Par exemple, certains proposent d'ores et déjà de mélanger des balises RFID[11] aux matériaux de construction (béton, peinture) ou d'en intégrer dans les meubles [Bohn 2004] ;

Communications ubiquitaires : tous ces microprocesseurs embarqués et éventuellement enfouis doivent pouvoir communiquer entre eux, sans fils, où qu'ils se trouvent, à

[8] Windows, Icons, Menus, Pointer. Le terme WIMP désigne les interfaces graphiques classiques, telles qu'introduites par le Star de Xerox.

[9] À condition toutefois qu'il soit équipé d'une puce d'identification par l'environnement.

[10] [Aarts 2003] identifie trois classes d'appareils :

- les appareils *autonomes* (ex. : étiquettes intelligentes) : on ne leur fournit jamais d'énergie au cours de leur vie. Ils sont donc *jetables*. Pour fonctionner pendant une durée raisonnable (plusieurs années), ils doivent donc avoir une consommation de l'ordre du microwatt ;
- les appareils *mobiles* (ex. : téléphones mobiles) : ils sont rechargeables ; ils peuvent consommer quelques milliwatts ;
- les appareils *fixes* (ex. : grands écrans muraux) : ils sont reliés au réseau d'électricité, donc ils peuvent consommer plusieurs watts. Leurs ressources sont « illimitées ».

[11] Radio-Frequency Identification, voir section 3.2.3.2 p. 39.

n'importe quel moment, et avec une autonomie suffisante, et surtout sans configuration manuelle préalable (*interopération spontanée* [Kindberg 2002]) ;

Interfaces utilisateurs intelligentes : les interactions doivent être naturelles (voix, gestes, etc.), et être personnalisées pour les utilisateurs. Ainsi, elles doivent tenir compte des préférences de ces derniers, ainsi que du contexte d'utilisation.

Ces systèmes doivent être capables, d'une part de capturer leur *contexte* d'utilisation, et d'autre part d'*agir* sur l'environnement. Leur réalisation implique de posséder des connaissances à la fois en logiciel et en matériel. C'est pourquoi des plates-formes matérielles ont été proposées, afin de permettre aux spécialistes du logiciel de se lancer dans l'ambiant [Gellersen 2004] sans pour autant suivre une formation approfondie quant aux aspects matériels.

La capture du contexte sera traitée en détails dans le chapitre 3. Par contre, nous ne nous intéressons pas dans ce travail aux *actions* sur l'environnement : il s'agit du domaine des *actionneurs* (micromécanique, nanotechnologies, automatisme, etc.). Dans le cadre de la réalisation de systèmes de présentation d'information, nous n'avons en effet pas besoin d'effectuer d'actions mécaniques.

2.5 Conclusion

Ce chapitre a présenté les concepts généraux d'IHM en termes de modèles d'architectures logicielles et de multimodalité. Nous avons souligné l'importance de la prise en compte des handicapés lors de la conception de systèmes informatiques et de lieux publics. Enfin, nous avons présenté la nouvelle tendance des IHM : la dissémination des équipements informatiques dans l'environnement des utilisateurs, de façon à proposer à ces derniers des interactions « naturelles » et discrètes.

Cette vision de l'*intelligence ambiante*, nous l'avons vu, repose notamment sur la capacité des systèmes informatiques à capter, interpréter et traiter des informations sur leur *contexte* d'utilisation. Le chapitre suivant est donc consacré au contexte. Il se donne comme objectif de le définir et d'indiquer les moyens dont disposent les concepteurs d'applications pour y accéder.

Chapitre 3

Le contexte : caractéristiques et accès

Dans le chapitre 2, nous avons traité des concepts généraux en IHM. Dans ce présent chapitre, nous introduisons une notion récente en IHM : la prise en compte du *contexte* d'interaction. Cependant, le mot *contexte* peut recouvrir tout une variété de définitions relativement différentes : c'est pourquoi nous commençons par en donner une définition. Nous étudierons ensuite plus en détails la *localisation*, qui est une composante importante du contexte. Enfin, nous donnerons des exemples d'applications contextuelles, et nous montrerons qu'il est souvent nécessaire de disposer de moyens d'accès au contexte indépendants des applications, de façon à pouvoir *utiliser* le contexte sans devoir forcément dialoguer avec les dispositifs de capture.

3.1 Définition

Historiquement, les ordinateurs ont été utilisés au bureau, par l'intermédiaire d'interfaces relativement figées : un écran, un clavier, une souris. En particulier, les ordinateurs n'étaient pas *mobiles*. Bien entendu, les ordinateurs portables ont introduit une composante de mobilité. Cependant, un ordinateur portable n'est à la base rien d'autre qu'une version transportable et allégée d'un ordinateur de bureau : une telle machine est insensible à son emplacement d'utilisation et au monde qui l'entoure, c'est-à-dire à son *contexte* d'utilisation.

Implicitement, nous venons d'utiliser une définition du contexte, donnée par Schilit [Schilit 1994] : selon ce dernier, le contexte serait caractérisé par le lieu d'utilisation, ainsi que les identités des personnes et des objets situés à proximité[1]. Cette définition est néanmoins trop limitée, c'est pourquoi de nombreuses autres définitions ont été proposées.

Certains reprennent cette définition et essaient de lui donner une tournure formelle : selon [Schmidt 1999], le contexte consiste en *« connaissances sur l'utilisateur et l'état des*

[1] *« where you are, who you are, what resources are nearby. »*

appareils informatiques, y compris ce qui se passe alentour, la situation et la localisation ». Souchon et al. [Souchon 2002] proposent ainsi de définir trois modèles pour le contexte : un modèle de l'utilisateur, un modèle de la plate-forme informatique, et un modèle de l'environnement (conditions physiques, localisation, environnement social et organisationnel).

D'autres proposent des extensions. Ainsi, Chen & Kotz [Chen 2000] proposent d'ajouter au contexte la situation *temporelle*, par exemple l'heure, la saison, l'année, etc.

Cependant, la définition du contexte la plus utilisée est due à Anind K. Dey [Dey 2000, Dey 2001] : *« le contexte comprend toute information qui peut être utilisée pour caractériser la situation d'une entité. Une entité est une personne ou un objet qui est considéré comme important vis-à-vis de l'interaction entre un utilisateur et une application, y compris l'utilisateur et l'application eux-mêmes. »* Cette définition est assez large ; elle se veut pragmatique car elle se base sur ce qui est pertinent pour les applications informatiques considérées.

Types de contexte La notion de contexte peut englober un certain nombre de facettes différentes. Chaque système dit « contextuel » prend donc en compte un sous-ensemble donné de ces facettes possibles. Nous dressons ici une liste des types de contexte qui peuvent être utilisés par des applications.

Localisation On appelle *localisation* le processus servant à déterminer la position d'un utilisateur ou d'un objet. Comme nous le verrons plus loin, la localisation peut inclure l'*orientation* des utilisateurs. Quasiment *tous* les systèmes contextuels que nous avons étudiés prennent *au moins* en compte la localisation. Il s'agit donc d'une composante majeure du contexte, bien plus exploitée que toutes les autres. C'est pourquoi la section 3.2 y sera spécialement consacrée.

Temps La forme la plus simple de *temps* envisageable est liée à l'heure et à la date. Ainsi, les informations pertinentes pour un utilisateur donné peuvent varier en fonction de l'heure, ou bien du jour de la semaine. Par exemple, le planning d'utilisation d'une pièce indique que les activités qui s'y déroulent sont différentes selon les créneaux horaires. On peut également imaginer que des systèmes contextuels changent de comportement en fonction des *fuseaux horaires*.

Objets situés à proximité Des systèmes peuvent utiliser dans leur fonctionnement des objets situés à proximité. Par exemple, une application fonctionnant sur un ordinateur portable peut chercher à utiliser l'imprimante la plus proche de l'utilisateur. Il existe également des applications qui font migrer les appels téléphoniques ou bien les fenêtres d'interfaces graphiques [Schilit 1993] sur le téléphone ou l'écran le plus proche de l'utilisateur.

Ressources informatiques Certains systèmes prennent en compte la charge du processeur des ordinateurs, ou bien l'occupation de la bande passante du réseau. Cela peut leur permettre de s'adapter aux ressources effectivement disponibles, pour ne pas contribuer à la surcharge (et donc à l'*inutilisabilité*) de systèmes déjà bien surchargés.

Conditions physiques Dans certaines situations, il peut être intéressant de mesurer certains paramètres physiques, par exemple la *température*, la *pression atmosphérique*, le *taux d'humidité*, le *niveau de luminosité*, le *niveau sonore*. Des mesures plus originales peuvent même être envisagées, comme la mesure du *taux de monoxyde de carbone*. Ainsi, une application domestique qui relèverait un taux de monoxyde de carbone anormalement élevé dans le contexte d'un utilisateur pourrait déclencher une alarme, voire prévenir les secours afin d'éviter une asphyxie.

Contexte de haut niveau Les facettes du contexte évoquées jusqu'ici sont d'assez bas niveau : il est possible de les mesurer avec des capteurs physiques. Cependant, on s'intéresse parfois à des contextes de plus haut niveau, comme l'activité des personnes, ou bien les tâches en cours dans une salle de réunion. Ce type de contexte n'est pas détectable directement, mais on peut chercher à l'inférer à partir des facettes de bas niveau énumérées ci-dessus.

Ces différentes facettes du contexte possèdent des propriétés [Gray 2001] :

- leurs formats de représentation ;
- la qualité des informations : résolution, précision, fiabilité, etc. ;
- une caractérisation des sources de contexte correspondantes : fiabilité, caractère intrusif, respect de la vie privée, coût, etc. ;
- transformation à effectuer sur les données brutes du contexte ;
- actions à effectuer en réponse : par exemple, réorienter une antenne GPS[2], éteindre un capteur défectueux, etc.

3.2 La localisation : une composante majeure du contexte

3.2.1 Introduction

La localisation est une composante très importante du contexte, car elle nous apprend bien plus que de simples coordonnées géographiques :

- en fonction du lieu où se trouve un utilisateur, il est possible de réaliser des inférences quant à son activité, sa tâche. Ainsi, si un employé de bureau se trouve assis à son bureau, on peut raisonnablement supposer qu'il est en train de travailler. Par contre, s'il se trouve sur une plage, il doit certainement être en période de repos ;
- en fonction du lieu, il est possible de faire des choix quant aux modalités utilisables pour dialoguer avec un utilisateur. Par exemple, si le propriétaire d'un téléphone portable se

[2]Global Positioning System, voir, section 3.2.2 p. 34.

trouve au bord d'une route fréquentée, il est souhaitable d'utiliser un volume de sonnerie élevé, car les bruits environnants sont importants et très gênants lors de la perception de sonneries de téléphones. Par contre, si le propriétaire se trouve au cinéma, il est souhaitable que le téléphone reste silencieux ;

- la localisation de l'utilisateur détermine également en partie les informations qui sont susceptibles de l'intéresser. Par exemple, lorsqu'une personne descend de son train et sort sur le parvis d'une gare, il est fort probable qu'elle soit intéressée par un plan détaillé des environs, un plan des lignes de transport en commun, ou bien des informations sur une correspondance qu'elle doit prendre.

De plus, la localisation de l'utilisateur est relativement aisée à obtenir, par rapport à d'autres aspects du contexte. Ainsi, essayer de déterminer dans l'absolu la tâche en cours de l'utilisateur sera bien plus difficile, et sujet à erreurs, que déterminer sa position géographique. Par exemple, certains travaux ont essayé de déterminer la tâche des utilisateurs à partir des mouvements de leurs corps [Randell 2000, Mäntyjärvi 2001], mais ces applications sont limitées à un type bien précis de tâches.

En conséquence, un grand nombre de systèmes d'accès à la localisation des utilisateurs ont été développés. Les premiers systèmes étaient d'ailleurs fortement liées aux applications, qui ne dépendaient principalement que de la localisation. En raison de l'importance donnée à la localisation, cette section se consacre à l'étude des techniques sous-jacentes.

Pour classifier ces différentes techniques de localisation, plusieurs axes et critères de comparaison ont été proposés dans [Hightower 2001] :

- *géographique contre symbolique*: certains systèmes effectuent un positionnement *géographique*. Ils fournissent des coordonnées géographiques par rapport à un système de coordonnées et un référentiel. À l'opposé, d'autres systèmes réalisent un positionnement *symbolique* : ils fournissent non pas des coordonnés brutes, mais des références à des objets connus, par exemple des identificateurs. Ainsi, si une utilisatrice appelée Alice se trouve dans son bureau, un système de positionnement géographique fournirait les coordonnées `45° 34’ 26’’ N, 1° 47’ 52’’ E`, tandis qu'un système de positionnement symbolique fournirait l'identificateur `bureauDAlice`. Il est à noter que des solutions ont été proposées pour passer d'un positionnement géographique à un positionnement symbolique, comme le LSI (Location Service Infrastructure) [Roth 2003] qui reprend des concepts proches de ceux du système DNS (Domain Name System) utilisé pour faire la correspondance entre noms d'ordinateurs et adresses réseau sur Internet ;
- *absolu contre relatif* : lorsqu'on effectue un positionnement *absolu*, les positions sont données par rapport à un système de coordonnées fixé. Par contre, en positionnement *relatif*, les positions des points successifs sont données les unes par rapport aux autres ;
- *localité* : les calculs peuvent être effectués localement, sur un appareil porté par l'utilisateur (*localisation locale*), ou bien au niveau de l'infrastructure environnante (*localisation globale*). Dans le premier cas, chaque dispositif de positionnement doit

maintenir à jour sa propre position, tandis que dans le second cas, l'infrastructure tient à jour la position de chacun des utilisateurs ;

- *précision* : il s'agit de l'erreur moyenne du système. Par exemple, un système donné peut avoir une précision de 10 mètres, ce qui signifie que *la plupart du temps*, les positions qu'il fournit sont situées dans un rayon de 10 mètres autour de la position réelle ;
- *fiabilité* : cette valeur indique si les positions fournies par le système sont souvent dans son intervalle de précision. Par exemple, le système précédent peut avoir une fiabilité de 95 %, ce qui signifie que les positions fournies se trouvent dans un rayon de 10 mètres autour des positions réelles *95 % du temps* ;
- *échelle* : les systèmes peuvent être conçus pour fonctionner à différentes échelles : au niveau mondial, au niveau d'un pays ou d'une région, à l'intérieur d'un bâtiment, etc. ;
- *coûts* : les coûts varient fortement d'un système à un autre. Les concepteurs doivent aboutir à un compromis entre coût et contraintes de précision et de fiabilité.

Examinons maintenant quelles sont les principales techniques de localisation disponibles, ainsi que leurs caractéristiques et leurs variantes technologiques : triangulation, proximité, reconnaissance d'empreintes, analyse de scènes et enfin navigation à l'estime. Nous verrons ensuite comment il est possible d'accroître la précision d'un système réel en combinant *plusieurs* de ces techniques.

3.2.2 Triangulation

La *triangulation* est une méthode basée sur le calcul de la position courante d'un mobile *par rapport* à la position de balises disposées dans l'environnement, appelées *amers*, et dont on connaît les positions. On peut mesurer deux informations différentes :

- la distance entre le mobile et les amers : on parle alors de *latération* ;
- les angles entre le mobile et les amers : on parle alors d'*angulation*.

Ces systèmes ont pour objectif de calculer les *coordonnées* des mobiles : on dit que ce sont des outils de *positionnement géographique*. De plus, ce positionnement est *absolu*.

On peut utiliser deux types d'amers : des *amers environnementaux* ou des amers *artificiels*. Dans le premier cas [der Hardt 1997], il s'agit de se repérer par rapport à des marques qui existent naturellement dans l'environnement. L'être humain fait régulièrement appel à cette technique : un promeneur se repère souvent par rapport aux collines ou montagnes qui l'entourent, aux villages qu'il aperçoit, aux arbres caractéristiques, etc.

En revanche, l'exploitation de ces marques naturelles est beaucoup plus difficile pour une machine, car leur identification est extrêmement délicate [Livatino 1994]. C'est pourquoi

on introduit souvent des *amers artificiels* : le principe reste le même, mais ces amers sont beaucoup plus facilement identifiables (forme géométrique particulière, utilisation d'ondes radio, etc.). On classe les amers artificiels en deux catégories :

- les balises *passives* [Volpe 1995, Strietzel 1999], par exemple des réflecteurs optiques qui présentent un motif particulier. Entre autres, on peut utiliser des marques p-similaires [Scharstein 1999], une sorte de codes-barres un peu particuliers. Comme on ne trouve habituellement pas d'objets p-similaires dans la nature, on peut être certain d'avoir localisé une balise du système de positionnement si on détecte un objet p-similaire ;

- les balises *actives*, qui possèdent leur propre source d'énergie. Par exemple, ce peut être des émetteurs de lumière infrarouge ou d'ondes radio. Le GPS rentre dans cette catégorie : les balises sont les satellites GPS, qui sont facilement identifiables et distinguables entre eux, par simple décodage de leurs codes d'identification.

Dans ces systèmes, les mobiles sont équipés d'un système de détection et de mesure des balises : un télémètre à laser pour un réflecteur optique, une caméra vidéo pour des marques p-similaires, un récepteur radio pour des balises à radiofréquences comme le GPS, etc. De cette façon, le mobile mesure à la fois la position angulaire d'une balise, ainsi que la distance qui l'en sépare. Cette dernière mesure peut par exemple se faire en fonction de l'atténuation du signal reçu de la balise, ou en fonction du temps de propagation des ondes radio.

Une fois les balises identifiées, et connaissant leurs positions dans un repère donné, le mobile peut en déduire sa position par triangulation [Cohen 1993, Madsen 1997, Betke 1994].

Exemple du GPS Le GPS (Global Positioning System) est un système mondial de positionnement par satellite, développé à l'origine par et pour l'armée des États-Unis. Il s'agit d'un système à balises, qui sont transportées par un ensemble de satellites à défilement. Les balises se déplacent donc, ce qui à première vue peut être problématique si l'on compte les utiliser pour effectuer des triangulations. Cependant, les récepteurs GPS connaissent précisément les trajectoires des satellites (éphémérides), ils peuvent donc calculer à chaque instant la position de ces derniers.

Les satellites sont précisément synchronisés entre eux et transportent des horloges atomiques. Ils émettent des signaux horodatés, ce qui permet aux récepteurs de mesurer des *pseudo-distances*, c'est-à-dire les différences de temps de propagation entre les différents signaux reçus des satellites. De cette façon, si l'on reçoit les signaux issus de trois satellites, il est possible de calculer sa position absolue à la surface du globe, en deux dimensions. Pour une localisation en trois dimensions, il est nécessaire de recevoir les signaux d'un quatrième satellite.

La précision offerte par le GPS est de quelques mètres, en particulier depuis que le mode de *disponibilité sélective*[3] a été désactivé. Cependant, le GPS ne fonctionne de façon précise et sûre qu'en plein air, d'où son succès auprès des randonneurs et des automobilistes [Philip 2002]. Malheureusement, dans les villes, lorsque les utilisateurs sont entourés de hauts immeubles, la réception des signaux émis des satellites devient vite difficile en raison de l'obstruction de l'horizon. Ce phénomène, appelé *effet canyon* [Chao 2001], empêche souvent un positionnement correct par GPS dans les environnements urbains.

À l'intérieur des bâtiments, la situation est encore pire, et généralement, les signaux GPS y sont très faibles, et donc inutilisables. Il est néanmoins possible d'installer une « constellation » de pseudo-satellites GPS (appelés *pseudolites*) à l'intérieur d'un bâtiment. On remplace ainsi les balises des vrais satellites, inaccessibles, par des balises comparables mais situées à l'intérieur du bâtiment. Si les caractéristiques radio de ces pseudolites sont identiques à celles des véritables satellites GPS, il est possible d'utiliser des récepteurs GPS du commerce. Cependant, il peut être coûteux de mettre en place une telle solution, car les signaux GPS sont assez difficiles à générer, ce qui demande l'installation d'un équipement onéreux. Pour contourner cet obstacle, certains proposent de retransmettre à l'intérieur les signaux des vrais satellites GPS, et de modifier légèrement les récepteurs [Samama 2005].

Téléphonie mobile et WiFi Un terminal de téléphonie mobile ou un récepteur WiFi[4] se trouve dans la même situation qu'un récepteur GPS : il reçoit des signaux à partir d'une balise, une station de base dans le premier cas, et un point d'accès dans le deuxième. On peut donc imaginer qu'une localisation du mobile soit possible.

Des systèmes de localisation des téléphones mobiles par triangulation vont être mis en œuvre sous peu aux États-Unis, car la FCC[5] oblige les opérateurs à déployer un service appelé Enhanced-911 (E-911 Phase II). Les compagnies de téléphone devront être capables de localiser les téléphones mobiles avec une précision de 50 à 300 mètres lorsque leurs utilisateurs appellent le 911, le numéro d'urgence.

Diverses techniques sont envisagées, comme la mesure de l'angle de réception des signaux issus des relais (AOA pour *Angle Of Arrival*), ou la différence de temps d'arrivée desdits signaux (TDOA pour *Time Difference Of Arrival*). Dans ces deux cas, il s'agit de positionnement *local*, effectué par les terminaux des usagers. Afin d'accroître la précision, certains imaginent d'installer des balises supplémentaires, en plus des stations de base des réseaux cellulaires. On parle alors d'E-ODT pour *Enhanced Observed Time Difference*. Cependant, il est à noter que certains constructeurs envisagent d'intégrer un récepteur GPS dans leurs téléphones, de façon à utiliser une source de positionnement jugée plus fiable que les balises des réseaux mobiles.

[3] La *disponibilité sélective* ou *Selective Availability* (SA) était une « fonctionnalité » du GPS qui interdisait aux récepteurs non accrédités par l'armée américaine d'accéder à toute la précision du GPS. En pratique, les utilisateurs civils n'avaient accès qu'à des signaux détériorés artificiellement.

[4] Wireless Fidelity. Le terme WiFi désigne les technologies de réseau sans fil par ondes radio conformes aux normes de type 802.11x.

[5] Federal Communications Commission.

On peut également se positionner en mesurant la puissance reçue à partir des stations de base du réseau WiFi ou de téléphonie mobile. De tels systèmes peuvent alors effectuer un positionnement local aussi bien que global. En effet, les points fixes (stations de base de la téléphonie mobile ou bien points d'accès WiFi) aussi bien que les équipements mobiles émettent des signaux. Par exemple, le système RADAR [Bahl 2000], proposé par Microsoft, effectue du positionnement local en mesurant la force des signaux et le rapport signal à bruit des signaux WiFi. Cependant, ces techniques restent relativement imprécises, en raison des incertitudes quant aux modèles de la propagation des ondes mis en œuvre [Anne 2005], et à l'imprécision des mesures de puissance.

UWB Toutes les solutions évoquées jusqu'ici effectuent un positionnement *absolu*. Cependant, des travaux ont récemment été menés dans le domaine des communications par UWB[6] [Daniele 2003]. Dans ces systèmes, un grand nombre d'utilisateurs portent des *émetteurs-récepteurs* UWB. Ces émetteurs-récepteurs communiquent entre eux, et peuvent ainsi mesurer les distances entre eux. Ils échangent ces mesures, ce qui permet à l'ensemble des émetteurs-récepteurs UWB de calculer leurs positions respectives [Fontana 2003].

Il s'agit d'un mode de *positionnement relatif* effectué *à un niveau global*, grâce à des algorithmes particuliers distribués sur l'ensemble du réseau des émetteurs-récepteurs UWB. Cependant, il est possible de transformer facilement ce positionnement en positionnement absolu, en fixant la position d'un certain nombre d'émetteurs-récepteurs UWB, et en enregistrant leurs positions absolues dans le système.

3.2.3 Détection de proximité

Plutôt que d'effectuer une triangulation fine par rapport à des points de repère, il est possible de connaître sa position en termes de proximité par rapport à ces points de repère. Dans ce cas, on parle de *détection de proximité*. Généralement, un émetteur envoie en permanence un signal bien particulier, et lorsque le récepteur correspondant reçoit ce signal, il en déduit qu'il se trouve *à proximité* de l'émetteur, la notion de proximité sous-jacente dépendant de la nature du signal transmis. Deux grandes stratégies existent : disposer d'émetteurs fixes, et placer les récepteurs sur les mobiles à positionner, ou au contraire, installer des récepteurs fixes dans l'environnement, et des balises émettrices sur les mobiles.

Dans cette section, nous étudions quelques types de signaux utilisables, et les technologies correspondantes.

3.2.3.1 Infrarouges, ultrasons et réseaux radio

Infrarouges Le premier système commercial de détection de proximité a été introduit par Olivetti au début des années 1990. Il était appelé *ActiveBadge* [Want 1992, Harter 1994] (voir fig. 3.1). Dans ce système, les utilisateurs doivent porter des badges qui émettent des

[6] Ultra Wide Band.

signaux infrarouges. Ainsi, un badge donné émet un identifiant unique toutes les dix secondes. Des récepteurs infrarouges peuvent alors être installés dans chaque pièce, de façon à recevoir les donnés d'identité émises par les badges des occupants éventuels. Il est ainsi possible de déterminer dans quelle pièce se trouve chaque utilisateur.

Ce système de positionnement est *symbolique* et *global*. En effet, l'identité et la position des utilisateurs sont déterminées grâce à des identifiants symboliques. De plus, cette identification se fait au niveau de l'infrastructure : les données reçues par les récepteurs sont centralisées et traitées par un service de positionnement.

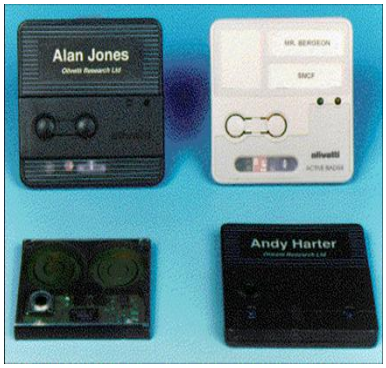


Figure 3.1 : *Quelques badges du système ActiveBadge d'Olivetti.*

Le système ParcTab d'informatique contextuelle, développé au centre de recherche de Palo Alto de Xerox, disposait d'un système de positionnement similaire [Schilit 1993, Schilit 1994]. Dans ParcTab, les utilisateurs portent des PDA[7] (voir fig. 3.2) qui communiquent par infrarouges avec des relais situés dans chaque pièce. Cette communication par infrarouges leur permet non seulement d'accéder à un réseau local, mais également d'être localisés par l'infrastructure. En effet, un système central mémorise quel est le relais actuellement utilisé par chaque PDA, ce qui permet de suivre les déplacements des utilisateurs, et ainsi, de leur fournir des services en fonction du lieu où ils se trouvent.

Figure 3.2 : *Un PDA du système ParcTab.*

Le CyberGuide [Long 1996] utilise la méthode inverse : des balises infrarouges sont installées dans toutes les pièces. Les dispositifs mobiles reçoivent les identifiants infrarouges transmis par les pièces, et en déduisent ainsi leur position par identification de la balise la plus proche.

[7]Personal Digital Assistant, *assistant numérique personnel*. Petit appareil de poche faisant office (au moins) d'agenda et de carnet d'adresses.

Ultrasons Il est possible d'utiliser des signaux à ultrasons à la place des infrarouges. Par exemple, dans le système *ActiveBat*, développé par AT&T (anciennement Olivetti Research) en 1998, les utilisateurs portent des badges qui émettent des ultrasons [Harter 1994, Harter 1999], voir figure 3.3. Les plafonds des pièces sont équipés de grilles de récepteurs d'ultrasons, ce qui permet de calculer la position des utilisateurs grâce à la mesure de temps de transmission des signaux ultrasonores. Il s'agit de localisation *géographique*. Dans ce cas-ci également, il est possible d'envisager la méthode inverse. Par exemple, dans le système *Cricket*, les récepteurs sont portés par les utilisateurs, tandis que les pièces sont équipées de rangées de balises ultrasonores.

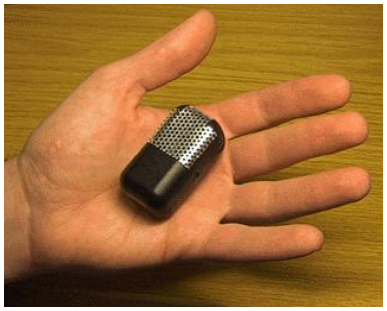


Figure 3.3 : *Un émetteur à ultrasons du système ActiveBat.*

Les systèmes à ultrasons se proposent d'installer des équipements au plafond. Dans le même ordre d'idée, il est possible d'installer des capteurs de pression dans le sol, qui sont capables de localiser et de d'identifier des utilisateurs en fonction du rythme des pressions exercées lorsqu'ils marchent. C'est par exemple ce que réalise le système *Smart Floor* [Orr 2000]. Cependant, l'inconvénient majeur à l'adoption de ces systèmes est leur coût : des travaux importants, ainsi que l'installation d'un matériel coûteux sont à prévoir dans tous les locaux où ils sont déployés.

Réseaux radio Dans la section précédente, nous avons vu que les réseaux radio (WiFi, téléphonie mobile) pouvaient être utilisés pour effectuer une localisation par triangulation. Ils peuvent également être utilisés pour effectuer une localisation symbolique au niveau de la cellule. Par exemple, la directive américaine E-911 Phase I impose une localisation d'un téléphone limitée à l'identifiant de la cellule dans laquelle il se trouve. Ainsi, il suffit d'identifier la station de base la plus proche pour effectuer la localisation. De même, un dispositif connecté à un réseau WiFi peut être localisé par l'identifiant de son point d'accès (ESSID, Extended Service Set Identification).

De façon plus symétrique, les périphériques Bluetooth peuvent se détecter mutuellement, en effectuant des balayages réguliers des dispositifs situés à proximité. Ainsi, un dispositif donné n'aura aucune idée de sa position géographique, mais il connaîtra les identifiants des autres périphériques Bluetooth considérés comme proches.

3.2.3.2 Systèmes RFID

Les technologies d'identification par radiofréquences, ou RFID[8], permettent de lire des valeurs stockées sur des *étiquettes*[9] spécialement conçues. Les étiquettes les plus répandues sont dites *passives*, car elles ne possèdent pas leur propre source d'énergie. Il existe également des étiquettes *actives*, mais elles sont beaucoup plus rares et plus onéreuses. La lecture (et éventuellement l'écriture) des étiquettes se fait grâce à un *lecteur* ou *interrogateur*, muni d'une antenne, et qui communique par radio avec les étiquettes.

Comme les étiquettes *passives* ne possèdent pas de source d'énergie propre, il est nécessaire de les alimenter lors de la lecture. Le lecteur envoie donc une onde radio suffisamment puissante pour que les étiquettes qui se situent dans sa portée puissent y puiser leur énergie et moduler ce signal. En pratique, les étiquettes sont équipées d'une antenne patch et d'un circuit résonant LC[10] accordé sur la fréquence émise par l'antenne de lecture. Il n'y a donc pas de problème de maintenance : de telles étiquettes sont virtuellement éternelles. Selon les modèles, on peut lire les données à une distance allant de quelques millimètres à 5 mètres environ.

À l'inverse, les étiquettes *actives* possèdent leur propre source d'énergie. Il s'agit en général d'une petite pile, ce qui augmente d'autant leur taille. De plus, la durée de vie de l'étiquette est alors limitée, quoique dans des proportions raisonnables : 5 à 10 ans. La distance de lecture est supérieure à celle des étiquettes passives : plus de 10 mètres.

La quantité de données portée par les étiquettes varie de 1 bit à quelques kilo-octets. En pratique, les étiquettes bon marché les plus courantes peuvent contenir quelques octets, ce qui est suffisant pour stocker un numéro d'identification comme le GUID[11].

Deux problèmes principaux se posent lors de la lecture des étiquettes RFID :

- présence de *faux positifs* : il peut arriver que l'on détecte une étiquette RFID alors qu'elle ne se trouve pas dans la zone « normale » de détection, mais à proximité ;
- présence de *faux négatifs* : il arrive que l'on ne détecte pas une étiquette qui pourtant se trouve dans la zone de détection. Les raisons sont diverses [Floerkemeier 2004] : collisions entre les émissions de plusieurs étiquettes, ou bien présence de capacités ou d'inductances parasites (métal, eau, champs magnétiques, autres étiquettes) qui désaccordent le circuit résonnant LC des étiquettes, ce qui les prive d'énergie.

Pour remédier à ces problèmes, il existe quelques règles simples [Brusey 2003], par exemple :

[8] En Anglais, *Radio-Frequency IDentification*.

[9] On parle également de *transpondeurs*.

[10] Circuit électronique composé d'une inductance et d'une capacité, qui entre en résonance à une fréquence donnée.

[11] *Global Unique Identifier*, implémentation par Microsoft du standard UUID, *Universal Unique Identifier*. Un GUID occupe 16 octets (128 bits).

- lorsqu'on veut détecter des objets, il ne faut pas faire une lecture ponctuelle, mais il est préférable d'attendre un intervalle de temps Δt, tout en veillant à ce que Δt ne soit pas trop long pour que le système reste réactif. De cette façon, on élimine les faux positifs, qui ne sont généralement détectés que très brièvement ;

- si dans une situation on n'est censé détecter qu'un seul objet, et si on en a détecté plusieurs, on peut pondérer les observations en fonction de la date d'observation (les observations anciennes pesant moins lourd), et intégrer sur Δt.

La technologie RFID est actuellement utilisée dans des domaines divers tels que l'identification de personnes lorsqu'ils pénètrent dans des endroits donnés, le suivi de colis [Römer 2004], des titres de transport sans contact, etc.

Ainsi, si l'on équipe une zone donnée de lecteurs RFID, il est possible de détecter les personnes qui passent dans le périmètre considéré, à condition (a) que tous les utilisateurs des lieux portent une étiquette RFID, et (b) que le lecteur soit suffisamment sensible. Dans ce cas, on peut réaliser un système de positionnement pris en charge par l'infrastructure. Des antennes de lecture peuvent être installées dans chaque lieu porteur d'une « sémantique spatiale » [Ni 2004] (un couloir donné, un bureau donné, etc.), et un serveur central peut alors tenir à jour la liste des positions de tous les utilisateurs.

À l'inverse, il est possible de positionner des étiquettes à des endroits clés de l'environnement, et de munir chaque utilisateur d'un lecteur RFID. Dans ce cas, la localisation est effectuée au niveau local, sur chaque utilisateur. Il n'est pas nécessaire d'installer d'équipements dans l'infrastructure, mais seulement de disséminer des étiquettes passives. Certains envisagent même d'inclure ces étiquettes en standard dans les revêtements des murs et des sols. Ainsi, lorsqu'un utilisateur détecte une étiquette RFID donnée, il sait qu'il pénètre dans la zone correspondante. Cependant, devoir porter un lecteur d'étiquettes est un inconvénient de taille, notamment en raison des dimensions de l'appareil lui-même, et de son antenne.

Dans ces deux exemples, des étiquettes RFID sont utilisées pour effectuer une localisation *symbolique*. On positionne une étiquette ou un lecteur à chaque endroit d'intérêt, ce qui permet de se repérer.

3.2.4 Reconnaissance d'empreintes

Nous avons vu que la mesure des signaux issus de bornes WiFi, combinée avec un modèle de propagation, pouvait être utilisée pour réaliser une localisation par triangulation. Cependant, nous avons vu que cette technique n'était pas très fiable, en particulier car il est très difficile d'obtenir un modèle de propagation conforme à la réalité. Nous allons voir qu'il est néanmoins possible d'utiliser la puissance de signaux WiFi pour se positionner, sans pour autant disposer d'un modèle de propagation, donc sans effectuer de triangulation.

Dans un environnement où plusieurs bornes WiFi sont installées, il est possible de mesurer la puissance issue de chacune d'entre elles en chaque point d'un bâtiment, ou de façon plus

réaliste selon un quadrillage donné. Ainsi, il est possible de constituer une base de données des « empreintes WiFi » en chaque point de mesure. Une empreinte sera donc un vecteur à n composantes, où n est le nombre total de bornes WiFi installées dans l'environnement.

On fait alors l'hypothèse que les empreintes sont le plus souvent uniques, et que des empreintes ressemblantes correspondront à des points proches les uns des autres. Un dispositif de positionnement effectuera donc des mesures d'empreintes en permanence, et comparera à chaque instant l'empreinte mesurée avec les empreintes de la base de données. Il pourra alors déduire qu'il se trouve au lieu correspondant à l'empreinte de la base de données la plus proche[12] de l'empreinte courante.

On peut ainsi réaliser une *localisation géographique* à l'aide d'un réseau WiFi classique, et pas seulement une identification de la borne la plus proche. Des systèmes commerciaux exploitent cette technique, comme ceux des sociétés Skyhook Wireless[13] et Ekahau[14] [Eissfeller 2004]. Le système Placelab[15] d'Intel utilise une technique similaire, mais elle est appliquée aussi bien aux stations de base de téléphonie cellulaire et aux appareils Bluetooth qu'aux points d'accès WiFi [LaMarca 2005].

Nous avons réalisé nos propres essais de localisation par WiFi à Supélec, où le rez-de-chaussée du bâtiment est équipé d'un réseau WiFi dense. Avec un logiciel de notre propre conception et en un minimum de temps (1 journée), nous avons pu obtenir des résultats corrects dans la plupart des cas (voir fig. 3.4). Cependant, en particulier lorsque les réseaux sont peu denses, il peut arriver que les empreintes mesurées en des endroits très différents soient relativement semblables, ce qui peut mener à des erreurs de positionnement.

Il est même possible d'utiliser les puissances reçues à partir des émetteurs de radio FM : un tel principe est mis en œuvre dans le système RightSPOT de Microsoft Research [Krumm 2003]. RightSPOT mesure la puissance des signaux de diverses radio FM commerciales, et les classe. Une base de données contient les probabilités des différents classements pour chaque lieu. Ainsi, pour un classement donné, un classificateur bayésien fournit la position la plus probable. Cependant, les expérimentations ont montré que la précision de ce système est limitée ; il est uniquement capable d'identifier le quartier où on se trouve.

On peut également utiliser la même technique, en mesurant la puissance reçue à partir de badges RFID [Hähnel 2004].

3.2.5 Analyse de scènes

Les systèmes d'analyse de scène sont basés sur une capture d'images à l'aide de caméras vidéos, et de traitement effectués à l'aide de logiciels de vision par ordinateur. Les caméras fournissent en permanence des images des pièces, et des serveurs centraux essaient de recon-

[12] On peut par exemple définir la proximité par rapport à la distance euclidienne dans un espace vectoriel de dimension n : $d\left(\vec{a},\vec{b}\right) = \sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2 + \cdots + (b_n - c_n)^2}$.

[13] `http://www.skyhookwireless.com/`.

[14] `http://www.ekahau.com`.

[15] `http://www.placelab.org/`.

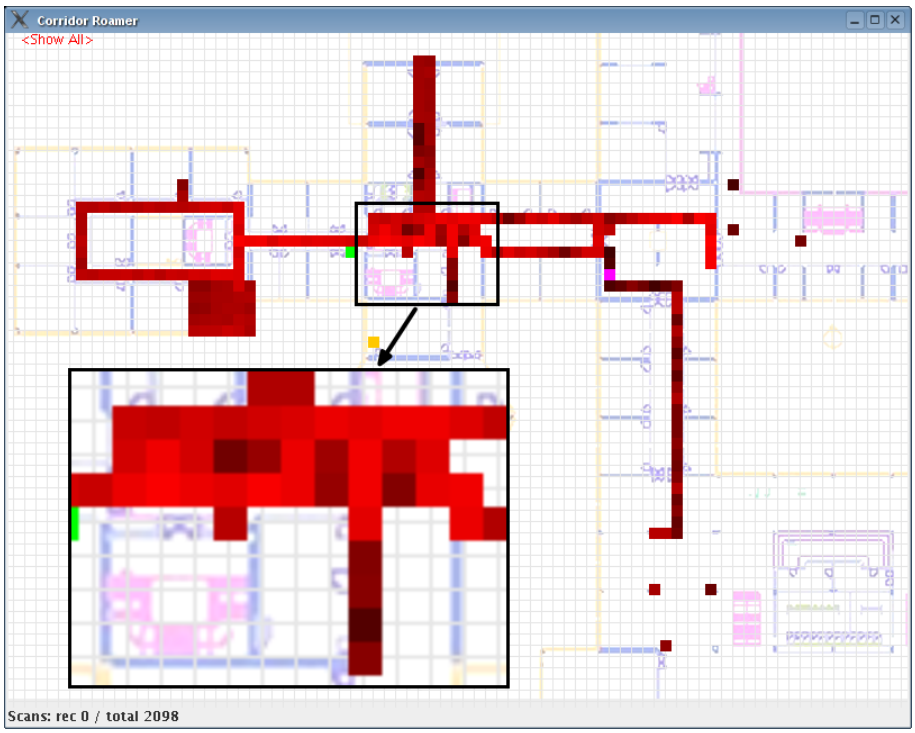


Figure 3.4 : *Localisation par empreintes WiFi à Supélec. Sur cette capture d'écran, la distance entre l'empreinte mesurée et les empreintes de référence d'un certain nombre de points est indiquée par une couleur rouge plus ou moins foncée (plus la distance est grande, plus le rouge est foncé), en superposition avec le plan de Supélec.*

naître les visages des utilisateurs, ce qui permet d'effectuer une localisation symbolique au niveau de la pièce. Une méthode de ce type est mise en œuvre dans le système *EasyLiving* de Microsoft Research [Hightower 2002].

Du point de vue des utilisateurs, la solution la plus pratique est la reconnaissance de visage, de silhouette, de motifs, etc. En effet, ils n'ont dans ce cas aucune mesure particulière à prendre : on parle de système non-intrusif. Cependant, il est possible d'améliorer la détection et d'éliminer les ambiguïtés si les utilisateurs portent des badges spécialement adaptés, par exemple des codes-barres en deux dimensions. Néanmoins, la situation est moins « naturelle » pour les utilisateurs, car ils doivent penser à se munir de leurs badges.

3.2.6 Navigation à l'estime

Lorsqu'aucune des méthodes citées ci-dessus n'est utilisable (par exemple, à l'intérieur des bâtiments il est généralement impossible d'utiliser le GPS), il est possible de recourir à la navigation à l'estime[16]. Il s'agit, à partir d'une position de départ connue, d'approximer régulièrement sa position courante en fonction du déplacement estimé depuis le dernier point.

[16]Dans la littérature, on rencontre fréquemment le terme anglais, *dead reckoning*.

Cette technique a longtemps été utilisée par les marins qui, connaissant leur cap grâce au compas et la vitesse de leur navire grâce au loch, pouvaient en déduire leur position approximative sur une carte de marine.

3.2.6.1 Odométrie et podométrie

Les véhicules à roues peuvent se baser sur l'*odométrie* [Borenstein 1996], dont le principe est de mesurer la distance parcourue par comptage du nombre de tours effectués par les roues. Connaissant le diamètre extérieur des roues, on accède théoriquement à la distance parcourue de façon fiable. C'est le système utilisé par les compteurs de distance des automobiles. Si l'on ajoute la connaissance à chaque instant du cap du véhicule, il est possible de reconstituer ses mouvements sur une carte.

Cependant, cette méthode est susceptible d'introduire des dérives, même lorsqu'on connaît de façon précise la géométrie d'un véhicule. Par exemple, les glissements introduisent des déplacements qui ne sont pas pris en compte, car ils n'induisent pas de rotation des roues.

L'odométrie convient donc aux véhicules, mais qu'en est-il du suivi des déplacements des piétons ? On peut adapter le principe de l'odométrie : on compte le nombre de pas, et on multiplie ce nombre par la longueur moyenne d'un pas. *En moyenne,* on doit donc obtenir une bonne estimation de la distance parcourue, de la même façon qu'en comptant le nombre de tours de roue d'un véhicule : c'est le principe du *podomètre*. Pour mesurer le cap, on peut envisager de placer une boussole qui serait solidaire du piéton. On disposerait alors des mêmes données que pour une mesure odométrique, et il devrait donc être possible d'effectuer un positionnement correct.

Malheureusement, il est illusoire de comparer un véhicule (automobile ou robot) à un être humain. En effet, un véhicule possède un nombre restreint de degrés de liberté, et il est muni d'un châssis qui le rend indéformable (dans certaines limites). Par opposition, le corps humain est souple, donc « déformable », et il possède un grand nombre de degrés de liberté. Il sera donc très difficile d'effectuer des mesures correctes, et l'on court le risque que les dérives soient très importantes, même pour de courtes distances. De plus, les boussoles sont fortement perturbées par les équipements électromagnétiques : leur utilisation devient très hasardeuse en intérieur.

Néanmoins, à défaut de pouvoir reconstituer précisément les déplacements de l'utilisateur, la podométrie peut se révéler utile. En effet, *à condition de disposer d'une bonne calibration de la longueur des pas de l'utilisateur*, cette méthode est susceptible de donner une bonne approximation des *distances parcourues*. Cette information peut être utile, à condition de la combiner avec d'autres informations complémentaires.

Ces méthodes de positionnement sont *relatives* : la position est estimée en fonction des déplacements effectués depuis la dernière position connue de façon absolue. Cette dernière *position absolue* pourrait être par exemple le lieu de la dernière réception des signaux GPS avant de rentrer dans un bâtiment ou un souterrain.

3.2.6.2 Centrales inertielles

Il est également possible d'utiliser une centrale inertielle : un tel dispositif mesure le vecteur accélération du mobile à l'aide de trois accéléromètres. On peut alors calculer sa position en intégrant deux fois le vecteur accélération. Le procédé peut être affiné si l'on connaît en plus les mouvements de rotation du mobile. Ceux-ci peuvent être mesurés par un *gyroscope*. Ici aussi, il s'agit de positionnement relatif.

Malheureusement, ce procédé est très sensible aux erreurs [Fusiello 1997]. Les positions calculées sont susceptibles de dévier assez rapidement des positions réelles, car les erreurs de mesure sur l'accélération, aussi petites soient-elles, s'accumulent lors des deux intégrations successives des signaux. Ainsi, cette méthode est catégoriquement rejetée par certains spécialistes du domaine[17] [Ladetto 2000].

En revanche, il semble que les signaux d'une centrale inertielle puissent être utilisés pour calibrer en permanence la longueur des pas de l'utilisateur [Konishi 2001], ce qui permet d'utiliser une méthode podométrique évoquée ci-dessus, tout en assurant une plus grande précision. En effet, au lieu de supposer que tous les pas font la même longueur, la position de chaque pas est estimée en fonction des paramètres mesurés par les accéléromètres.

Pour surmonter ces problèmes, on peut utiliser des méthodes de correction, par exemple en recalant les positions brutes mesurées sur les chemins indiqués par une carte (voir section 3.2.8.1).

3.2.7 Récapitulatif

Le tableau 3.1 récapitule les caractéristiques des différentes techniques vues jusqu'ici.

3.2.8 Amélioration des résultats par des méthodes mathématiques

Les méthodes évoquées jusqu'ici fournissent des informations de positionnement *brutes*. Cependant, en combinant ces informations avec des modèles probabilistes de déplacement ou des cartes où sont indiquées les positions les plus probables pour les mobiles, il doit être possible d'améliorer les résultats des méthodes de positionnement. Cette section traite de ces méthodes *mathématiques*.

3.2.8.1 Recalage sur une carte

Pour corriger les dérives inhérentes à la navigation à l'estime, ainsi que pour améliorer la précision du GPS, il est possible d'essayer de se *recaler par rapport à une carte* en permanence. Par exemple, une voiture ne peut circuler que sur des routes, ce qui limite considérablement

[17] *« Dead reckoning for on-foot navigation applications cannot be computed by double integration of the antero-posterior acceleration. »*, dans [Ladetto 2000].

		Géographique/Symbolique	Absolu/Relatif	Niveau : Local/Global	Précision	Échelle	Coût
Triangulation	GPS	G	A	L	1 m	monde	$
	téléph. mob.	G	A	L/G	50 à 300 m	région	$
	WiFi	G	A	L/G	quelques m	bâtiment, ville	$$
	UWB	G	R	G	quelques cm	bâtiment	$$
Proximité	infrarouges	S	A	G	quelques m	pièce	$$$
	ultrasons	S	A	G	quelques cm	pièce	$$$$
	réseau radio	S	A	L/G	cellule	réseau	$
	RFID	S	A	G	qqs cm ou m	pièce	$$$
Reconnaissance d'empreintes	WiFi	G	A	L	quelques m	bâtiment	$$
	bande FM	G	A	L	quelques km	zone urbaine	$
Analyse de scènes		S	A	G	quelques cm	pièce	$$
Navigation à l'estime		G	R	L	variable	pays	$$$

Tableau 3.1 : *Caractéristiques des techniques de localisation présentées dans la section 3.2.*

ses déplacements. En comparant les mesures effectuées avec les mouvements *possibles* d'après la carte, il est souvent possible d'améliorer grandement la précision des systèmes de positionnement. Cette technique est d'ailleurs utilisée de façon naturelle par les conducteurs et les randonneurs, qui comparent régulièrement les panneaux et la forme des routes et des chemins avec ce qu'indique la carte.

Cette technique de recalage est appelée *map matching* [Bernstein 1996, White 2000]. Plusieurs variantes existent :

- on peut travailler à un niveau global : il s'agit alors soit simplement de chercher sur la carte quel est le segment de route le plus proche de sa position mesurée, soit de rechercher parmi les différents chemins possibles indiqués sur la carte quel est celui qui correspond à la séquence d'*événements* (tournants, lignes droites, etc.) observés [Kitazawa 2000] ;
- on peut également travailler au niveau local : dans ce cas, après qu'on a déduit sa position x_n le long d'une route, on cherche sa position suivante x_{n+1} dans un petit cercle autour de x_n. Le rayon de ce cercle est un paramètre important qui conditionne la précision de l'algorithme [Yoshimura 2003].

Les systèmes de navigation pour automobiles utilisent le *map matching*, aussi bien les systèmes basés sur le GPS que ceux construits autour de centrales inertielles. Cela fonctionne bien pour positionner des voitures, car ces dernières sont forcées de suivre des routes.

Le *map matching* est également utilisé en robotique : même lorsque les robots n'embarquent pas de centrale inertielle, leur centrale de contrôle peut reconstituer leurs déplacements théoriques en fonction des *actions* qu'ils ont effectuées. En effet, les *commandes* qui ont été envoyées aux actionneurs (changement d'orientation, avancées, etc.) sont gardées en mémoire. Cette trajectoire reconstituée peut alors simplement être recalée par rapport à une carte.

Cependant, ces techniques ne peuvent pas fonctionner de façon sûre pour des piétons, car les mouvements de ces derniers sont beaucoup moins prédictibles, notamment lorsqu'ils ne se trouvent pas sur des chemins. Par exemple, vouloir faire du *map matching* pour repérer la position de piétons sur l'Esplanade de la Défense à Paris n'a pas de sens, car il n'existe pas à cet endroit de notion de *chemin* ; les piétons disposent de tous leurs degrés de liberté dans le plan horizontal.

De plus, même lorsqu'un chemin est bien délimité, les piétons ont souvent tendance à zigzaguer. De plus, les utilisateurs aveugles (qu'il est impensable de laisser pour compte) ont tendance à s'éloigner régulièrement des chemins, ce qui rend caduques les techniques de *map matching*.

3.2.8.2 Localisation markovienne

La *localisation markovienne* [Fox 1998] permet d'estimer la position d'un robot, connaissant son *modèle d'action* et son *modèle de perception*.

Le modèle d'action représente la probabilité de se retrouver en une localisation ℓ', sachant qu'on se trouvait auparavant en une localisation ℓ, et que l'on a effectué une action a :

$$P(\ell|\ell', a)$$

Pour un robot, l'action a représente l'une des actions élémentaires qu'il peut accomplir : avancer, reculer, tourner, etc. La localisation inclut toujours la position, mais elle peut également inclure d'autres paramètres, comme l'orientation. Ainsi, il est tout à fait possible qu'une localisation soit notée par un triplet $\langle x, y, \theta \rangle$: x et y représentent la position proprement dite, et θ l'orientation.

Le modèle de perception représente la probabilité de percevoir v sachant qu'on se trouve en une localisation ℓ :

$$P(v|\ell)$$

v représente les valeurs lues sur les capteurs à un moment donné. Ce peut aussi bien être un type de données très simple (par exemple, la distance à l'obstacle le plus proche) qu'un type de données plus complexe (par exemple, une image transmise par une caméra). On note

qu'avec ce type de modélisation, il est possible que les modèles des capteurs dépendent de la position ℓ.

On cherche alors à estimer la position du système, en fonction des actions effectuées et des valeurs relevées par les capteurs. La méthode repose sur la mise à jour incrémentale d'une distribution $Bel(\ell)$, qui représente, pour chaque localisation ℓ, la « croyance » du système de se trouver en ℓ. Si la position initiale est connue, alors à $t = t_0$, $Bel(\ell)$ est centrée sur cette position.

On fait l'hypothèse que si à un instant t_1 donné, on connaît la valeur de $Bel(\ell)$, alors cette valeur résume toutes les mesures passées, toutes les valeurs de $Bel(\ell)$ pour $t < t_1$. Il s'agit de l'hypothèse markovienne, d'où le nom de cette méthode de localisation.

La mise à jour de $Bel(l)$ est réalisée dans deux situations :

- lorsque le robot fait une action a, on effectue une *phase de prédiction* ;
- lorsqu'on lit une valeur v d'un capteur, on effectue une *phase de mise à jour*.

Phase de prédiction : lorsque le robot effectue un mouvement a, on modifie $Bel(\ell)$ selon l'équation suivante, tirée de la théorie des chaînes de Markov :

$$Bel(\ell) \longleftarrow \int_E P(\ell|\ell', a) Bel(\ell') d\ell'$$

L'intégrale est calculée sur toutes les localisations ℓ' de l'espace E des localisations.

Phase de mise à jour : lorsqu'on lit une valeur v issue d'un capteur, on modifie $Bel(\ell)$ selon la formule de Bayes :

$$Bel(\ell) \longleftarrow \alpha P(v|\ell) Bel(\ell)$$

La valeur α est une constante de normalisation, calculée de façon à ce que l'on ait toujours

$$\int_E Bel(\ell) = 1$$

Cette solution est a priori adaptée à un espace des localisations *continu*. Cependant, dans une mise en œuvre pratique, on a généralement besoin de discrétiser cet espace. Cette discrétisation peut être soit basée sur une grille (les localisations sont des points régulièrement disposés), soit topologique (les localisations sont des points d'intérêt distingués[18]). On peut alors effectuer simplement les calculs ci-dessus, en réalisant des intégrations numériques.

Dans le cas discret, si l'on connaît le modèle des capteurs utilisés, il est possible d'accélérer les calculs en pré-calculant toutes les valeurs de $P(v|\ell)$. Par exemple, dans [Jensfelt 2001],

[18] En Anglais : *landmarks*.

les auteurs pré-calculent les $P(v|\ell)$ en lançant des simulations de déplacement dans une modélisation de l'environnement de type CAO[19].

3.2.8.3 Localisation Monte-Carlo

Le principe de la méthode de localisation dite « Monte-Carlo[20] » [Rekleitis 2004] est de simuler un grand nombre de fois les déplacements potentiels du mobile dont on cherche à suivre les déplacements. Chaque instance de cette simulation est appelée une *particule*. À chaque instant, un comptage des particules en fonction des zones de l'espace permet d'évaluer la densité de probabilité de présence du mobile, ainsi que sa position la plus probable.

Au départ, les particules sont tirées au sort, selon une distribution qui peut être :

- soit répartie de façon uniforme sur tout l'espace si la position de départ est inconnue ;
- soit centrée sur la position de départ si elle est connue.

On note N le nombre de particules. Ces dernières sont des couples localisation-poids notés s_i :

$$s_i = \langle \langle x_i, y_i, \theta_i \rangle , p_i \rangle = \langle \ell_i, p_i \rangle$$

Les facteurs de poids p_i sont normalisés : ils sont choisis de sorte que $\sum_{i=1}^{N} p_i = 1$.

Comme dans le cas de la localisation markovienne, on modifie le modèle lorsque (a) le mobile effectue une action (i.e. lorsqu'il se déplace), ou bien lorsque (b) les capteurs fournissent une valeur.

(a) Lorsque le mobile se déplace, c'est-à-dire qu'il effectue une action notée a, on commence par choisir N échantillons dans l'ensemble $\{s_i, i \in [1; N]\}$, avec une probabilité de choisir s_i valant p_i. Cela signifie que certains des s_i seront choisis plusieurs fois (ceux de poids le plus fort), tandis que d'autres ne seront jamais choisis (ceux de poids plus faible).

Pour chaque $s_i = \langle \ell'_i, p_i \rangle$ choisi, on construit un nouvel échantillon :

$$\left\langle \ell_i, \frac{1}{N} \right\rangle$$

Dans cet échantillon, ℓ_i est choisi aléatoirement, selon la loi de probabilité $P(\ell|\ell', a)$. Cette première opération est nommée *rééchantillonnage*. Comme on le voit sur la figure 3.5, des rééchantillonnages successifs provoquent une dilution de la densité de probabilité de présence du mobile. En effet, à chaque fois que l'on effectue un rééchantillonnage, les poids de toutes

[19] Conception Assistée par Ordinateur.

[20] La méthode Monte-Carlo est également nommée *bootstrap filter*, *condensation algorithm*, *survival of the fittest*, ou *filtres à particules*. Les termes diffèrent, mais le principe reste le même.

les particules sont réinitialisés à $\frac{1}{N}$, ce qui élimine une bonne partie des informations de densité.

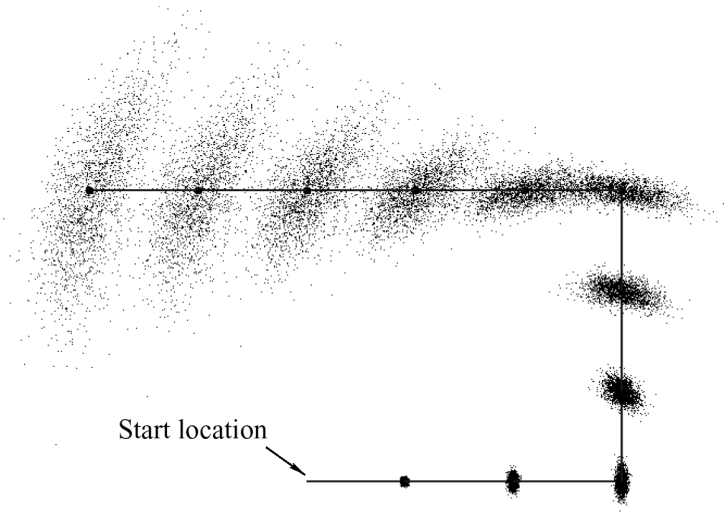


Figure 3.5 : *Filtre à particules : effet de rééchantillonnages successifs.*

En pratique, on n'effectue jamais plusieurs rééchantillonnages successifs, ce qui n'aurait de toute façon pas beaucoup de sens. On alterne les phases de rééchantillonnage avec les phases de mise à jour des mesures.

(b) Lorsqu'un capteur fournit une valeur de mesure v, on dit que l'on met à jour les mesures. Pour chaque échantillon $s_i = \langle \ell_i, p_i \rangle$, on met à jour son poids, selon la formule suivante :

$$p_i \longleftarrow p_i \times P(v|\ell_i)$$

Ensuite, on effectue une normalisation de tous les poids : on divise tous les p_i par $\sum_{i=1}^{N} p_i$.

Sur la figure 3.6, on peut voir un exemple de fonctionnement complet de la localisation Monte-Carlo. Le plan (a) présente la distribution initiale des particules : elles ont été réparties aléatoirement, avec une probabilité uniforme sur tout le plan. Dès que le mobile se déplace un peu, les particules commencent à converger vers sa position réelle.

Dans cet exemple, on constate donc que lorsque le mobile est un robot, dont on connaît précisément le modèle d'action (i.e. les déplacements a effectués), la localisation Monte-Carlo est une méthode très efficace. Deux extensions classiques de cet algorithme permettent d'en améliorer la performance et la fiabilité.

Il est possible d'en concevoir une version *anytime*. Au lieu d'avoir un nombre N d'échantillons fixé par conception (ce qui implique un temps de calcul constant à chaque itération), on s'autorise à travailler sur un nombre d'échantillons variable. Après chaque mesure, on génère des échantillons en continu, jusqu'à ce qu'une nouvelle mesure survienne. Ainsi, lorsque c'est possible, on peut générer un grand nombre d'échantillons, ce qui accroît la précision du système. Par contre, lorsque les valeurs mesurées sont reçues avec une grande fréquence, on

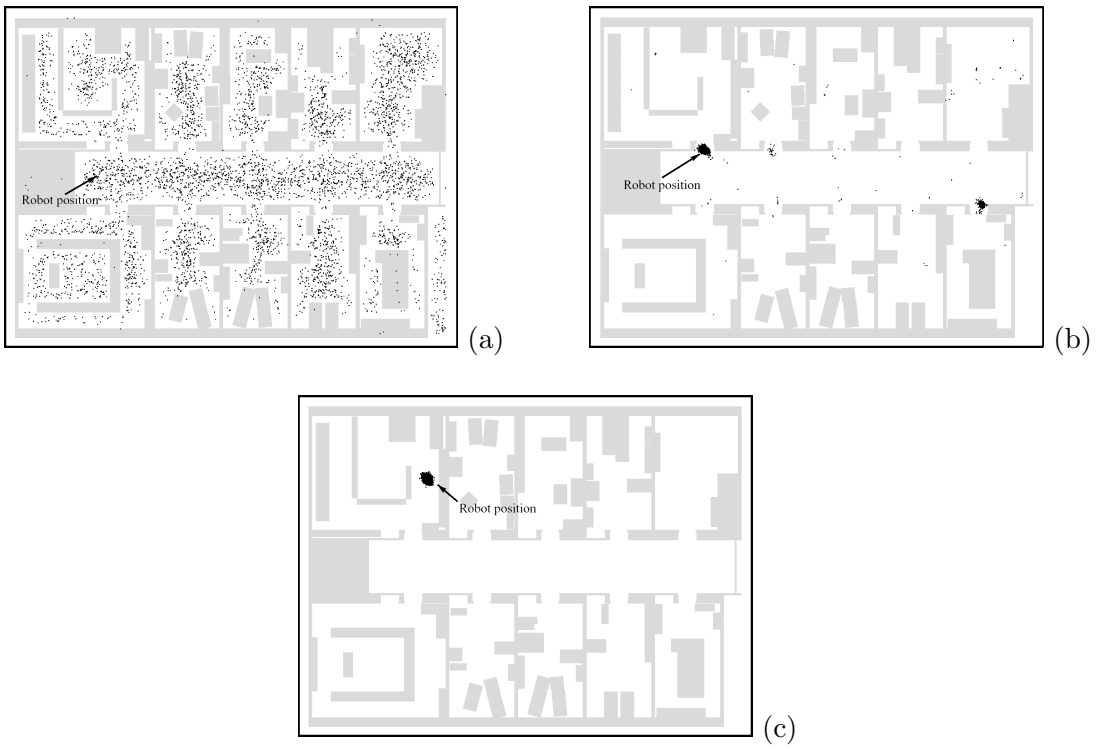


Figure 3.6 : *Localisation par une méthode de type Monte-Carlo. Au départ, les particules sont distribuées de façon aléatoire (a). Après seulement quelques déplacements du mobile, toutes les particules convergent vers la position réelle du mobile (b, c).*

diminue le nombre d'échantillons (et donc la précision), mais le système continue à réagir en temps réel.

Sous certaines conditions, il arrive qu'un filtre à particules « choisisse » une solution qui n'est pas la bonne. Si à un moment donné, une solution est très probable, il est possible que toutes les particules soient « attirées » autour de cette solution, même si elle ne correspond pas à la localisation réelle du mobile. Lorsque toutes les particules se trouvent autour de cette fausse solution, elles n'ont plus vraiment de possibilité de revenir vers la vraie solution : on parle alors de *kidnapping*. Pour surmonter ce problème, on peut alors décider d'introduire systématiquement de la variabilité à chaque étape. On ajoute ainsi quelques échantillons aléatoires répartis uniformément, de façon à pouvoir sortir d'une situation de kidnapping. Cette méthode est baptisée *jittering* ou *roughening*.

3.2.9 Combinaison de plusieurs techniques

Chacune des méthodes de localisation présentées ci-dessus possède ses propres avantages et inconvénients. Cependant, lorsque plusieurs techniques de positionnement sont disponibles simultanément, il est possible d'augmenter la précision et la fiabilité globales, en combinant leurs résultats [Lallement 1999].

Ainsi, dans la *location stack* [Hightower 2002], qui propose une approche par couches à la capture du contexte lié à la localisation, les données brutes des capteurs (couche 1) sont d'abord converties en *mesures* (couche 2), avant d'être combinées par des algorithmes de *fusion* (couche 3). La redondance entre les capteurs est au cœur du système : elle garantit la fiabilité de l'ensemble, ainsi que la capacité d'une réalisation à s'adapter à différentes applications contextuelles.

Si plusieurs techniques de positionnement géographiques sont disponibles en même temps, par exemple par GPS et par triangulation par rapport aux points d'accès WiFi, on peut réaliser la fusion de ces informations. On peut se contenter de les moyenner, ce qui doit permettre d'éliminer un certain nombre d'erreurs, mais on peut également utiliser des méthodes mathématiques plus évoluées, comme l'une de celles citées ci-dessus, ou bien un filtre de Kalman[21].

De même, les informations de positionnement géographique et symbolique peuvent être utilisées de façon complémentaire. Ainsi, il est possible de suivre la position d'objets grâce à un système de vision par ordinateur (localisation géographique), tout en attribuant une étiquette symbolique aux objets repérés lorsqu'on détecte leurs badges RFID, ou bien les points d'accès WiFi qui leur sont attachés [Anne 2005].

3.3 Applications contextuelles

Nous appelons *application contextuelle* un dispositif informatique qui exploite le *contexte*, tel qu'il a été défini au cours de la section 3.1. Dans cette section, nous donnons quelques exemples emblématiques de telles applications.

Assistants mobiles Dans de nombreux cas, le but des applications est de fournir des informations contextuelles aux utilisateurs. Pour ce faire, elles utilisent souvent de petits dispositifs portables : par exemple, le Cyberguide [Long 1996], un guide touristique pour les musées, était basé sur l'assistant personnel (PDA) d'Apple (Newton). Plus généralement, la plupart des systèmes sensibles au contexte sont censés fournir des informations aux utilisateurs sur leur environnement, par exemple, *« où se trouve la pizzeria la plus proche ? »* [Hull 1997].

Dans le même ordre idée, on peut concevoir des assistants pour faire ses courses : un tel appareil peut établir un comparatif des prix entre les magasins, et guider l'utilisateur dans les rayons [Asthana 1994]. Pour la désignation et l'identification d'objets dans ces conditions, CoolTown [Kindberg 2001] suggère d'identifier les objets du monde réel par des URL[22], et permet ainsi les échanges d'information entre objets. La capture du contexte est entièrement *ad hoc*, et elle est limitée à l'identification du lieu courant et des objets désignés par l'utilisateur. CoolTown fournit à ce dernier des pages Web en fonction de sa position et des objets d'intérêt.

[21] Voir `http://www.cs.unc.edu/~welch/kalman/`.

[22] Uniform Resource Locator.

De façon plus générale, le système *Stick e-notes* [Pascoe 1997] se propose d'associer des objets multimédias à divers contextes. Un Stick e-note se compose de deux parties :

1. un contexte auquel il est rattaché : par exemple, *« Je me trouve au bureau et la température est 25°C »* ;
2. un contenu : des informations ou des actions. Par exemple, *« Note : il faut aller à la plage. »*

L'architecture Stick e-notes vérifie constamment si les conditions (i.e. le contexte) des Stick e-notes sont vérifiées. Lorsque c'est le cas pour un Stick e-note, le contenu correspondant est déclenché. Dans cette architecture, les contenus sont visionnables à l'aide d'une *lentille d'information personnelle*, que l'on peut assimiler à un PDA.

On rejoint ainsi l'idée de Fitzmaurice qui propose d'utiliser les objets du monde réel comme des *ancres* vers des contenus informatiques [Fitzmaurice 1993].

Systèmes fixes Cependant, tous les systèmes d'information n'exigent pas des utilisateurs qu'ils portent des PDA. Il est ainsi possible d'utiliser des systèmes publics d'affichage. C'est ce que fait, par exemple, le Gossip Wall [Streitz 2003]. Dans ce type de réalisations, les systèmes publics d'affichage jouent plusieurs rôles : fournir des informations d'intérêt général quand personne n'est à proximité immédiate, et fournir des informations plus personnelles lorsqu'un utilisateur engage une interaction explicite. Ce comportement n'est pas sans soulever des problèmes de respect de la vie privée, puisque des informations potentiellement personnelles peuvent être affichées sur une surface publique [Vogel 2004].

Migration de services Certains systèmes ont pour but de faire « migrer » des services utiles lorsque l'utilisateur se déplace : par exemple, une application de ParcTab permet de faire migrer des fenêtres X-Window vers l'écran le plus proche. De façon analogue, grâce à ActiveBadge, il est possible de transmettre les appels téléphoniques vers le téléphone le plus proche du correspondant demandé [Want 1992].

Utilisation du contexte dans des systèmes de bureau Dans la mesure où il permet le suivi des utilisateurs, le système ParcTab permet la construction d'une application d'affichage de leur position. L'application ne s'utilise pas *en contexte*, mais elle *utilise* le contexte (dans ce cas limité à la localisation) de tous les utilisateurs du système, pour les représenter au bon endroit sur un plan [Weiser 1993]. De même, le navigateur Mobisaic [Voelker 1994] est consultable depuis un poste de travail fixe. Par contre, les pages affichées peuvent *contenir* des informations qui elles, dépendent fortement du contexte.

Réalité et virtualité Les applications contextuelles que nous venons de citer fonctionnent dans le monde physique. Cependant, elles ajoutent des objets informatiques à l'intérieur

de ce substrat physique de base. Ainsi, on parle dans certains cas de systèmes de *réalité augmentée*, car les interactions se situent principalement dans le monde physique, avec enrichissement des objets physiques par des propriétés virtuelles. Par exemple, lors d'une intervention chirurgicale, un modèle informatique du tracé idéal du bistouri peut apparaître en superposition avec les images de la caméra de contrôle de l'intervention, de façon à guider le chirurgien [Dubois 1999]. Cette approche semble a priori opposée à ce que propose le paradigme de la *réalité virtuelle*, où il s'agit de créer de toutes pièces un monde virtuel dans lequel se déroulent les interactions.

En fait, Milgram a montré qu'il est très difficile de définir de façon précise les concepts de *réalité*, *réalité augmentée*, ou *réalité virtuelle*. Il leur préfère un continuum qui s'étend de la réalité pure (le monde physique) à la virtualité pure (les mondes virtuels) : le *continuum réalité-virtualité* [Milgram 1994] (fig. 3.7).

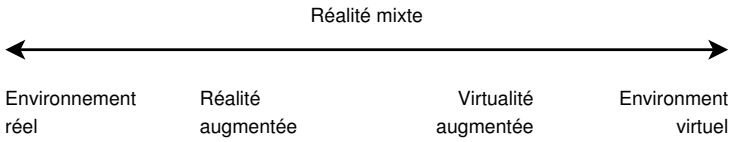


Figure 3.7 : *Représentation du continuum réalité-virtualité.*

Dans ce continuum, on voit apparaître une nouvelle notion : la *virtualité augmentée*. Dans ce cas, des interactions dans un monde purement informatique (virtuel) sont enrichies par des éléments issus du monde physique [Nigay 2001]. Par exemple, dans un bâtiment virtuel, une fenêtre (virtuelle) peut donner sur un paysage filmé dans le monde réel.

3.4 Plates-formes d'accès au contexte

Les premiers systèmes sensibles au contexte réalisaient la capture du contexte de façon ad hoc. En fonction des aspects du contexte dont ils avaient besoin, ils s'interfaçaient directement avec des capteurs physiques qui réalisaient l'acquisition des données du contexte. Des traitements bien spécifiques étaient ensuite appliquées à ces données brutes. Il y avait donc un couplage très fort entre l'*utilisation* du contexte (c'est-à-dire les applications, du point de vue des utilisateurs) d'une part, et sa *capture* d'autre part. En effet, ces deux facettes étaient très fortement interdépendantes, au point de ne former qu'une seule entité.

Cette approche présente un inconvénient majeur : il est nécessaire de repenser entièrement la capture du contexte, de ses aspects logiciels de haut niveau (interface utilisateur) jusqu'à ses aspects matériels de bas niveau, voire physiques, à chaque fois que l'on veut concevoir un nouveau système sensible au contexte. Dans ces conditions, les compétences requises sont nombreuses, car elles mêlent des domaines très divers : manipulation de capteurs physiques (température, éclairement, localisation, etc.), agrégation de données (algorithmes d'inférence d'informations de haut niveau à partir de capteurs de bas niveau), interfaces homme-machine, etc. De plus, le lien intime entre la capture du contexte et les applications rend très difficile la capitalisation du travail réalisé pour sa réexploitation ultérieure.

Au contraire, il serait souhaitable de séparer la capture du contexte de son utilisation. Mieux, il faudrait que ces deux aspects soient *orthogonaux*. De cette façon, le contexte serait vu comme une ressource ou un service, que les applications pourraient interroger lorsque ce serait nécessaire. Le même service pourrait être utilisé pour alimenter plusieurs applications différentes, qui viendraient chacune puiser dans l'ensemble des données contextuelles. Et réciproquement, une même application dépendante du contexte pourrait utiliser, au choix, un certain nombre de services différents pour l'accès au contexte, par exemple en fonction de la disponibilité en temps réel de ces derniers.

Pour ces raisons, des plates-formes génériques pour l'accès au contexte ont été proposées après les applications pionnières des débuts.

3.4.1 Plates-formes existantes

Nous présentons ici un certain nombre de plates-formes d'accès au contexte qui ont été proposées dans la littérature. Nous présentons brièvement chacune d'entre elles, et nous indiquons quelles sont ses particularités. Nous classons ces plates-formes en deux grandes catégories : d'une part, celles qui font appel à un mécanisme de type « tableau noir » pour partager les informations contextuelles, et d'autre part, celles qui sont basées sur la notion de composant.

3.4.1.1 Approches de type « tableau noir »

Le principe de ces systèmes est de disposer d'un espace d'échange commun où tout le monde (i.e. tous les processus en cours de fonctionnement sur un système) peut lire et écrire : le tableau noir. Les informations écrites peuvent être structurées, par exemple sous forme de *tuples*. On parle alors d'*espace de tuples* [Ahuja 1986, Carriero 1989], dont le langage Linda est un exemple majeur. Les informations lues et écrites sont des tuples, parmi lesquels il est possible de faire des recherches par motifs (*pattern-matching*).

Un tuple est de la forme suivante :

$$\langle \texttt{coordinates}, 34, -65 \rangle$$

Ce tuple peut être enregistré dans l'espace de tuples. Il peut ensuite être retrouvé par une recherche de motif du type :

$$\langle \texttt{coordinates}, x?, y? \rangle$$

Ces notions sont reprises au sein de JavaSpaces[23], qui fait partie de la plate-forme Jini[24] pour le langage Java. JavaSpaces introduit en Java les concepts des espaces de tuples et du langage Linda.

[23] Voir `http://www.jini.org/nonav/standards/davis/doc/specs/html/js-spec.html`.
[24] Voir `http://java.sun.com/products/jini/2_1index.html`.

Le projet ParcTab Dans sa thèse soutenue en 1995, Bill Schilit [Schilit 1995] pose les premières fondations des systèmes d'abstraction du contexte. Les informations contextuelles sont mises à disposition des applications par des *dynamic environment servers* auprès desquels il est possible de s'inscrire. Ces serveurs accumulent des informations sur des objets (*dynamic environment objects*), sous forme de couples clé-valeur, appelés *attributs*. Les valeurs sont soit des chaînes de caractères, soit des listes de valeurs.

Les objets sont groupés en *dynamic environnement sets*. Une API[25] permet d'interroger et de s'inscrire auprès d'un *dynamic environment set*. Lors de l'interrogation, le système renvoie les objets qui correspondent à une requête exprimée dans un langage spécifique. Après inscription, le système informe les applications (via des fonctions de rappel) des changements intervenus dans les objets d'intérêt.

Il existe des *dynamic environment servers* pour au moins trois sortes d'entités :

- l'utilisateur, qui est décrit par un *user agent* ;
- le contexte, modélisé par l'*active map service* ;
- les périphériques interactifs, par exemple les PDA ParcTab [Schilit 1993], modélisés par des *device agents*.

ActiveBat Le système ActiveBat [Harter 1999], même s'il est plus un système d'accès à la position géographique (voir section 3.2.3.1) qu'une plate-forme complète d'accès au contexte en général, est basé sur un espace commun de partage des informations.

Il a recours à une modélisation de l'environnement basée sur des objets CORBA[26], rendus persistants grâce à une base de données relationnelle. Les applications peuvent accéder à ces objets en lecture et écriture. Il existe un système d'événements basé sur les modifications des positions géographiques des objets physiques : tous les objets d'intérêt sont suivis grâce à un système de localisation absolue, ce qui permet d'envoyer des notifications aux applications intéressées.

Context Fabric Conçu comme une évolution de ParcTab, le *Context Fabric* [Hong 2002] est centré sur la modélisation des entités du monde réel, sous forme d'*infospaces*. Un *infospace* peut comporter :

- des informations à propos de l'entité elle-même : *« Mon prénom est Christophe »* ;
- des informations perçues par l'entité : *« Je me trouve dans le bureau D2.11 »* ;
- des informations récupérées par l'entité : *« On m'a dit que... »*.

[25] Application Programming Interface.

[26] Common Object Resource Broker Architecture, intergiciel standardisé permettant de construire des applications distribuées en réseau.

Un infospace contient des tuples, qui regroupent des données du contexte, des métadonnées, un historique, et des informations portant sur les droits d'accès. Il est possible de lire les valeurs d'un tuple ponctuellement, périodiquement, ou bien de s'inscrire pour recevoir des notifications lorsque les données changent.

EQUIP EQUIP [Greenhalgh 2002] est une plate-forme pour la réalité mixte, qui offre la possibilité de développer des applications distribuées en Java ou C++. Un service spécifique permet le partage d'informations entre applications : le *dataspace*. Il peut contenir des informations typées, auxquelles on accède soit au travers de notifications (événements), soit par lecture de valeurs. Il est inspiré de la notion d'espace de tuples, voire de tableau noir. L'abonnement aux événements se fait par la donnée d'un *motif* (*pattern-matching*).

Le *dataspace* permet à des applications qui ne se connaissent pas (i.e. qui ne savent pas où elles sont mutuellement localisées) de s'échanger des données. La découverte du *dataspace* en elle-même peut se faire par l'envoi de requêtes réseau de type multicast.

EQUIP a été utilisé pour construire des guides touristiques pour des villes ou des musées, des systèmes de visualisation, des environnements immersifs en 3D, ainsi que des environnements ludiques pour enfants.

Aura Le projet Aura [Judd 2003] se propose de fournir aux applications contextuelles un moyen simple pour accéder aux informations du contexte. Le système se base sur le *service d'information contextuelle* (CIS), qui regroupe une base de données et un service de calcul du contexte. Cependant, la base de données n'est pas statique, comme le sont les bases de données classiques. C'est en réalité une base de données virtuelle, qui joue le rôle d'interface entre les applications et un système de récolte des informations contextuelles au fur et à mesure de leur mise à disposition ou bien sur demande, de façon à ne pas être forcé de stocker toutes les informations en permanence, ce qui est jugé inefficace. De plus, les informations contextuelles sont étiquetées par des métadonnées qui portent sur leur précision, le niveau de confiance à leur accorder, etc.

Les objets du monde réel sont regroupés en classes. Pour chaque classe il existe un type de *synthétiseur de contexte*, qui se charge d'agréger diverses informations contextuelles. Le système comporte quatre classes de base (personnes, appareils électroniques, espaces physiques et réseaux informatiques), mais laisse la possibilité aux applications d'ajouter si besoin leurs propres classes.

Event Heap Dans le cadre du développement d'une salle de réunion interactive (projet iRoom[27]), l'université de Stanford a mis en place un intergiciel dédié au développement des applications collaboratives et multi-supports : iROS [Johanson 2002] (Interactive Room Operating System). Il ne s'agit pas à proprement parler d'une plate-forme générique d'accès au contexte, mais l'un de ses composants, l'*Event Heap* pourrait être utilisé comme telle.

[27] Voir `http://iwork.stanford.edu/`.

En effet, l'Event Heap est une évolution des espaces de tuples : il stocke des messages, appelés *événements*, qui sont des ensembles de couples clé-valeur typés. Par rapport aux espaces de tuples originaux, l'Event Heap introduit la notion de péremption des événements : après un certain délai, les événements non consommés sont retirés du tas.

De cette façon, les applications sont découplées les unes des autres. Elles ne connaissent que l'Event Heap vers lequel elles publient, et à partir duquel elles lisent, des informations. Le découplage est également temporel, car la lecture des informations peut se faire à un moment quelconque après leur publication, tant qu'elles ne sont pas périmées.

3.4.1.2 Plates-formes à composants

Context Tookit Dans son travail de 2001, Anind K. Dey propose une plate-forme générique adaptée à la réalisation de systèmes sensibles au contexte : le *Context Toolkit* [Dey 2001]. Son approche est différente de celle de Schilit : Dey se focalise non pas sur la façon dont les applications peuvent accéder au contexte, mais plutôt sur les moyens d'obtenir des informations contextuelles de haut niveau à partir d'informations brutes issues de capteurs.

Dey propose une architecture à composants, appelés *context widgets*. De même que dans une boîte à outils graphique, un widget représente une unité élémentaire d'interaction, dont la sémantique et le comportement sont définis, un *context widget* encapsule un capteur élémentaire bien précis, qu'il soit matériel ou logiciel. Ainsi, une partie de la logique des applications sensibles au contexte pourrait être conçue comme un assemblage de widgets, comme cela est possible pour l'interface utilisateur des logiciels.

Cependant, il existe une particularité significative des *context widgets*. Dans une interface graphique, un widget *appartient* à une application donnée (par exemple, le bouton « égal » d'un accessoire « calculatrice » lui appartient en propre ; il n'est partagé avec aucune autre application). Au contraire, les *context widgets* existent indépendamment des applications, et sont donc partagés par toutes les applications qui ont besoin de leurs informations. Ceci permet une nette séparation entre la capture du contexte et son utilisation.

Le Context Toolkit fournit d'autres entités de base, qui ressemblent aux *context widgets* dans leur conception, mais qui sont chargées d'accomplir des tâches diverses là où les widgets se contentent d'encapsuler des capteurs :

- les *interpréteurs* convertissent des données brutes en informations de plus haut niveau ;
- les *agrégateurs* rassemblent des informations contextuelles relatives à une entité donnée (personne, lieu ou objet), à partir d'autres widgets ou agrégateurs ;
- les *services* encapsulent les effecteurs, et donc permettent aux applications d'agir sur le monde physique ;
- les *découvreurs* permettent d'accéder aux autres composants. On peut donc les assimiler à des services d'annuaire.

Le Context Toolkit ne fournit pas un catalogue de composants prédéfinis, mais une architecture logicielle au sein de laquelle il est facile de créer des composants correspondant par exemple à un capteur physique, à un effecteur, ou encore à un traitement particulier sur les données. Le concepteur d'un composant n'a pas à se soucier des aspects réseau, ni des moyens d'accès à son composant : tout ceci est pris en charge par le Context Toolkit.

Il existe un mécanisme par lequel les applications peuvent découvrir quels composants sont disponibles pour répondre à une requête donnée. Elles peuvent alors s'inscrire auprès des composants, et recevoir une notification lorsqu'une information intéressante est générée. Cependant, cela implique une certaine manipulation, et éventuellement une certaine connaissance, des composants.

Pour remédier à ce problème, Dey introduit une abstraction supplémentaire, appelée *situation abstraction*. Cette abstraction englobe l'ensemble des composants de contexte. Une application peut donc dialoguer uniquement avec elle, et tout ignorer des composants eux-mêmes. Lorsqu'une application lui envoie une requête, la *situation abstraction* se charge de localiser le composant qui saura y répondre, et lui transmet la requête sous une forme adaptée. Ainsi, les applications traitent toute l'infrastructure de capture et d'agrégation du contexte comme s'il s'agissait d'une entité unique.

Contexteurs La théorie des contexteurs [Coutaz 2002, Rey 2002, Rey 2004] propose une architecture à composants pour la construction d'applications sensibles au contexte. Un *contexteur* est un composant qui peut encapsuler des traitements informatiques, ou bien un capteur physique. En entrée et en sortie, un contexteur reçoit et émet :

- des données : issues des capteurs, elles sont petit à petit transformées de façon à être exploitées après identification du contexte ;
- des métadonnées : elles décrivent les données. Par exemple, elles peuvent fournir des indications de qualité ou de confiance ;
- des contrôles : ils permettent de modifier les paramètres de fonctionnement des contexteurs.

Les contexteurs sont organisés en une série de couches hiérarchiques, en partant des capteurs jusqu'à l'application utilisatrice. Il est également prévu un mécanisme d'encapsulation, qui permet de construire un nouveau contexteur par assemblage de contexteurs existants.

JCAF JCAF [Bardram 2005], pour *Java Context Awareness Framework*, fournit une API pour l'implémentation d'applications sensibles au contexte, ainsi qu'une infrastructure d'exécution. La structure de JCAF est très proche de celle du Context Toolkit. L'API JCAF fournit des classes Java représentant un certain nombre d'objets génériques. Les concepteurs d'applications doivent donc créer les sous-classes concrètes correspondant aux objets mis en jeu dans leurs applications.

La plate-forme JCAF fonctionne de façon distribuée sur un réseau IP[28]. Les capteurs sont encapsulés dans des *context monitors* ; les effecteurs dans des *context actuators*. Les objets du monde réel (utilisateurs, objets d'intérêt) sont représentés par des *entités*. Il existe des *transformateurs* de contexte, qui peuvent traduire ou agréger des informations de contexte.

Le fonctionnement de base est le suivant : les *context monitors* décèlent les modifications du monde réel, et les reportent dans les entités, sous forme de *context items*. Les applications peuvent alors s'inscrire en tant qu'observateurs auprès de *services de contexte*, de façon à obtenir les informations dont elles ont besoin. Toutes les communications sont authentifiées, de façon à assurer sécurité et respect de la vie privée, et peuvent se faire sur deux modes : synchrone (recherche d'une valeur) ou asynchrone (abonnement à un certain type d'événements).

JCAF a été utilisée pour un certain nombre de projets d'enseignement et de recherche.

Mobile Gaia La plate-forme Mobile Gaia [Chetan 2004, Chetan 2005] est centrée sur les utilisateurs. Son objectif est de faciliter le développement d'applications qui mettent en jeu des appareils portés par les utilisateurs, ou situés à proximité immédiate. On parle alors de WPAN, pour *Wireless Private Area Networking*. Cette plate-forme est basée sur la notion de composants, qui peuvent être déployés sur des dispositifs divers.

Au niveau de l'intergiciel, un ensemble de services facilitent le fonctionnement des applications. Notamment, on trouve un service de localisation qui effectue de la fusion de données des capteurs, ainsi qu'un service d'événements qui envoie lorsque c'est nécessaire des notifications aux processus qui se sont abonnés. Il existe également un service chargé de veiller à la sécurité (authentification des dispositifs à l'aide d'un système de signatures à clés asymétriques), ainsi que d'un service responsable du déploiement et du démarrage des applications.

Un espace personnel est « construit » autour de chaque utilisateur. Cependant, deux espaces personnels peuvent se mettre en relation : dans ce cas, des informations peuvent passer de l'un à l'autre, des requêtes peuvent être transmises, etc.

Semantic Space Le projet *Semantic Space* [Wang 2004] a été nommé ainsi par analogie avec le Web sémantique. Ce travail met l'accent sur une représentation de la *sémantique* du contexte, qui est réalisée à l'aide de RDF[29]. Des ontologies du contexte sont décrites en langage OWL[30]. Une ontologie de base de haut niveau est incluse dans le modèle. Elle couvre trois classes d'objets du monde réel (utilisateurs, localisations et appareils informatiques), ainsi que la classe des activités que peuvent effectuer les utilisateurs.

Des *context wrappers* encapsulent les capteurs matériels et logiciels, et communiquent leurs informations à un *context agregator* dont le rôle est d'ajouter les assertions[31] RDF qui corres-

[28] Protocole Internet.
[29] Resource Description Framework, voir p. 66.
[30] Web Ontology Language, voir p. 66.
[31] Une *assertion* RDF est un triplet ⟨ sujet, propriété, objet ⟩.

pondent à une base de connaissances centrale. Un moteur de requêtes permet d'interroger la base de connaissances à l'aide d'un langage de requêtes adapté à RDF. Un moteur d'inférences est également disponible, mais ses résultats ne sont pas stockés dans la base de connaissances de façon à éviter les incohérences.

Sentient Object Model Le *Sentient Object Model* [Biegel 2004] est basé sur trois types de composants :

- les *capteurs* qui *produisent* des événements ;
- les *effecteurs* qui *consomment* des événements ;
- les *sentient objects* qui à la fois *produisent et consomment* des événements.

Les sentient objects effectuent de la fusion de données issues des capteurs et des inférences sur le contexte. Ils sont chargés de la transformation des événements entre les capteurs et les effecteurs.

Résumé Parmi les architectures à composants présentés ci-dessus, certaines proposent l'accès aux informations du contexte directement par requêtes auprès de composants : c'est le cas du Context Toolkit, des contexteurs, de Mobile Gaia et du Sentient Object Model. Cependant, d'autres font appel à une base de données ou de connaissances, qui rassemble les informations sur les objets d'intérêt. C'est le cas de JCAF et Semantic Space. En ce sens, ils rejoignent quelque peu les approches de type « tableau noir ».

Il existe également des plates-formes *génériques* à composants, c'est-à-dire non spécifiques à un domaine en particulier, mais qui peuvent être utilisées pour la capture du contexte. C'est par exemple le cas du CORBA Component Model (CCM) [Merle 2002]. Cependant, dans ce cas, les applications ne peuvent pas tirer parti de fonctionnalités de la plate-forme spécialement dédiées à la gestion du contexte.

Les plates-formes existantes ne fournissent pas toutes des moyens pour accéder au contexte de haut niveau d'une façon reconfigurable dynamiquement, ni de vérifier de façon statique la cohérence entre les entrées et sorties de leurs composants constitutifs. Dans le cadre de l'étude du contexte, nous avons donc proposé notre propre plate-forme, qui intègre ces deux aspects.

3.4.2 Notre proposition de plate-forme

À titre d'exemple, et afin de donner une idée plus précise de la façon dont peut être conçue une plate-forme pour l'accès au contexte, nous décrivons de façon plus détaillée une plate-forme que nous avons spécifiée et implémentée. Celle-ci s'inspire des travaux introduits ci-dessus, et ajoute quelques aspects originaux [Jacquet 2005, Jacquet 2006e].

Cette plate-forme est basée sur deux aspects principaux :

1. la notion de *ruche d'objets*, une abstraction pour l'accès au contexte, qui propose un couplage faible entre capture et utilisation du contexte ainsi que des possibilités de reconfiguration dynamique des applications. Ainsi, la capture du contexte est réalisée de façon *générique*, sans se préoccuper des applications utilisatrices. Ceci permet, d'une part, à n'importe quelle application d'accéder au contexte, et d'autre part, de pouvoir modifier les modalités de capture du contexte sans recompiler ni même arrêter l'exécution des applications existantes. Cette *ruche d'objets* s'inspire des approches de type « tableau noir » ;

2. un *typage fort* de toutes les communications entre composants logiciels, ce qui permet de réaliser des vérifications statiques de cohérence.

3.4.2.1 Modèle et plate-forme

L'objectif de cette plate-forme est de fournir une modélisation (partielle) du monde réel. À chaque objet d'intérêt du monde correspond donc un objet du modèle, et réciproquement, chaque objet du modèle décrit un objet du monde. La plate-forme se charge donc de maintenir le modèle en conformité avec l'état courant du monde (voir fig. 3.8).

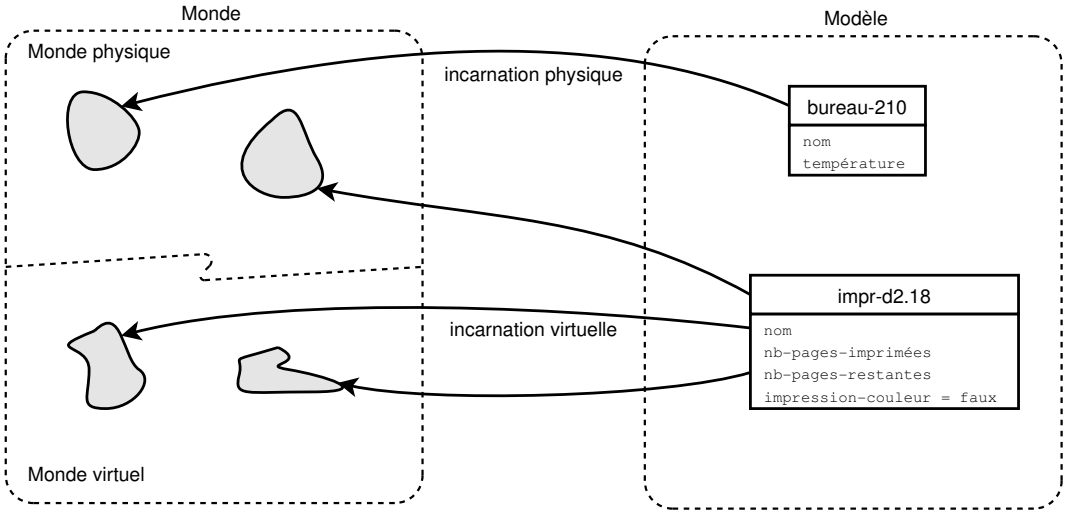


Figure 3.8 : *Relations entre objets du monde et objets du modèle.*

La structure générale de la plate-forme est présentée sur la figure 3.9. Elle est organisée en couches : (a) les *capteurs* sont à la frontière entre le monde et la plate-forme logicielle. Ils scrutent le monde en permanence de façon à maintenir le modèle à jour. (b) La couche d'*agrégation du contexte* se charge de combiner des informations contextuelles, de façon à en déduire du contexte de niveau de plus en plus élevé. (c) Les informations contextuelles de haut niveau qui caractérisent les objets du modèle sont *assignées* aux attributs de ces objets. (d) La *ruche d'objets* matérialise le modèle au sein de la plate-forme.

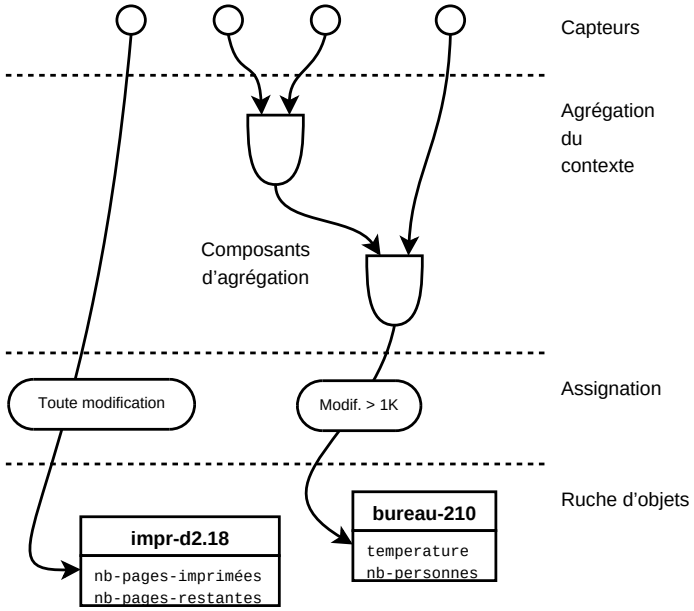


Figure 3.9 : *Capture du contexte dans notre plate-forme d'accès au contexte.*

3.4.2.2 Composants et typage

Notre plate-forme est basée sur la notion de composant de contexte (qui correspond à un *context widget* [Dey 2001] ou à un *contexteur* [Rey 2002]). Il s'agit de composants logiciels relativement autonomes qui disposent d'entrées et de sorties. Il en existe de deux sortes : (a) les *capteurs* qui ne disposent pas d'entrée et encapsulent un capteur physique, et (b) les composants d'agrégation qui effectuent un traitement sur leurs entrées et modifient leurs sorties en conséquence.

Les composants d'agrégation peuvent être assemblés en cascade, de façon à transformer petit à petit les informations contextuelles : une sortie d'un composant dit *producteur* est alors connectée sur une entrée d'un composant dit *consommateur*. Une taxonomie en a été proposée par Rey et al. [Rey 2002].

Afin d'assurer la cohérence des données manipulées par le système, il est utile de typer les entrées et les sorties. Ainsi, pour qu'une connexion soit possible entre deux composants de contexte, il est nécessaire que le type de la sortie du premier soit compatible avec le type de l'entrée du second. Une telle règle permet la vérification *statique* de type et donc évite certaines erreurs au moment de l'exécution, ce qui facilite la mise au point d'une application utilisatrice.

Exemple : sur une entrée destinée à recevoir l'accélération de la pesanteur (g, mesurée en $\mathrm{m{\cdot}s^{-2}}$ [mètres par seconde par seconde]), il sera impossible de connecter une sortie fournissant la température (mesurée en K [kelvins]).

Dans cet exemple, on voit un premier typage possible : le typage des grandeurs mesurées à l'aide des unités correspondantes du système international. En réalité, nous envisageons

deux grandes classes d'entrées/sorties : les entrées/sorties de *valeurs* et les entrées/sorties d'*événements*.

Entrées/sorties de valeurs Les entrées/sorties de *valeurs* peuvent servir à véhiculer des informations aussi diverses que des grandeurs physiques ou des structures de données informatiques complexes (photographies, échantillons sonores, etc.)

Une sortie de valeurs possède une valeur à chaque instant. Par exemple, un capteur physique, comme un capteur de température, mesure sa grandeur en permanence, donc cette dernière peut être lue à tout moment. Un composant consommateur connecté sur cette sortie disposera de plusieurs moyens d'obtenir les informations : (a) demander la valeur courante de la sortie à un moment donné, (b) s'abonner auprès du producteur de façon à recevoir une notification lors de certains changements (spécifiés lors de l'abonnement), et (c) demander à recevoir la valeur à une fréquence fixée.

Entrées/sorties d'événements Les sorties d'*événements* obéissent à une logique différente. Elles ne possèdent pas de valeur à chaque instant, donc on ne peut pas les interroger à tout moment. En contrepartie, elles émettent ponctuellement des messages ou *événements* à l'attention des consommateurs qui leur sont connectés (et qui se sont abonnés).

Chaque type d'événement est nommé, c'est-à-dire qu'il doit être identifié par un nom unique dans le système. Par exemple, un capteur de passage (de type *barrière lumineuse*) enverra un événement `passage-détecté` lors de chaque franchissement.

Les types d'événements peuvent être classés dans une hiérarchie, au sein de laquelle tous les événements sont des descendants d'un ancêtre commun. Ainsi, une sortie d'événements de type T_1 pourra être connectée sur une entrée d'événements de type T_2 d'un autre composant si T_1 est un sous-type de T_2.

Discussion Pour ne pas compliquer inutilement le système par multiplication du nombre de composants, il est possible d'introduire deux formes de *polymorphisme*, de sorte qu'un même composant soit capable de travailler de façon générique sur des types différents. Nous introduisons donc du polymorphisme paramétrique, comme dans ML [Appel 1991], ainsi que du polymorphisme par sous typage, comme dans les langages à objets.

Cependant, il reste fondamentalement deux sortes d'entrées et de sorties : les entrées/-sorties de valeurs et d'événements. Il est légitime de se demander si cette distinction est réellement utile. En effet, pourquoi ne pas remplacer les sorties d'événements par des valeurs booléennes inversées lors de chaque occurrence des événements considérés, les entrées d'événements s'abonnant alors simplement aux changements des sorties ?

Exemple : ainsi, un détecteur de passage fournirait une valeur de sortie valant tantôt `vrai` et tantôt `faux`, le changement s'effectuant lors des passages. Pour être notifié des passages, il suffirait de s'abonner aux changements de cette sortie booléenne.

Cette solution peut sembler séduisante car elle élimine la distinction entre entrées/sorties de valeurs et entrées/sorties d'événements. Cependant : (a) elle conduit à un appauvrissement de la sémantique des entrées/sorties : plus rien ne permet de vérifier la cohérence sémantique entre entrées/sorties d'événements. De plus, les sorties n'ont plus forcément de sens : la valeur booléenne de sortie ne signifie rien ; seuls ses changements sont importants ; et (b) elle considère que les événements ne peuvent pas être vecteurs d'information. Or, l'architecture STEAM[32] [Meier 2003] nous montre qu'il est naturel de munir les événements d'*attributs*.

Exemple : il peut être nécessaire de mesurer le temps de franchissement de notre barrière lumineuse. Ce temps de franchissement doit être associé à chaque passage détecté. Il est donc naturel de définir un type d'événements `passage-mesuré`, sous-événement de `passage-détecté`, qui possède un attribut `temps-de-passage`, grandeur physique mesurée en secondes.

En bref, s'il est possible *en principe* de se passer d'événements, ceux-ci permettent de représenter la sémantique des entrées/sorties avec élégance, tout en offrant la possibilité d'une vérification plus fine de la cohérence. Le choix entre entrées/sorties de valeurs et d'événements apporte une grande souplesse dans la spécification des composants et leur donne une sémantique riche.

3.4.2.3 Ruche d'objets

Ruche et assignation Les objets du modèle sont matérialisés au sein d'un *service* particulier, appelé *« ruche »* d'objets, qui contient une description de tous les objets d'intérêt. Les applications peuvent s'abonner auprès de la ruche afin d'être tenues informées des changements intervenus sur le *contexte de haut niveau*, c'est-à-dire sur les attributs des objets du modèle.

Exemple : une application peut s'abonner à l'attribut `nb-personnes` de l'objet `bureau-210` de façon à être tenue informée des allées et venues dans le bureau en question.

La mise à jour des attributs de la ruche peut être réalisée de façon très simple, en « connectant » ces attributs en sortie de composants d'agrégation. Nous nommons cette connexion *assignation*. Ainsi, les attributs de la ruche se comportent comme des *consommateurs* vis-à-vis des sorties des composants de contexte qui leur sont assignées. Notons que seules les sorties de *valeurs* peuvent être assignées à des attributs, et que la compatibilité de types est vérifiée de la même façon qu'entre composants.

En conséquence, les sorties d'*événements* ne peuvent pas être directement assignées à des attributs. Il est nécessaire de passer par un composant qui effectuera en quelque-sorte une *conversion*, par exemple un compteur d'occurences d'un événement.

Exemple : la sortie du composant de mesure de température, obtenue par agrégation d'informations issues de trois capteurs physiques différents, peut être assignée à l'attribut `température` de l'objet `bureau-210` de la ruche.

[32] Scalable Timed Events And Mobility.

Importance de la ruche La ruche ne fournit pas seulement un moyen de *nommer* les attributs dans un modèle objet. En effet, elle introduit une couche d'abstraction qui matérialise un couplage faible entre deux processus bien différents : (a) d'une part, la *capture et l'agrégation du contexte* par des chaînes de composants, en amont de la ruche ; et (b) d'autre part, l'*utilisation du contexte* par des applications, en aval de la ruche.

Sans la ruche, les applications seraient *directement* reliées aux chaînes de composants de contexte. En conséquence, lors d'une modification de ces composants (par exemple, lors d'une reconfiguration du système), les applications devraient elles aussi être modifiées.

Exemple : une application installée dans un bâtiment du nord de l'Essonne a besoin de connaître la température extérieure. Dans un premier temps, on ne dispose pas de sonde de température. On récupère donc l'information de température par le réseau, à travers des données METAR[33] transmises par l'aéroport d'Orly. Dans un deuxième temps, on finit par installer un capteur de température à l'extérieur, qui donnera donc une information plus pertinente que celle transmise par Orly, situé à quelques kilomètres de là.

Cas n°1, sans la ruche : au départ, l'application était abonnée au composant d'accès aux données METAR. Lors du changement, le code source correspondant a été remplacé par un abonnement au composant encapsulant la sonde de température extérieure. *Après recompilation*, l'application fonctionne à nouveau, mais avec des données plus précises.

Cas n°2, avec la ruche : au départ, la sortie du composant d'accès aux données METAR, était assignée à l'attribut `température-extérieure` de l'objet représentant le bâtiment d'intérêt. Lors de sa mise à disposition, le composant correspondant à la sonde extérieure a simplement été assigné à l'attribut `température-extérieure`, en lieu et place du précédent. L'application, située *de l'autre côté* de la ruche, n'a même pas été arrêtée. Elle a continué à venir piocher ses informations dans la ruche *sans se rendre compte de rien, mais en profitant du nouveau capteur dès sa mise en place.*

Ainsi, le code source des applications ne fait aucune référence aux capteurs. Toutes les dépendances sont indiquées de façon *déclarative* et peuvent être modifiées au cours de l'exécution, ce qui autorise une *reconfiguration dynamique* des applications.

De plus, toute la couche de capture et d'agrégation du contexte peut être construite de façon indépendante du reste, sans forcément avoir en tête toutes les applications possibles. Les applications en question peuvent n'être connectées qu'après coup, et sans modifier l'architecture de capture.

3.4.2.4 Solutions technologiques

Nous avons réalisé une implémentation en Java de cette plate-forme. Les principaux objectifs de cette implémentation sont les suivants :

[33] METeorological Aerodrome Routine Weather Report.

- système distribué sur un réseau, lequel doit être transparent à la fois lors de la conception de composants de capture et d'agrégation du contexte, et lors de la réalisation d'applications ;
- système basé sur des protocoles simples et ouverts.

D'après ce qui précède, une application contextuelle conforme à notre architecture sera constitué d'un certain nombre de *composants de contexte*. Notre implémentation propose donc de faire fonctionner ces composants sur les ordinateurs d'un réseau. Chaque ordinateur sera équipé d'un logiciel serveur, chargé de permettre aux composants qu'il héberge de communiquer avec les autres composants, qu'ils soient locaux ou distants.

Pour la communication entre composants, il nous a semblé pertinent d'échanger des fragments de graphes RDF, de façon à pouvoir décrire formellement la sémantique des vocabulaires grâce à OWL. RDF est un modèle conceptuel, et il existe plusieurs formats de représentation correspondants. Cependant, la représentation RDF/XML[34] nous semble la plus pratique ; c'est donc elle que nous avons utilisée.

De cette façon, il est possible de considérer les serveurs situés sur chacun des ordinateurs d'une application comme des *services Web* qui communiquent par protocole HTTP[35]. Nous nous sommes tournés vers des services Web très légers implémentés grâce au protocole XML-RPC[36], réputé pour sa simplicité. Cependant, il serait possible d'utiliser des standards de services Web plus élaborés, comme SOAP[37]

Composants et identification Comme nous l'avons vu précédemment, le rôle des serveurs est d'héberger les composants et de permettre la communication entre eux. La ruche peut être considérée comme l'un de ces composants. Elle est hébergée sur l'un des ordinateurs du système, de la même façon que n'importe quel autre composant.

Il peut être utile de pouvoir obtenir la liste de tous les composants disponibles. C'est pourquoi chaque serveur dispose d'un composant particulier, appelé `annuaire`, qui est capable de donner la liste des autres composants disponibles localement sur ce serveur. Il pourra être envisagé dans le futur de donner aux annuaires la possibilité de dialoguer entre eux, afin d'obtenir la liste des composants disponibles sur *tout* le réseau, et pas seulement localement [Bettstetter 2000].

Ainsi, l'architecture générale de notre implémentation ressemble à l'exemple de la figure 3.10. Chaque serveur héberge des composants (dont certains correspondent à des capteurs, les seuls représentés ici). Un serveur est identifié par un URI[38] qui reprend l'adresse physique du serveur et son numéro de port. Au sein d'un serveur donné, chaque composant est identifié par un nom unique.

[34] RDF sur eXtensible Markup Language.
[35] HyperText Transfer Protocol.
[36] XML-Remote Procedure Call.
[37] Simple Object Access Protocol.
[38] Uniform Resource Identifier

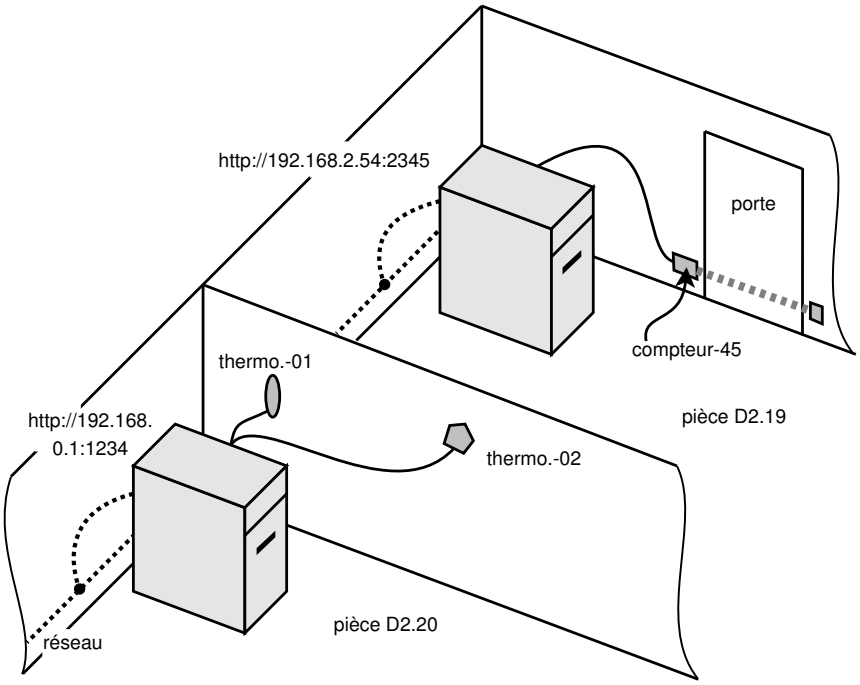


Figure 3.10 : *Exemple de déploiement de système d'architecture contextuelle.*

De cette façon, il est possible de définir des URI pour chacun des composants du système, en concaténant l'URI du serveur et le nom local du composant, séparés par une barre oblique.

Exemple : le composant `thermomètre-02`, situé sur le serveur identifié par l'URI `http://192.168.0.1:1234`, sera à son tour identifié par l'URI suivant : `http://192.168.0.1:1234/thermomètre-02`.

Ainsi, on peut facilement désigner un composant quelconque par son URI, ce qui est très important lorsqu'on utilise RDF, car tous les objets (ou *ressources* pour reprendre la terminologie dédiée) y sont désignés par des URI. Ainsi, les URI de notre système sont *à la fois* des identifiants *logiques* (vis-à-vis de RDF) *et* des identifiants *physiques*, permettant l'accès aux composants par la plate-forme à travers les couches réseau.

Construction dynamique d'une application Un éditeur graphique permet d'assigner des sorties de composants à des attributs dans la ruche (voir fig. 3.11). De cette façon, il est possible de construire une architecture de capture du contexte sans même écrire une seule ligne code.

3.4.2.5 Conclusion sur notre plate-forme

Cette plate-forme à composants présente deux aspects originaux :

- un typage fort des messages échangés entre composants ;

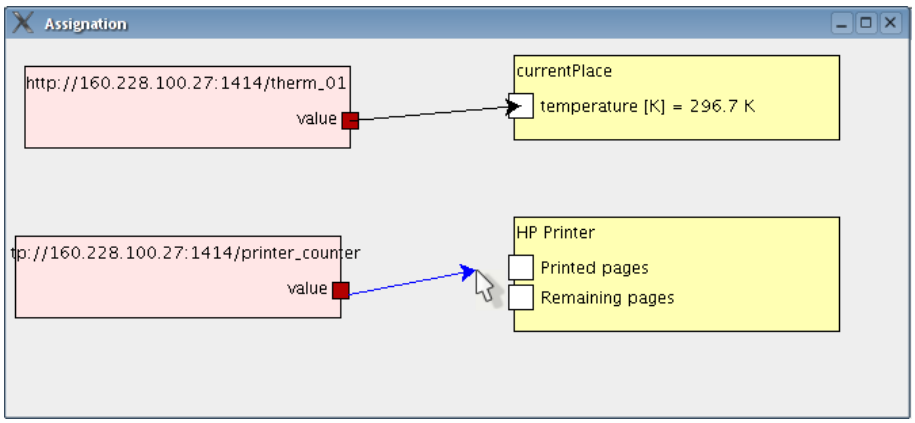


Figure 3.11 : *Assignation à la souris de valeurs en sortie à des attributs de la ruche.*

- le concept de *ruche d'objets*, un mécanisme de haut niveau pour l'accès au contexte, qui permet de reconfigurer dynamiquement les applications.

Parmi les voies d'amélioration possibles, on peut citer :

- une gestion de l'historique du contexte ;
- une meilleure formalisation des types de données, ainsi qu'une méta-descriptiption des composants ;
- la capacité à gérer des *contextes dynamiques*, par exemple en permettant un changement des objets d'intérêt au gré des déplacements des utilisateurs. Ainsi, l'objet *« imprimante la plus proche »* pourrait désigner successivement des imprimantes différentes lors des mouvements d'un utilisateur.

3.5 Conclusion

Dans ce chapitre, nous avons donné une définition du contexte, et traité en détails l'une de ses composantes principales, la *localisation*. Dans la dernière partie de cette thèse (« Implémentation »), nous décrirons une application contextuelle, pour laquelle le contexte possède deux composantes :

- les capacités des utilisateurs et des dispositifs en termes d'interaction multimodale. Ces points ont été détaillés au chapitre 2 ;
- la localisation symbolique. Dans nos expérimentations, nous utiliserons une méthode de capture de la localisation basée sur des badges à infrarouges (voir annexe C), et inspirée de systèmes qui ont été présentés dans le présent chapitre.

Deuxième partie

Le modèle

Chapitre 4

Motivations

4.1 Introduction

Dans ce chapitre, nous indiquons quels sont les buts de notre étude et quels sont les objectifs à atteindre. Nous soulevons un certain nombre de problèmes dont nous jugeons l'étude intéressante et importante. Notre modèle, exposé au chapitre 5, traitera une partie des problèmes et objectifs évoqués ici.

Nous nous fixons comme objectif de présenter des informations *en contexte* à des utilisateurs mobiles. Un premier paragraphe traitera donc des moyens d'effectuer cette présentation, puis nous nous pencherons sur le choix d'un lieu d'étude qui servira de fil conducteur pour la suite. Dans une seconde section, nous identifierons quels sont les problèmes induits par cette approche, avant d'indiquer ceux qui ont été traités dans notre travail.

Présentation d'informations en contexte Nous voulons être capables de fournir des informations aux utilisateurs d'un lieu donné, là où ils en ont besoin, lors de leurs déplacements dans ce lieu. Notre but est d'utiliser de façon *opportuniste* les dispositifs appropriés que les utilisateurs peuvent rencontrer au cours de leurs déplacements.

Ces dispositifs peuvent être par exemple des écrans publics, des haut-parleurs, ou même des appareils privés (PDA, Tablet PC, écran de téléphone mobile, tablette Braille). De façon générique, nous les appelons *dispositifs de présentation*. Ce sont des appareils capables de présenter une information selon une ou plusieurs modalités données (visuelle, auditive ou TPK pour les exemples cités ci-dessus). Le tableau 4.1 présente une taxonomie des dispositifs de présentation. Nous distinguons les dispositifs *publics*, perceptibles par tous les utilisateurs situés à proximité, et les dispositifs *privés*, qui appartiennent en propre à un utilisateur donné et ne sont perceptibles que par celui-ci.

Dans ce tableau, l'*étiquette* évoquée en tant que dispositif de présentation visuel et public correspond à une étiquette physique, statique, apposée dans un environnement. Par exemple, ce peut être le nom du propriétaire d'un bureau dans une entreprise. Bien entendu, une telle

Mode	Dispositifs publics	Dispositifs privés
Visuel	Écran Panneau d'affichage Journal lumineux Étiquette	PDA Écran de téléphone mobile Tablet PC Montre
Auditif	Haut-parleur	Oreillette Sonnerie de téléphone mobile
TPK	—	Tablette Braille Vibreur fixé au doigt Vibreur de téléphone mobile

Tableau 4.1 : *Taxonomie des dispositifs de présentation.*

étiquette n'est pas un dispositif électronique, mais elle permet néanmoins de présenter une information, même s'il s'agit toujours de la même (caractère *statique* de l'étiquette). Pour cette raison, il peut être intéressant de modéliser explicitement de telles étiquettes dans le système de fourniture d'information.

Choix d'un lieu d'étude Le but général est de fournir des informations à des utilisateurs mobiles, dans des environnements où il est essentiel de disposer d'informations. Nous devons donc choisir un lieu *paradigmatique*, qui sera utilisé pour construire nos exemples. A priori, notre plate-forme doit pouvoir être déployée dans toutes les catégories de lieux publics, et pour des utilisations variées. Le tableau 4.2 liste quelques lieux typiques dans lesquels notre système pourrait être utilisé.

Lieu	Taille	Identification des besoins des utilisateurs
Station de métro ou de RER	−	facile, voire triviale
Gare	+ à ++	facile
École	+ à ++	difficile
Centre commercial	++	difficile
Aéroport	++	relativement facile
Ville tout entière	+ + +	très difficile

Tableau 4.2 : *Quelques lieux de déploiement possibles pour notre système, classés par tailles croissantes.*

La taille des lieux ne nous semble pas appeler de commentaire particulier. Intéressons-nous par contre à la colonne appelée *« identification des besoins des utilisateurs »*. Elle indique à quel point il est facile de capturer les besoins des utilisateurs en termes d'information. Dans les lieux liés au transport, cette capture est relativement facile. En effet, le but global des utilisateurs *en partance* est clairement identifié : il s'agit de prendre le métro, le train ou l'avion. Ce but est commun à tous les utilisateurs de ces lieux. Les buts individuels,

plus précis (*« prendre le train pour Lyon »*, *« prendre l'avion pour Nice »*) sont également identifiables. Par exemple, il est possible d'équiper les billets de train ou d'avion d'étiquettes RFID, qui pourraient être lues par des lecteurs appropriés. Dans une station de métro, la détection de la destination peut même être complètement triviale, car réduite à une seule possibilité. Par contre, les besoins des utilisateurs ne sont pas capturables aussi facilement dans des environnements polyvalents tels que les centre commerciaux ou, a fortiori, les villes.

Pour que des simulations soient réalisables, l'identification des utilisateurs doit être relativement facile, et au moins réaliste. De plus, nous voulions un lieu d'une taille suffisante pour générer des situations intéressantes. Nous avons donc choisi de situer nos exemples dans un aéroport. D'une part, les buts des utilisateurs sont facilement inférables à partir du lieu où ils se trouvent et de leurs titres de transport. Et d'autre part, il existe plusieurs zones différentes dans un aéroport (salles d'attente, cafétéria, etc.), ce qui nous servira à démontrer certains aspects de notre système.

De plus, la grande taille d'un aéroport provoque chez les utilisateurs un besoin d'information plus grand que dans une structure de plus petite taille. Ainsi, dans une petite station de métro, les passagers ne se posent pas de question particulière : leur quai est indiqué de façon non ambiguë ; ils savent que le prochain métro arrivera sous peu, etc. Il en est tout autrement dans un aéroport, où de multiples destinations sont desservies par de multiples compagnies, et où les passagers sont amenés à se rendre dans des lieux bien différents.

4.2 Problèmes à traiter

4.2.1 Mise en contexte dynamique

4.2.1.1 Mobilité des utilisateurs

Les utilisateurs sont considérés comme étant extrêmement mobiles. Ils sont susceptibles de se déplacer sans arrêt pour des motifs variés, et nous voulons éviter qu'ils se déplacent spécifiquement pour aller chercher de l'information. Au contraire, nous voulons leur présenter des informations lors de leurs déplacements habituels, là où ils en ont besoin.

Ainsi, lorsqu'un utilisateur se trouvera à proximité d'un dispositif de présentation[1], ce dernier effectuera la présentation des informations réputées pertinentes pour l'utilisateur.

Par exemple, imaginons qu'un passager ordinaire arrive dans un aéroport. Les informations pertinentes pour le passager sont les renseignements relatifs à son vol (comptoir d'embarquement, heure de départ, etc.). Au cours de ses déplacements, lorsque l'utilisateur s'approchera d'un écran public, ces informations s'y afficheront donc. Il est également possible que ce passager possède des dispositifs de présentation privés, par exemple une oreillette

[1] C'est-à-dire lorsque l'utilisateur sera capable de *percevoir* les informations présentées par le dispositif en question. La notion de *proximité* sera définie formellement au chapitre 5.

reliée à son téléphone portable. Dans ce cas, on peut imaginer que les informations lui soient fournies par synthèse vocale dans son oreillette.

De cette façon, les dispositifs publics d'affichage ne présenteraient plus un ensemble d'informations censées convenir à tout le monde, comme ils le font de façon classique, mais seulement les informations intéressantes pour les utilisateurs situés à proximité. Par exemple, dans un aéroport se trouvent des panneaux d'affichage qui donnent les horaires des prochains vols. Comme ils affichent une grande quantité d'informations, il est relativement difficile d'y trouver l'information dont on a besoin (voir fig. 4.1). Ils seraient beaucoup moins chargés, et donc la recherche serait bien plus facile, s'ils n'affichaient que les informations utiles aux voyageurs situés devant eux.

Figure 4.1 : *À l'aéroport de Roissy, un mur d'écrans affiche en tout 160 vols, même si seulement trois passagers sont en train d'y rechercher des informations.*

4.2.1.2 Mobilité des dispositifs de présentation

Agrégation d'écrans Nous venons de voir que les dispositifs de présentation pourraient ne fournir des informations qu'aux utilisateurs situés à proximité, ce qui limiterait généralement la quantité d'informations présentées, et donc faciliterait la recherche. Cependant, si de nombreuses personnes se rassemblent devant un dispositif, par exemple un écran, celui-ci va rapidement subir une surcharge d'informations. En conséquence, les utilisateurs vont devoir rechercher leur information parmi une longue liste d'informations non pertinentes. Notons que ce n'est pas un défaut du système : il s'agit seulement d'une situation dans laquelle il n'est pas meilleur que les systèmes classiques.

Néanmoins, dans une telle situation, on peut penser que les responsables du lieu considéré chercheront à trouver une solution à ce problème de surcharge des écrans. Il paraît ainsi relativement intuitif d'apporter un *deuxième* écran, à côté du premier, de façon à augmenter la surface disponible. Cependant, sans prise en compte particulière de cette situation, les deux écrans ne vont pas coopérer entre eux : le second va se contenter de recopier le contenu

du premier, ne diminuant ainsi en rien la surcharge. En d'autres termes, le deuxième écran sera quasiment inutile.

La solution de ce problème réside dans une judicieuse *répartition du contenu* [Mansoux 2005] entre les deux écrans (voir fig. 4.2). Il faudrait qu'une partie des informations soient affichées sur le premier écran, et que les autres soient affichées sur le deuxième. De cette façon, la quantité d'informations de chaque écran serait à peu près diminuée de moitié.

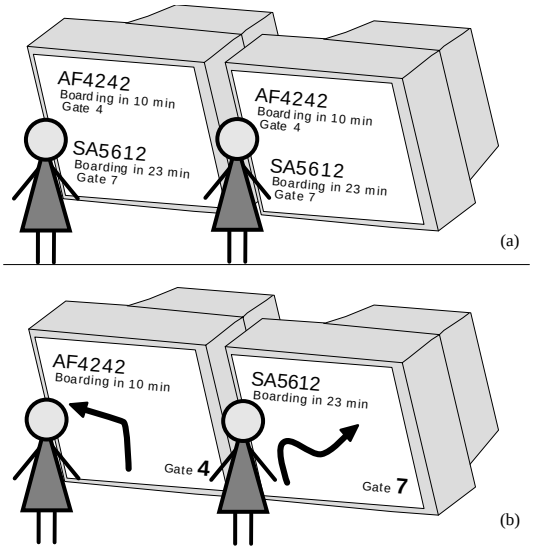


Figure 4.2 : *Deux écrans dans un aéroport. Simplement voisins dans le cas (a), ils recopient leur contenu. Dans le cas (b), ils coopèrent entre eux, ce qui permet de répartir le contenu.*

Fonctionnement sans configuration Comme nous venons de le voir, des dispositifs de présentation proches doivent pouvoir se répartir un ensemble d'informations à afficher. L'une des solutions serait d'implémenter la collaboration entre les deux dispositifs par un ensemble de paramètres de configuration donnés explicitement.

Si cette solution présente une certaine facilité lors de la conception du système, elle empêche cependant une reconfiguration dynamique simple des dispositifs. En effet, dans ce cas, si l'on déplace physiquement les écrans, il est nécessaire de modifier explicitement les paramètres de configuration du système, de façon à refléter les modifications apportées à la réalité physique. Cette approche présente deux inconvénients majeurs :

- ce n'est pas très pratique, car il faut effectuer *deux* modifications, l'une physique (le déplacement de l'écran), l'autre logicielle (la reconfiguration du système) ;
- si l'on oublie de reconfigurer le logiciel, cela crée une *incohérence* profonde entre le monde physique et sa modélisation logicielle. En pratique, deux écrans éloignés continueront à collaborer comme s'ils étaient toujours côté à côte, ce qui peut poser de sérieux problèmes. Par exemple, il est ainsi possible que des utilisateurs ne voient pas l'information qui leur est destinée, car elle s'afficherait sur un écran situé complètement

à l'opposé. Réciproquement, deux écrans approchés l'un de l'autre ne collaboreraient pas spontanément.

Au final, il ne semble pas raisonnable d'avoir recours à un tel mécanisme de configuration manuelle. Nous proposons donc que les dispositifs de présentation n'aient pas de connaissance *a priori* de leurs conditions de fonctionnement. Nous souhaitons qu'ils prennent connaissance par eux-mêmes, et en permanence, de leur environnement, et donc en particulier de quels autres dispositifs sont situés à proximité. Ainsi, ils pourront être placés (et déplacés) à n'importe quel endroit sans qu'il y ait besoin de (re-)configurer quoi ce soit de façon manuelle. Où qu'ils soient, ils pourront fournir des informations pertinentes aux utilisateurs situés à proximité, et collaborer avec les dispositifs voisins de façon à améliorer le confort des utilisateurs.

Par exemple, supposons que dans un endroit donné soit installé un haut-parleur. Ce haut-parleur pourra donc naturellement être utilisé pour fournir des informations à une personne non-voyante. Cependant, si un message est en cours de diffusion lors de l'arrivée de la personne aveugle, et si cette dernière dispose de ses propres écouteurs, il serait intéressant d'utiliser ces écouteurs pour lui transmettre de l'information. Ainsi, sans aucune configuration préalable, des dispositifs supplémentaires, même privés, pourront être utilisés de façon spontanée.

4.2.1.3 Accrochage sémantique entre informations

Prenons l'exemple d'un restaurant situé dans un aéroport. Dans l'enceinte du restaurant, les utilisateurs sont donc susceptibles de recevoir des messages sur deux sujets différents :

- le départ des vols de l'aéroport ;
- les menus du restaurant.

De façon basique, on peut imaginer que ces deux types d'informations soient présentées conjointement aux utilisateurs. Cependant, il serait intéressant de les mettre en relation les unes avec les autres. Par exemple, si un passager doit se présenter à son comptoir d'embarquement dans 40 minutes, il est inutile de lui fournir les informations concernant les menus qui ne peuvent pas être servis dans ce laps de temps. C'est ce que nous appelons *accrochage sémantique entre informations*.

Pour ce faire, il faut que la *sémantique* des différentes informations soit décrite sous une forme exploitable par la machine. En effet, il est nécessaire d'appliquer des règles de raisonnement sur les informations de façon à réaliser ces *accrochages*. Cela soulève deux problèmes principaux :

- le langage de description doit être suffisamment expressif pour exprimer d'une part toute la sémantique des informations, et d'autre part les règles de raisonnement ;

- il est nécessaire de disposer d'une ontologie suffisamment complète des domaines sur lesquels portent les informations, ce qui est un problème ouvert. En effet, s'il est envisageable de proposer des ontologies pour des domaines restreints [Mizoguchi 2001], il est extrêmement difficile de déterminer des ontologies sur des domaines larges.

4.2.1.4 Priorités entre informations

Nous avons vu qu'il peut être pertinent de faire interagir des informations entre elles, de façon à faire émerger de nouvelles informations. Un autre façon d'analyser les interactions dynamiques entre informations consiste à introduire des priorités au sein de l'ensemble des informations à présenter.

Par exemple, dressons une liste de quelques informations qui peuvent être diffusées à l'intérieur d'un aéroport :

- annonce d'un embarquement qui se termine dans une heure ;
- dernier appel pour un embarquement qui se termine dans cinq minutes ;
- menu d'un restaurant ;
- annonce d'une disparition d'enfant dans l'enceinte de l'aérogare.

Clairement, toutes ces informations n'ont pas la même *priorité*. Ainsi, lors de la présentation de ces informations, il paraît souhaitable que l'accent soit mis sur la disparition d'enfant et l'embarquement dont la clôture est proche, par rapport aux deux autres informations.

Le processus de présentation pourra utiliser ces niveaux de priorité. Par exemple, les informations les plus prioritaires pourront être mises en exergue, c'est-à-dire affichées en premier, en plus gros, d'une couleur différente, etc. De plus, s'il n'est pas possible de présenter simultanément *toutes* les informations dans une situation donnée, le système pourra ainsi privilégier les plus prioritaires.

4.2.2 Différences entre utilisateurs

4.2.2.1 Identification des différences

Notre ambition est d'offrir des services au plus grand nombre possible d'utilisateurs. Cela signifie qu'il faut tenir compte de leurs caractéristiques propres, et leur offrir des modes de présentation adaptés en conséquence. Il est donc nécessaire de disposer d'une *modélisation* de l'utilisateur, même partielle [Bunt 2005]. Marucci [Marucci 2002] dénombre quatre aspects de la modélisation d'un utilisateur : il s'agit de représenter son niveau de connaissance, ses préférences, ses buts et sa position géographique. Dey, quant à lui, note que la pertinence des informations fournies à un utilisateur par un système sensible au contexte dépend de la *tâche* qu'il est en train d'accomplir [Dey 2001].

Nous considérons que la définition de Marucci est trop restrictive : elle ne fait pas mention des *capacités* de l'utilisateur. En effet, il est possible qu'un système soit utilisé par des non-voyants, des malentendants, ou des personnes à mobilité réduite. Il ne pourra pas avoir avec chacune de ces catégories d'utilisateurs les mêmes comportements qu'avec des personnes non handicapées. On pourrait considérer que les handicaps éventuels soient pris en compte dans les *préférences* de la classification de Marucci, mais dans ce cas, ce terme semble pour le moins mal choisi.

Nous préférons donc introduire une notion de *profil* des utilisateurs : un profil correspond aussi bien aux capacités de l'utilisateur (par exemple, il peut être capable d'entendre, mais à condition de respecter un certain volume sonore minimum) qu'à ses préférences (par exemple, un utilisateur sans problèmes de vue particuliers peut néanmoins préférer le mode auditif au mode visuel). Il est alors nécessaire de définir les domaines qui seront décrits par un profil d'utilisateur. On peut penser aux caractéristiques suivantes :

- capacités et préférences sensorielles : vue, ouïe, etc. (cas des handicaps sensoriels) ;
- capacités motrices (cas des personnes à mobilité réduite) ;
- capacités mentales et cognitives : cette catégorie de différence peut aussi bien regrouper les handicaps mentaux traditionnellement identifiés comme tels (difficulté à comprendre une information par exemple), que les variations de compétences dans divers domaines, notamment le domaine linguistique. Ainsi, une personne qui se trouve dans un pays dont elle ne comprend pas du tout la langue se trouve dans une situation de *handicap* bien réel ;
- niveau dynamique de stress des utilisateurs, peut-être lié à la tâche qu'ils sont en train d'accomplir. Par exemple, dans un aéroport certains utilisateurs sont pressés (ceux qui doivent se rendre *immédiatement* à leur comptoir d'embarquement), et d'autres sont moins pressés ;
- éventuellement, des renseignements sur ses maladies chroniques, ses allergies, etc.

Comme notre but est de concevoir un système qui présente des informations sur des dispositifs situés à proximité des utilisateurs, les handicaps moteurs n'ont pas besoin d'être traités de façon explicite : si un utilisateur se trouve devant un écran, peu importe comment il y est arrivé, que ce soit sur ses jambes ou en fauteuil roulant. Tout ce qui compte alors, c'est de lui fournir les informations dont il a besoin. De même, nous considérons que la tâche est étroitement liée aux lieux visités par les utilisateur : si on se trouve dans un aéroport, on sera intéressé par les informations relatives aux avions ; si on se trouve dans un restaurant, on sera intéressé par le menu, etc.

Dans ce travail, nous ne traitons délibérément pas des handicaps mentaux proprement dits. Par contre, les différences de compétences, notamment linguistiques, nous semblent intéressantes, en particulier vis-à-vis de notre exemple de l'aéroport. En effet, un aéroport regroupe

dans un même lieu des personnes de langue et de culture différentes. Nous jugeons donc important de traiter explicitement ce type de différences.

Le niveau de stress est un facteur dynamique important, qui serait intéressant à prendre en compte. Cependant, la mesure d'un tel facteur est extrêmement délicate. Nous n'en tiendrons pas compte, mais cela constituerait une perspective intéressante. Par exemple, les informations pourraient être présentées avec moins de détails aux personnes en situation de stress. En effet, lorsqu'on est stressé, on risque de paniquer s'il faut lire ou écouter trop de détails : on préfère généralement une information qui aille droit au but, quitte à ce qu'elle soit moins précise.

Au final, nous pensons que les paramètres suivants sont pertinents pour notre application :

- le profil sensoriel de l'utilisateur : capacités et préférences ;
- sa position géographique, à la fois par rapport aux dispositifs d'interaction, et par rapport aux zones où il se trouve, et où peuvent être diffusées des informations différentes ;
- ses compétences linguistiques, qui peuvent varier selon les modalités (par exemple, un Français peut être capable de lire correctement l'Anglais, mais avoir du mal à comprendre l'Anglais oral) ;
- les connaissances qu'il a déjà acquises, son historique.

4.2.2.2 Multimodalité

En introduction, nous avons vu que les informations doivent pouvoir être présentées par divers dispositifs. Chacun de ces dispositifs peut utiliser des modalités en sortie : mode visuel pour un écran, mode auditif pour un haut-parleur, mode TPK pour une tablette Braille.

Nous voulons intégrer l'aspect multimodal au cœur de notre système. Plutôt que d'utiliser *au hasard* tel ou tel dispositif de présentation, nous voulons que le dispositif soit choisi en fonction des modalités qu'il peut utiliser en sortie, et des modalités dont disposent les utilisateurs en entrée (en fonction de leurs éventuels handicaps, de leurs compétences linguistiques, etc.)

Supposons par exemple que dans le hall d'un aéroport soient installés des moniteurs et des haut-parleurs. Si une personne sourde pénètre dans le hall, le système devrait lui fournir les informations concernant son vol à travers un moniteur situé à proximité. Si ensuite une personne aveugle arrive dans le hall, le haut-parleur devrait lire à voix haute les informations concernant le nouvel utilisateur.

Que se passerait-il en théorie si une personne sourde et aveugle arrive ? À première vue, on peut imaginer que dans ce cas précis, le système ne puisse rien faire. Cependant, si ce nouvel utilisateur dispose de sa propre tablette Braille (qu'il transporte avec lui), le système devrait être capable de l'utiliser pour transmettre les informations à l'utilisateur. De cette façon, l'ensemble des modalités pouvant être utilisées n'est pas limité par l'ensemble des dispositifs

de présentation installés en un lieu donné, car les utilisateurs peuvent amener avec eux leurs propres dispositifs (privés) capable de fonctionner selon des modalités supplémentaires.

Nous spécifions donc un système *multimodal*, car la présentation d'informations peut faire appel à des modalités très diverses. Cependant, dans ce travail et afin de ne pas mélanger les problèmes, nous avons choisi de nous restreindre au cas de la *multimodalité exclusive* (voir section 2.2.3.2), c'est-à-dire qu'une information donnée ne sera présentée qu'à travers *une seule* modalité à la fois.

4.2.2.3 Priorités entre utilisateurs

Nous avons vu plus haut qu'il est possible de distinguer un certain nombre de différences entre utilisateurs. Selon chacun de ces axes, il est envisageable d'établir une notion de *priorité* entre utilisateurs. De la sorte, lors de la présentation des informations, les utilisateurs les plus prioritaires pourraient être « avantagés » par rapport aux autres. Par exemple, leurs informations pourraient leur être fournies plus rapidement, ou bien être mises en évidence sur les dispositifs de présentation : affichage en haut d'un écran, lecture en premier d'un message audio, etc. De plus, dans le cas « extrême » où il n'est pas possible de fournir à chaque utilisateur son information (en raison par exemple d'une saturation de tous les dispositifs de présentation), les besoins des utilisateurs les plus prioritaires pourraient être satisfaits en premier.

Dans ces conditions, il est même possible de distinguer un axe de différenciation supplémentaire, entre usagers « standards », et usagers ayant souscrit une formule d'abonnement privilégié. Ces derniers pourraient être avantagés en termes de visibilité, de précision et de niveau de détail des informations fournies.

Afin de voir la façon dont peuvent être définies des priorités entre utilisateurs, reprenons certains des axes de distinction entre utilisateurs évoqués ci-dessus :

- les utilisateurs pressés (c'est-à-dire, ceux dont l'heure d'embarquement est proche) sont raisonnablement plus prioritaires que les utilisateurs moins pressés ;

- il paraît normal que les utilisateurs handicapés soient plus prioritaires que les non handicapés, car ces derniers sont bien plus à même d'aller chercher les informations dont ils ont besoin si elles ne leur sont pas fournies d'emblée, ce qui est relativement difficile pour des handicapés, qu'ils soient handicapés sensoriels ou moteurs ;

- le fait qu'une classe d'utilisateurs soient jugés prioritaires par rapport aux autres ne résulte pas d'une organisation sociale de base, mais cela peut avoir des justifications *commerciales* pour les opérateurs de certains lieux publics. En effet, un service de base serait fourni à tous les utilisateurs, mais ceux qui souscriraient un abonnement spécifique verraient leurs conditions d'utilisation améliorées.

4.2.3 Aspects relatifs à la vie privée

Nous nous proposons de concevoir un système capable d'utiliser, éventuellement, des dispositifs publics de présentation (par exemple des écrans publics ou des haut-parleurs) afin de fournir des informations personnalisées aux utilisateurs. On peut alors légitimement se poser la question du respect de la vie privée de ces derniers.

Cependant, il ne s'agit pas de fournir des informations *privées* aux utilisateurs via des dispositifs publics, mais plutôt d'*opérer une sélection* parmi des informations publiques, de façon à améliorer les interactions en diminuant la charge cognitive[2] des utilisateurs lorsqu'ils recherchent des informations. De cette façon, on ne révèle *a priori* rien de leur vie privée.

Néanmoins, il est possible d'imaginer un type de situation dans laquelle des informations privées seraient diffusées. Nous avons vu que notre système se propose d'effectuer une sélection parmi des informations. En conséquence, si un seul utilisateur est présent en un endroit donné, un individu dissimulé à proximité peut inférer des informations privées à partir de la présentation qui sera faite par le système, *car il sait qu'elles ne concerneront que l'utilisateur en question.* Notons que ce problème disparaît à partir de deux ou trois utilisateurs, pour peu que ceux-ci soient concernés par des informations différentes : il n'est alors plus possible d'attribuer les informations présentées à telle ou telle personne.

Par exemple, supposons qu'un moniteur affiche les destinations des voyageurs situés à proximité. Si plusieurs personnes sont présentes, le moniteur affichera quelques destinations, et il n'est pas possible d'en inférer quoi que ce soit sur quiconque. Mais si une seule personne est présente, alors *sa destination sera seule affichée*, ce qui peut poser problème.

Pour remédier à cet inconvénient, il est possible d'introduire un « brouillage » des informations. Dans le cas évoqué ci-dessus, on peut par exemple décider qu'en-deçà de deux ou trois informations pertinentes à présenter, une ou deux informations supplémentaires, aléatoires et non pertinentes, seront présentées en plus. De cette façon, on évite la divulgation des informations relatives à un passager qui se présente seul devant l'écran.

4.3 Conclusion : problèmes retenus dans ce travail

Dans ce travail, nous ne traitons pas de tous les problèmes évoqués ci-dessus, mais uniquement d'un certain nombre d'entre eux :

- la mobilité des utilisateurs, et donc l'utilisation *opportuniste* de dispositifs de présentation au cours de leurs déplacements ;
- les différentes capacités et préférences sensorielles des utilisateurs, et leur impact sur les aspects multimodaux du système ;

[2] La *charge cognitive* d'un utilisateur est le niveau d'effort intellectuel requis pour effectuer une tâche donnée.

- l'agrégation d'écrans et leur fonctionnement sans configuration manuelle.

Le chapitre 5 définit un modèle pour résoudre ces problèmes, et le chapitre 6 détaille les algorithmes correspondants.

Chapitre 5

Le modèle KUP

Dans le chapitre 4, nous avons dressé une liste de fonctionnalités souhaitées pour un système dynamique de présentation d'informations. Dans le présent chapitre, nous décrivons un modèle qui permet de réaliser une partie des objectifs précédemment évoqués.

5.1 Information (Knowledge), Utilisateur et dispositif de Présentation

Dans la section 2.1 nous avons détaillé les modèles d'architecture logicielle pour les interfaces homme-machine les plus répandus, avant de conclure que leur principe général était le même (figure 2.5) : un noyau fonctionnel dialogue avec une interface. L'utilisateur humain n'est en général pas représenté explicitement dans le système ; il se contente d'interagir avec l'interface.

Pour identifier les autres formes d'organisation possibles, nous proposons de caractériser les relations que peuvent entretenir le noyau fonctionnel, l'interface et les utilisateurs.

Précisons que nous nous intéressons aux interactions *en sortie*, c'est-à-dire que l'interface fournit des informations aux utilisateurs, mais ceux-ci n'ont pas la possibilité d'effectuer des actions en entrée. Cependant, il serait intéressant de compléter le modèle exposé ici pour le rendre symétrique.

Nous définissons un certain nombre d'entités, à la fois physiques et logicielles. Les entités physiques seront suffixées par la lettre φ, tandis que les entités logicielles seront suffixées par la lettre ℓ. Nous noterons donc :

- K_ℓ l'entité logicielle correspondant aux sources d'information et de connaissance (d'où le K pour *knowledge*). Ces sources d'information sont chargées de communiquer des informations aux utilisateurs ;
- P_φ l'entité physique correspondant à l'interface physique (en sortie), par exemple un écran, et P_ℓ l'entité logicielle responsable de la *présentation* des informations ;

- U_φ l'entité physique correspondant à l'utilisateur, et U_ℓ sa représentation logicielle.

Dans le système informatique, chaque couple d'entités *logicielles* peut être mis en communication ou non (voir tableau 5.1).

$K_\ell \leftrightarrow P_\ell$	$P_\ell \leftrightarrow U_\ell$	$U_\ell \leftrightarrow K_\ell$	Nom de la situation
			Impossible, toutes les entités sont isolées
•			Linéaire, $K_\ell \leftrightarrow P_\ell$
	•		*Impossible, K_ℓ isolé*
•	•		Linéaire, $K \leftrightarrow P \leftrightarrow U$
		•	*Impossible, P_ℓ isolé*
•		•	Linéaire, $U_\ell \leftrightarrow K_\ell \leftrightarrow P_\ell$
	•	•	Linéaire, $K_\ell \leftrightarrow U_\ell \leftrightarrow P_\ell$
•	•	•	Triangulaire

Tableau 5.1 : *Situations de communication possibles entre entités.*

D'emblée, il est possible d'éliminer trois situations (marquées *« impossibles »* dans le tableau 5.1) : dans ces situations, le noyau fonctionnel et/ou l'entité logicielle de présentation sont complètement isolés du reste du système, ce qui n'est pas concevable. Cinq situations effectives sont donc envisageables à ce stade. Examinons-les successivement.

Situation linéaire $K_\ell \leftrightarrow P_\ell$. Elle correspond aux modèles classiques évoqués plus haut et détaillés dans la section 2.1. L'utilisateur n'est pas représenté logiciellement ; le noyau fonctionnel communique avec le module de présentation.

Situation linéaire $K_\ell \leftrightarrow P_\ell \leftrightarrow U_\ell$. Il s'agit d'un prolongement du cas précédent, qui correspond également aux modèles d'architecture classiques, mais dans lequel l'utilisateur est modélisé, par exemple par un profil.

Situation linéaire $U_\ell \leftrightarrow K_\ell \leftrightarrow P_\ell$. Elle ne semble pas très réaliste, car l'entité utilisateur est directement reliée au noyau fonctionnel, mais pas du tout à l'interface. On peut alors se demander quel est l'intérêt de ce module d'interface... En conséquence, nous n'examinerons pas plus en détails cette situation.

Situation linéaire $K_\ell \leftrightarrow U_\ell \leftrightarrow P_\ell$. Ici, la représentation logicielle de l'utilisateur communique directement avec le noyau fonctionnel, et également avec l'entité d'interface. Contrairement aux deux premières situations, où l'utilisateur soit n'est pas représenté, soit est modélisé par une entité qui se contente d'interagir avec l'interface, l'entité logicielle associée à l'utilisateur se trouve ici au centre du système.

Situation triangulaire. Elle peut être vue comme une extension de chacune des trois situations ci-dessus (ou plus exactement, des deux situations jugées viables parmi celles-ci), où l'on rajoute une communication entre les deux entités qui ne communiquaient pas. Des échanges peuvent avoir lieu sans restriction entre chacune des trois entités logicielles.

Dans le cadre de la conception de notre système mobile et opportuniste, il nous semble judicieux de séparer deux types d'actions, deux phases différentes de fonctionnement :

- d'une part, la *fourniture* d'une information par le noyau fonctionnel (K_ℓ) à destination du modèle informatique de l'utilisateur (U_ℓ) ;

- d'autre part, la *présentation* de cette information (par un dispositif de présentation physique P_φ) à l'utilisateur (U_φ). Bien entendu, cette présentation effectuée dans le monde physique doit être au préalable négociée dans le système (i.e. entre U_ℓ et P_ℓ).

De cette façon, le système peut « récolter » de façon opportuniste des informations au fur et à mesure de leur découverte, les mémoriser, *même s'il n'y a pas de dispositif de présentation disponible à ce moment-là*. La présentation peut se faire plus tard, de façon opportuniste elle aussi, lorsque l'utilisateur se trouvera à proximité d'un dispositif de présentation approprié. La décorrélation entre les deux phases est la condition du fonctionnement doublement opportuniste du système.

Comment obtenir cette décorrélation ? Il ne faut pas que le noyau fonctionnel (K_ℓ) soit directement relié à l'entité de présentation (P_ℓ). Au contraire, il est nécessaire qu'il existe un intermédiaire entre les deux. Si ce n'est pas le cas, la fourniture et la présentation de l'information seront forcément liées.

Parmi les situations examinées ci-dessus, la situation linéaire $K_\ell \leftrightarrow U_\ell \leftrightarrow P_\ell$ correspond à ce cas de figure. L'intermédiaire entre le noyau fonctionnel et le dispositif de présentation est l'entité utilisateur. Notons que la situation triangulaire propose le même schéma, à ceci près qu'il est *possible* que le noyau fonctionnel communique directement avec le dispositif de présentation, sans pour autant que ce soit forcément toujours le cas.

Dans notre modèle, nous nous basons donc principalement sur la situation linéaire $K_\ell \leftrightarrow U_\ell \leftrightarrow P_\ell$, et pour cette raison nous appelons notre modèle KUP, de façon à indiquer clairement que l'entité utilisateur se trouve au centre, entre le noyau fonctionnel et le dispositif de présentation. Néanmoins, dans certaines situations qui seront indiquées plus loin, nous nous réservons la possibilité d'une communication directe entre noyau fonctionnel et dispositifs de présentation, comme la situation triangulaire le prévoit. Cependant, l'immense majorité des cas envisagés dans notre travail se placeront dans le cadre de la situation linéaire $K_\ell \leftrightarrow U_\ell \leftrightarrow P_\ell$.

En conséquence, le modèle KUP introduit un changement radical de paradigme par rapport aux modèles d'architecture logicielle classiques en IHM. Alors que dans ces derniers seuls sont représentés explicitement le noyau fonctionnel et les dispositifs de présentation, KUP place l'utilisateur au centre du système. L'utilisateur doit donc être représenté explicitement sous forme d'une entité logicielle. Par sa position, cette dernière sera appelée à jouer un rôle central dans l'architecture.

Récapitulatif Nous venons de définir deux entités physiques, l'utilisateur U_φ et le dispositif de présentation P_φ. À ces entités physiques nous associons des entités logicielles, respectivement U_ℓ et P_ℓ.

Nous avons également introduit une entité logicielle correspondant au noyau fonctionnel ou aux sources d'information : K_ℓ. En réalité, nous associons également une entité physique à chaque source d'information, que nous notons K_φ. Ce n'est pas une entité physique *tangible* comme U_φ et P_φ, mais nous verrons qu'elle correspond aux « zones » (physiques) pour lesquelles l'information fournie par les sources d'information est valide. Les entités physiques K_φ seront traités en détails dans la section 5.2.5.

La figure 5.1 récapitule les relations qui existent entre les entités physiques et les entités logiques du modèle KUP.

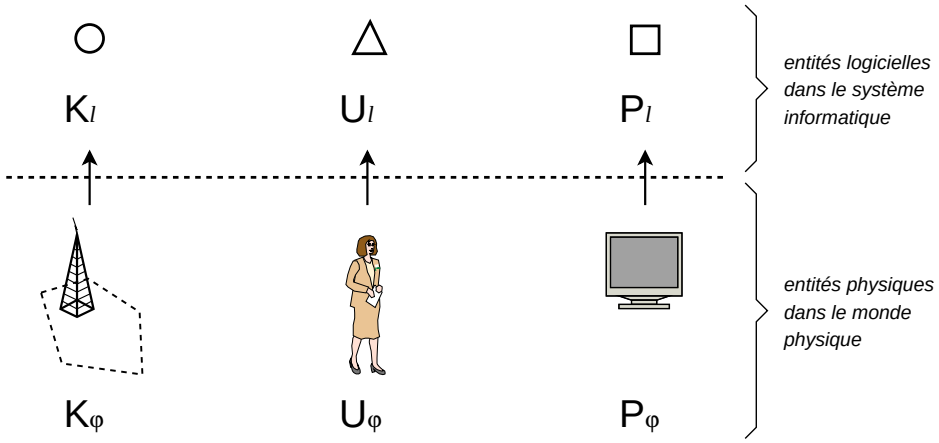


Figure 5.1 : *Relations entre entités physiques et entités logiques.*

Dans la suite, lorsqu'il n'y a pas ambigüité, nous notons simplement K, U, P au lieu de K_φ, U_φ, P_φ ou K_ℓ, U_ℓ, P_ℓ.

5.2 Proximité

5.2.1 Espace, positions

Les entités physiques tangibles (dispositifs de présentation, utilisateurs) sont *situées*, i.e. *localisées* dans un espace. Ce dernier est noté $\mathcal{E}$; il s'agit d'un ensemble de points. Cette définition se veut suffisamment générale, donc elle ne donne pas de contraintes supplémentaires. $\mathcal{E}$ pourrait par exemple être un ensemble discret de points, $\mathbb{R}^2$ si les entités peuvent se déplacer sur un plan avec des coordonnées cartésiennes continues, ou bien $\mathbb{R}^3$ pour des déplacements en trois dimensions, etc.

Il est par ailleurs possible que certaines dimensions de l'espace $\mathcal{E}$ correspondent à l'*orientation* des entités. Par exemple, si on pose $\mathcal{E} = \mathbb{R}^2 \times [0, 360[$ la position d'une entité est donnée par trois coordonnées :

- deux coordonnées $\langle x, y \rangle$ de positionnement dans un plan (coordonnées cartésiennes par exemple) ;
- une coordonnée d'orientation θ (un azimut), par exemple donnée en degrés par rapport à la direction du nord.

La position d'une entité donnée est un élément x de $\mathcal{E}$; on note $x \in \mathcal{E}$.

Enfin, on définit également les ensembles suivants :

- ensemble des sources d'information : $\mathcal{K}$;
- ensemble des utilisateurs : $\mathcal{U}$;
- ensemble des dispositifs de présentation : $\mathcal{P}$.

5.2.2 Espace perceptuel

De façon informelle, nous souhaitons définir l'*espace perceptuel* d'une entité physique tangible $e \in \mathcal{U} \cup \mathcal{P}$ comme étant l'ensemble des points de l'espace tels que si une autre entité y est située, elle peut être *perçue* par e. Par exemple, pour une entité utilisateur, l'espace perceptuel pourrait correspondre à son champ de vision (à condition que cet utilisateur soit voyant).

Cependant, cette définition est trop restrictive. En effet :

1. un utilisateur dispose généralement de plusieurs sens[1] en état de fonctionnement, et avec des perceptions différentes. Par exemple, le champ de vision d'un être humain ne correspond pas à sa zone de perception auditive. Ainsi, un écran situé à 2 m derrière un utilisateur ne sera pas perçu par ce dernier, tandis qu'un dispositif de synthèse vocale situé au même endroit le sera ;
2. la perception dépend des *attributs des modalités*. Par exemple, une sonnerie de téléphone mobile émise à cinquante mètres ne sera pas perçue par un être humain, tandis que le son d'une sirène le sera sans problème. Ainsi, pour un même mode (sonore), et même une même modalité (sonnerie), l'espace perceptuel semble dépendre de la façon dont est instanciée la modalité.

En conséquence, la définition informelle de la notion d'espace perceptuel introduite ci-dessus est trop limitée. Elle doit être complétée pour tenir compte des modalités et de leurs instanciations.

Pour ce faire, nous introduisons une notion supplémentaire : la notion d'*espace multimodalisé*, ou en abrégé de m-espace. Un m-espace est égal au produit cartésien de l'espace $\mathcal{E}$ par l'ensemble des instanciations des modalités utilisables.

Par exemple, supposons que les modalités utilisables par une entité soient :

[1] « *Sens* » comme dans les « *cinq sens* », voir page 11.

- *« texte »*, avec un attribut *« taille »* variant continûment entre 10 et 60 points et un attribut *« couleur »* pouvant prendre les trois valeurs discrètes *rouge*, *vert* et *bleu* ;
- et *« sonnerie »*, avec un attribut *« volume »* variant continûment entre 0 et 100.

Dans ce cas, le m-espace serait :

$$\begin{aligned}\mathcal{M} \;=\; & \mathcal{E} \times \\ & (\{\texttt{texte}\,\langle \texttt{taille}, \texttt{couleur}\rangle\,,\ (\texttt{taille},\texttt{couleur}) \in [10{,}60] \times \{\texttt{r},\texttt{v},\texttt{b}\}\} \cup \\ & \{\texttt{sonnerie}\,\langle \texttt{volume}\rangle\,,\ \texttt{volume} \in [0{,}100]\})\end{aligned}$$

Voici quelques exemples d'éléments de ce m-espace :

- le point de coordonnées 46°23”32’ N, 1°02”56’E, avec un texte de taille 23 et de couleur verte ;
- le point de coordonnées 45°07”19’ N, 2°01”32’E, avec un texte de taille 59 et de couleur bleue ;
- le point de coordonnées 46°23”32’ N, 1°02”56’E, avec une sonnerie de volume 61.

Avec cette définition, on rend bien compte des caractéristiques de la perception des utilisateurs humains. Cependant, quel sens peut-on donner à la « perception » d'un dispositif de présentation ? Dans ce cas, il ne s'agit pas bien entendu d'une perception réalisée à l'aide de capteurs biologiques. Par contre, un dispositif peut disposer de capteurs multimodaux en entrée, par exemple un microphone, une caméra vidéo, un clavier, etc. Dans ce cas, la perception est tout simplement définie par rapport aux caractéristiques techniques de ces capteurs.

Cependant, on peut imaginer d'autres capteurs, qui ne rentrent pas dans le cadre classique de la multimodalité. Par exemple, on peut supposer qu'un dispositif de présentation dispose d'un appareillage électronique lui permettant de détecter les utilisateurs situés à proximité, tel qu'un détecteur d'étiquettes RFID. On peut dire dans ce cas que la méthode de détection représente une « modalité » supplémentaire. Ce n'est pas une modalité au sens classique du terme, car les modalités ont été définies par rapport aux sens des êtres humains (voir section 2.2.1 page 11). Cependant, pour un dispositif électronique, ces moyens de détection jouent exactement le même rôle que les modalités pour les êtres humains. Nous les classifierons donc en tant que *modalités*. Par exemple, on peut envisager les modalités « détection par infrarouges », « détection par RFID », etc. Dans le cas de la détection RFID, la modalité posséderait un attribut « puissance nécessaire pour effectuer la détection ».

Notons toutefois que si la perception des utilisateurs humains est définie par rapport à leurs organes biologiques, il est impossible pour un système informatique d'accéder de façon exacte à cette perception. En pratique, un système informatique devra *effectuer une mesure* à l'aide

d'un appareil électronique afin d'évaluer la perception d'un être humain. La mesure ainsi effectuée sera *assimilée* à la perception réelle de l'utilisateur, mais il est bon de se souvenir qu'il ne s'agit que d'une *approximation.* Cette approximation n'est valable que dans la mesure où le capteur électronique reproduit fidèlement les caractéristiques de la perception de chaque utilisateur humain.

Par exemple, une détection de balises infrarouges permet de modéliser relativement fidèlement la perception visuelle, car :

- les infrarouges ont une portée limitée, ce qui correspond à la distance maximale à laquelle il est possible de distinguer un texte ou un objet ;
- la détection des infrarouges tient compte de l'orientation de l'utilisateur, ce qui rend compte du caractère directif de la perception visuelle (on ne peut percevoir que les objets situés devant soi, pas les objets situés à l'arrière).

Globalement, cette méthode représente donc une mesure plutôt bonne de la perception visuelle. Cependant, il est clair qu'elle ne sera pas fidèle de façon absolue : par exemple, il est impossible que la distance maximale et l'angle maximal de perception correspondent *exactement* aux caractéristiques visuelles de l'utilisateur humain.

Il est maintenant possible de définir formellement l'*espace perceptuel* d'une entité physique. Il s'agit d'un sous-ensemble d'un m-espace $\mathcal{M}$, qui correspond à l'ensemble des « points » perceptibles par l'entité, la perception étant définie comme indiqué ci-dessus. Si une entité se déplace, son espace perceptuel sera modifié : dans la plupart des cas, il va « suivre » son entité de façon naturelle. L'espace perceptuel recouvre certes une proximité spatiale, mais comme nous l'avons vu plus haut, cette notion est conditionnée par les modalités (voir fig. 5.2). On note l'espace perceptuel d'une entité e située en $x \in \mathcal{E}$ de la façon suivante :

$$\mathcal{EP}(e, x) \subset \mathcal{M}$$

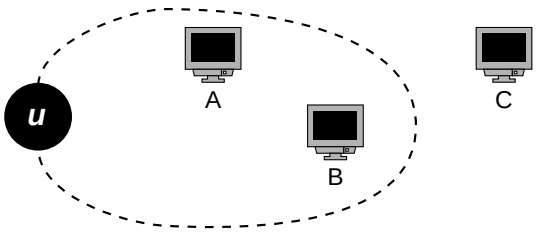


Figure 5.2 : *Si on effectue une projection de l'espace perceptuel d'un utilisateur u, selon la modalité visuelle et à attributs constants (par exemple, à taille de caractères donnée), on obtient un* champ visuel. *Ici, les informations affichées sur les écrans A et B seraient perçues par l'utilisateur, mais pas celles affichées par C. Les limites du champ visuel de u sont indiquées par les pointillés.*

Nous avons vu ci-dessus que la *perception* d'une entité physique peut être décrite dans un espace multimodalisé ou m-espace. Réciproquement, une telle entité peut être vue comme une *source* de contenu multimodal[2] localisée dans l'espace $\mathcal{E}$. Cela revient à dire qu'une entité est localisée dans un m-espace $\mathcal{M}$. Comme une entité peut éventuellement diffuser du contenu selon plusieurs modalités, sa localisation n'est pas forcément réduite à un seul point du m-espace ; au contraire, il s'agit d'un *sous-ensemble* du m-espace. Formellement, la localisation d'une entité e est notée :

$$\ell\,(e) \subset \mathcal{M}$$

La localisation d'une entité e est donc un ensemble de couples (x_i, m_i), où x_i est une position géographique (un élément de $\mathcal{E}$), et m_i une modalité instanciée. Doit-on imposer des restrictions particulières sur ces couples ? Si l'on décide qu'une entité doit être positionnée de façon ponctuelle dans l'espace $\mathcal{E}$, alors tous les x_i doivent être égaux. Plus formellement, il existe alors un élément x de $\mathcal{E}$ tel que pour tout i, $x_i = x$. Cependant, si l'on conçoit que les entités ont une certaine *étendue* dans l'espace, alors une telle restriction n'est pas nécessaire. On peut alors se contenter d'imposer que l'ensemble $\{x_i\}$ soit borné, pour éviter que l'étendue puisse être infinie.

Il est alors possible de définir l'*ensemble perceptuel* d'une entité e : c'est l'ensemble des entités qui se trouvent à l'intérieur de son espace perceptuel (au sens de la relation de localisation ℓ). Autrement dit, il s'agit de l'ensemble des entités que e est capable de percevoir. L'ensemble perceptuel de e varie en permanence, au fur et à mesure que des entités entrent et sortent de son espace perceptuel. L'inclusion dans l'ensemble perceptuel dépend de la proximité, ainsi que des caractéristiques multimodales au sens large des deux entités. Ainsi, un écran situé à 1 m devant un utilisateur *aveugle* ne sera pas dans son ensemble perceptuel, au contraire d'un haut-parleur situé à la même distance. De même, un utilisateur ne se trouvera dans l'ensemble perceptuel d'un écran capable de détecter la proximité par RFID qu'à la condition qu'il porte lui-même un badge RFID.

Formellement, l'ensemble perceptuel d'une entité e, noté $\mathcal{SP}(e)$, est ainsi défini :

$$\mathcal{SP}(e) = \{d \in \mathcal{U} \cup \mathcal{P} \mid \ell(d) \cap \mathcal{EP}(e, \ell(e)) \neq \emptyset\}$$

5.2.3 Espace de rayonnement

Nous disposons maintenant de toutes les notions nécessaires à la définition de la notion *réciproque* de l'espace perceptuel. L'espace perceptuel caractérise les perceptions d'une entité, autrement dit ses *entrées*, c'est-à-dire son comportement en tant que *récepteur multimodal*. Cependant, comme nous venons de le voir, une entité peut également être un *émetteur multimodal*, c'est-à-dire présenter des caractéristiques multimodales en sortie. De même que les

[2] « *Multimodal* » au sens large. Comme nous l'avons vu plus haut, pour un utilisateur, le contenu multimodal peut consister en rayonnement infrarouge ou RFID.

caractéristiques en entrée sont décrites par l'espace perceptuel, les caractéristiques en sortie sont décrites par ce que nous appelons l'*espace de rayonnement*.

Formellement, on définit l'*espace de rayonnement* d'une entité e vis-à-vis d'une entité d comme étant l'ensemble des points x de l'espace $\mathcal{E}$ desquels d peut percevoir e, c'est-à-dire pour lesquels e appartient à l'espace perceptuel de d situé en x :

$$\mathcal{ER}(e|d) = \{x \in \mathcal{E} \mid \ell(e) \cap \mathcal{EP}(d, x) \neq \emptyset\}$$

La figure 5.3 explique comment est défini l'espace de rayonnement, dans un cas simple où la seule modalité considérée est le texte visuel.

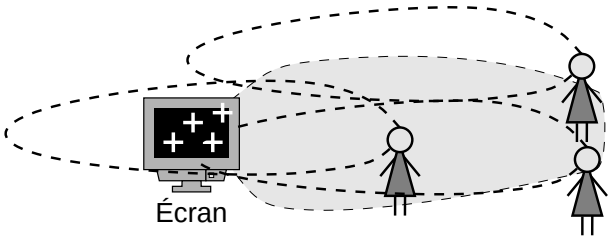


Figure 5.3 : *Construction de l'espace de rayonnement d'un écran vis-à-vis d'un utilisateur. L'utilisateur est fictivement déplacé en chaque point du m-espace. L'espace de rayonnement (ici en gris) est constitué des points en lesquels l'intersection entre l'espace perceptuel de l'utilisateur (ici son champ visuel, en pointillés) et la localisation du dispositif (croix blanches) est non vide.*

Notons bien que dans le cas général, l'espace de rayonnement est forcément défini *par rapport à une entité « réceptrice »*. En effet, les espaces perceptuels étant potentiellement différents pour chaque entité, la perception d'un « émetteur » en un point donné de l'espace sera ou non possible selon l'entité considérée. Il est donc impossible de définir un espace de rayonnement dans l'absolu.

Remarquons qu'il en est de même pour la zone de couverture des satellites (voir fig. 5.4). La zone de couverture d'un satellite est l'ensemble des points de la surface terrestre où la réception des signaux du satellite est possible *avec un type d'antenne donné*. En effet, les antennes présentent des caractéristiques différentes, que ce soit en termes de gain ou de directivité. Il est donc possible de donner autant de zones de couverture qu'il existe d'antennes réceptrices. Ainsi, sur la figure 5.4, plusieurs zones de couverture sont données, en fonction de la taille des antennes paraboliques utilisées.

Dans des cas particuliers cependant, il est tout à fait envisageable que l'espace de rayonnement d'une entité donnée e puisse être le même vis-à-vis de n'importe quelle autre entité. Dans ce cas-ci, on note simplement $\mathcal{ER}(e)$.

Nous avons défini plus haut la notion d'ensemble de perception. De même, nous définissons l'*ensemble de rayonnement* d'une entité e (à un instant t) comme étant l'ensemble des entités

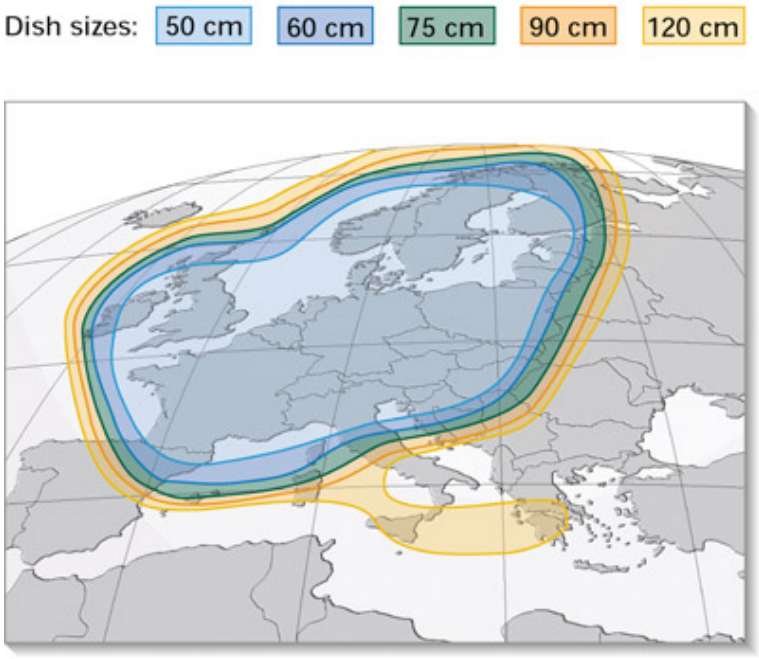


Figure 5.4 : *Plusieurs zones de couverture pour le faisceau nord du satellite Astra 2A, en fonction de la taille des antennes de réception.* Document Astra.

qui se trouvent à l'instant t à l'intérieur (au moins partiellement) de l'espace de rayonnement de e par rapport à elles. Plus formellement :

$$\mathcal{SR}(e) = \{d \in \mathcal{U} \cup \mathcal{P} \mid \ell(e) \cap \mathcal{EP}(d, \ell(d)) \neq \emptyset\}$$

5.2.4 Récapitulatif sur les espaces perceptuels et de rayonnement

La figure 5.5 récapitule de façon synthétique les définitions des espaces perceptuels et des espaces de rayonnement. L'espace perceptuel d'une entité caractérise ses « entrées », car il détermine la façon dont elle va *percevoir* les autres entités. De même, on peut dire que l'espace de rayonnement caractérise ses « sorties », car il détermine la façon dont les autres entités vont percevoir l'entité en question. L'espace de rayonnement d'une entité dépend de sa *localisation* au sein d'un m-espace, qui décrit aussi bien sa position géographique que les modalités selon lesquelles elle sait communiquer avec les autres entités.

Ces définitions ont introduit une notion de *proximité*. La proximité dépend certes de la position géographique (dans l'espace $\mathcal{E}$, ce qui peut inclure non seulement les coordonnées de l'endroit où se trouve une entité, mais également son orientation), mais également des capacités multimodales des entités (localisation dans un m-espace). Ainsi, même si dans la suite nous utilisons parfois (un peu abusivement) les termes *proche* et *proximité* sans plus de qualificatifs, ils doivent être compris au sens de la proximité sensorielle décrite ci-dessus.

Par exemple, si un utilisateur aveugle se trouve juste devant un écran, on ne considérera pas que ces deux entités sont proches l'une de l'autre. De même, si un utilisateur voyant se

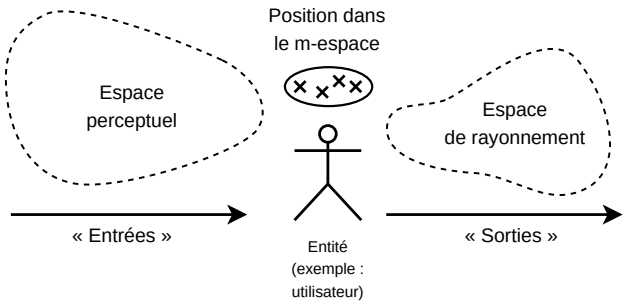


Figure 5.5 : *Récapitulatif : l'espace perceptuel caractérise les « entrées », tandis que l'espace de rayonnement caractérise les « sorties » d'une entité.*

trouve devant un écran, mais lui tourne manifestement le dos, les deux entités ne seront pas non plus considérées comme étant proches.

5.2.5 Cas des sources d'information

Dans ce qui précède, nous avons traité de la localisation et de la perception des entités physiques *tangibles* (U_φ et P_φ). Qu'en est-il des entités physiques *non tangibles* que sont les sources d'information (K_φ) ? Nous proposons tout simplement d'*étendre* toutes les définitions ci-dessus aux sources d'information.

Ainsi, nous considérons que les sources d'information sont localisées, et qu'elles possèdent des espaces perceptuels et de rayonnement (les « zones » évoquées dans la section 5.1). Cependant, nous ajoutons quelques spécificités à ces définitions :

- l'espace de rayonnement d'une source d'information est défini comme étant la zone géographique dans laquelle elle doit diffuser des informations. Autrement dit, son espace de rayonnement vis-à-vis d'un utilisateur u est la partie de l'espace au sein de laquelle les informations qu'elle détient sont réputées pertinentes pour u ;
- l'espace perceptuel d'une source d'information est défini comme étant égal à son espace de rayonnement. En effet, il découle du point précédent qu'une source d'information diffuse des informations à des entités (utilisateurs dans la plupart des cas) qui se situent dans son espace de rayonnement. La *détection* de ces entités doit donc se produire dans exactement le même périmètre que la diffusion d'informations à leur intention ;
- il paraît raisonnable que l'espace de rayonnement d'une source d'information k ainsi définie soit le même vis-à-vis de n'importe quelle entité u. Au lieu de noter son espace de rayonnement $\mathcal{ER}(i|u)$, on note alors $\mathcal{ER}(i)$. On a toujours $\mathcal{ER}(i) \subset \mathcal{E}$.

Notons bien qu'une source d'information (ou de connaissance) représente une *interface* (au sens protocolaire du terme) entre le noyau fonctionnel de l'application et les autres entités du système. *De l'extérieur*, tout se passe *comme si* une source d'information *était* un noyau fonctionnel ou une partie d'un noyau fonctionnel. En réalité toutefois, il est tout à fait possible

que le noyau fonctionnel corresponde à un enchaînement complexe de traitements, éventuellement effectués à l'échelle d'un réseau entier, et dont finalement l'entité source d'information n'en soit qu'un point d'accès.

Pour une source de connaissances, la diffusion des informations ne fait pas appel à la notion de modalité : une source est seulement censée diffuser des informations à l'intérieur d'une *zone géographique*. Or, comme pour toute autre entité, la définition de ses espaces perceptuel et de rayonnement fait appel à la notion de m-espace. Comment peut-on alors définir un m-espace pour une source d'informations, alors qu'il n'existe pas de notion pertinente de modalité ?

Nous proposons tout simplement d'introduire une unique *pseudo-modalité*, notée $\perp$. Le m-espace correspondant $\mathcal{M}$ sera alors défini par :

$$\mathcal{M} = \mathcal{E} \times \{\perp\}$$

On définit de cette façon un véritable m-espace, sans « épaisseur » toutefois selon l'axe des modalités. Il est trivialement en bijection avec l'espace géographique $\mathcal{E}$ [3].

La figure 5.6 récapitule le fonctionnement d'une source d'information, par rapport à notre objectif global de fourniture et de présentation opportuniste d'informations. Où qu'elles se trouvent dans le périmètre de l'espace de rayonnement d'une source d'information, les entités utilisateurs reçoivent des informations. En revanche, les dispositifs de présentation utilisés pour effectuer la présentation des information peuvent varier, en fonction de la position des utilisateurs vis-à-vis desdits dispositifs.

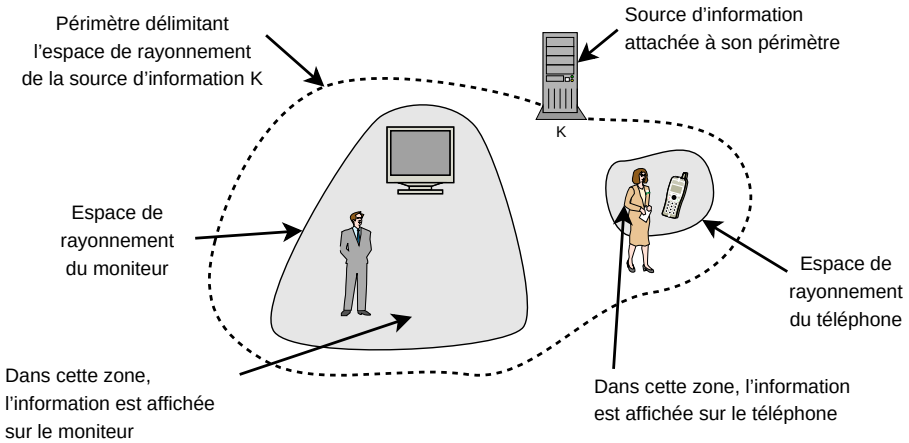


Figure 5.6 : *Les utilisateurs reçoivent des informations dans la totalité de l'espace de rayonnement de la source d'information K, mais les dispositifs de présentation peuvent varier.*

Il semble donc logique que les sources d'information soient liées à l'infrastructure, alors que les dispositifs de présentation peuvent y être liés (par exemple des écrans publics fixes) ou non (par exemple, un téléphone portable ou un PDA porté par l'utilisateur). Dans ces

[3] La bijection correspondante est la fonction $\psi : x \in \mathcal{E} \longmapsto (x, \perp) \in \mathcal{M}$.

conditions, quel est donc le lien entre les espaces de rayonnement des sources d'information et l'architecture sous-jacente des lieux ?

Il paraît logique que les périmètres de diffusion des informations correspondent d'une façon ou d'une autre à l'architecture des bâtiments. Par exemple, les informations liées au menu d'un restaurant devront être diffusées à l'entrée et dans la salle du restaurant. Les informations sur des vols d'avion devront être diffusées dans le hall d'un aéroport. Dans ces cas, on voit que les périmètres de diffusion correspondent à des notions architecturales, par exemple des pièces, des couloirs ou encore des ensembles de pièces. Ainsi, la figure 5.7 donne, sur un plan de Supélec, les position possibles d'espaces de rayonnement de sources d'informations.

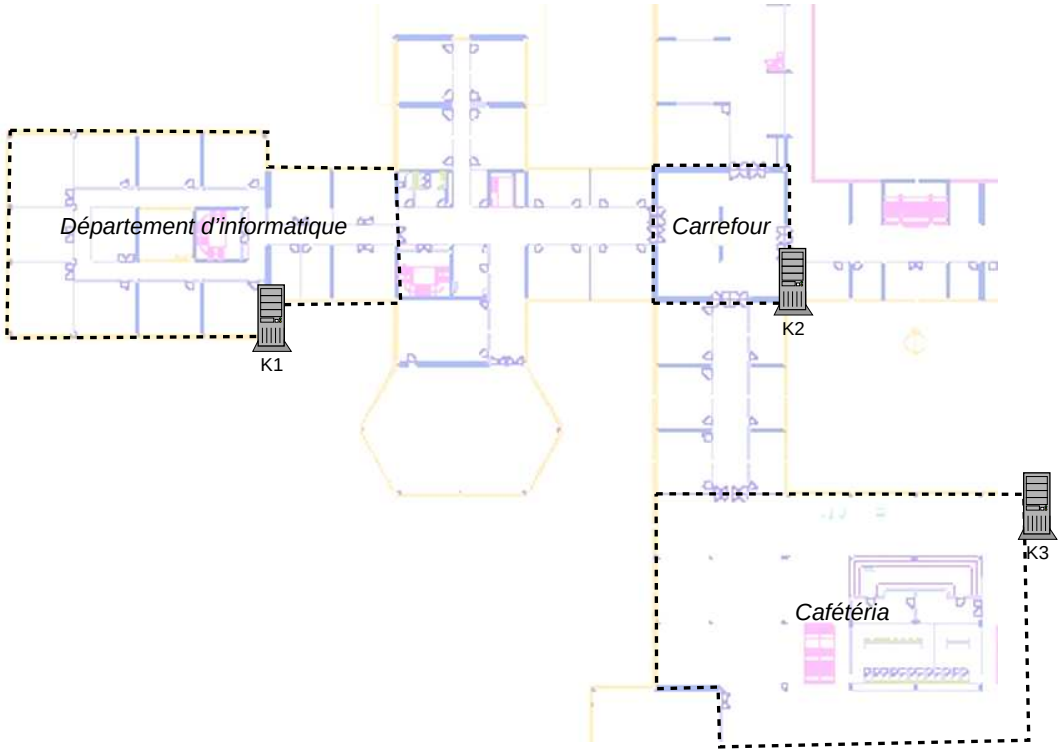


Figure 5.7 : *Il est possible de calquer sur le plan de Supélec la position des espaces de rayonnement de sources d'information.*

Afin de préserver toute la souplesse et la généricité de notre modèle, il n'est pas question de faire figurer explicitement ces notions architecturales à l'intérieur du modèle. Cependant, la personne qui concevra les périmètres de diffusion des informations dans un bâtiment donné, pourra, si elle le désire, directement dériver ces périmètres des informations architecturales. À cet effet, il serait souhaitable que les outils de conception permettent une « importation » de plans d'architecture afin de faciliter le processus de définition des espaces de rayonnement des sources d'information.

5.3 Unités sémantiques

5.3.1 Définition

Les informations émises par les sources d'information sont appelées *unités sémantiques*, ou en abrégé u.s. Une unité sémantique correspond à une information relativement élémentaire, que l'on peut transmettre sur un réseau, et qui peut s'exprimer dans un certain nombre de modalités (une au minimum). Les unités sémantiques proviennent des sources d'information (K) ; elles sont destinées aux utilisateurs (U), et sont présentées par les dispositifs de présentation (P).

Voici quelques exemples d'unités sémantiques :

- le numéro de la porte d'embarquement d'un passager dans un aéroport ;
- les menus d'un restaurant ;
- l'horaire du prochain train allant dans la direction souhaitée.

5.3.2 Contenu concret des u.s.

Comme indiqué en introduction, les u.s. ont pour vocation à être exprimées sur un dispositif de présentation, et selon une modalité donnée. Il est donc nécessaire de leur associer un contenu concret dans la modalité en question. Comme nous l'avons déjà vu (section 2.2.4.2), la génération *automatique* de contenu fait l'objet de recherches spécifiques : nous ne détaillerons donc pas ces processus.

Dans notre conception des unités sémantiques, une fois qu'une modalité concrète a été instanciée, une unité sémantique donnée est (éventuellement) capable de s'exprimer selon cette modalité instanciée. Cela signifie que l'unité sémantique est elle-même capable de produire un contenu concret associé afin de s'exprimer. Ce contenu peut être :

- soit tout simplement choisi dans une liste de contenus spécifiés à l'avance (ex : contenus textuels différents définis pour des longueurs de texte différentes) ;
- soit dérivé d'un contenu spécifié à l'avance, dont certaines parties ou propriétés peuvent être fixées dynamiquement en fonction des attributs de l'instanciation (ex : modification de la taille des caractères d'un texte en fonction de la place disponible à l'écran) ;
- soit généré entièrement automatiquement à l'aide de méthodes spécialisées (ex : génération automatique de texte en langue naturelle).

Que l'une ou l'autre de ces méthodes soit utilisée, peu importe de notre point de vue. Le processus de génération du contenu concret d'une unité sémantique est vu comme une *boîte noire* dont on spécifie l'entrée (modalité concrète instanciée), et dont on utilise la sortie

(contenu concret). Il reste cependant à définir de façon précise ce qu'est en pratique ce *contenu concret*.

Sur le plan pratique, et de façon générique, le contenu concret peut être vu comme un attribut, susceptible de contenir des données binaires brutes. Pour chaque modalité existante, il faut donc normaliser un *format* dans lequel le contenu concret doit être exprimé. À partir d'une spécification des attributs de la modalité, ainsi que d'une description du contenu concret dans ce format, les dispositifs de présentation devront alors être capables de générer les sorties physiques correspondantes. Voici quelques exemples de formats qui pourraient être utilisés :

- pour la modalité « texte visuel » : le texte lui-même, codé sous forme d'une chaîne de caractères UTF-8[4] ;
- pour la modalité « photo » : l'image représentée sous forme d'un fichier JPEG ;
- pour la modalité « synthèse vocale » : une chaîne de caractères UTF-8 contenant les phonèmes à prononcer, codés par des symboles de l'Alphabet Phonétique International ;
- etc.

La figure 5.8 résume le processus de génération de contenu concret par une unité sémantique donnée, selon différentes modalités.

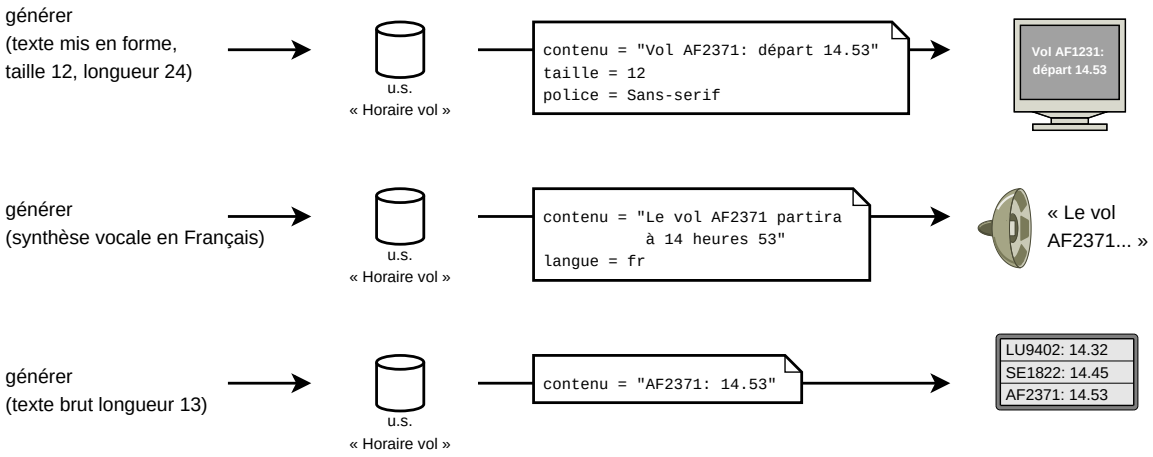


Figure 5.8 : *Génération de contenu concret. Une même u.s. génère des contenus concrets différents pour chaque modalité instanciée. Ces contenus sont sont exprimés dans des* formats *normalisés.*

5.3.3 Métadonnées

Les unités sémantiques portent donc une information qui n'est utile que pour certains types d'utilisateurs. Afin de déterminer les conditions dans lesquelles cette information est réelle-

[4] *8-bit Unicode Transformation Format*, méthode de représentation des caractères Unicode sur un nombre variable d'octets, qui possède l'avantage d'être identique au code ASCII pour les caractères ASCII.

ment pertinente, il peut être utile de munir les unités sémantiques de *métadonnées*. Dans cette section, nous présentons quelques types de métadonnées de base, mais il serait bien entendu possible d'en envisager d'autres.

Sujet et type Le rôle de la source d'information est de communiquer des informations (u.s.) potentiellement intéressantes à un utilisateur. Cependant, ce dernier peut ponctuellement ne pas être intéressé par une catégorie particulière d'unités sémantiques.

Par exemple, on peut imaginer qu'une source d'information donne systématiquement le menu d'un restaurant d'entreprise à tous les membres du personnel qui empruntent un couloir situé à proximité. Supposons alors que Bernard, un employé de l'entreprise, est pressé et n'a pas le temps de manger. Il ne sera donc pas intéressé par les informations concernant le menu. Il faudrait donc que l'entité utilisateur associée à Bernard puisse d'elle-même faire le tri, et éliminer d'emblée les unités sémantiques non pertinentes.

Pour cette raison, il est intéressant d'indiquer pour chaque unité sémantique quel est son *type*, ou encore son *sujet*. De cette façon, l'entité utilisateur pourra alors sélectionner les u.s. réellement utiles, et éventuellement les classer par ordre de priorité. Ainsi, elle pourrait s'occuper en priorité de la présentation des u.s. les plus importantes.

Cependant, pour pouvoir indiquer les types et les sujets des unités sémantiques en vue d'un traitement automatique, il faut disposer d'ontologies correspondantes, ce qui pose toujours problème dans le cas général. En effet, il est impossible de concevoir une ontologie universelle... Cependant, dans le cadre d'applications restreintes à des domaines relativement limités, il est possible de concevoir de petites ontologies adaptées. Ainsi, on peut dresser une typologie des unités sémantiques, grâce à laquelle les entités utilisateurs pourront opérer des sélections.

Date d'expiration Le modèle KUP permet de découpler la fourniture et la présentation d'une unité sémantique. Par conséquent, la présentation d'une u.s. peut se faire à un moment différent de sa récolte. Or on imagine aisément que les informations portées par une u.s. ont potentiellement une validité limitée dans le temps. Il faut donc empêcher la présentation des u.s. qui seraient devenues invalides depuis leur acquisition.

On peut donc munir les u.s. d'une date d'expiration, c'est-à-dire un instant au-delà duquel leur présentation n'est plus utile, ni souhaitée, voire néfaste. Cependant, il doit également exister des u.s. valables sans limitation de durée. De telles u.s. ne seront pas munies de date d'expiration.

Par exemple, les informations sur les menus d'un restaurant d'entreprise ne sont valables que jusqu'à 13h30, l'heure de fermeture. Au-delà, elles n'ont plus aucun sens, car on ne peut plus se faire servir le menu. Par contre, les informations sur la localisation des bureaux dans la même entreprise peuvent être considérées comme permanentes (du moins, à l'échelle de temps d'une visite d'une personne extérieure, c'est-à-dire moins d'une journée).

5.4 Un système opportuniste pour la présentation d'informations

5.4.1 Fonctionnement global

Au cours de la section 5.1, nous avons envisagé deux phases lors d'une interaction : la *fourniture* d'une information, puis sa *présentation*. Cette section décrit (à un haut niveau d'abstraction) comment se comporte le système lors de ces deux phases.

Tout d'abord, nous sommes désormais capables de préciser ce que nous entendons par « information » : concrètement, une information est modélisée par une unité sémantique, comme cela a été décrit dans la section 5.3. Ainsi, ce sont des unités sémantiques qui sont fournies et présentées. Mais à quelles entités le sont-elles ? Nous avons expliqué en section 5.1 qu'un modèle linéaire, dans lequel l'utilisateur U_ℓ se trouve au centre, nous paraît la meilleure solution. Les u.s. sont donc fournies à l'entité logicielle utilisateur (interaction $K_\ell \rightarrow U_\ell$), qui se charge alors de leur présentation (interaction $U_\ell \rightarrow P_\ell$). Ce comportement est résumé sur les diagrammes UML des figures 5.9 et 5.10.

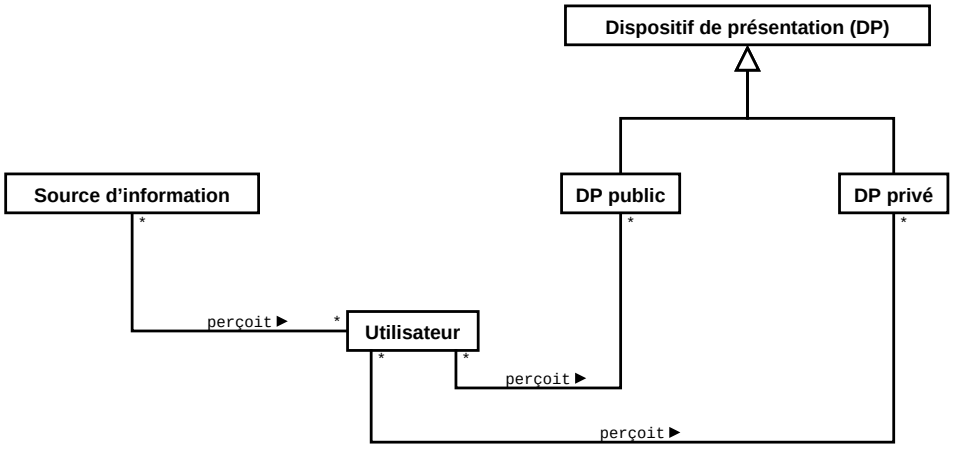


Figure 5.9 : *Les différentes entités du modèle KUP, représentées sous forme d'un* diagramme de classes *UML.*

Fourniture d'une unité sémantique Cette première phase est très simple : lorsqu'un utilisateur pénètre dans l'espace de rayonnement d'une source d'information, celle-ci lui fournit éventuellement une ou plusieurs u.s. pertinentes. Réciproquement, lorsqu'un utilisateur quitte un tel espace de rayonnement, il est tout à fait possible qu'une ou plusieurs des u.s. qu'il a acquises perdent leur pertinence. Dans ce cas, on peut imaginer que lors de la sortie d'un utilisateur de son espace de rayonnement, une source d'information demande à celui-ci de *retirer* une u.s. de sa liste d'u.s. acquises.

Par exemple, imaginons qu'une source d'information soit chargée de la diffusion du menu d'un restaurant. L'espace de rayonnement correspondant englobe les salles du restaurant, la terrasse, ainsi que la portion de trottoir correspondante. Lorsqu'un utilisateur pénètre dans

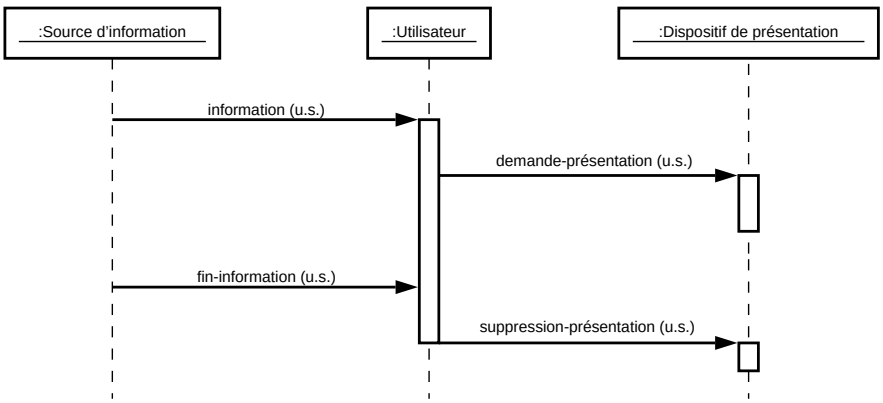


Figure 5.10 : *Les principales interactions entre entités du modèle, sous forme d'un* diagramme de séquence *UML.*

cet espace, il reçoit une u.s. correspondant au menu ; et réciproquement, lorsqu'il quitte cet espace, la source d'information lui indique que cette u.s. n'est plus pertinente.

Le mécanisme de *péremption géographique* que nous introduisons ici s'ajoute au mécanisme de *péremption temporelle* que nous avons introduit en section 5.3.3.

Présentation d'une unité sémantique Les utilisateurs sont mobiles : lorsque l'un d'entre eux reçoit une unité sémantique, il n'y a pas forcément de dispositif de présentation disponible à proximité. Mais lorsqu'un tel dispositif se présente ultérieurement, l'entité utilisateur va essayer d'y faire présenter ses unités sémantiques. À ce moment-là, on peut identifier deux sous-problèmes imbriqués :

1. pour un dispositif de présentation donné, l'utilisateur et ce dispositif doivent se mettre d'accord sur *la* modalité à utiliser pour la présentation de l'u.s. En effet, nous avons indiqué dans le cahier des charges (section 4.2.2.2, page 79) que notre système devait être conçu pour fonctionner en multimodalité *exclusive* ;

2. si plusieurs dispositifs sont disponibles pour la présentation d'une u.s., il faut choisir l'un d'entre eux.

Le chapitre 6 traite successivement ces deux problèmes. En ce qui concerne le mode de fonctionnement global du système, nous voyons qu'il est opportuniste par deux aspects différents :

- par rapport à la transmission des informations : l'utilisateur reçoit des unités sémantiques lorsqu'il pénètre dans des zones spécifiques lors de ses déplacements ;

- par rapport à la présentation d'information : les unités sémantiques sont mémorisées par l'entité utilisateur, pour être éventuellement présentées plus tard, lorsque l'utilisateur rencontre un dispositif de présentation au hasard de ses déplacements.

Lien avec les espaces perceptuels et de rayonnement La première phase (fourniture d'une u.s.) répond à la question *« quelle information présenter ? »*. Cette réponse dépend de l'endroit où se trouve l'utilisateur par rapport aux espaces de rayonnement des sources d'information. La réponse n'est que partielle, car un filtrage des informations peut ensuite être nécessaire, de façon à ce que l'utilisateur dispose d'informations réellement utiles. Ce filtrage peut se baser sur les métadonnées des unités sémantiques.

La deuxième phase (présentation d'une u.s.) répond à la question *« comment et où présenter l'information »*. Cette réponse est fonction de la position de l'utilisateur par rapport aux différents dispositifs de présentation, c'est-à-dire du contenu de l'ensemble perceptuel de l'utilisateur.

Le tableau 5.2 synthétise ce résultat.

Question		Outil pour la solution
quoi ?	$\longrightarrow$	$\mathcal{ER}$, espace de rayonnement des sources d'information
comment et où ?	$\longrightarrow$	$\mathcal{EP}$, l'espace perceptuel de l'utilisateur

Tableau 5.2 : *Outils pour répondre aux questions sur les entités.*

5.4.2 Précisions sur les interactions entre entités logicielles

Trois sortes d'entités logicielles sont présentes dans le modèle KUP : informateurs (K_ℓ), utilisateurs (U_ℓ) et présentateurs (P_ℓ). Dans la section précédente, nous avons vu les deux interactions *majeures* entre ces trois entités. Cependant, à côté de ces interactions majeures, on peut imaginer d'autres interactions d'ordre mineur. Dans cette section, nous nous proposons d'analyser de façon exhaustive la signification des interactions entre deux entités quelconques. Si l'on prend en compte la *direction* des interactions (i.e. quelle entité est émettrice d'un message ; quelle entité est réceptrice), on a $3 \times 3 = 9$ cas à examiner. La figure 5.11 résume les différents cas, qui sont détaillés ci-après.

[K $\rightarrow$ K] *Interaction d'une source d'information vers une autre source d'information.* Cela pourrait correspondre à la notion d'*accrochage sémantique* qui a été définie en section 4.2.1.3. Cependant, se pose alors la question de savoir si les accrochages sémantiques ont lieu :

- soit au niveau des sources d'information, donc à un niveau *global*, en une seule fois pour un certain nombre d'u.s. à venir ;
- soit au niveau des unités sémantiques elles-mêmes, donc à un niveau extrêmement *local*, au cas par cas.

Nous n'avons pas traité des accrochages sémantiques dans notre travail. Nous n'avons donc pas tranché cette question.

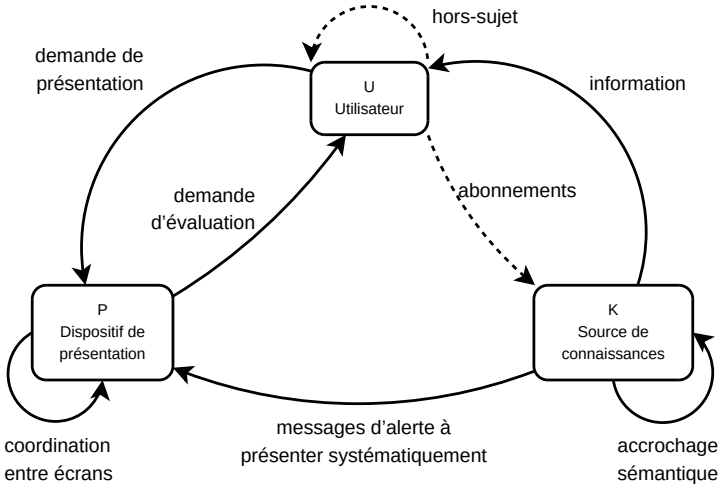


Figure 5.11 : *Interactions entre entités.*

[K → U] *Interaction d'une source d'information vers un utilisateur.* Il s'agit de la relation d'*information* : lorsqu'un utilisateur susceptible d'être intéressé par une information pénètre dans l'espace de rayonnement d'une source d'information, cette dernière lui fournit l'unité sémantique correspondante.

[K → P] *Interaction d'une source d'information vers un dispositif de présentation.* A priori, les unités sémantiques sont d'abord envoyées par une source d'information à une entité utilisateur, laquelle demande ensuite à des dispositifs de présentation de les présenter. Dans ce schéma, il n'y a donc pas d'interaction directe entre une source d'information et un dispositif de présentation : tout passe par l'entité utilisateur, centre névralgique du système. Cependant, il existe des situations dans lesquelles il pourrait être souhaitable de permettre aux informateurs de communiquer directement avec les présentateurs. Par exemple, si un ordre d'évacuation est donné dans un aéroport, l'ordre d'évacuation *doit* être présenté par *tous* les dispositifs disponibles (écrans, haut-parleurs, etc.). Dans ce cas, aucune sélection n'est à opérer dans les informations présentées, donc il n'est pas utile que les entités utilisateurs entrent en jeu. On peut donc imaginer que dans ce cas précis, l'information soit directement transmise aux dispositifs, ce qui éviterait d'ailleurs d'éventuels problèmes de congestion des réseaux.

[U → K] *Interaction d'un utilisateur vers une source d'information.* Il n'est pas prévu que les entités utilisateurs aient un comportement proactif vis-à-vis des sources d'information. Nous avons vu plus haut que les sources d'information fournissent des u.s. aux entités utilisateurs : c'est le seul sens de circulation de l'information. Cependant, on pourrait imaginer qu'il existe un système d'abonnement à tel ou tel type de message, ce qui permettrait aux sources d'information de n'envoyer certaines informations qu'aux entités utilisateurs qui en auraient spécifiquement fait la demande, ou réciproquement, de ne pas envoyer certains types d'informations à ceux qui les auraient explicitement refusés. Dans ce cas, l'action de s'abonner ou de se désabonner serait une interaction d'une entité utilisateur vers une source d'information.

[U → U] *Interaction d'un utilisateur vers un autre utilisateur.* Dans ce cas, les interactions entre utilisateurs du monde réel seraient *médiées* par les entités qui leurs correspondent. Nous ne traitons pas de ces possibilités dans ce travail.

[U → P] *Interaction d'un utilisateur vers un dispositif de présentation.* Après qu'une entité utilisateur a reçu une unité sémantique, son but est d'obtenir sa *présentation* à destination de l'utilisateur humain qu'elle représente. Ainsi, lorsque l'utilisateur s'approche d'un dispositif de présentation, l'entité demande à ce dernier de négocier la présentation des unités sémantiques qu'il a éventuellement reçues.

[P → K] *Interaction d'un dispositif de présentation vers une source d'information.* Dans notre modèle, les informations sont fournies par les sources d'information, normalement aux entités utilisateurs, et éventuellement directement aux dispositifs de présentation, mais uniquement dans des circonstances bien particulières. Il n'y a aucune raison pour qu'un dispositif de présentation interroge une source d'information. Nous pensons donc que ce type d'interaction ne peut pas exister dans notre modèle.

[P → U] *Interaction d'un dispositif de présentation vers un utilisateur.* Il arrive qu'un dispositif de présentation ait à modifier la présentation des unités sémantiques dont il a la charge. Pour cela, il doit déterminer quelle est l'option la moins perturbante pour les utilisateurs. Il va donc négocier une telle modification avec chacun des utilisateurs intéressés par l'une de ses unités sémantiques. Ce mécanisme est détaillé dans la section 6.3.

[P → P] *Interaction d'un dispositif de présentation vers un autre dispositif de présentation.* Les dispositifs de présentation peuvent être amenés à interagir entre eux, notamment lorsqu'il faut se coordonner pour optimiser l'espace d'affichage (s'il s'agit d'écrans), et faire migrer des unités sémantiques d'un dispositif à un autre. Ce mécanisme est lui aussi détaillé dans la section 6.3.

5.5 Modèle à agents

5.5.1 Introduction

Dans cette section, nous allons voir comment le modèle abstrait KUP peut être concrétisé sous la forme d'une architecture informatique. Bien entendu, nous pourrions envisager de construire une solution basée sur une architecture *centralisée*. Ainsi, toutes les entités correspondraient à des processus (ou plus généralement à des segments de code) s'exécutant sur un système informatique central. Cependant, nous pensons qu'une telle méthode présente un certain nombre d'inconvénients, principalement :

- sa *fragilité* : si les serveurs centraux tombent en panne, le système entier cesse de fonctionner ;
- et sa *rigidité* : il est impossible de déplacer les dispositifs de présentation à volonté, du moins sans travail de reconfiguration.

À l'inverse, nous souhaitons donner au personnel des lieux dans lesquels notre système sera déployé la possibilité de déplacer les dispositifs de présentation, d'en apporter de nouveaux si un événement particulier survient, etc., tout ceci sans être obligé de configurer quoi que ce soit. Les dispositifs de présentation doivent être capables de s'adapter d'eux-mêmes aux changements, sans qu'une intervention humaine soit nécessaire.

Nous proposons donc de bâtir une architecture logicielle distribuée, basée sur la notion d'*agent*. Ainsi, chacun des trois types d'entités logicielles évoqués précédemment correspondra à un type d'agents :

- les *agents-utilisateurs* (U) : ils correspondent aux utilisateurs ; ils connaissent leurs préférences ;
- les *agents-informateurs* (K) : ils correspondent aux sources d'information ; ils fournissent des informations aux agents-utilisateurs ;
- les *agents présentateurs* (P) : ils correspondent aux dispositifs de présentation ; ils sont capables de négocier la présentation d'une u.s. sur un dispositif, et d'effectuer cette présentation.

Ainsi, le monde des agents constitue le « miroir » du monde réel, du moins en ce qui concerne nos trois types d'entités d'intérêt.

5.5.2 Caractéristiques des agents

Communication Nous supposons que tous les agents peuvent communiquer entre eux. Cette supposition nous semble réaliste : les communications peuvent passer par des réseaux sans fils de type WiFi, qui sont maintenant courants. De plus, de tels réseaux sont largement déployés dans certains bâtiments, par exemple à Supélec, qui est couvert par un réseau au maillage très dense. Lorsque des réseaux de type WiFi ne sont pas disponibles, il est possible d'avoir recours à des réseaux de téléphonie mobile, dont les zones de couverture sont beaucoup plus étendues.

Perception, proximité Des notions de perception et de rayonnement ont été définies pour les entités physiques et abstraites (voir section 5.2). Ces relations sont alors *projetées* dans le monde des agents. Ainsi, si par exemple une entité utilisateur u perçoit une entité dispositif de présentation p, alors la même relation existera entre les agents associés : nous dirons par abus de langage que l'agent u perçoit l'agent p. De façon homonyme, nous dirons également que l'agent u est *proche* de l'agent p.

Mais comment le système informatique accédera-t-il à cette notion de perception ? Par exemple, comment pourra-t-on savoir en pratique que l'utilisateur a perçoit le dispositif b ? Comme nous l'avons déjà évoqué, il faudra effectuer des *mesures*. Par exemple, la proximité peut être détectée par des lecteurs d'étiquettes RFID (voir section 3.2.3.2), ou encore des systèmes basés sur la réception de signaux Bluetooth ou infrarouges (voir section 3.2.3.1).

Réactivité Les agents sont de type *réactif* : ils restent en sommeil la plupart du temps, et réagissent lorsque des événements particuliers se produisent. En pratique, un agent donné a peut réagir à trois sortes d'événements :

1. un autre agent b vient de s'approcher de a ;
2. un agent b, auparavant proche de a, vient de s'en éloigner ;
3. a vient de recevoir un message par le réseau, en provenance d'un agent c quelconque, qui n'est pas nécessairement proche de a.

Ainsi, si les agents se trouvaient seuls dans le système informatique, *il ne se passerait jamais rien*. Les agents ont des comportements réactifs *lorsque se déplacent les entités qu'ils représentent* (i.e. les humains et les dispositifs de présentation). Cela signifie que toute la proactivité du système est assurée par les utilisateurs humains : ce sont ces derniers qui vont se déplacer (ou déplacer des dispositifs de présentation !) et de là, déclencher des cascades de réactions dans le système. Au final, on ne construit pas réellement un *monde d'agents*, mais plutôt une *agentification du monde réel*.

On rejoint ainsi la vision de l'intelligence ambiante, dans laquelle des systèmes informatiques sont *discrètement* à l'écoute des actions des êtres humains, et peuvent ainsi intervenir de façon opportune et non intrusive.

Notons bien que les intentions des êtres humains ne sont absolument *pas* modélisés par le système. Tout ce que nous retenons des êtres humains, ce sont leurs *comportements* effectifs, tels qu'observés à travers l'évolution des relations de proximité. Nous avons ainsi une vision *behavioriste* des utilisateurs.

Notons également que la dynamique des trois catégories d'agents est différente :

- les agents utilisateurs sont clairement les plus rapides. Ils correspondent à des êtres humains qui se déplacent sans cesse, car par hypothèse, ils sont en situation de mobilité (gare, aéroport) ;
- viennent ensuite les agents présentateurs, car nous concevons spécifiquement un système dans lequel il doit être facile de déplacer des dispositifs de présentation. Par exemple, il est possible d'apporter un écran supplémentaire dans un hall d'embarquement si tous les écrans qui s'y trouvent sont saturés. De plus, les utilisateurs peuvent éventuellement se déplacer avec leurs propres dispositifs de présentation (téléphones, PDA), d'où une dynamique assez élevée ;
- les agents informateurs nous semblent être les moins rapides, car leurs déplacements correspondent à la réorganisation de zones informationnelles. Il nous semble que ces zones ne devraient être modifiées qu'en cas de changement structurel dans un environnement (par exemple, changement de l'affectation de certains locaux), ou en cas de modifications durables (par exemple, travaux dans une aile d'un bâtiment). Dans tous les cas, ces modifications possèdent des constantes de temps assez longues.

À la lumière de cette organisation sous forme d'agents, il est possible de détailler un peu plus les étapes de la procédure de *fourniture et présentation* d'information. Les étapes sont alors les suivantes :

1. un agent utilisateur pénètre dans l'espace de rayonnement d'un agent informateur ;
2. *éventuellement*, et *à un moment donné, non spécifié*, l'agent informateur envoie une unité sémantique à l'agent utilisateur ;
3. celui-ci va chercher à effectuer la présentation de l'u.s. pour l'utilisateur (réel, humain, physique). Il prend donc contact avec chaque agent présentateur dont le dispositif se trouve dans l'espace perceptuel de l'utilisateur, et négocie avec lui la présentation de l'u.s. Si possible, il choisit l'un des agents présentateurs et lui confie la présentation de l'u.s. ;
4. par la suite, lors des entrées et sorties d'agents présentateurs de l'espace perceptuel de l'être humain, on reconsidère la présentation de l'u.s. et on fait éventuellement migrer cette dernière (ou tout simplement, on la fait se présenter si elle ne l'était pas auparavant) sur un nouveau dispositif.

Les agents mis en jeu sont représentés sur le diagramme de classes de la figure 5.12. On note que *conceptuellement*, tout est basé sur la perception de l'agent utilisateur, même si, en pratique, ce sont les agents informateurs qui détectent la présence des agents utilisateurs. Cependant, cette subtilité pratique ne change rien, car les agents informateurs sont définis de telle façon que leur espace perceptuel soit égal à leur espace de rayonnement.

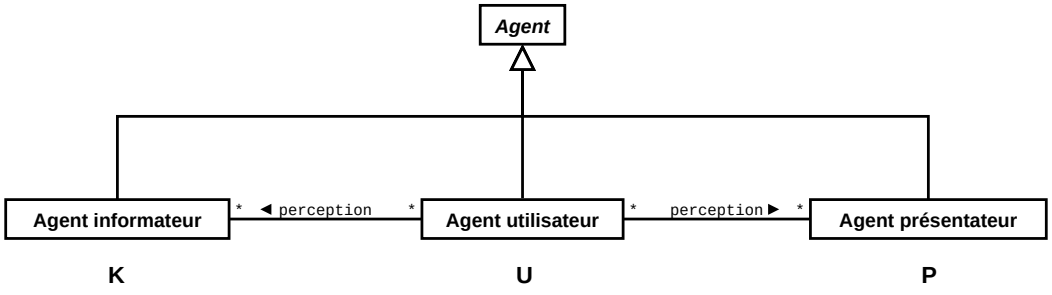


Figure 5.12 : *Le modèle de la figure 5.9 se* projette *de cette façon dans le monde des agents.*

5.5.3 Retour sur la structure du modèle

Nous l'avons déjà vu au cours de la section 5.1 : le modèle KUP propose un changement de paradigme par rapport aux modèles d'architecture logicielle classiques en IHM. KUP est un modèle conceptuel qui peut être dérivé en modèles d'architecture dans lesquels l'utilisateur est placé au centre du système, alors qu'il est relégué en périphérie par les modèles classiques. La concrétisation des entités de KUP sous forme d'agents, et en particulier la présence d'un agent utilisateur, renforcent ce changement de paradigme.

En effet, même si on « agentifie » un modèle d'architecture logicielle classique d'IHM (par exemple MVC ou PAC), l'utilisateur U n'est jamais représenté explicitement par le modèle. Dans notre approche au contraire, l'élément le plus actif est l'agent utilisateur : c'est en effet à ce dernier que l'on *fournit* les unités sémantiques, et c'est lui qui se charge de négocier leur *présentation* sur des dispositifs situés à proximité perceptuelle de l'utilisateur.

La figure 5.13 montre le passage d'un modèle d'architecture logicielle classique au modèle KUP, aussi bien au niveau des entités d'intérêt qu'au niveau des agents.

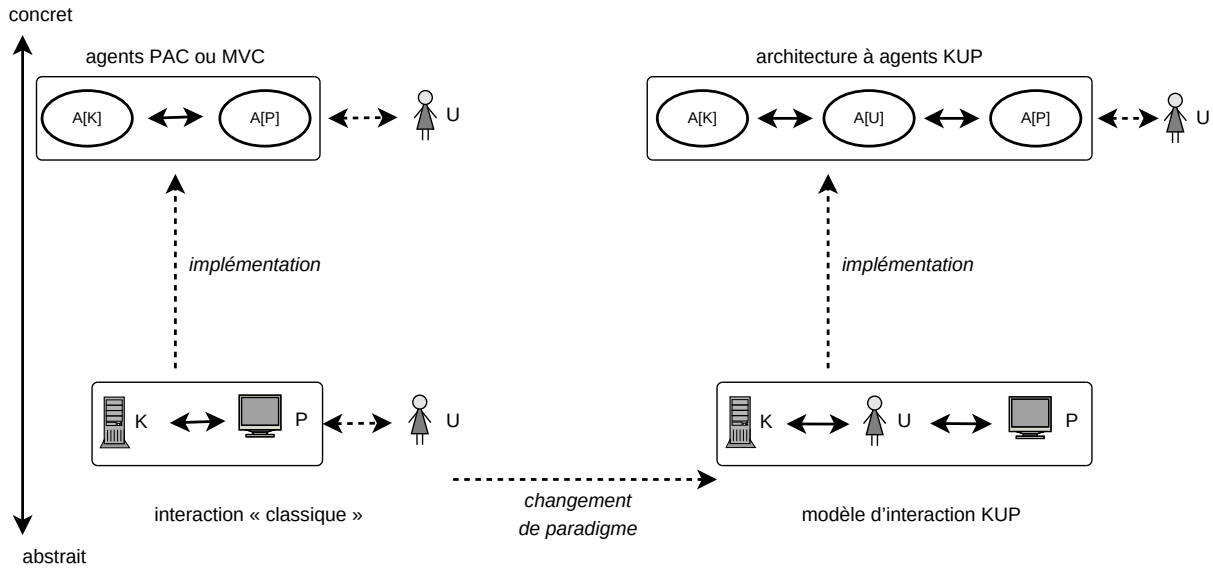


Figure 5.13 : *Modèle d'architecture logicielle classique en IHM, modèle KUP, et implémentations à agents.*

Problèmes pratiques Le problème qui se pose en pratique est celui de la détection par les agents informateurs des utilisateurs présents dans leur périmètre d'information.

Certains cas sont simples : par exemple, dans un aéroport, on peut placer un seul agent informateur pour toute l'aérogare. L'entrée et la sortie des utilisateurs du périmètre en question se fait simplement au niveau des portiques d'accès. Autre exemple, le cas où l'on assimile le périmètre d'information d'un agent informateur à l'espace de rayonnement d'un dispositif de présentation : dans ce cas, le détecteur de proximité du dispositif de présentation permet également l'entrée des utilisateurs dans le périmètre d'information.

On pourrait même simplifier le fonctionnement à l'extrême, et supposer que le périmètre d'information de l'agent informateur s'étend à *l'espace en entier*. Ainsi, on communique son u.s. à l'agent utilisateur dès lors que l'on peut communiquer avec lui, sans se préoccuper de sa localisation physique. En fait, cela signifie que l'on confond le périmètre d'information avec la zone de couverture du réseau physique.

Mais dans le cas général, le problème reste ouvert, et on est obligé d'utiliser des techniques classiques de localisation des utilisateurs. Par exemple, les PDA éventuellement portés par des utilisateurs sont capables de se positionner par rapport aux bornes WiFi situées dans

l'environnement. Ils peuvent alors communiquer leurs positions à un service global de localisation, qui à son tour avertira les agents informateurs de l'arrivée et du départ des utilisateurs.

Localisation concrète des agents Nous venons de voir que les agents correspondent aux entités du modèle KUP, donc d'un point de vue *logique*, on confond leur localisation avec celle des entités physiques qu'ils représentent. Cependant, d'un point de vue *concret*, un agent est un processus logiciel, et il convient de déterminer précisément sur quel système informatique il doit fonctionner. C'est ce que nous appelons la *localisation concrète* des agents.

Nous n'introduisons pas de contrainte particulière sur cette localisation concrète. Il est possible par exemple de la calquer sur la localisation logique des agents. Dans ce cas, chaque utilisateur porterait une petite unité de traitement (un PDA par exemple) qui exécuterait le code correspondant à son agent ; de même les agents présentateurs fonctionneraient sur des processeurs installés à l'intérieur des dispositifs de présentation.

Si une telle architecture est techniquement envisageable, elle est difficile à mettre en œuvre en pratique, par exemple dans le cas d'un aéroport. En effet, il faudrait dans ce cas que chaque passager dispose d'un PDA, ce qui n'est pas très réaliste. On peut alors imaginer une architecture concrète totalement différente, dans laquelle *tous* les agents en jeu dans l'aéroport (informateurs, utilisateurs et présentateurs) fonctionneraient sur des serveurs centraux, ce qui élimine le problème de la dissémination d'unités de traitement informatiques. Dans ce cas, par exemple, les agents utilisateurs pourraient être créés sur les serveurs de l'aéroport lors de l'achat des billets correspondants.

Entre les deux cas extrêmes que nous venons d'évoquer, il est bien entendu possible d'imaginer toute une série de configurations intermédiaires. En pratique, les personnes chargées de l'implémentation d'une application donnée auront toute latitude pour décider de l'emplacement concret des agents. Dans la suite, nous ne nous préoccuperons donc plus que de la localisation *logique* des agents : celle des entités physiques correspondantes.

5.6 Exemples

5.6.1 L'aérogare

Nous supposons que dans un aéroport soit installé un système de fourniture d'information aux utilisateurs mobiles. Le système fournit aux passagers le numéro de leur porte d'embarquement, ainsi que l'heure de fin de l'embarquement. Ainsi, les unités sémantiques correspondent aux triplets ⟨ numéro de vol, heure d'embarquement, numéro de porte ⟩.

On suppose que ces informations sont pertinentes dans tout le périmètre de l'aéroport, donc il existe une unique source d'information dont l'espace de rayonnement s'étend à l'aérogare en entier. Ainsi, dès qu'ils y pénètrent, les agents utilisateurs reçoivent immédiatement leurs unités sémantiques. Au hasard des déplacements des utilisateur, ces u.s. pourront alors être présentées par les écrans et haut-parleurs répartis dans les halls de l'aérogare.

5.6.2 La porte intelligente

La *porte intelligente* est une porte conventionnelle, augmentée de capacités de communication avec les utilisateurs. Par exemple, elle peut les renseigner sur ce sur quoi elle donne accès, indiquer qui a le droit de la franchir, etc. On suppose que la porte dispose d'une étiquette statique qui indique sa fonction (i.e. une petite plaque vissée sur la porte), ainsi que d'un écran et d'un système de synthèse vocale, qui permet d'émettre des informations à l'attention des non-voyants.

Nous supposons alors qu'un utilisateur aveugle s'approche de la porte. Cet utilisateur possède une tablette Braille et un PDA, qu'il porte sur lui en permanence. Nous nous posons alors la question de savoir comment cette situation peut être modélisée par le système, et en particulier comment est prise en compte la tablette Braille.

La notion de dispositifs de présentation *privés* et *publics* permet de résoudre ce problème de façon élégante. En effet, on considère que l'on a trois dispositifs publics (l'étiquette, l'écran, le haut-parleur) et un dispositif privé (la tablette Braille). Lorsque l'agent informateur lié à la porte a une information à transmettre, il contacte l'agent de l'utilisateur (situé sur son PDA) et lui transmet une unité sémantique. Celui-ci choisit un dispositif de présentation approprié (i.e. compatible avec l'u.s.), et ce dernier présente l'u.s. Il n'y a pas de distinction particulière faite à l'égard de la tablette Braille : on utilise exactement les mêmes mécanismes que pour les autres dispositifs de présentation.

Le diagramme d'objets correspondant est présenté sur la figure 5.14. Il est conforme au diagramme de classes de la figure 5.9.

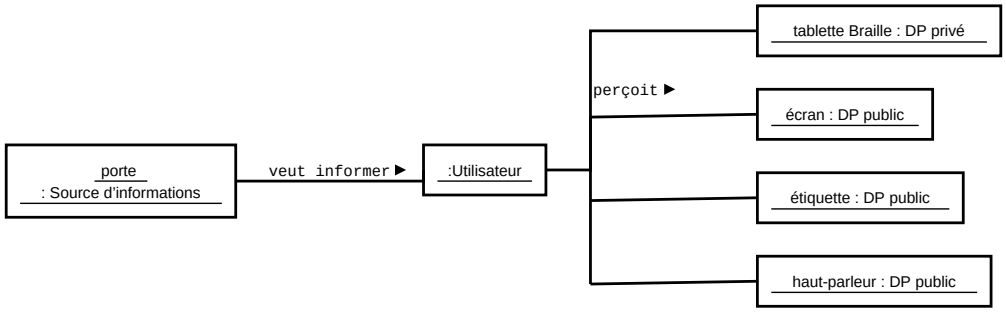


Figure 5.14 : *L'exemple de la porte exprimé sous forme d'un* diagramme d'objets *UML.*

5.6.3 Décomposition des entités physiques

Le modèle KUP gère la situation où plusieurs sources de connaissances fournissent des informations à présenter sur un ou des dispositifs portés par l'utilisateur. Par exemple, la figure 5.15 représente la situation d'un utilisateur situé à proximité d'une photocopieuse, et qui dispose d'un téléphone, d'un PDA, d'une montre et d'une oreillette. Le PDA est décomposé selon le modèle KUP : d'un côté la source d'information « emploi du temps », et de l'autre le dispositif de présentation « écran PDA ». De même, la montre est décomposée selon

une source d'information « heure » et un dispositif de présentation « écran de la montre » (que l'on suppose être un afficheur à cristaux liquides en mode point).

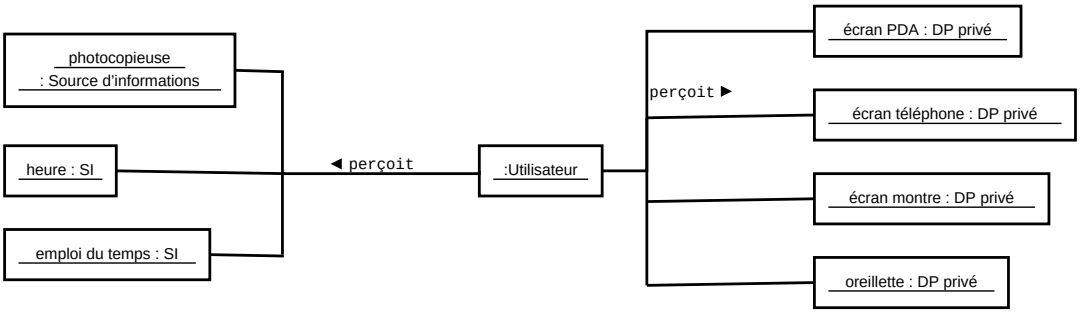


Figure 5.15 : *Exemple de décomposition d'objets courants en sources d'information et dispositifs de présentation.*

Le problème de cette approche, c'est qu'elle autoriserait a priori des combinaisons *peu souhaitables*, par exemple : affichage de l'heure (donnée par la montre) sur l'écran du PDA, de l'emploi du temps sur le téléphone, et du panneau de contrôle de la photocopieuse sur la montre. Cependant, on peut se dire qu'en pratique, les contraintes portant sur la taille feront qu'une telle situation aberrante ne devrait pas se produire : il est physiquement impossible d'afficher une information aussi complexe qu'un panneau de commande de photocopieuse sur une montre (ces questions seront détaillées dans le prochain chapitre).

Néanmoins, cet exemple soulève la question du *niveau de décomposition* souhaitable : même s'il est effectivement possible de décomposer toutes les entités existantes, ce n'est pas la finalité du modèle. En effet, le modèle KUP est conçu pour créer de nouvelles possibilités d'interaction, pas pour compliquer les interactions avec les dispositifs existants. En l'occurrence, il ne paraît pas du tout pertinent de séparer la montre en une source de données d'une part, et un dispositif de présentation d'autre part. Il paraît logique que la montre affiche directement l'heure, sans passer par une procédure relativement complexe de choix de dispositif.

Par contre, *en plus* d'afficher l'heure, la montre pourrait tout à fait offrir une petite zone de message, utilisable pour la présentation d'u.s. *supplémentaires*. Dans ce cas-ci, on ne modifie pas la fonction de base de la montre, qui reste gérée en interne par la logique habituelle et sans algorithme particulier, mais on lui *ajoute* une nouvelle fonctionnalité, qui n'interfère pas d'ailleurs avec sa fonctionnalité de base.

En conclusion, il semble que la décomposition en sources d'information et dispositifs de présentation ne doive se faire que pour permettre de nouvelles conditions d'interaction, ou l'utilisation de nouveaux dispositifs ou de nouvelles modalités. Par contre, il ne faut pas décomposer à outrance et ainsi briser les associations *naturelles*.

5.7 Conclusion

Dans ce chapitre, nous avons présenté un nouveau modèle pour les interactions homme-machine dans un cadre ambiant : le modèle KUP. Il nous a permis de définir un système opportuniste pour la fourniture et la présentation d'informations à des utilisateurs mobiles. Il propose une approche complètement décentralisée, qui s'adapte particulièrement bien à une instanciation sous forme d'agents logiciels.

Une telle approche décentralisée peut alors fonctionner de manière *spontanée*, au gré des déplacements des entités physiques, et sans qu'il y ait besoin d'une quelconque configuration manuelle.

Cependant, le présent chapitre n'a pas détaillé certains points, qui seront traités au chapitre suivant :

1. la façon dont peut être sélectionnée une modalité d'interaction, et surtout la façon dont est ensuite instanciée cette modalité ;
2. la façon dont est opéré le choix d'un dispositif de présentation pour une unité sémantique, lorsqu'a priori plusieurs dispositifs pourraient convenir. Par la même occasion, nous expliquerons comment une forme de *coopération* peut naître entre dispositifs de présentation voisins, de façon à optimiser l'espace de présentation et à éviter la présentation d'informations redondantes.

Chapitre 6

Algorithmes

Dans la section 5.4.1, nous avons vu que pour présenter une unité sémantique donnée, il est nécessaire de résoudre deux sous-problèmes imbriqués :

- pour un dispositif de présentation donné, il faut choisir une modalité, ainsi qu'une instanciation de cette modalité qui convienne à l'utilisateur, selon laquelle l'unité sémantique doit savoir s'exprimer, et qui respecte les contraintes de partage du dispositif entre plusieurs unités sémantiques ;
- entre plusieurs dispositifs de présentation utilisables, il faut décider quel dispositif utiliser.

Ces deux sous-problèmes doivent être résolus lorsque deux ou plusieurs entités entrent en proximité. Après quelques précisions sur la façon dont cette proximité peut être mesurée, le présent chapitre se propose d'aborder successivement ces deux problèmes et d'indiquer les solutions correspondantes.

6.1 Mesure de la proximité

Dans la section 5.2.2, nous avons défini formellement la notion d'espace perceptuel. Cette notion tient compte des modalités utilisées par les entités, d'où sa définition comme sous-ensemble d'un *m-espace*. Ainsi, l'espace perceptuel d'un utilisateur situé à proximité d'un haut-parleur peut contenir ou non ce haut-parleur en fonction de la valeur donnée au volume sonore de ce dernier.

Cependant, cette approche est délicate en pratique, car elle nous a conduits à définir les espaces perceptuels comme étant des fonctions sur l'ensemble produit des domaines de définition des attributs (qui peut être très grand). On aboutirait ainsi à des algorithmes très complexes, et surtout à des temps de décision probablement grands, car l'espace des combinaisons à examiner serait gigantesque.

En fait, manipuler des espaces perceptuels fonctions de l'instanciation serait raisonnablement envisageable *si l'instanciation était déjà faite*. Or en pratique, les espaces perceptuels ne sont pas calculés ou déduits : ils sont *mesurés*, et le but de cette mesure est justement de permettre la détermination de l'instanciation. Ainsi, la paramétrisation des espaces perceptuels par les variables d'instanciation conduit à multiplier le nombre d'inconnues à déterminer.

Nous avons choisi une option plus simple : l'espace perceptuel d'un utilisateur (ou l'espace de rayonnement d'un dispositif de présentation) est pragmatiquement défini en tenant compte des valeurs *maximales* des paramètres du dispositif (le plus haut volume sonore, la plus grande taille de caractères, etc.) Ceci permet de prendre en considération tous les dispositifs *potentiels* lors de la recherche d'un dispositif de présentation. Si ensuite il n'est pas possible de donner les valeurs maximales à certains paramètres d'un dispositif donné, ce dernier sera tout simplement éliminé par le processus de négociation (voir ci-dessous).

En d'autres termes, on élargit au maximum le champ des dispositifs considérés, quitte à les éliminer plus tard s'ils ne conviennent pas. Cette approche nous semble la plus simple, car elle n'introduit *aucune* complexité supplémentaire dans les algorithmes (contrairement à l'approche qui consisterait à considérer les espaces perceptuels comme fonctions de l'instanciation). Dans le pire des cas, on est amené à prendre initialement en considération certains dispositifs de sortie qui seront rapidement éliminés car ils ne conviendraient pas.

En pratique, l'ensemble perceptuel *pragmatique* d'une entité sera donc défini comme étant l'ensemble des entités *détectables* par son dispositif de mesure physique (lecteur RFID, détecteur de badges, détecteurs d'orientation, etc.). On introduit ainsi une notion simple de *proximité* binaire.

6.2 Choix et instanciation d'une modalité pour la présentation d'une unité sémantique

6.2.1 Modélisation de l'utilisateur et de l'environnement

Souchon définit le contexte comme étant *l'environnement complet dans lequel une tâche est effectuée* [Souchon 2002]. Cet environnement est représenté par trois modèles : modèle de l'utilisateur, modèle de la plate-forme (i.e. des équipements informatiques), et modèle de l'environnement (conditions physiques[1], lieu, environnement social et organisationnel).

Dans la section 4.2.2.1, nous nous sommes intéressés à la modélisation de l'utilisateur. Nous ne traiterons pas ici de l'historique des informations qui lui ont déjà été communiquées. L'utilisateur sera donc modélisé par un profil sensorio-cognitif d'une part, et par sa position géographique d'autre part. Cette information géographique est double :

[1] Exemples : luminosité, pression, etc.

- position de l'utilisateur par rapport à un dispositif d'interaction : cet aspect est modélisé par la notion de proximité, ainsi que par une mesure de la distance, qui sera utilisée lors du processus d'instanciation ;

- position de l'utilisateur par rapport à des zones ciblées par des informations différentes : cet aspect est tout simplement modélisé par la localisation des agents informateurs, qui fournissent des unités sémantiques aux utilisateurs qui passent dans leur zone.

Le profil sensorio-cognitif de l'utilisateur caractérise ses capacités ou incapacités physiques (vue, ouïe, etc.), ainsi que ses compétences cognitives (par exemple, langues parlées et lues). Ces caractéristiques permettent ou interdisent l'utilisation de certaines modalités. Ce profil est relativement peu dynamique. Il peut évoluer au cours du temps (vieillissement des organes, maladies ou au contraire guérisons, apprentissage de nouvelles langues, etc.), mais ces évolutions sont relativement lentes. À l'échelle de temps d'utilisation de notre système, on peut le considérer comme étant statique.

Ce profil sensorio-cognitif doit également permettre à l'utilisateur d'exprimer ses préférences : lorsqu'il est possible d'utiliser plusieurs modalités pour présenter une u.s., il est normal d'utiliser celle que l'utilisateur préfère. Ces préférences peuvent concerner un mode, une modalité, ou même un attribut. Par exemple, on peut imaginer qu'un locuteur francophone de naissance, mais qui comprend l'Anglais, préfère néanmoins écouter des messages en Français plutôt qu'en Anglais.

De la même façon qu'un profil sensoriel est associé à chaque utilisateur pour modéliser ses capacités et préférences d'interaction, les caractéristiques d'interaction des dispositifs de présentation doivent elles aussi être précisées. Par exemple, il est important de noter qu'un écran sait utiliser la modalité textuelle, avec une taille de texte pouvant prendre un certain nombre de valeurs ; qu'un haut-parleur sait utiliser la modalité auditive avec certains volumes sonores, etc. Ceci permet donc d'établir un profil pour chaque dispositif de présentation, chargé de recenser ses capacités multimodales en sortie.

Le profil d'un dispositif de présentation peut comprendre des caractéristiques statiques aussi bien que dynamiques, par exemple :

- un écran ne prend en charge que des modalités visuelles, un écran alphanumérique ne prend pas en charge les modalités graphiques : ce sont des caractéristiques *statiques*.

- un téléphone dont la charge de la batterie passe en-dessous d'un seuil critique peut désactiver sa fonction vibreur : le fait que cette modalité soit utilisable ou non est donc une caractéristique *dynamique*. De même, il est possible qu'un composant donné d'un dispositif de présentation (par exemple le haut parleur d'un système « écran + synthèse vocale ») tombe en panne, sans pour autant compromettre l'ensemble. Dans ce cas, s'il diagnostique la panne, le système supprimera la ou les modalités en question de sa liste de possibilités.

L'environnement, tel que défini plus haut, nous intéresse dans la mesure où il peut perturber les interactions entre les dispositifs et les utilisateurs. Il s'agit donc de modéliser ces perturbations, de façon à les contourner. Par exemple, dans un environnement bruyant, un système à haut-parleur devra augmenter son volume sonore de façon à couvrir le bruit ambiant. Nous pourrions donc introduire un profil de l'environnement, qui contiendrait des contraintes supplémentaires liées aux perturbations de cet environnement.

Cependant, une telle organisation laisserait supposer qu'il existe quelque part une entité « environnement » chargée de maintenir à jour ce profil. Le profil d'environnement serait alors utilisé lors de chaque présentation d'information. Cela nous paraît relativement compliqué, et nous préférons proposer une solution plus simple : intégrer les caractéristiques pertinentes du profil de l'environnement aux caractéristiques dynamiques des profils des dispositifs de présentation. Par exemple, s'il y a du bruit aux alentours, le profil d'un dispositif haut-parleur indiquera simplement que le niveau sonore doit être supérieur à une valeur donnée, voire que la modalité sonore n'est pas du tout utilisable sur ce dispositif.

Cette solution présente l'avantage de la simplicité, et de plus, elle nous semble correspondre à une implémentation assez naturelle. En effet, il paraît raisonnable qu'un dispositif de présentation se charge lui-même de vérifier les conditions dans lesquelles ses messages pourront être diffusés. Par exemple, un haut-parleur disposera d'un microphone chargé de mesurer le bruit ambiant, de façon à vérifier qu'il n'est pas trop élevé ; un écran disposera d'un capteur de luminosité afin de s'assurer que le contraste par rapport à la lumière ambiante est bon, etc. Dans ces conditions, il est donc naturel que les contraintes dynamiques imposées par l'environnement soient représentées au sein du profil du *dispositif*.

Ainsi, le profil du dispositif permet de tenir compte dans l'environnement au niveau du choix de la modalité (au cas où une modalité soit inutilisable dans un contexte donné), du dispositif (car si on indique qu'une modalité correspondant au dispositif courant est inutilisable, cela va influer sur le choix du dispositif), et enfin de l'instanciation de la modalité (choix de valeurs d'attributs différentes selon le contexte).

Notons cependant que cette solution fait apparaître au niveau de certains dispositifs des *boucles réflexes*, c'est-à-dire des traitements locaux des entrées, dans le but de produire des effets locaux eux aussi (modification du profil des dispositifs considérés). Nous verrons en perspectives que de façon beaucoup plus générale, le modèle pourrait être étendu pour tenir compte des entrées.

Enfin, dans la définition d'une unité sémantique (voir section 5.3), nous avons indiqué qu'une u.s. est capable de produire un contenu concret dans un certain nombre de modalités. Il est donc nécessaire de pouvoir indiquer quelles sont ces modalités « utilisables » par chaque unité sémantique. Pour cette raison, nous décidons d'attribuer un profil à chaque unité sémantique. Ce profil est statique. Il indique simplement quelles sont les modalités selon lesquelles l'u.s. sait générer un contenu concret.

En résumé, il existe donc trois sortes de profils différents :

- *profil d'un utilisateur :* il décrit les capacités et préférences d'un utilisateur donné. Il possède une composante dynamique, mais qui varie lentement ;
- *profil d'un dispositif de présentation :* il décrit les capacités du dispositif, en relation avec les contraintes extérieures imposées par l'environnement. Les pannes du dispositif ainsi que la variation des conditions de l'environnement imposent à ce profil une grande dynamicité ;
- *profil d'une unité sémantique :* ce profil statique décrit les modalités selon lesquelles l'u.s. sait générer un contenu concret.

Lors de la présentation d'une u.s. donnée, pour un utilisateur donné, sur un dispositif de présentation donné, il est donc nécessaire de *mettre en relation* ces trois profils. Les modalités et valeurs d'attributs utilisables sont les modalités et valeurs d'attributs *communs* à chacun des trois profils. Autrement dit, les solutions se trouvent à l'*intersection* de ces profils.

Les sections suivantes se fixent donc comme objectif de définir formellement la notion de *profil*, ainsi que la notion d'*intersection de profils*. Une fois ces notions posées, il sera possible de donner l'algorithme de choix et d'instanciation de modalité proprement dit. Une vue d'ensemble de cet algorithme est représentée sur la figure 6.1.

6.2.2 Arbres taxonomiques

Avant de passer à l'expression des profils, nous devons définir la notion préliminaire d'*arbre taxonomique*. Un arbre taxonomique permet de définir formellement une taxonomie des modalités telle que celle de la figure 2.6 (page 13). Nous définissons tout d'abord la notion d'attribut, puis nous introduisons les arbres taxonomiques eux-mêmes.

Un attribut est un couple $\langle n, E\rangle$ où n est une chaîne de caractères (*nom* de l'attribut) et E un ensemble quelconque (*domaine* de l'attribut). Nous avons vu précédemment que l'on envisage généralement deux grands cas : soit E est un ensemble fini, soit E est un intervalle de $\mathbb{R}$. Cependant, il est bien entendu possible d'envisager d'autres situations. On note $\mathcal{A}$ l'ensemble de tous les attributs.

Un arbre taxonomique décrit une taxonomie des modalités. Chaque modalité est munie d'attributs tels que définis ci-dessus. De façon informelle, un (nœud d'un) arbre taxonomique est un triplet ⟨ nom de modalité, ensemble d'attributs, ensemble de sous-arbres ⟩. On note $\mathcal{AT}$ l'ensemble des *arbres taxonomiques*. Plus formellement, on utilise la définition récursive suivante :

Si $S = \{s_1, s_2, \ldots, s_m\} \subset \mathcal{AT}$ (S est l'ensemble des sous-arbres du nœud courant ; les s_i sont donc des arbres taxonomiques, étant entendu que l'ensemble S peut être vide),
si $n \in \mathbb{S}$ (*nom* du nœud courant[2]),
si $A = \{a_1, a_2, \ldots, a_p\} \subset \mathcal{A}$ (les a_i sont des attributs)
alors $\langle n, A, S\rangle$ est un arbre taxonomique.

[2]On note $\mathbb{S}$ l'ensemble des *chaînes de caractères*.

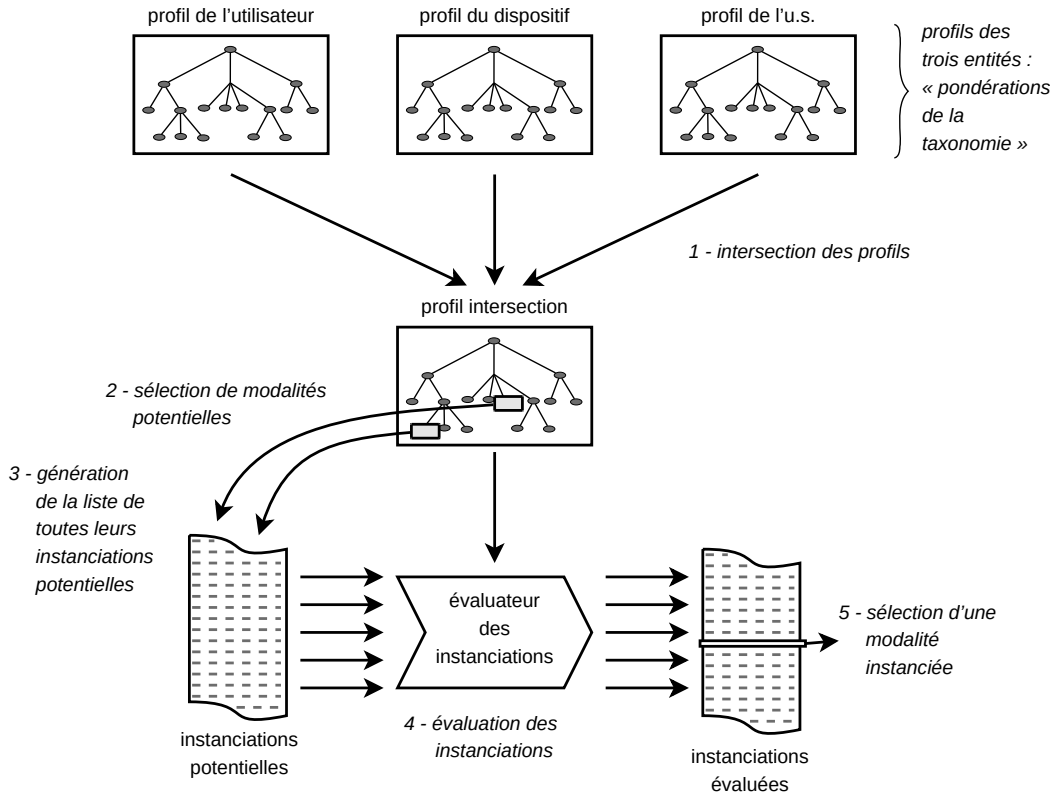


Figure 6.1 : *Vue d'ensemble de l'algorithme. On réalise tout d'abord l'intersection des profils de l'utilisateur, du dispositif de présentation et de l'unité sémantique. Ceci permet d'obtenir la liste des modalités utilisables. Pour ces modalités, on établit la liste des instanciations potentielles, qui sont alors évaluées par rapport au profil intersection. De cette façon, on peut choisir la « meilleure » modalité instanciée.*

La racine de cet arbre taxonomique a pour nom n. Ses attributs sont les a_i ; ses arbres fils sont les s_i. Si $S = \emptyset$ on dit que l'on a affaire à une *feuille*.

En notant $\mathbb{S}$ l'ensemble des chaînes de caractères, et pour un ensemble E quelconque donné, $\wp(E)$ l'ensemble des parties (sous-ensembles) de E, on a :

$$\mathcal{AT} = \mathbb{S} \times \wp(\mathcal{A}) \times \wp(\mathcal{AT})$$

6.2.3 Arbres de pondération

6.2.3.1 Introduction

Nous pouvons maintenant passer plus directement à l'expression des profils. D'après ce que nous avons expliqué plus haut, un profil indique les capacités et préférences d'une entité en termes de modalités. Pour ce faire, nous introduisons la notion d'*arbre de pondération*, dont le principe est le suivant : il s'agit d'ajouter des pondérations à un arbre taxonomique, qui

permettront d'exprimer les capacités, préférences et contraintes de l'entité décrite (utilisateur, dispositif de présentation ou unité sémantique).

Une pondération est un nombre réel compris entre 0 (inclus) et 1 (inclus également). Une pondération peut être située à deux endroits différents :

- au niveau d'un nœud : la pondération s'applique alors au sous-arbre ayant ce nœud pour racine. Un *poids* à 1 signifie que les modalités du sous-arbre sont acceptées, voire souhaitées, tandis qu'un poids à 0 signifie que les modalités correspondantes sont refusées, ou non prises en charge. Les valeurs intermédiaires permettent de nuancer ces deux extrêmes, et ainsi d'exprimer des *niveaux de préférence* ;

- au niveau d'un attribut : on donne alors une fonction[3] de pondération définie sur l'ensemble des valeurs possibles de cet attribut, et à valeurs dans $[0, 1]$. Cette fonction indique la pondération accordée à chaque valeur possible de l'attribut. La signification des pondérations est la même que précédemment. Ainsi, les valeurs de l'attribut proches de 1 seront souhaitées, tandis que les valeurs proches de 0 ne le seront pas, voire seront refusées pour une pondération à 0.

Un *profil* (tel qu'évoqué précédemment) est défini comme étant un arbre de pondération dont la racine correspond à la racine de la taxonomie des modalités, c'est-à-dire la modalité abstraite mère des trois modes visuel, auditif et TPK. Par opposition, les arbres de pondération dont la racine est située plus bas dans la taxonomie des modalités sont considérés comme des *arbres de pondération partiels*.

La figure 6.2 donne un exemple d'arbre de pondération, basé (pour des raisons de lisibilité) sur une taxonomie partielle. Cet arbre pourrait correspondre à un utilisateur malvoyant, qui préférerait donc largement les modes auditifs aux modes visuels : les pondérations correspondantes sont indiquées en blanc sur fond noir, à proximité des nœuds. Les fonctions de pondération sont indiquées (dans des rectangles arrondis) pour quelques attributs : selon que les attributs sont à variations continues ou à valeurs discrètes, les fonctions de pondération sont continues ou discrètes.

Cet arbre de pondération correspond à un utilisateur malvoyant. Ainsi, la modalité visuelle est pondérée (pondération de nœud) par 0,2, ce qui signifie que cette modalité est peu souhaitée par l'utilisateur. Par contre, la modalité auditive est pondérée par 1, ce qui signifie que ce dernière modalité est souhaitée. Les valeurs possibles du volume de la modalité auditive reçoivent un poids entre 0 et 1 (pondération d'attribut) : les valeurs inférieures à 20 dB sont pondérées par 0 (donc refusées), par contre, les valeurs entre 20 dB et 65 dB sont souhaitées, d'autant plus qu'elles sont élevées. Au-delà de 65 dB, les valeurs ne sont plus acceptées, car le volume trop élevé devient gênant.

[3] Une fonction mathématique.

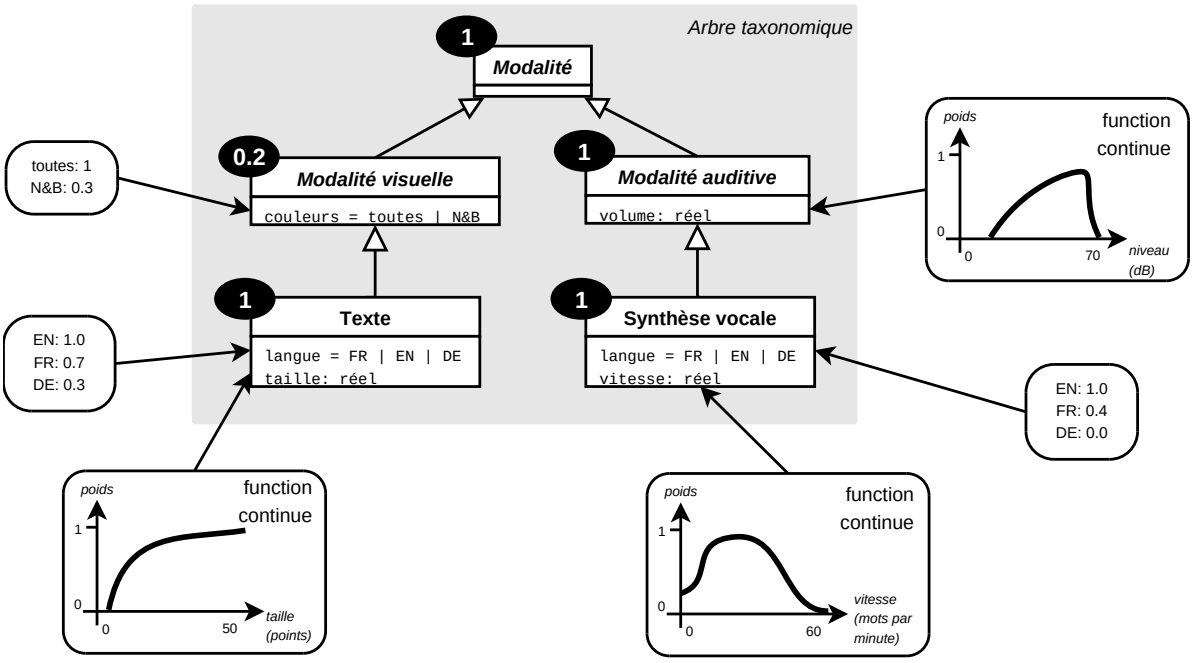


Figure 6.2 : *Exemple de profil (arbre de pondération).*

6.2.3.2 Interdépendance entre attributs

Dans ce qui précède, nous avons pondéré de façon indépendante chacun des attributs des nœuds. Nous allons voir que cela peut parfois s'avérer insuffisant.

Examinons l'extrait de taxonomie des modalités présenté sur la figure 6.3. Supposons que pour afficher une unité sémantique sur un écran, nous voulions déterminer les valeurs des attributs de la modalité *texte*. Ceux-ci sont au nombre de deux :

- *taille* : taille d'un caractère de texte, exprimée en centimètres. Pour des raisons de simplification, nous supposons que tous les caractères sont inscrits dans un carré dont le côté a pour longueur cet attribut *taille* ;
- *longueur* : longueur du texte, exprimée en nombre de caractères.

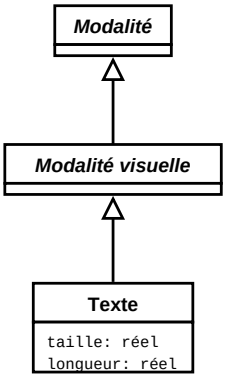


Figure 6.3 : *Exemple de taxonomie partielle dont certains attributs sont interdépendants.*

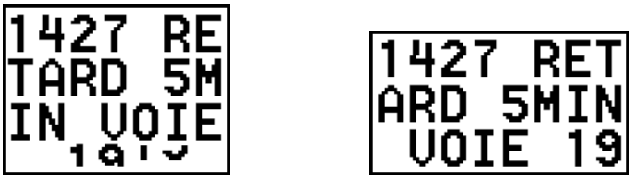
Nous constatons immédiatement que ces deux attributs ne sont pas indépendants. En effet, même en première approximation, ces deux attributs doivent respecter une contrainte où ils apparaissent tous les deux liés. Si on appelle *hauteur* et *largeur* respectivement la hauteur et la largeur de l'écran, il vient :

$$longueur \times (taille)^2 \leq hauteur \times largeur$$

Bien entendu, une telle contrainte n'est pas suffisante, car il faut de plus s'assurer que les caractères ne sont pas tronqués (voir fig. 6.4, a). On en vient donc à remplacer la contrainte ci-dessous par la contrainte suivante[4], qui permet de s'assurer que tous les caractères peuvent être logés dans le rectangle :

$$\left\lceil \frac{longueur}{\left\lfloor \frac{largeur}{taille} \right\rfloor} \right\rceil \times taille \leq hauteur$$

Cette contrainte est plus réaliste que la précédente, mais elle n'est toujours pas parfaite... En effet, si l'on est cette fois-ci assuré que tous les caractères entrent dans la zone de texte sans être tronqués, il est tout à fait possible que, par contre, des *mots* soient tronqués ! Sans précaution, le système peut être amené à choisir un nombre de lignes et de colonnes tel qu'il soit impossible de couper harmonieusement les mots, ce qui peut mener à de gros problèmes de compréhension de la part des utilisateurs (figure 6.4, b).



(a) (b)

Figure 6.4 : *On veut afficher le texte* « 1427 RETARD 5MIN VOIE 19 »*, soit 24 caractères. Chaque caractère est inscrit dans un carré de* 10×10*. Dans le cas (a), la zone de texte mesure* 70×35*. La première contrainte est respectée :* $70 \times 35 = 2450 \geq 2400 = 24 \times 10^2$*. Cependant, on est obligé de « découper » des caractères (ici, le 1 et le 9) pour les faire tenir dans la zone, d'où un résultat quelque peu illisible. La deuxième contrainte n'est pas vérifiée par l'exemple (a), mais elle l'est par l'exemple (b) où la zone est de taille* 80×30*. Dans ce cas, aucun* caractère *n'est tronqué, mais il reste un* mot *tronqué.*

Nous venons de voir qu'il existe des dépendances entre attributs : lorsqu'un attribut prend une valeur donnée, il peut arriver que les variations d'autres attributs soient limitées. En conséquence, il semble donc impossible de pondérer les valeurs de chaque attribut indépendamment des valeurs des autres. Supposons par exemple que l'on souhaite privilégier les

[4]En effet, le nombre maximum de caractères par ligne est $\left\lfloor \frac{largeur}{taille} \right\rfloor$. On en déduit de façon analogue le nombre de lignes nécessaires, on multiplie par la hauteur d'une ligne, et on vérifie que cette quantité est bien inférieure à la hauteur de la zone de texte.

grandes tailles de texte. Cela peut sembler une bonne idée (pour le confort de lecture des utilisateurs), mais cela ne peut pas se faire indépendamment de la prise en compte de l'attribut *longueur*. En effet, à quoi cela pourrait-il mener d'afficher d'énormes caractères, si cela devait conduire à limiter la taille du texte à deux ou trois de ces caractères, ce qui est généralement insuffisant pour faire passer un message élaboré ?

En conclusion, les fonctions de pondération ne doivent pas être données sur des *attributs isolés*, car ceux-ci ne sont pas forcément indépendants, mais sur des *groupes d'attributs liés*. Il reste à déterminer la façon dont ces groupes sont définis. Tout d'abord, nous notons qu'en un nœud donné, les attributs ne peuvent dépendre que d'attributs situés *plus haut* dans l'arbre, car les attributs situés sur l'une des branches inférieures n'existeront pas forcément lorsque l'une des modalités concrètes correspondantes sera choisie.

Après examen de la taxonomie des modalités relativement complète de la figure 2.6, il apparaît que les attributs interdépendants se situent en réalité *au sein d'un même nœud*. Dans la suite, nous définiront donc les fonctions de pondération *sur l'ensemble des attributs d'un nœud*, ce qui autorisera tout ou partie de ceux-ci à être dépendants les uns des autres.

6.2.3.3 Définition formelle

Après cette introduction informelle aux arbres de pondération, nous pouvons en donner une définition précise. Soit un arbre taxonomique donné T. Nous cherchons donc à définir les arbres de pondération associés à la taxonomie T. On note $\mathcal{AP}_T$ l'ensemble de ces arbres. Un arbre de pondération reprend la structure de T (même nœuds, mêmes sous-arbres), mais ajoute un *poids* et une *fonction de pondération* à chaque nœud.

Formellement, nous définissons un arbre de pondération de T par une fonction φ, définie sur l'ensemble des sous-arbres de T, qui associe un sous-arbre de pondération à chaque sous-arbre de T. Un sous-arbre (ou nœud) de pondération est un quadruplet $\langle$ nom de modalité, poids, fonction de pondération sur les attributs de cette modalité, sous-arbres de pondération $\rangle$. Comme expliqué précédemment, la fonction de pondération sur les attributs permet d'associer un niveau de satisfaction (compris entre 0 et 1) à chaque combinaison possible des attributs. Une valeur proche de 1 indiquera que la combinaison est satisfaisante, donc souhaitée, tandis qu'une valeur proche de 0 indiquera que la combinaison est peu satisfaisante, donc non souhaitée.

φ est de la forme suivante :

$$\varphi\left(\langle n, \{\langle n_1, E_1\rangle, \ldots, \langle n_m, E_m\rangle\}, \{s_1, \ldots s_p\}\rangle\right) = \langle n, x, f, \{\varphi(s_1), \ldots, \varphi(s_p)\}\rangle$$

où $x \in [0,1]$ (*poids*), et f est une fonction de $E_1 \times \cdots \times E_m$ dans $[0,1]$ (*fonction de pondération* sur les attributs).

En d'autres termes, une pondératrice φ associe à chaque nœud de T un poids x, ainsi qu'une fonction de pondération f, définie sur les attributs. L'ensemble des fonctions de pondération

sur un ensemble E (E étant égal au produit cartésien d'un certain nombre de domaines d'attributs) est noté $\mathcal{FP}_E$.

6.2.3.4 Intersection d'arbres de pondération

Nous disposons maintenant du formalisme nécessaire pour définir l'*intersection* des arbres de pondération (et donc des profils), telle qu'elle a été évoquée à la fin de la section 6.2.1.

Nous proposons tout simplement d'effectuer le produit des poids et des fonctions de pondération qui se correspondent. De cette façon, les pondérations défavorables se retrouveront sur l'intersection (notamment en raison du caractère absorbant de 0 pour la multiplication), tandis que les pondérations neutres (i.e. proches de 1) ne l'affecteront pas.

Formellement, soit T une taxonomie fixée, et A_1 et A_2 deux éléments de $\mathcal{AP}_T$. Il existe donc une fonction φ_1 telle que $A_1 = \varphi_1(T)$, et une fonction φ_2 telle que $A_2 = \varphi_2(T)$.

On définit alors une nouvelle pondératrice ψ sur l'ensemble des sous-arbres (au sens large) de T de la façon suivante :

Soit N un nœud de T : $N = \langle n, \{\langle n_1, E_1\rangle, \ldots, \langle n_m, E_m\rangle\}, \{s_1, \ldots s_p\}\rangle$,
sa pondération dans A_1 : $\varphi_1(N) = \langle n, x_1, f_1, \{\varphi_1(s_1), \ldots \varphi_1(s_p)\}\rangle$,
et sa pondération dans A_2 : $\varphi_2(N) = \langle n, x_2, f_2, \{\varphi_2(s_1), \ldots \varphi_2(s_p)\}\rangle$.

Alors on pose :

$$\psi(N) = \langle n, x_1 \cdot x_2, f_1 \cdot f_2, \{\psi(s_1), \ldots, \psi(s_p)\}\rangle$$

Donc $\psi(T) \in \mathcal{AP}_T$: c'est un arbre de pondération. On alors dit que l'arbre de pondération $B = \psi(T)$ est l'*intersection* des arbres de pondération A_1 et A_2. On note :

$$B = A_1 \sqcap A_2$$

$$\psi(T) = \varphi_1(T) \sqcap \varphi_2(T)$$

Par extension de l'écriture, on note aussi : $\psi = \varphi_1 \sqcap \varphi_2$.

Sur ce qui précède, il est clair que la loi $\sqcap$ est associative et commutative (elle hérite ces propriétés de la loi de multiplication dans $[0, 1]$, à partir de laquelle elle est définie).

6.2.4 Évaluation d'un arbre de pondération

Voici où nous en sommes dans le cheminement global :

1. nous avons vu que la prise en compte des diverses contraintes (introduites par le dispositif de présentation, l'utilisateur, et l'unité sémantique) conduisait à l'obtention de trois arbres de pondération ;

2. il est alors possible de réaliser l'intersection de ces trois arbres (au sens de la loi $\sqcap$), afin d'obtenir un nouvel arbre de pondération : nous venons de voir les règles de calcul correspondantes. Cet arbre représente les modalités et les valeurs d'attributs compatibles avec chacun des intervenants ;

3. il reste alors à utiliser cet arbre afin de déterminer *in fine* quelle est la modalité à utiliser. La notion d'*évaluation*, exposée dans cette section, permet de répondre à cette question.

Les modalités concrètes sont les feuilles des arbres (taxonomiques aussi bien que de pondération). Nous proposons donc de donner une évaluation (un poids) à chaque feuille de l'arbre de pondération, et choisir la feuille qui aura le meilleur poids.

Schématiquement, pour chaque feuille on suit le chemin qui mène de la racine de l'arbre à cette feuille, et on calcule un poids qui dépend des pondérations des nœuds rencontrés sur le chemin. Pour le calcul de ce poids, il faut :

- que les feuilles situées sur un sous-arbre qui possède un nœud parent dont la pondération est 0 aient un poids de 0 (caractère absorbant de 0), et qu'une pondération à 1 n'influe pas négativement le résultat (caractère neutre de 1) : on a donc envie de faire le *produit* des pondérations le long des chemins de l'arbre ;

- que la longueur du chemin n'influe cependant pas sur le calcul : on peut alors songer à faire non pas le produit, mais la moyenne géométrique des pondérations. En effet, si on n'y prenait pas garde, les produits ayant plus de facteurs sur les chemins les plus longs, le poids correspondant aurait d'autant plus de chance d'être faible. On constaterait donc *statistiquement* un déséquilibre de poids entre les chemins les plus longs et les chemins les plus courts ;

- que les deux points précédents s'appliquent aussi bien aux fonctions de pondération qu'aux poids.

Ainsi donc, si sur un chemin, et pour un choix donné d'attributs, on a q nœuds de pondérations $x_1, \ldots, x_q$ et de fonctions de pondération $f_1, \ldots, f_q$, on donnera à ce chemin (et donc à la feuille correspondante) un poids du type :

$$w = \sqrt[q]{f_1 \times \cdots \times f_q} \times \sqrt[q]{x_1 \times \cdots \times x_q} = (f_1 \times \cdots \times f_q)^{\frac{1}{q}} \times (x_1 \times \cdots \times x_q)^{\frac{1}{q}}$$

Ceci est très schématique : s'il est effectivement possible d'effectuer le produit des pondérations des nœuds (ce sont des réels compris entre 0 et 1), il n'est en revanche pas possible d'effectuer aussi simplement le produit des fonctions de pondération. En effet, ces fonctions

de pondération sont potentiellement définies sur des ensembles complètement différents. Leur produit n'est donc pas défini.

Afin de lever ce problème, définissons le produit de deux fonctions de pondération. Soient donc deux fonctions de pondération f et g ainsi définies :

- f est une fonction de pondération sur un ensemble F : $f : F \longrightarrow [0,1]$. $f \in \mathcal{FP}_F$;
- g est une fonction de pondération sur un ensemble G : $g : G \longrightarrow [0,1]$. $g \in \mathcal{FP}_G$.

On définit alors la fonction h suivante :

$$\begin{array}{rcl} h : F \times G & \longrightarrow & [0,1] \\ (\vec{y}, \vec{z}) & \longmapsto & f(\vec{y}) \cdot g(\vec{z}) \end{array}$$

On note $h = f * g$. On définit ainsi la loi $*$ (*« produit »*) sur les fonctions de pondération. $f * g$ est une fonction de pondération sur l'ensemble $F \times G$.

La loi $*$ est associative (en raison de l'associativité de la multiplication des réels sur $[0,1]$ et de celle du produit cartésien des ensembles), mais elle n'est pas commutative (car pour deux ensembles F et G, $F \times G \neq G \times F$).

Ainsi, de façon précise, le « poids » donné à la feuille évoquée plus haut *est une fonction de pondération* ainsi définie :

$$w = (x_1 \times x_2 \times \cdots \times x_q)^{\frac{1}{q}} \cdot (f_1 * f_2 * \cdots * f_q)^{\frac{1}{q}}$$

Ou encore :

$$w(\vec{y_1}, \vec{y_2}, \ldots, \vec{y_q}) = (x_1 \times x_2 \times \cdots \times x_q)^{\frac{1}{q}} \cdot (f_1(\vec{y_1}) \cdot f_2(\vec{y_2}) \cdot \cdots \cdot f_q(\vec{y_q}))^{\frac{1}{q}}$$

Les $\vec{y_i}$ sont des combinaisons d'attributs au niveau des nœuds. Le vecteur $\langle \vec{y_1}, \vec{y_2}, \ldots, \vec{y_q} \rangle$ correspond donc à une combinaison de *tous* les attributs relatifs à la modalité considérée. En ce sens, on peut qualifier cette combinaison de *totale*, par rapport aux précédentes, que l'on peut considérer comme étant *partielles*.

Le poids w est donc une fonction sur E, produit cartésien $E_1 \times E_2 \times \cdots \times E_q$ des domaines de définition des fonctions de pondération des différents nœuds. Notons que E est également un produit cartésien (dans un ordre bien précis) de tous les domaines des attributs de la modalité considérée. La fonction w est à valeurs dans l'intervalle $[0,1]$. Plus le poids est élevé (i.e. proche de 1), plus l'instance de modalité correspondante est jugée *« intéressante »*.

La figure 6.5 résume les processus d'intersection et d'évaluation d'arbres de pondération. Pour des raisons de simplicité, nous n'avons pas représenté les attributs. En pratique, ceux-

ci recevraient des fonctions de pondération, et donc au final, les poids des feuilles seraient fonction des valeurs des attributs.

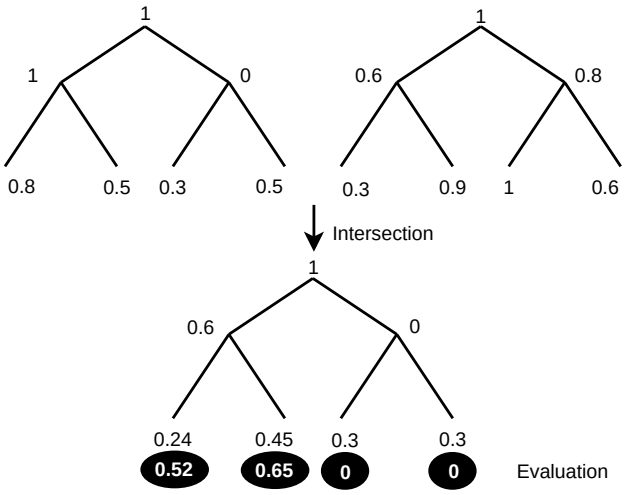


Figure 6.5 : *Intersection et évaluation d'arbres de pondération*

6.2.5 Instanciation

Grâce à la méthode décrite ci-dessus, on obtient donc un ensemble de feuilles (i.e. de modalités) « évaluées ». L'évaluation d'une modalité donnée est une fonction w définie sur l'ensemble de ses attributs.

D'emblée, il se peut que toutes ces évaluations soient nulles, ce qui est le cas si les profils sont incompatibles (par exemple, utilisateur non-voyant qui s'approche d'un écran). Dans ce cas, il n'est pas possible d'effectuer la présentation de l'unité sémantique.

Dans la suite, nous nous intéressons au cas où certaines de ces évaluations sont non nulles. Plus particulièrement, nous allons supposer dans la suite qu'une seule évaluation est non nulle : ceci ne réduit pas la généralité du raisonnement, car si un dispositif de présentation donné présente plusieurs modalités (par exemple un ensemble écran + haut-parleur), il est toujours possible de le scinder en plusieurs dispositifs logiques ne disposant chacun que d'une seule modalité utilisable. Notons qu'alors, il est possible d'interpréter la procédure d'intersection ainsi : il s'agit de déterminer si *la* modalité du dispositif de présentation est compatible à la fois avec l'utilisateur et l'unité sémantique.

Cette réflexion en amène une autre : *l'instanciation s'effectue du point de vue du dispositif.* En effet, c'est le dispositif qui va effectuer une présentation de l'unité sémantique en utilisant les valeurs d'attributs issues de l'instanciation. Ces valeurs doivent donc tenir compte de *toutes* les unités sémantiques présentées par le dispositif, et donc de *tous* les utilisateurs concernés. Par exemple, sur un écran qui présenterait trois unités sémantiques, il n'est pas concevable que l'une d'entre elles occupe 90 % de la place disponible, et laisse seulement 10 % de l'espace aux deux autres, les rendant par là même illisibles.

Il est donc nécessaire de réaliser un compromis entre les attributs de toutes les u.s. présentées sur le dispositif, en fonction des desiderata des utilisateurs correspondant à chacune d'entre elles. Les paragraphes suivants donnent une méthode afin d'obtenir un compromis raisonnable.

6.2.5.1 Espace des combinaisons possibles d'attributs

Soit donc un dispositif donné, qui permet la présentation d'unités sémantiques selon *une* modalité donnée. Cette modalité est caractérisée par un certain nombre d'attributs, dont les domaines de variation sont notés $E_1, \ldots, E_p$. On note $\Omega = E_1 \times \cdots \times E_p$. Ces attributs sont tous les attributs présents dans l'arbre taxonomique sur le chemin menant de la racine à la feuille correspondant à la modalité considérée.

Sur ce dispositif, on souhaite présenter n unités sémantiques $u_1, \ldots, u_n$. Pour chacune de ces n u.s., il faut donc déterminer une *instanciation* de la modalité, c'est-à-dire une valeur pour chacun des p attributs. Autrement dit, pour chaque u.s., on doit choisir un élément $\omega \in \Omega$. Au final, cela revient à choisir $n \cdot p$ valeurs (p valeurs d'attributs pour chacune des n u.s.), c'est-à-dire un élément π de $\Pi = \Omega^n$.

Nous supposons que le dispositif présente des contraintes, notamment des contraintes de place. On introduit donc une *fonction de contraintes*, z, ainsi définie :

$$\begin{array}{rcl} z : \Pi & \longrightarrow & \{0,1\} \\ \pi & \longmapsto & \begin{cases} 1 & \text{si } \pi \text{ est une combinaison d'attributs compatible avec le dispositif} \\ 0 & \text{sinon} \end{cases} \end{array}$$

Autrement dit, pour un choix π des attributs des u.s., $z(\pi)$ vaut 1 si, et seulement si, il est possible de présenter les u.s. sur le dispositif avec ce choix d'attributs.

Par exemple, sur un écran, il n'est pas possible d'affecter une surface arbitraire à chaque unité sémantique, car la somme des surfaces individuellement affectées à chaque u.s. doit être inférieure à la surface de l'écran S. Dans ce cas, la fonction z vaudrait 1 pour les combinaisons d'attributs telles que la surface occupée par l'ensemble des u.s. soit inférieure à S, et 0 sinon.

On introduit l'ensemble $\widetilde{\Pi}$ des éléments de Π qui satisfont les contraintes du dispositif : $\widetilde{\Pi} = \{\pi \in \Pi | z(\pi) = 1\}$. $\widetilde{\Pi} \subset \Pi$.

6.2.5.2 Évaluation des combinaisons d'attributs

Introduction informelle On cherche alors à *classer* les combinaisons possibles d'attributs, de façon à déterminer quel est *globalement* la combinaison la *plus satisfaisante* pour l'ensemble des utilisateurs concernés. Nous introduisons ce classement en deux temps :

- tout d'abord, dans cette section, nous expliquons comment associer à chaque combinaison d'attributs (i.e. à chaque élément de $\widetilde{\Pi}$) une mesure de la satisfaction des utilisateurs. Nous appelons cette mesure *évaluation* ;
- ensuite, dans la section suivante, nous expliquerons comment utiliser ces évaluations pour déterminer quelle combinaison d'attributs est globalement la plus satisfaisante, de façon à faire un choix. Notamment, nous définirons précisément ce que nous entendons par la notion de *« meilleure satisfaction globale »*.

Pour commencer, voyons de façon informelle comment sont définies les évaluations des choix d'attributs. À un choix π d'attributs, les utilisateurs concernés associent chacun une évaluation, i.e. un « degré de satisfaction », calculé à l'aide des fonctions de pondération. Ces évaluations (individuelles) sont des nombres compris entre 0 et 1. L'évaluation (globale) de π est simplement définie comme étant le vecteur de ces évaluations individuelles classées par ordre croissant[5].

Exemple Supposons qu'un dispositif doive présenter trois u.s. (appelées u_1, u_2 et u_3) pour trois utilisateurs A, B, et C. L'espace des combinaisons d'attributs est (dans ce cas) composé de quintuplets de la forme $\langle$ langue u_1, taille u_1, langue u_2, longueur u_2, durée u_3 $\rangle$. Dans cet espace, nous considérons la combinaison de valeurs d'attributs suivante :

$$\pi = \langle \texttt{fr},\ 4,\ \texttt{de},\ 6,\ 7 \rangle$$

Les évaluations données individuellement par chaque utilisateur sont les suivantes :

- A évalue π à 0,7 ;
- B évalue π à 0,8 ;
- C évalue π à 0,6 ;

L'évaluation globale est le vecteur de ces trois évaluations individuelles, classées par ordre croissant, c'est-à-dire : $\langle$0,6, 0,7, 0,8$\rangle$. Cet exemple sera ultérieurement repris (et poursuivi) dans le tableau 6.1 (p. 131).

Formalisation Définissons maintenant formellement cette notion d'évaluation d'un élément de $\widetilde{\Pi}$. On suppose qu'il y a m utilisateurs. Il n'y a aucune raison pour que m soit égal à n, car un utilisateur peut être intéressé par plusieurs u.s., et réciproquement, plusieurs utilisateurs peuvent être intéressés par la même u.s. On suppose que l'utilisateur numéro i est intéressé par k u.s., d'indices j_1 à j_k. À un choix π des attributs, et à une u.s. d'indice j_ℓ ($\ell \in [1,k]$) l'utilisateur numéro i associe donc une évaluation notée $w_{i,\ell}(\pi)$ (voir 6.2.5.4).

[5] L'intérêt de ce classement par ordre croissant apparaîtra lorsqu'il s'agira de faire un choix (voir section 6.2.5.3).

Bien entendu, le nombre de ces évaluations dépend de l'utilisateur, mais chacun en fournit au moins une[6].

Pour l'ensemble des utilisateurs, on obtient donc au total q évaluations, avec $q \geq m$. Le cas où $q = m$ est obtenu lorsque chaque utilisateur est intéressé par exactement une u.s. Le cas où $q = n$ est obtenu lorsque chaque u.s. intéresse exactement un utilisateur.

On définit alors une fonction d'évaluation e sur $\widetilde{\Pi}$ comme suit : pour un élément π, on obtient exactement q évaluations d'utilisateurs. On les classe par ordre croissant, et on en forme ainsi un q-uplet de la forme $\langle w_1, w_2, \ldots, w_q \rangle$ avec $w_1 \leq w_2 \leq \cdots \leq w_q$. Ce q-uplet est noté $e(\pi)$. On dit que c'est l'*évaluation* de la combinaison d'attributs π. Ce q-uplet commence par l'évaluation donnée par l'utilisateur le moins bien loti, et se termine par celle de l'utilisateur le mieux loti.

6.2.5.3 Choix d'une combinaison d'attributs

Il s'agit maintenant de déterminer quelle est la *« meilleure »* combinaison d'attributs. Comment la définir ?

Le but du système est de donner un confort maximal à un maximum d'utilisateurs. Bien entendu, dès l'instant où il y a plusieurs utilisateurs, le niveau de satisfaction atteint ne pourra pas être le même pour tout le monde : nous venons de voir que certains seront mieux lotis que d'autres. Il s'agit donc de trouver un compromis entre les désirs des uns et des autres.

Examinons un exemple de façon détaillée. On suppose que deux utilisateurs souhaitent que leurs u.s. respectives soient affichées sur un même écran, sous forme textuelle. Ils souhaitent tous les deux que la taille des caractères soit la plus grande possible. Les tailles peuvent varier entre 0 et 10, mais la surface disponible à l'écran impose que la somme des tailles soit inférieure à 12. En conséquence, le poids que les deux utilisateurs donnent à l'attribut `taille` est proportionnel à cette dernière : il vaut 0 pour un texte en taille 0, et 1 pour un texte en taille 10. Sur la figure 6.6a, on a fait varier en abscisse la taille allouée à l'u.s. du premier utilisateur, appelée `t1`. On a alors tracé, en fonction de `t1`, la « meilleure » taille allouable à l'u.s. du deuxième utilisateur[7], notée `t2`.

Sur la figure 6.6b, on a tracé le maximum des deux satisfactions, c'est-à-dire le poids accordé aux choix de la taille par l'utilisateur le plus favorisé. Selon ce critère, il faudrait soit choisir `t1` entre 0 et 2 (et donc `t2` = 10), ou bien `t1` = 10 et `t2` = 0. Certes, ces solutions donnent entière satisfaction à l'un des deux utilisateurs, mais l'autre se trouve « sacrifié » : dans certains cas, son u.s. n'est même pas lisible !

Pour éviter ce problème, on pourrait imaginer, pour tenir compte de tous les utilisateurs, de maximiser la *moyenne* des évaluations : « en moyenne », tous devraient y trouver leur

[6] Si un utilisateur ne fournissait pas d'évaluation, cela voudrait dire qu'il n'est intéressé par aucune u.s. de l'écran, donc il n'aurait rien à faire dans la liste des utilisateurs considérés.

[7] Selon les critères de ce dernier, c'est-à-dire la plus grande.

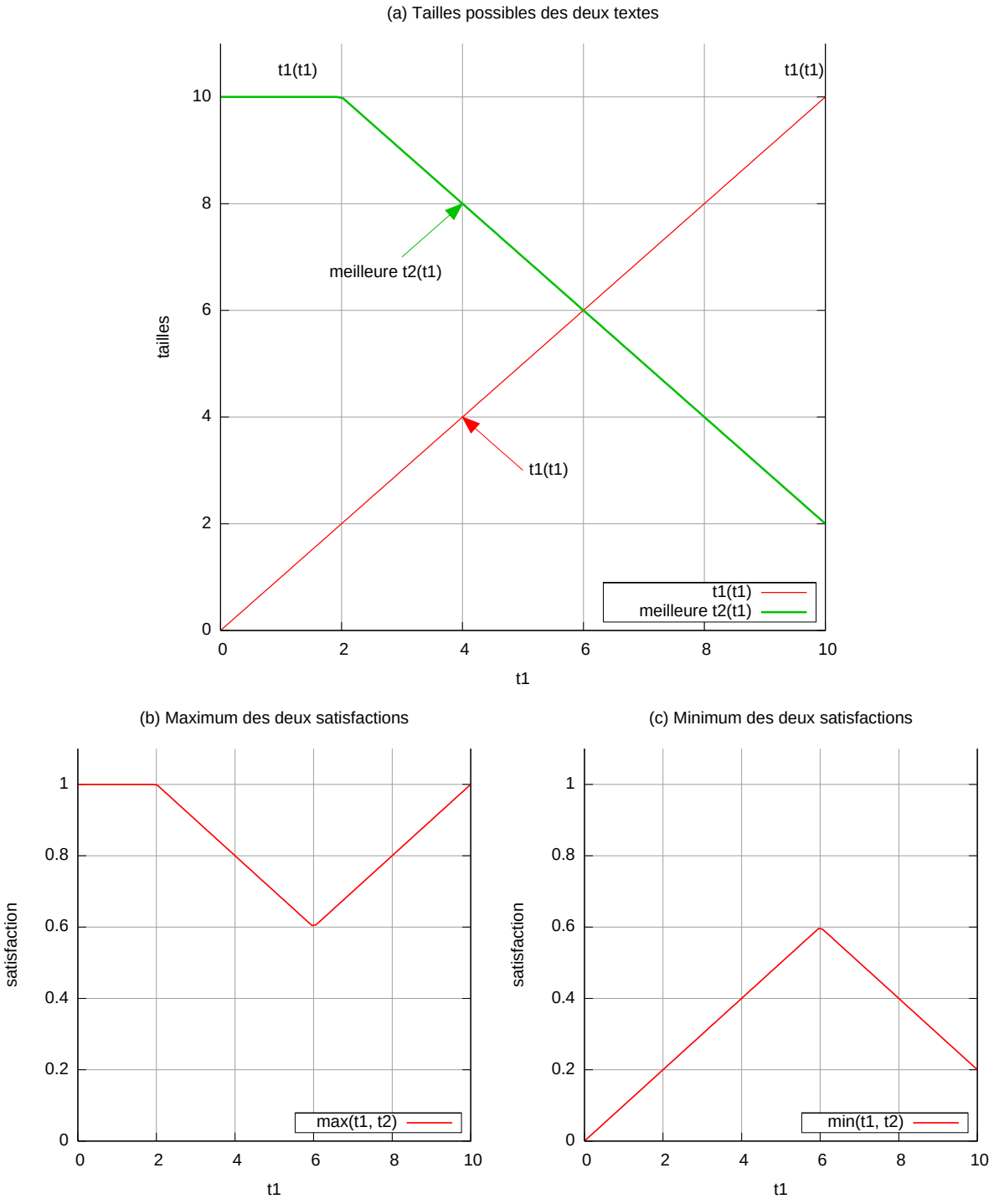


Figure 6.6 : *Tailles respectives de deux textes affichés sur un même écran, et préférences correspondantes des utilisateurs. Le graphe (a) donne la meilleure taille* `t2` *utilisable pour chaque valeur possible de* `t1`*. Les graphes (b) et (c) présentent respectivement le maximum et le minimum des préférences des deux utilisateurs, en fonction de la valeur de* `t1`*.*

compte. Cependant, dès que le nombre d'utilisateurs croît, le poids relatif de l'un d'entre eux devient insignifiant. Dès lors, il est tout à fait possible qu'une *bonne solution en moyenne* soit tout à fait catastrophique pour l'un des utilisateurs.

Il nous semble donc important d'assurer un niveau de satisfaction *minimum* pour tout le monde. Si on s'assure que le niveau de satisfaction *atteint par l'utilisateur le moins bien loti* dépasse une valeur minimale, alors on est sûr que personne ne peut être sacrifié au profit des autres. Cette propriété est intéressante, car notre but est de fournir un système qui soit utile à *tous* ses utilisateurs, pas un système qui en privilégie certains au détriment des autres. Sur la figure 6.6c, on a tracé le minimum des évaluations des deux utilisateurs. Si l'on cherche à maximiser ce minimum, on trouve que la « meilleure » solution est obtenue pour `t1` $= 6$, et donc `t2` $= 6$. De cette façon, chacun des deux utilisateurs obtient au moins un niveau de satisfaction minimal. La solution semble bien meilleure que celle obtenue par maximisation des maxima.

Comme les $e(\pi)$ sont des q-uplets de préférences classées par ordre croissant, le raisonnement qui précède nous pousse donc à choisir les q-uplets dont le *premier élément* est maximal (i.e. pour lesquels la préférence la moins bonne est maximale). De la sorte, on est assuré que toutes les préférences vaudront *au moins* la valeur de ce premier élément. Si *plusieurs* combinaisons π sont évaluées en des q-uplets de premier élément égal, on pourra alors les départager selon leur deuxième élément, et ainsi de suite.

Ceci revient à choisir la combinaison π d'attributs telle que $e(\pi)$ soit maximal *au sens de l'ordre lexicographique* (voir section 6.2.5.5).

Par exemple, supposons qu'un dispositif doive présenter trois u.s. pour trois utilisateurs A, B et C. Le système doit déterminer les valeurs de cinq attributs, étant données les évaluations de trois utilisateurs. On dispose des données du tableau 6.1.

π – **Valeurs**	e_A	e_B	e_C	$e(\pi)$ – **Évaluation**
⟨`fr`, 4, `de`, 6, 7⟩	0,7	0,8	0,6	⟨0,6, 0,7, 0,8⟩
⟨`it`, 2, `en`, 9, 1⟩	0,9	0,3	0,7	⟨0,3, 0,7, 0,9⟩
⟨`en`, 2, `de`, 3, 5⟩	0,8	0,7	0,9	⟨0,7, 0,8, 0,9⟩
⟨`es`, 8, `fr`, 1, 3⟩	0,6	0,9	0,5	⟨0,5, 0,6, 0,9⟩
⟨`de`, 3, `es`, 7, 5⟩	0,2	0,4	0,95	⟨0,2, 0,4, 0,95⟩

Tableau 6.1 : *Classement des évaluations.*

La première colonne contient les combinaisons de valeurs d'attributs. Les trois colonnes suivantes contiennent les évaluations d'utilisateurs correspondantes, et la dernière colonne le vecteur d'évaluation global, composé des valeurs des trois colonnes précédentes classées par ordre croissant. La solution choisie est la troisième, parce qu'elle maximise le taux de satisfaction de l'utilisateur le moins satisfait. Ainsi, toutes les évaluations d'utilisateurs valent au moins 0,7.

Il peut arriver que le maximum (au sens de l'ordre lexicographique) soit obtenu pour *plusieurs* valeurs de π. Dans ce cas, on garde toutes ces valeurs potentielles, et on choisit au final celle pour laquelle le *coût* (voir section 6.3) est le plus faible.

6.2.5.4 Précision sur les évaluations $w_{i,\ell}(\pi)$

Le calcul des $w_{i,\ell}(\pi)$ mérite quelques précisions. Nous donnons ici une idée de ce « calcul », sans la formaliser à outrance.

On a vu que π est un vecteur à $n \cdot p$ coordonnées. On extrait alors de π les p attributs correspondant à l'u.s. concernée par $w_{i,\ell}$ (*projection*). On obtient donc un p-uplet, auquel on peut appliquer la fonction d'évaluation de l'utilisateur numéro i. Le résultat de cette application est $w_{i,\ell}(\pi)$.

De façon simple, on peut dire que la fonction w de chaque utilisateur s'applique, pour chaque unité sémantique, aux p attributs correspondant à cette u.s. Cette fonction w est alors prolongée en une fonction $w_{i,\ell}$ à $n \cdot p$ variables, les $(n-1) \cdot p$ variables supplémentaires n'intervenant tout simplement pas dans son calcul.

Exemple Supposons qu'on évalue sur un dispositif la présentation de trois u.s. sur un dispositif, numérotées de 1 à 3. La modalité en question possède deux attributs, `lang` et `taille`. L'ensemble Π des combinaisons d'attributs possède donc six coordonnées, que nous noterons `lang1`, `taille1` (langue et taille de l'u.s. 1), `lang2`, `taille2` (langue et taille de l'u.s. 2), et enfin `lang3` et `taille3` (langue et taille de l'u.s. 3).

Trois utilisateurs, numérotés de a à c, sont intéressés par les u.s. : a par 1, b par 2, et c par 1 et 3. Chacun d'entre eux donne donc une fonction d'évaluation sur les attributs de chacune des u.s. par laquelle il est intéressé, ce qui est représenté sur le tableau 6.2.

Utilisateur	u.s. n°1 { `lang1`, `taille1` }	u.s. n°2 { `lang2`, `taille2` }	u.s. n°3 { `lang3`, `taille3` }
a	$w_{a,1}(\texttt{lang1}, \texttt{taille1})$	—	—
b	—	$w_{b,1}(\texttt{lang2}, \texttt{taille2})$	—
c	$w_{c,1}(\texttt{lang1}, \texttt{taille1})$	—	$w_{c,2}(\texttt{lang3}, \texttt{taille3})$

Tableau 6.2 : *Relation entre fonctions d'évaluation et espace des combinaisons d'attributs.*

Ces quatre fonctions se prolongent de façon naturelle sur Π. Par exemple :

$$w_{b,1}(\texttt{lang1}, \texttt{taille1}, \texttt{lang2}, \texttt{taille2}, \texttt{lang3}, \texttt{taille3}) = w_{b,1}(\texttt{lang2}, \texttt{taille2})$$

6.2.5.5 Précisions sur l'ordre lexicographique

On travaille ici sur des t-uplets de réels de l'intervalle [0,1]. Entre de tels t-uplets, on définit une relation d'ordre strict notée $\prec_t$ par :

$$\begin{aligned}&\langle a_1, a_2, \ldots, a_t\rangle \prec_t \langle b_1, b_2, \ldots, b_t\rangle \text{ ssi}\\ &\quad a_1 < b_1\\ &\quad \text{ou}\\ &\quad (t > 1) \quad \text{et} \quad (a_1 = b_1) \quad \text{et} \quad (\langle a_2, \ldots, a_t\rangle \prec_{t-1} \langle b_2, \ldots, b_t\rangle)\end{aligned}$$

On appelle la relation $\prec_t$ *ordre t-lexicographique*, ou tout simplement *ordre lexicographique* car si l'on compare des t-uplets, l'ordre ne peut être autre que t-lexicographique.

Remarque : l'ordre 1-lexicographique est l'ordre naturel strict sur [0,1].

6.2.6 Compléments

6.2.6.1 Multiples contenus pour une même modalité

Pour une unité sémantique donnée, il peut arriver que l'on puisse générer plusieurs contenus différents pour la même modalité. Par exemple, soient l'u.s. *« heure et lieu de départ du train n°5634 »*, et la modalité textuelle, avec l'attribut `langue` fixé (Français). On peut envisager les contenus suivants :

1. `5634 Dép 10h32 V 09` ;
2. `Train n°5634 : départ 10h32, voie 09.`

Ces deux contenus diffèrent par la longueur de leur texte (19 caractères dans le premier cas, 36 dans le deuxième).

Comment prendre en compte cette situation dans le modèle ?

Il serait possible de complexifier le modèle, et de permettre à une même modalité d'apparaître plusieurs fois dans la liste des modalités possibles pour une u.s. Cette approche nous semble assez compliquée, et nous proposons donc une solution beaucoup plus simple.

Nous proposons que les différents contenus soient distinguées par des valeurs différentes données à certains des attributs de la modalité. En fonction des valeurs choisies pour ces attributs lors de la phase d'instanciation, l'un ou l'autre des contenus sera généré.

Ainsi, dans l'exemple ci-dessus, la longueur est un attribut de la modalité textuelle. Sa valeur devra donc être fixée lors de la phase d'instanciation. Dans l'arbre de pondération associé à l'unité sémantique, il sera indiqué que l'u.s. sait générer deux longueurs de texte différentes,

à l'aide d'une fonction de pondération toujours nulle, sauf pour les deux valeurs utilisables (voir figure 6.7).

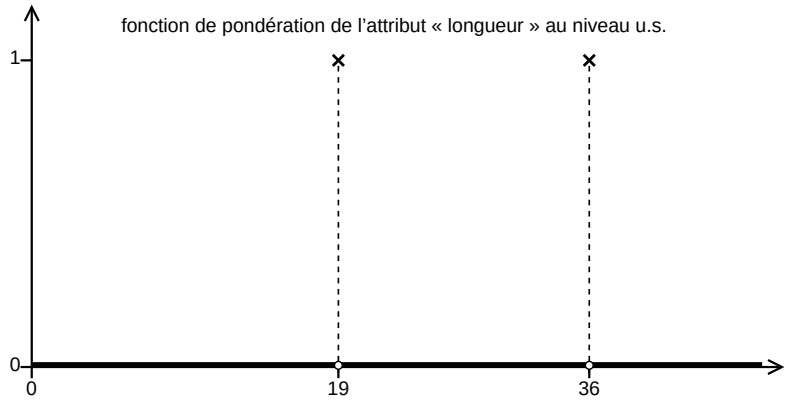


Figure 6.7 : *Au niveau de la modalité textuelle de l'u.s., la fonction de pondération de l'attribut* longueur *vaut zéro tout le temps, sauf aux deux longueurs possibles, où elle vaut 1.*

Ensuite, on effectuera le produit de cette fonction de pondération avec, entre autres, la fonction de pondération correspondante du dispositif de présentation. Imaginons par exemple que ce dernier dispose d'un afficheur de 32 caractères, et tente de réserver quelques caractères pour l'affichage de l'heure. La fonction de pondération correspondante est présentée sur la figure 6.8.

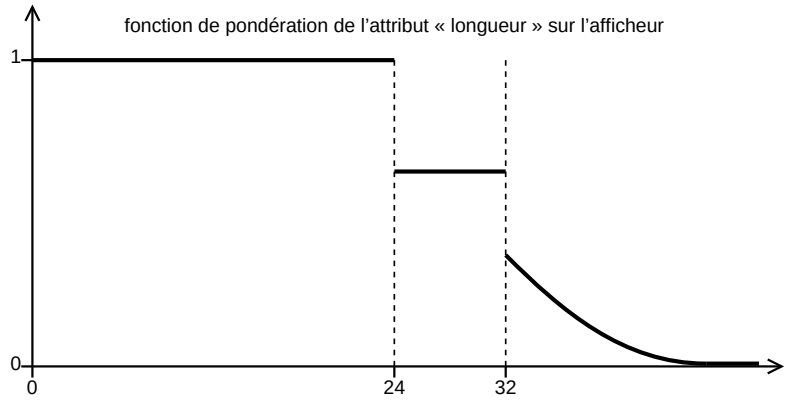


Figure 6.8 : *Au niveau de l'afficheur, l'attribut* taille *est pondéré par 1 pour les petites valeurs (l'affichage ne pose aucun problème). La pondération décroît ensuite à 0,6 car de telles tailles empêchent l'afficheur de donner l'heure. Au-delà de 32, la pondération tombe rapidement à 0 car l'afficheur est limité à 32 caractères. Cependant, la chute n'est pas immédiate, car l'afficheur est capable de faire défiler un texte qui serait trop long. Cependant, lors des défilements, la gêne est telle pour les utilisateurs que les pondérations correspondantes sont très faibles.*

Lorsqu'on fait le produit des deux fonctions de pondération précédentes, on obtient la fonction de la figure 6.9. Il est important de noter que *les seules valeurs de la taille pour lesquelles la pondération est non nulle correspondent aux tailles auxquelles l'u.s. sait générer un contenu.*

En conséquence, le code de génération de contenu de l'unité sémantique consistera en un branchement en fonction de la taille choisie lors de l'instanciation.

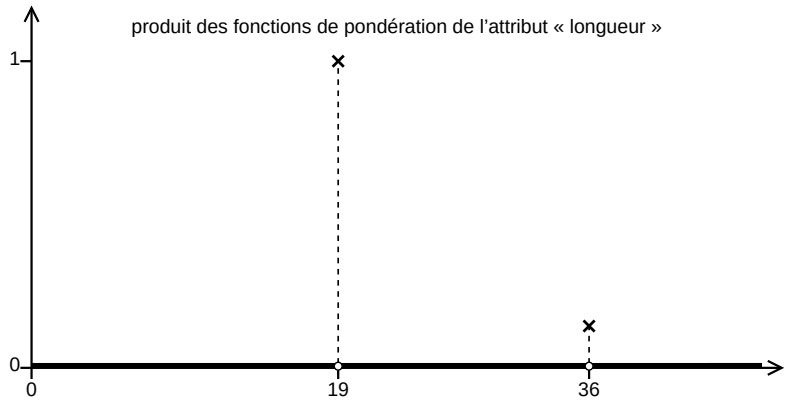


Figure 6.9 : *La fonction de pondération produit ne possède (éventuellement) des valeurs non nulles qu'aux endroits où la fonction de pondération de l'u.s. est elle-même non nulle.*

Dans l'exemple, il est encore possible à ce stade que la version « longue » (36 caractères) du texte soit utilisée, mais on constate d'ores et déjà qu'elle est très nettement désavantagée par rapport à la version courte. Dans ce cas, le système des fonctions de pondération arrive donc bien à modéliser une logique de choix *raisonnable*, dans la mesure où l'on considère qu'un texte défilant est contraignant pour les utilisateurs.

6.2.6.2 Unités pour l'expression des valeurs d'attributs

Pour le moment, nous n'avons pas précisé les unités utilisées pour exprimer les valeurs des attributs, et plus particulièrement les variables des fonctions de pondération. Cette section se propose de donner quelques précisions sur le sujet.

Cas du mode visuel Lorsqu'on manipule des modalités visuelles, il est nécessaire de spécifier des tailles (taille d'un texte, dimensions d'une image, etc.). Se pose alors la question du moyen d'expression des contraintes portant sur ces tailles.

Lorsqu'il est question de texte, la façon la plus évidente de procéder est d'exprimer la taille du corps typographique en mètres ou en points typographiques[8] . Cependant, cette solution ne permet pas de tenir compte de la distance entre l'utilisateur et l'écran. Ainsi, si un utilisateur peut déchiffrer sans difficulté un texte écrit en corps 30 sur un écran situé à 1 m de lui, il en est tout autre si l'écran se trouve à 5 m.

Du point de vue de l'utilisateur, il faut donc pouvoir exprimer les contraintes sur les tailles de façon indépendante de la distance.

Ceci est possible, à condition d'utiliser des *tailles angulaires*, comme indiqué sur la figure 6.10.

[8]Un point typographique (pica) mesure 352,777 μm.

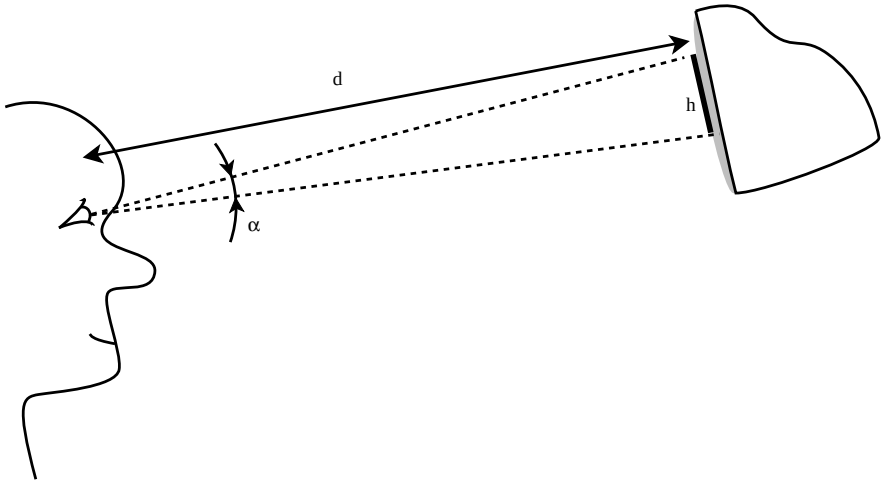


Figure 6.10 : *La taille angulaire α permet de donner des contraintes indépendantes de la distance entre un écran et l'utilisateur.*

De cette façon, un utilisateur peut indiquer la taille *apparente* α des objets qu'il souhaite percevoir. Connaissant la distance d entre l'utilisateur et l'écran, il est alors possible de déterminer très simplement la taille d'affichage (en mètres, et par conséquent également en points, pixels, etc.). On considère que l'œil de l'utilisateur se trouve en un sommet d'angle $\frac{\alpha}{2}$ d'un triangle-rectangle dont le côté opposé vaut $\frac{h}{2}$ (voir figure 6.11). On en déduit la formule :

$$h = 2d \tan \frac{\alpha}{2} \qquad \text{formule } (*)$$

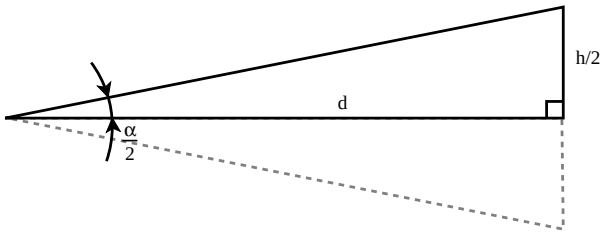


Figure 6.11 : *Étant donnée une taille angulaire α et la distance d entre l'utilisateur et l'écran, il est très facile de calculer la taille physique h.*

On considère généralement que la résolution angulaire de l'œil humain est de l'ordre d'1 à 3 minutes d'angle. En pratique, des angles plus importants sont nécessaires pour distinguer des caractères. De plus, un utilisateur peut souhaiter voir des textes sous des angles plus grands de façon à augmenter le confort de lecture, surtout dans des conditions *difficiles* (stress lorsqu'il faut prendre son train rapidement, foule très dense, bousculades, etc.)

Ainsi, un utilisateur pourrait par exemple spécifier qu'il *« souhaite percevoir des caractères d'au moins 1° »*, ce qui correspond à une hauteur de 1,7 cm à 1 m de distance (soit des caractères en corps 48), et à une hauteur de 8,7 cm à 5 m de distance (corps 246).

En résumé, il existe deux façons différentes de manipuler les tailles pour ce qui est des modalités visuelles :

- des tailles du point de vue de l'utilisateur : des tailles angulaires ;
- des tailles du point de vue de l'écran : des tailles linéaires, exprimées dans une unité de longueur.

Cas du mode auditif On se trouve dans une situation similaire si l'on considère les modalités auditives : l'intensité acoustique perçue par un utilisateur situé à 5 mètres d'un haut-parleur n'est pas la même que celle perçue à proximité immédiate du haut-parleur. En effet, l'intensité des ondes sonores décroît selon le carré de la distance à la source, d'où une atténuation importante. Les contraintes doivent donc ici aussi s'exprimer de deux points de vue différents :

- au niveau des sources sonores ;
- au niveau des utilisateurs.

En acoustique, on manipule les grandeurs suivantes [Hartmann 1998, Pireaux 1964] :

- la pression acoustique efficace (RMS[9]) : moyenne quadratique de la pression en un point, mesurée en Pa, et notée x ;
- intensité acoustique (efficace) en un point, mesurée en $\mathrm{W} \cdot \mathrm{m}^{-2}$. Notée I, elle est reliée à x par la relation $I = \frac{x^2}{\rho c}$, où ρ est la densité du milieu, et c la célérité du son dans ce milieu ;
- niveau de pression acoustique en décibels : $L = 20 \log_{10}\left(\frac{x}{x_0}\right) = 10 \log_{10}\left(\frac{I}{I_0}\right)$.

P_0 est choisi de sorte que ce soit le seuil d'audition[10] soit $P_0 = 2 \cdot 10^{-5}$ Pa. Étant donnée cette définition, les êtres humains perçoivent des niveaux sonores entre 0 dB et 120 dB (« seuil de la douleur »). Dans l'air, les valeurs des constantes ρ et c sont telles que $I_0 \approx 10^{-12}$ $\mathrm{W} \cdot \mathrm{m}^{-2}$.

Voyons comment il est possible de convertir la puissance émise en niveau de pression acoustique perçu par l'utilisateur. En première approximation, on peut négliger l'absorption par le milieu ; on ne considère que l'atténuation des ondes acoustiques en fonction de la distance. On considère également que les ondes se propagent sans limite dans toutes les directions. En pratique, le modèle simple proposé ici devrait être modifié si on voulait l'appliquer à l'acoustique en intérieur.

Si P est la puissance émise par un haut-parleur, alors l'intensité à une distance r sera « répartie » sur une sphère de rayon r, d'où :

[9] Root Mean Square.
[10] Pour des individus jeunes et en bonne santé.

$$I(r) = \frac{P}{4\pi r^2}$$
$$L(r) = 10 \log_{10} \left(\frac{P}{4\pi r^2} \cdot \frac{1}{I_0} \right) \qquad \text{formule } (**)$$

On obtient ainsi le niveau de pression acoustique à une distance r en fonction de la puissance émise par un haut-parleur. On peut aussi envisager de mesurer le niveau sonore engendré par un haut-parleur à une distance fixée r_1, et extrapoler ensuite cette valeur pour d'autres distances r_2.

On se place d'abord en un point 1 situé à distance r_1 de la source et on mesure une intensité acoustique I_1.

On se place ensuite en un point 2 situé à distance r_2 de la source et on mesure une intensité acoustique I_2. De la conservation de la puissance totale sur les sphères de rayon r_1 et r_2, il vient :

$$I_1 \times 4\pi r_1^2 = I_2 \times 4\pi r_2^2$$
$$I_2 = I_1 \times \left(\frac{r_1}{r_2} \right)^2$$
$$L_2 = L_1 + 20 \log_{10} \left(\frac{r_1}{r_2} \right) \qquad \text{formule } (***)$$

Par exemple, cette formule montre que le volume décroît de 6 dB si on double la distance à la source[11].

Cas du mode TPK En ce qui concerne les modalités visuelles et auditives, les utilisateurs peuvent se trouver à des distances variables des dispositifs. A priori, la situation du mode TPK (tactilo-proprio-kinesthésique) est plus simple car les dispositifs sont *touchés* ou *sentis* par les utilisateurs, donc à distance fixée. Il n'est donc pas nécessaire de prendre en compte des points de vue différents.

Conséquences Le modèle général n'est pas affecté par les considérations ci-dessus, mais cependant, il devient nécessaire de distinguer deux points de vue (celui de l'utilisateur ou celui du dispositif) lors de l'expression des profils. Le tableau 6.3 résume les points de vue pertinents.

Le point de vue pertinent pour les unités sémantiques est celui du dispositif, car les u.s. doivent générer un *contenu concret* pour une modalité donnée. Elles doivent ainsi exprimer les contraintes portant sur ce contenu concret en termes d'unités de mesure liées au dispositif.

[11] En effet, $20 \log_{10} \left(\frac{1}{2} \right) \approx -6$.

Profil	Point de vue dispositif	Point de vue utilisateur
Utilisateur		×
Dispositif	×	
Unité sémantique	×	

Tableau 6.3 : *Différents points de vue.*

Comme au final, le but de nos opérations sur les arbres de contraintes est de déterminer une instanciation de modalité à utiliser pour la présentation de l'u.s., l'arbre résultat doit naturellement être exprimé du point de vue du dispositif.

Le plus simple est donc d'effectuer un *changement de point de vue* sur l'arbre qui est exprimé du point de vue de l'utilisateur (l'arbre correspondant au profil de l'utilisateur), et d'effectuer ensuite l'intersection du point de vue du dispositif.

Ce *changement de point de vue* est relativement simple à effectuer si l'on connaît la distance entre l'utilisateur et le dispositif :

- les dimensions angulaires sont converties en tailles à l'aide de la formule $(*)$;
- les niveaux sonores sont convertis à l'aide de la formule $(***)$ (ou, de façon équivalente, les puissances sont converties en niveaux à l'aide de la formule $(**)$) ;
- d'autres grandeurs pourraient être converties de façon analogue.

6.2.6.3 Préférences contradictoires

Il peut arriver que lors de l'ajout d'un nouvel utilisateur à un dispositif, les préférences de ce dernier soient *incompatibles* avec celles des autres utilisateurs intéressés par la même unité sémantique que lui. Par exemple, supposons qu'un écran présente une u.s. sous forme textuelle. Un premier utilisateur exige que la taille du texte soit supérieure à 30 (évaluations nulles en-deçà), tandis qu'un second exige que cette taille soit inférieure à 25 (évaluations nulles au-delà). Dans ce cas, quelle que soit la combinaison d'attributs choisie, l'évaluation de l'un au moins des deux utilisateurs sera forcément nulle.

Dans ce cas, l'unique solution est de présenter l'unité sémantique en question *plusieurs fois*, avec des attributs différents à chaque fois. En pratique, lors de l'ajout d'un nouvel utilisateur, si les préférences de ce dernier sont incompatibles avec celles des autres utilisateurs, l'u.s. sera dupliquée spécialement pour lui (sous réserve que cet ajout soit possible). De cette façon, la perturbation sera moindre pour les utilisateurs déjà présents, et le nouvel utilisateur pourra accéder à ses informations.

Cependant, de telles situations nous semblent *extrêmes*, et nous pouvons raisonnablement supposer qu'elles surviendront relativement rarement.

6.3 Choix d'un dispositif parmi plusieurs

Dans la section précédente, nous avons vu que pour un dispositif de présentation donné, nous sommes capables d'instancier de façon *optimale* une modalité pour la présentation d'une unité sémantique. Cependant, si plusieurs dispositifs de présentation compatibles[12] se trouvent dans l'espace perceptuel de l'utilisateur, il est nécessaire de choisir lequel utiliser. Cette section explique donc comment il est possible d'effectuer ce choix.

6.3.1 Position du problème

Nous considérons que lorsqu'un utilisateur est proche d'au moins un dispositif de présentation, il doit nécessairement pouvoir consulter toutes ses unités sémantiques. C'est ce que nous appelons la *contrainte de complétude*.

Nous souhaitons également que les informations soient distribuées de façon optimale entre les différents dispositifs. Dans ce but, nous introduisons la notion de *charge informationnelle* d'un dispositif. La charge informationnelle est égale au nombre d'unités sémantiques présentées par le dispositif. Par exemple, la charge informationnelle d'un écran est tout simplement égale au nombre d'unités sémantiques qui y sont affichées. Pour un dispositif de présentation p, on note ℓ_p sa charge.

Pour que les unités sémantiques soient distribuées de façon optimale entre les dispositifs, la charge de chacun d'entre eux doit donc être minimale. Ainsi, il doit être possible de permettre aux utilisateurs d'accéder plus rapidement à leurs informations, et donc de limiter leur confusion et leur surcharge cognitive. C'est ce que nous appelons la *contrainte d'optimisation* de l'espace de présentation.

Si nous ne tenions compte que de ces deux contraintes, le problème se résumerait à la résolution d'un système de contraintes distribuées (complétude), tout en minimisant un paramètre de charge (optimisation). Ainsi, le problème pourrait être considéré comme étant une instance de DCOP (Distributed Constraint Optimization Problem) ; plusieurs algorithmes existent pour résoudre un tel problème [Mailler 2004].

Par exemple, supposons que trois utilisateurs, a, b et c se trouvent face à deux écrans appelés 1 et 2. Les unités sémantiques respectives de a, b et c sont α, α et β. Nous supposons que a voit l'écran 1, et que b et c voient les deux écrans. Cela conduit au système de contraintes suivant :

$$(1 \text{ affiche } \alpha) \wedge ((1 \text{ affiche } \alpha) \vee (2 \text{ affiche } \alpha)) \wedge ((1 \text{ affiche } \beta) \vee (2 \text{ affiche } \beta))$$

Une solution possible est que 1 et 2 affichent tous les deux α et β. Dans ce cas, la charge totale serait 4. Or, une meilleure solution existe, dans laquelle 1 affiche α et 2 affiche β, pour

[12] Du point de vue des modalités supportées par les dispositifs et l'utilisateur.

une charge totale valant 2. Cette dernière solution est visiblement optimale, car pour afficher deux u.s., la charge totale doit être au moins égale à deux.

Les algorithmes généraux de résolution de contraintes que nous venons d'évoquer sont conçus pour trouver en une seule fois une solution à un problème, donné lui aussi en une seule fois. À l'opposé, notre problème est construit pas à pas à partir d'une situation initiale. En effet, nous pouvons supposer qu'au commencement aucun utilisateur n'est proche d'aucun dispositif de présentation (i.e. l'espace perceptuel de tous les utilisateurs est vide). À partir de là, deux sortes d'événements peuvent survenir :

1. un utilisateur s'approche d'un dispositif dont il était éloigné ;

2. un utilisateur s'éloigne d'un dispositif dont il était proche.

De cette façon, toute situation peut être construite par une suite d'événements n°1 et n°2 (commençant par un événement n°1). Si nous supposons que nous disposons d'une solution convenable à un moment donné, et si nous sommes capables de construire une nouvelle solution après qu'un événement n°1 ou n°2 soit survenu, alors nous sommes capables de résoudre le problème à tout moment (principe de récursion). Ainsi, un algorithme incrémental sera très efficace dans cette situation. Nous verrons que la solution que nous introduisons à la section 6.3.3 est incrémentale.

Optimiser la charge des dispositifs est certes un but important, mais le suivre aveuglément peut conduire à la réalisation de systèmes inutilisables. En effet, reprenons l'exemple de moniteurs situés à proximité d'un groupe d'utilisateurs. Si on décide d'éviter absolument toute surcharge des écrans, on finit par réorganiser constamment l'affichage des informations. Ceci a une conséquence néfaste : les utilisateurs risquent de voir leurs u.s. passer d'un écran à l'autre à chaque fois qu'un autre utilisateur s'approche ou s'éloigne d'un écran. Ils passeraient alors leur temps à rechercher l'information qu'ils sont en train de lire, perdraient le fil, et finiraient par ne plus s'y retrouver du tout. En résumé, ce serait bien plus fastidieux et perturbant d'utiliser un tel système que de devoir rechercher une u.s. donnée sur un écran surchargé...

Au contraire donc, la présentation des unités sémantiques devrait rester *stable* lorsque des événements se produisent, pour ne pas perturber les utilisateurs. En effet, si un utilisateur donné ne se déplace pas, il est fort probable qu'il soit en train de percevoir (lire, écouter, sentir avec les doigts) une u.s., et donc il s'attend à ce que celle-ci reste en place, et non qu'elle disparaisse brusquement pour réapparaître ailleurs. Inversement, lorsqu'un utilisateur se déplace, il est peu probable qu'il soit en train de consulter des informations en même temps, et donc on peut déplacer une u.s. sans que cela le dérange de trop.

Ces considérations nous conduisent à prendre en considération les trois contraintes suivantes :

- **Contrainte C_1 (complétude).** Lorsqu'un utilisateur est proche d'un certain nombre de dispositifs de présentation, ses u.s. (s'il en possède) doivent lui être fournies par l'un (au moins) d'entre eux ;
- **Contrainte C_2 (stabilité).** Lorsqu'un utilisateur ne se déplace pas, ses u.s. doivent rester fixes. En particulier, elles ne doivent pas passer d'un dispositif à un autre ;
- **Contrainte C_3 (optimisation).** Pour éviter que les dispositifs ne soient surchargés, on évite au maximum la duplication des informations. Cela signifie que la somme des charges des dispositifs utilisés doit être minimale.

Pour des raisons ergonomiques, nous considérons que la contrainte C_1 est plus forte que la contrainte C_2, qui à son tour est plus forte que la contrainte C_3. En conséquence, la charge des dispositifs ne sera généralement pas complètement optimisée, de façon à préserver la stabilité des présentations. De même, lorsqu'un dispositif commencera à être saturé, certaines de ses u.s. seront déplacées sur un autre, afin qu'il puisse présenter une nouvelle u.s. à destination d'un nouvel utilisateur. Ainsi, on viole la contrainte de stabilité de façon à respecter la contrainte de complétude. Ceci permet d'introduire une forme de *coopération* entre dispositifs de présentation.

Avant d'étudier un algorithme incrémental capable de résoudre le problème que nous venons d'énoncer (section 6.3.3), nous allons tout d'abord définir un certain nombre de *coûts* associés à des opérations élémentaires sur les unités sémantiques (section 6.3.2). Ces coûts nous seront utiles pour la définition de l'algorithme.

6.3.2 Formalisation mathématique

La notion de coût nous sert à mesurer le degré de satisfaction ou de dissatisfaction des utilisateurs, en fonction des opération élémentaires effectuées sur les unités sémantiques. En conséquence, nous définissons plusieurs sortes différentes de coûts.

Coût statique de présentation Le coût statique d'un dispositif doit mesurer la difficulté que peut avoir un utilisateur à obtenir une unité sémantique à partir de ce dispositif. Ce coût est donc lié au nombre d'unités sémantiques actuellement présentées par ce dispositif. Le coût statique doit donc être une fonction strictement croissante de la *charge informationnelle* du dispositif, telle que définie plus haut.

De plus, il paraît raisonnable de considérer que plus il y a d'u.s. présentes sur un dispositif, plus l'ajout d'une nouvelle u.s. doit faire augmenter le coût statique, car plus il devient difficile d'obtenir une u.s. donnée. Par exemple, la figure 6.12 montre comment l'affichage de deux écrans est modifié lors de l'ajout de la même unité sémantique. Lorsque l'affichage ne contient auparavant qu'une seule u.s., l'ajout ne rend pas fondamentalement plus difficile la recherche d'une information (fig. 6.12a). Par contre, lorsque plusieurs u.s. sont déjà affichées, en chercher une nouvelle est extrêmement fastidieux (fig. 6.12b).

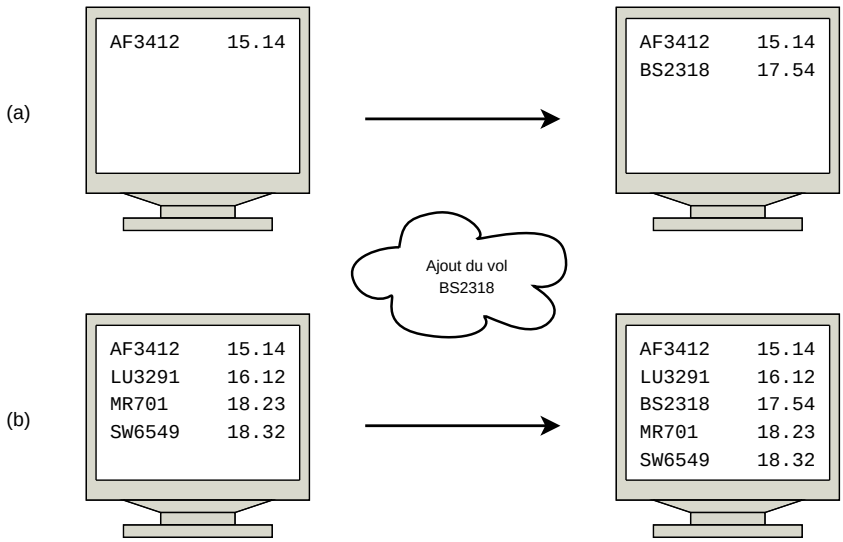


Figure 6.12 : *Effet de l'ajout d'une unité sémantique sur un écran peu chargé (a), ou déjà très chargé (b).*

Formellement, nous cherchons donc une fonction c définie sur $\mathbb{R}^+$, telle que, pour tous x et y de $\mathbb{R}^+$, on ait :

$$(x < y) \Rightarrow \big(\forall \delta \in \mathbb{R}^{+*}, c(y+\delta) - c(y) > c(x+\delta) - c(x)\big)$$

Si maintenant on cherche une telle fonction c *dérivable*, l'implication peut être transformée en :

$$\forall \delta \in \mathbb{R}^{+*}, \frac{c(y+\delta) - c(y)}{\delta} > \frac{c(x+\delta) - c(x)}{\delta}$$

D'où finalement :

$$(x < y) \Rightarrow \big(c'(y) > c'(x)\big)$$

Ceci montre que la dérivée c' de c est croissante sur $\mathbb{R}^{+*}$, donc c doit être une fonction *convexe*.

Pour un dispositif p donné dont la charge est ℓ_p, le *coût statique* est donc $c(\ell_p)$.

En première approximation, on peut donc par exemple prendre pour c des fonctions du type $\ell \longmapsto \ell^\alpha$ (avec $\alpha > 1$). On peut cependant envisager des expressions plus complexes, dans lesquelles on ne se contenterait pas du *nombre* des u.s. (i.e. de la charge) mais d'une *mesure de la complexité cognitive* de l'ensemble des u.s.

Nous avons réalisé une expérience cognitive auprès d'utilisateurs qui montre l'augmentation du temps de recherche d'une information dans une liste, en fonction du nombre total d'items dans la liste. Le protocole expérimental, ainsi que les résultats résultats de cette expérience

sont présentés en annexe A. Ils montrent qu'en-deçà d'un certain seuil (relativement élevé), l'augmentation du nombre d'items fait augmenter le temps de recherche d'une façon relativement linéaire. Cette expérience conforte donc ce qui vient d'être exposé. En effet, il paraît raisonnable que le degré de dissatisfaction des utilisateurs augmente au moins linéairement avec le temps passé à rechercher l'information : dans le cas linéaire, on aurait $\alpha = 1$ (cas limite d'une fonction convexe). Cependant, il paraît probable que plus le temps est élevé, plus une augmentation élémentaire de ce temps fasse augmenter le degré de dissatisfaction. Dans ce cas, on aurait une fonction strictement convexe ($\alpha > 1$).

Cette notion de coût statique correspond à la notion de *coût de rendu* introduite par Thevenin [Thevenin 1999].

Coût cognitif dynamique Nous appelons *coût cognitif dynamique* une mesure de la perturbation qu'introduit chez des utilisateurs la modification d'une présentation par l'ajout ou le retrait d'une unité sémantique. Nous supposons donc ici que l'on veut ajouter δ u.s. ($\delta \neq 0$) à un dispositif de présentation p donné (ou bien lui retirer $-\delta$ u.s. si $\delta < 0$).

Un coût cognitif est induit sur chaque utilisateur intéressé par au moins une u.s. du dispositif considéré. Le coût cognitif global est la somme des coûts cognitifs induits sur chaque utilisateur considéré individuellement.

Pour un utilisateur u donné, nous distinguons deux composantes différentes du coût cognitif dynamique :

- le coût induit par la modification de coût statique : il caractérise à quel point il lui sera plus difficile, ou bien plus facile, de retrouver ses informations après la modification de présentation. Nous définissons simplement ce coût comme étant égal à la différence des coûts statiques ;
- le coût engendré par la modification de la présentation, qui, quelle qu'elle soit, perturbe obligatoirement l'utilisateur. Pour un utilisateur u donné, ce coût de *perturbation* est noté P_u.

Ce coût dynamique est noté $d_c(p,u,\delta)$. Nous avons donc :

$$\begin{aligned} d_c(p,u,\delta) &= c_{\text{apres}} - c_{\text{avant}} + P_u \\ d_c(p,u,\delta) &= c(\ell_p + \delta) - c_(\ell_p) + P_u \end{aligned}$$

Notons que vue la définition du coût statique, le coût cognitif dynamique croît lorsque ℓ_p augmente. Ainsi, δ étant donné, si $\ell_2 > \ell_1$ alors $c(\ell_2 + \delta) - c(\ell_2) > c(\ell_1 + \delta) - c(\ell_1)$. Autrement dit, *plus il y a d'u.s. présentées sur un dispositif, plus il est coûteux de présenter une u.s. supplémentaire.*

Par exemple, supposons que sur un système donné, c soit défini par $c(x) = x^2$ pour deux écrans numérotés 1 et 2. Ce choix pour c est complètement arbitraire : en pratique, c doit être choisi pour chaque moniteur, de façon à refléter la difficulté de trouver des informations sur l'écran au fur et à mesure que sa charge augmente.

Si à un moment donné, l'écran n°1 affiche deux u.s., alors $c(\ell_1) = 2^2 = 4$; si l'écran n°2 affiche au même moment quatre u.s., alors $c(\ell_2) = 4^2 = 16$. Si l'on veut ajouter une u.s. à ces écrans, quels sont les coûts cognitifs dynamiques associés ? Par le calcul, on trouve $d_c(1,1) = (2+1)^2 - 2^2 = 9 - 4 = 5$; de même, $d_c(2,1) = (4+1)^2 - 4^2 = 25 - 16 = 9$.

Ainsi, si on a le choix, on préférera que la nouvelle u.s. soit affichée sur l'écran n°1, car le coût cognitif dynamique est moindre. Ceci correspond à l'intuition que l'on peut avoir de la situation. Notons bien qu'ici, les deux écrans partagent la même fonction de coût statique. On peut cependant imaginer des situations où ce ne serait pas le cas : chaque dispositif de présentation peut définir sa propre fonction de coût statique, correspondant à ses caractéristiques propres.

En cas d'une *diminution* du nombre d'u.s. ($\delta < 0$), la situation s'améliore d'un point de vue cognitif, mais il reste néanmoins une *perturbation* induite ponctuellement sur les utilisateurs. Le terme P_u est chargé de modéliser cet aspect.

Coût dynamique d'instanciation Un coût est calculé pour l'utilisateur qui voit une u.s. nouvellement présentée sur un dispositif. Ce coût mesure le degré de satisfaction (ou plus exactement, de dissatisfaction) de l'utilisateur. Il doit donc décroître lorsque croît le niveau de satisfaction de l'utilisateur. Par exemple, si l'évaluation donnée par l'utilisateur u à l'instanciation de la modalité est notée w, nous pouvons envisager des expressions de la forme suivante :

$$d_i(u) = \alpha \cdot (1 - w)$$

Coût dynamique de ré-instanciation Pour chaque utilisateur affecté par une modification des attributs d'une u.s., on calcule un coût lié à la modification de son niveau de satisfaction (i.e. de l'évaluation qu'il donne à l'instanciation de la modalité). Si son niveau de satisfaction a augmenté, le coût est négatif, sinon il est positif. Par exemple, si l'évaluation donnée à l'instanciation par un utilisateur u passe de w_{initial} à w_{final}, on peut envisager une expression du type :

$$d_r(u) = \beta \cdot (w_{\text{initial}} - w_{\text{final}}) + P_u$$

P_u est une constante fixée pour un utilisateur donné. Il modélise la perturbation liée à la ré-instanciation. En effet, même si la nouvelle instanciation est plus satisfaisante que la précédente, une perturbation est de toute manière induite sur l'utilisateur.

Coûts des opérations Nous venons de définir trois *coûts dynamiques* élémentaires. Ils sont en effet *élémentaires* à plusieurs égards :

- ils ne modélisent les perturbations induites que sur un seul utilisateur ;
- ils ne prennent en compte que des opérations atomiques (ajout ou suppression d'une unité sémantique sur un dispositif). En pratique, on peut imaginer des opérations plus complexes, comme par exemple des migrations.

Nous proposons donc des coûts *composés* pour deux types d'opération : l'ajout simple d'une unité sémantique à un dispositif, et la migration d'une unité sémantique d'un dispositif à un autre.

Ajout d'une u.s. On suppose qu'une u.s. est ajoutée pour l'utilisateur u, sur un dispositif p déjà utilisé par les utilisateurs v_i. On doit donc comptabiliser un coût cognitif et un coût de ré-instanciation pour chacun des v_i, ainsi qu'un coût d'instanciation pour u :

$$d_i(u) + \sum_i \left(d_c(p, v_i, +1) + r(v_i)\right)$$

Migration d'une u.s. On suppose qu'une u.s. utilisée par une personne u migre d'un dispositif p_1 utilisé par les utilisateurs v_i à un dispositif p_2 utilisé par les utilisateurs w_j. Il faut comptabiliser le coût de suppression de l'u.s. de p_1, son ajout sur p_2, et la perturbation imposée à l'utilisateur :

$$\underbrace{\sum_i \left(d_c(p_1, v_i, -1) + d_r(v_i)\right)}_{\text{Suppression de l'u.s. de } p_1} + \underbrace{\sum_j \left(d_c(p_2, w_j, +1) + d_r(w_j)\right)}_{\text{Ajout sur } p_2} + d_i(u) + P'_u$$

P'_u est un facteur chargé de modéliser la perturbation induite sur u par la migration.

Notons bien que les coûts n'interviennent pas dans l'instanciation (sauf pour départager des instanciations dont les évaluations seraient égales). Pour un dispositif donné, les attributs sont déterminés de façon à trouver le meilleur compromis entre utilisateurs. Le coût total de la présentation est alors évalué pour ce dispositif, de même que pour les autres dispositifs situés dans l'espace perceptuel de l'utilisateur. Sera finalement choisi le dispositif pour lequel le coût est minimum. Bien que ce ne soit qu'un facteur parmi d'autres, l'instanciation choisie intervient dans le calcul du coût pour un dispositif donné. De cette façon, les solutions pour lesquelles l'instanciation satisfait le plus l'utilisateur sont privilégiées.

6.3.3 Algorithme

Grâce aux coûts définis ci-dessus, nous pouvons proposer un algorithme permettant la résolution incrémentale du problème de répartition des unités sémantiques sur des dispositifs de présentation divers.

Tout d'abord, nous spécifions les agents utilisateurs de telle sorte qu'ils puissent attribuer à chacune de leurs unités sémantiques d'intérêt un *dispositif principal*, c'est-à-dire le dispositif de présentation où l'u.s. en question est présentée. Pour simplifier la description de l'algorithme, nous supposons que chaque utilisateur ne demande la présentation que d'une unique unité sémantique. Au démarrage tous les dispositifs principaux sont non définis.

Au niveau global, l'algorithme fonctionne ainsi : lorsqu'un agent utilisateur s'approche (i) ou s'éloigne (ii) d'un dispositif, il envoie des *requêtes d'évaluation* aux agents présentateurs situés à proximité, afin de connaître le *coût* de certaines opérations (présentation d'une nouvelle u.s., migration d'u.s.), que nous décrivons plus loin. Les agents présentateurs répondent à ces requêtes (iii, iv) et mémorisent les opérations soumises à évaluation. L'agent utilisateur a alors le choix entre *confirmer* ou *annuler* chacune de ces opérations évaluées.

Dans les figures ci-dessous, les agents sont représentés par des ovales. Le lien entre un agent utilisateur et l'agent qui correspond à son dispositif principal est représenté par une ligne pointillée. Les messages entre agents sont représentés par des flèches, généralement étiquetées. Les flèches en pointillés représentent les réponses aux requêtes.

[i] Lorsqu'un agent utilisateur doté d'une u.s. u s'approche d'un dispositif p :

- *s'il possède déjà un dispositif principal*, il envoie une requête `évaluation-migration`(s) à son dispositif principal (voir fig. 6.13). Si le résultat (coût) de la requête est négatif, alors l'agent la confirme. Sinon, il l'annule ;
- *sinon*, il envoie une requête `évaluation-présentation`(u) à s, et la confirme systématiquement, de façon à satisfaire la contrainte C_1 (complétude), voir fig. 6.14.

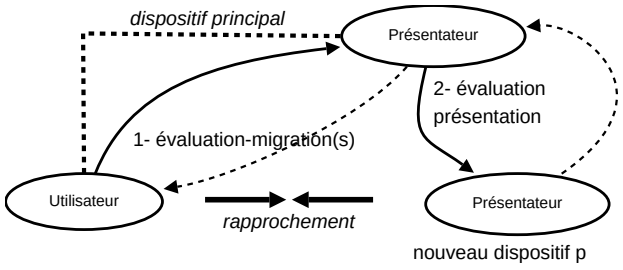


Figure 6.13 : *Lorsqu'un utilisateur s'approche d'un dispositif, il essaie d'y faire migrer une u.s.*

[ii] Lorsqu'un agent utilisateur (dont l'u.s. est u) s'éloigne de son *dispositif principal*, il commence par envoyer une notification `éloignement` à son dispositif principal, puis :

- *si d'autres dispositifs sont situés à proximité*, il envoie une requête `évaluation-présentation`(u) à chacun d'entre eux, et choisit celui pour lequel le coût est le plus faible (voir fig. 6.15). Il en fait son dispositif principal (contrainte C_3, optimisation de l'espace de présentation). Il envoie alors un message de confirmation à ce dispositif, et un message d'annulation à tous les autres ;

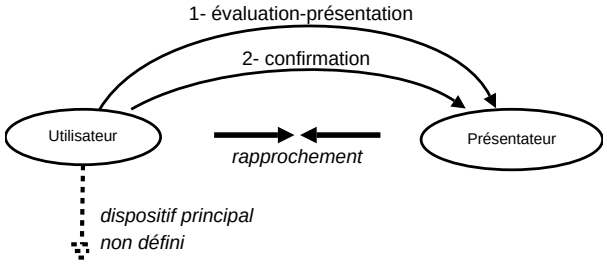


Figure 6.14 : *Lorsqu'un utilisateur qui n'a pas de dispositif principal s'approche d'un dispositif, il y demande systématiquement la présentation de ses u.s.*

- *sinon*, son dispositif principal est réputé non défini (et par conséquent son u.s. n'est plus présentée nulle part).

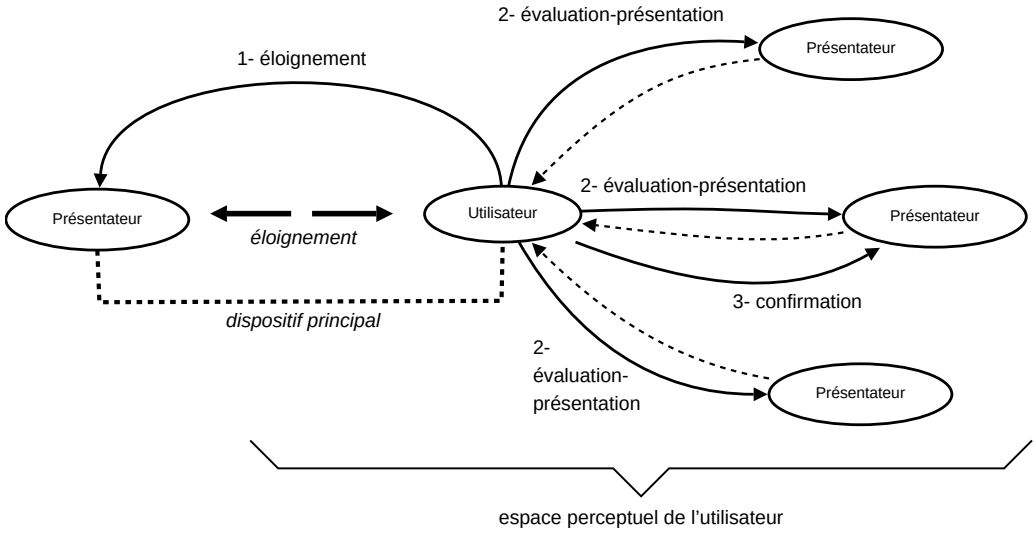


Figure 6.15 : *Lorsqu'un utilisateur s'éloigne de son dispositif principal, il essaie de faire présenter son u.s. par un autre dispositif situé à proximité.*

[*iii*] Lorsqu'un agent présentateur reçoit une requête `évaluation-présentation`(u) :

- *s'il y a encore de la place pour l'u.s. u*, il l'ajoute à la liste des u.s. à présenter : lorsque la contrainte C_1 (complétude) est satisfiable, le dispositif essaie de satisfaire la contrainte C_2 (stabilité) ;
- *sinon*, il essaie de déplacer sur un autre dispositif l'une des u.s. qu'il présente. Plus précisément, pour chaque u.s. présentée v, il envoie récursivement une requête `évaluation-présentation`(v) à chaque dispositif vu par tous les agents-utilisateurs a_i intéressés par v (voir fig. 6.16). Le coût d'une migration possible (si l'appel récursif correspondant n'échoue pas) est alors défini comme étant le coût renvoyé (noté d), plus le coût de suppression de v.

  Si au moins l'un de ces appels récursifs n'échoue pas, l'agent choisi le moins coûteux, le confirme, et annule les autres. Sinon, il déclare à son tour échouer. Si la

contrainte C_2 (stabilité) n'est pas satisfiable, le dispositif essaie toujours de respecter la contrainte C_1 (complétude), mais ce faisant, il continue à optimiser l'espace de présentation (contrainte C_3). La règle C_1 n'est enfreinte qu'en dernier lieu, c'est-à-dire si tous les dispositifs à proximité n'ont plus d'espace disponible.

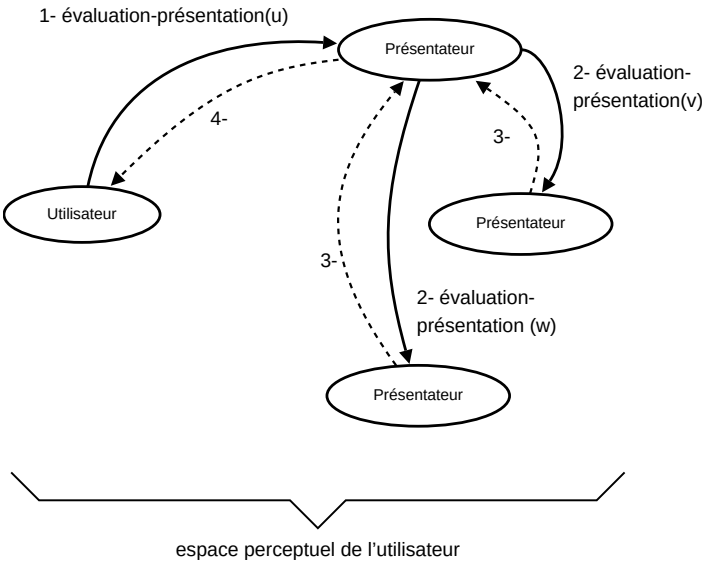


Figure 6.16 : *Lorsqu'un dispositif est surchargé, il essaie de faire migrer certaines des u.s. qu'il présente sur des dispositifs inclus dans les espaces perceptuels de tous les utilisateurs intéressés.*

[*iv*] Lorsqu'un agent présentateur reçoit une requête `évaluation-migration`(s) concernant une u.s. u :

- si u est le dispositif principal de plus d'un agent utilisateur, la requête échoue (complétude et stabilité) ;
- sinon, l'agent présentateur envoie à s une requête `évaluation-présentation`(u), et note le coût associé d_1 (voir fig. 6.13). Il évalue le « coût » de la suppression de u de son espace de présentation. Ce coût, négatif, est appelé d_2. Il calcule alors le coût associé à la perturbation induite par les utilisateurs, et le note P_u. Alors, il renvoie $d_1+d_2+P_u$. On considère que la migration est souhaitable si cette quantité est négative.

Confirmation et annulation Nous avons décrit ici le fonctionnement de l'algorithme en termes d'*évaluation* d'opérations. Les évaluations préparent une opération, sans pour autant l'exécuter. Les fonctions d'évaluation renvoient à l'entité appelante le coût de l'opération, accompagné d'un identifiant. Cet identifiant permet de se référer de façon unique à une opération qui a été évaluée. Ainsi, l'entité potentiellement intéressée par l'opération peut décider soit de la *confirmer*, ce qui commande son exécution, soit de l'*annuler*, ce qui la supprime simplement de la mémoire des entités considérées.

6.3.4 Résumé

Par l'application de trois contraintes ergonomiques, cet algorithme permet de résoudre deux problèmes :

1. celui du choix d'un dispositif lors de la présentation initiale d'une unité sémantique. Par exemple, lorsqu'un utilisateur se trouve devant trois écrans et qu'il reçoit une unité sémantique, cet algorithme permet de décider lequel des trois écrans il est souhaitable d'utiliser ;
2. celui de la migration des unités sémantiques au gré des déplacements des utilisateurs. Par exemple, si un utilisateur s'éloigne de l'écran où était affichée son unité sémantique, cet algorithme permet de déterminer sur quel dispositif il faut la faire migrer.

Le chapitre 7 donnera des précisions sur l'implémentation pratique de cet algorithme, ainsi qu'une démonstration réalisée à l'aide d'un émulateur spécifique (voir section 7.5).

6.4 Conclusion

Le chapitre 5 a présenté le modèle KUP, un nouveau modèle d'architecture logicielle pour les systèmes interactifs, qui propose de découpler les phases de *fourniture* et de *présentation* des informations à destination des utilisateurs. Le présent chapitre a donné des précisions quant aux méthodes mises en jeu lors du choix des dispositifs de présentation et de l'instanciation des modalités.

Le choix des dispositifs de présentation est effectué grâce à un algorithme incrémental de négociation entre agents présentateurs et utilisateurs. L'instanciation des modalités se fait par le choix d'une solution la plus satisfaisante possible dans l'intersection des profils de l'utilisateur, du dispositif de présentation et de l'unité sémantique à présenter.

Le chapitre suivant explique comment les concepts introduits jusqu'ici peuvent être mis en œuvre dans la réalisation d'une plate-forme concrète pour le développement d'applications de fourniture contextuelle d'informations à des utilisateurs mobiles.

Troisième partie

Implémentation

Chapitre 7

Implémentation : la plate-forme PRIAM

7.1 Introduction

Dans les chapitres 5 et 6, nous avons présenté un modèle et des algorithmes qui permettent la spécification d'un système opportuniste de présentation d'information pour des utilisateurs mobiles. Le présent chapitre introduit la plate-forme PRIAM, pour PRésentation d'Informations dans l'AMbiant, qui constitue une implémentation des concepts évoqués plus haut.

L'objectif principal de cette plate-forme est de permettre la réalisation en pratique d'applications de fourniture et de présentation d'information à des utilisateurs dans des environnements publics. La plate-forme doit prendre en charge tous les aspects de bas niveau (gestion du réseau, échange de données entre agents, etc.) de façon à en décharger d'autant les concepteurs d'applications. Ainsi, nous nous donnons comme objectif de factoriser *au sein de la plate-forme* tous les mécanismes communs à tous les agents et à toutes les applications. De cette façon, les applications n'auront plus qu'à spécifier explicitement les comportements qui leurs seront particuliers.

Selon les cas, les agents de la plate-forme peuvent être amenés à fonctionner dans des environnements matériels et logiciels très différents : dans certaines applications, ils fonctionneront tous sur un serveur centralisé, tandis que dans d'autres, si certains agents résideront sur des serveurs, d'autres devront fonctionner sur des dispositifs individuels portables de type PDA. En conséquence, nous avons choisi de réaliser notre implémentation en langage Java, de façon à bénéficier d'un maximum de portabilité. Ainsi, grâce à Java, un même *bytecode* pourra être exécuté sur n'importe quelle plate-forme, à condition bien entendu qu'une machine virtuelle Java y soit disponible.

Nous supposons qu'un réseau IP est accessible à tous les agents, et donc que chacun d'entre eux peut communiquer sans problème avec chacun des autres. Dans les cas où tous les agents fonctionneraient sur un même serveur central, ceci demeurera bien entendu possible même en

l'absence de réseau physique[1]. À l'opposé, lorsque tout ou partie des agents fonctionneront sur des dispositifs mobiles, un réseau sans fil de type WiFi devra être disponible. Nous ne traitons pas des éventuels problèmes d'accès au réseau, notamment liés à l'authentification : nous supposons simplement que tous les dispositifs informatiques sur lesquels fonctionnent les agents sont capables d'accéder au réseau ambiant et de disposer d'une connectivité IP.

7.2 Taxonomie et profils

Avant de passer à la description de l'implémentation des agents eux-mêmes, voyons d'abord comment peuvent être représentés leurs profils. Tout d'abord, nous avons vu plus haut qu'un profil n'a de sens que par rapport à une *taxonomie des modalités* : nous devons donc dans un premier temps donner un formalisme utilisable pour décrire une telle taxonomie.

7.2.1 Taxonomie des modalité

Une taxonomie est définie par :

- un arbre de modalités. Chaque nœud de l'arbre correspond à une modalité. On y liste ses attributs (et leurs domaines), ainsi que les sous-modalités ;
- une liste d'ensembles discrets, qui correspondent aux *domaines* possibles pour les différents attributs.

Nous avons décidé de décrire les taxonomies de modalités en XML, de façon à pouvoir utiliser les outils standards de manipulation de ce type de fichiers. Notamment, nous avons eu recours à la bibliothèque Xerces[2], une implémentation de l'API[3] DOM[4] disponible pour Java.

La DTD[5] correspondante est donnée en annexe B.1, et un exemple de fichier XML de description de taxonomie est donné en annexe B.2.

Dans ce formalisme, on décrit tout simplement une hiérarchie de taxonomies par un ensemble d'éléments XML `modality` imbriqués. Chaque modalité possède un nom, et surtout un identifiant (attribut `id`), qui est utilisé en interne pour y faire référence. Notamment, lors de l'expression des profils, il permettra de faire le lien avec les modalités, et ainsi d'associer des pondérations à chaque nœud de l'arbre de modalités.

Les attributs des modalités sont indiqués à l'aide d'éléments XML `attribute`, inclus à l'intérieur des éléments `modality`. Un attribut peut être de deux types :

[1] Grâce à l'interface de *loopback* `localhost`.
[2] Voir `http://xerces.apache.org/`.
[3] Application Programming Interface.
[4] Document Object Model, voir `http://www.w3.org/DOM/`.
[5] Définition de type de document XML.

- *discret* : dans ce cas, les valeurs possibles de l'ensemble sont données à part, à l'aide d'un élément `finiteSet`. Ceci permet de définir un ensemble fini une fois pour toutes, et de le réutiliser plusieurs fois ensuite sans devoir dupliquer sa description. Par exemple, les deux attributs `lang` de l'exemple de l'annexe B.2 sont de type `Lang`, dont les valeurs peuvent être `fr` ou `en` ;

- *continu* : dans ce cas, on indique les bornes de l'intervalle de $\mathbb{R}$ correspondant. Cependant, dans l'état actuel de notre implémentation, nous *discrétisons* les intervalles réels. C'est pourquoi un attribut nommé `step` indique le pas de parcours discret de l'intervalle. Par exemple, l'attribut `volume` varie entre 0 et 10, avec un pas de 2 : cela signifie que lors de la recherche d'une valeur pour cet attribut, seules les valeurs 0, 2, 4, 6, 8 et 10 seront examinées. Dans une version plus élaborée, cette restriction sera levée, et il sera possible de prendre en compte de « vrais » intervalles continus.

La structure de l'arbre XML est reprise en Java par des classes correspondantes : `Taxonomy` (taxonomie tout entière), `TaxonomicTree` (un arbre de modalités), `Attribute`, et une classe abstraite `AttributeType`, qui possède deux sous-classes concrètes, `FiniteSet` et `RealInterval`.

7.2.2 Profils

Pour décrire le profil d'un utilisateur, d'un dispositif de présentation ou d'une unité sémantique, nous avons suivi les mêmes principes que pour la description d'une taxonomie : les profils sont donnés sous forme de fichiers XML, dans un dialecte pour lequel il existe une DTD. Cette DTD est donnée en annexe B.3. Un exemple de profil exprimé en XML est donné en annexe B.4.

Ce profil donne la pondération, ainsi que la fonction de pondération, associées à chaque nœud. Au niveau du programme, un profil est représenté par une classe `WeightedTree`, qui contient un certain nombre d'objets de type `WeightedNode`. Comme son nom le laisse supposer, un `WeightedNode` contient le poids et la fonction de pondération associés à un nœud de la taxonomie.

Notons que nous aurions pu décrire les profils en utilisant la norme CC/PP[6], publiée par le W3C. Dans ce cas, la taxonomie des modalités correspondrait à un *vocabulaire* CC/PP, qui permettrait ensuite d'écrire des *profils*. Cependant, nous n'avons pas trouvé de réel avantage à utiliser CC/PP :

- nous n'avons pas de besoin d'interopérabilité pour le moment, donc utiliser une norme n'est pas une priorité. De plus, l'adoption industrielle de CC/PP est relativement limitée pour le moment, donc le gain en termes d'interopérabilité aurait été très faible ;

[6] Composite Capability/Preferences Profiles, voir *CC/PP Structure and Vocabularies 1.0* (2004), `http://www.w3.org/TR/CCPP-struct/vocab/`.

- notre propre format de description des profils, comme on peut le voir en annexe B.4, est très concis et spécialement adapté à notre application. Le format CC/PP se serait révélé plus lourd, à la fois en termes de taille des descriptions, et en termes de traitement à effectuer pour les interpréter.

Cependant, si cela s'avérait nécessaire à l'avenir, il serait possible de passer à CC/PP sans grande modification de la plate-forme.

7.2.2.1 Fonctions de pondération

Afin de simplifier la conception de cette implémentation, nous n'exprimons pas directement les fonctions de pondération dans le fichier XML. Nous nous contentons d'indiquer le nom qualifié d'une classe Java chargée d'implémenter cette fonction. Cette classe doit être une sous-classe d'une classe abstraite prédéfinie, `WeightFunction`. Elle doit redéfinir la méthode `apply()`, qui est chargée de calculer la valeur de la fonction en un point donné. Un exemple est donné dans l'annexe B.4.

Cette approche possède un avantage évident : il est possible de bénéficier de toute la puissance expressive de Java pour décrire les fonctions. De plus, au niveau de l'implémentation, il n'est pas nécessaire de définir un interpréteur pour les fonctions, étant donné que la machine virtuelle Java peut jouer ce rôle.

Cependant, l'inconvénient de ce choix est qu'il est alors impossible d'étudier les variations de ces fonctions (pour rechercher un maximum) en utilisant les résultats de l'Analyse. Par exemple, si on lui donne sous forme d'expression algébrique la fonction $f(x) = 1-(x-3)^2$, un programme peut facilement en déterminer le maximum : par dérivation, on obtient $f'(x) = -2 \cdot (x-3)$, et par étude des zéros de cette dérivée, on voit qu'un extremum est atteint pour $x = 3$.

Or si on fournit juste une référence à une classe Java dont la méthode `apply()` peut *calculer des valeurs* de la fonction f en des points donnés, il n'est plus possible d'effectuer cette étude. Il faut alors parcourir le domaine de définition, demander à la classe Java de calculer la fonction en un certain nombre de points, et essayer d'en déduire la position des extrema. En un mot, la fonction est ici une véritable *boîte noire* qu'il n'est pas possible d'étudier de façon algébrique.

Ceci est la raison pour laquelle nous avons introduit plus haut une notion de « pas » pour les intervalles de $\mathbb{R}$: le programme d'optimisation parcourt alors les intervalles selon ces pas.

Les fonctions de pondération doivent pouvoir transiter par le réseau. Il est donc nécessaire de les *sérialiser*, mais cette sérialisation ne doit pas se limiter aux données d'instance : elle doit inclure le *code* de la méthode `apply()`, qui constitue l'information la plus importante dans la définition d'une fonction de pondération. L'annexe B.5 donne plus de détails sur cette opération.

7.2.2.2 Recherche pratique des extrema

Nous a vu que lorsqu'il s'agit de réaliser l'instanciation de n unités sémantiques chacune spécifiée par p variables, cela revient à maximiser une fonction dans un espace de dimension $n \cdot p$ (voir section 6.2.5.1). Cependant, la plupart du temps, les variables ne peuvent prendre que quelques valeurs discrètes (par exemple, longueur d'un texte). On peut donc les parcourir exhaustivement sans que le calcul ne soit trop long. Et même lorsqu'on a affaire à des valeurs continues (par exemple, taille d'une fonte), si on évite de choisir un pas de discrétisation trop petit, on limite le nombre de cas à examiner. Par exemple, dans le cas de la taille d'une fonte, si on prend un pas de 2 mm, et si l'intervalle de variation s'étend de 6 mm à 3 cm, cela laisse 13 cas à examiner, ce qui est tout à fait réaliste.

De plus, les valeurs de n et p restent relativement basses :

- en ce qui concerne n (le nombre d'u.s. présentées) il ne peut pas raisonnablement être trop élevé, pour deux raisons : la place disponible sur le dispositif, et la (sur)charge cognitive des utilisateurs. Une valeur de $n = 10$ semble difficilement dépassable ;
- en ce qui concerne p, le nombre d'attributs pour une modalité donnée, il ne dépasse pas 6 dans notre taxonomie d'exemple ;

Il doit donc être possible de rechercher rapidement un optimum, même si en première approximation cet optimum est assez grossier.

Une autre piste pourrait être l'utilisation d'un moteur de calcul mathématique, par exemple le noyau de Maple. Cependant, cette solution oblige à disposer d'un serveur de calcul, ce qui est contraire à l'approche décentralisée du travail. De plus, ce type de logiciels n'est pas toujours capable de trouver tout seul les optimums ; il faut parfois le guider un peu en restreignant les intervalles de recherche.

Il faudrait donc introduire un formalisme de description des fonctions de pondération, ce qui permettrait en outre de séparer les fonctions en produits ou sommes de fonctions indépendantes sur des sous-ensembles stricts de l'ensemble des paramètres, ce qui permet de traiter chacune des fonctions partielles à part, et réduit ainsi la combinatoire. Par exemple, si la fonction $f(a,b,c,d)$ peut s'écrire sous forme $f_1(a,b) \times f_2(c,d)$, et si chacun des quatre paramètres prend ses valeurs dans un ensemble discret de n éléments, la combinatoire passe de n^4 à $2n^2$ lorsqu'on décompose la fonction f et que l'on traite indépendamment f_1 et f_2.

7.2.2.3 Éditeur de profils

L'édition d'une taxonomie n'est pas une tâche fréquente. Elle n'est réalisée qu'une seule fois pour un ensemble de systèmes donné. Par exemple, dans toutes nos expérimentations, nous avons utilisé une seule taxonomie, très simple, donnée par le fichier XML de l'annexe B.2. De plus, pour la plupart des applications, la taxonomie de la figure 2.6 (page 13) devrait convenir, donc il est envisageable que la taxonomie soit intégrée à la plate-forme elle même, et en tout

état de cause, il ne sera pas nécessaire d'en redéfinir une nouvelle pour chaque application. Il n'est donc pas gênant que la spécification de la taxonomie se fasse manuellement, c'est-à-dire que l'on doive rédiger à la main le fichier XML correspondant.

Cependant, pour une taxonomie donnée, il est régulièrement nécessaire de spécifier des profils, par exemple pour décrire un utilisateur, une unité sémantique, un dispositif de présentation, etc. Il est donc utile de disposer d'un éditeur de profils qui permette de réaliser cette tâche relativement simplement. Nous avons donc implémenté un tel éditeur. À partir d'une taxonomie spécifiée sous forme d'un fichier XML, il permet de créer et de modifier des arbres de pondération. Il permet d'indiquer le poids ainsi que la fonction de pondération associée à chaque nœud de la taxonomie. La fenêtre de l'éditeur est présentée sur la figure 7.1.

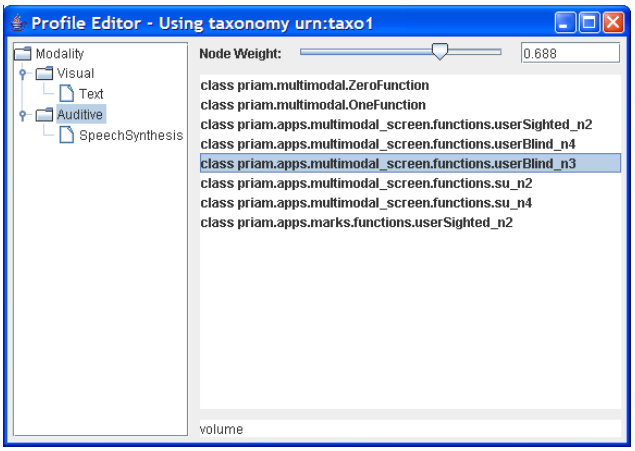


Figure 7.1 : *Fenêtre de l'éditeur de profils.*

Le panneau de gauche permet la sélection des modalités parmi l'arbre de modalités défini dans la taxonomie. Le curseur à glissière du haut permet de régler le poids de la modalité sélectionnée. Dans la zone centrale s'affichent la liste des fonctions de pondération disponibles, i.e. des sous-classes de `WeightFunction`. Le concepteur de profil choisit alors dans cette liste la fonction de pondération à associer au nœud sélectionné.

Bien entendu, comme nous l'avons déjà précisé, il est nécessaire de spécifier les fonctions de pondération elles-mêmes sous forme de classes Java. L'une de nos perspectives d'implémentation concerne la spécification des fonctions de pondération non plus en Java, mais dans un formalisme spécifique, tel que l'édition des fonctions puisse se faire dans l'éditeur. Nous nous sommes par exemple intéressés au cas particulier où une fonction de pondération sur tous les attributs d'un nœud peut se décomposer en produit de fonctions élémentaires, portant chacune sur un attribut, et où ces fonctions élémentaires sont soit discrètes, soit polynomiales par morceaux. Nous avons dans ce cas implémenté un prototype d'éditeur de profil qui permet l'édition directe des fonctions. Une copie d'écran de ce prototype est présentée sur la figure 7.2.

Dans ce cas, la partie droite de la fenêtre permet de spécifier la fonction de pondération de l'attribut sélectionné :

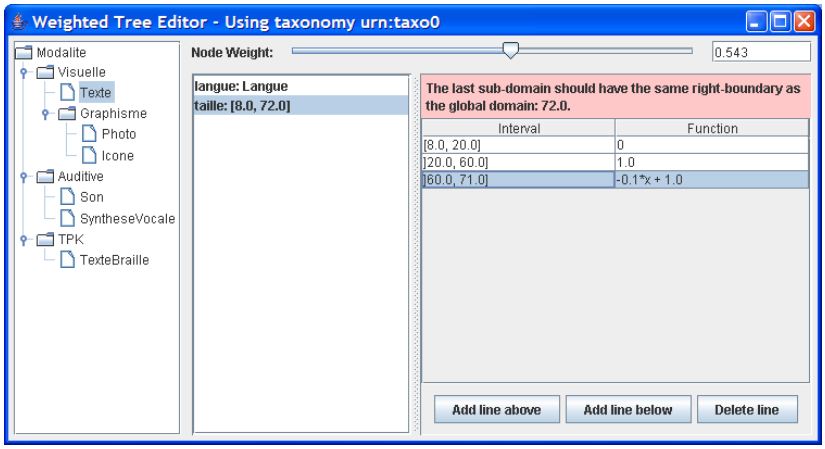


Figure 7.2 : *Prototype de l'éditeur de profils dans le cas particulier où tous les attributs sont indépendants, et où leurs variations dans le cas continu suivent des fonctions polynomiales par morceaux.*

- pour un attribut à valeurs dans un ensemble discret, on donne le poids associé à chaque élément de l'ensemble ;

- pour un attribut défini sur un intervalle de $\mathbb{R}$, on peut découper l'intervalle en question en un nombre arbitraire de sous-intervalles, et indiquer une fonction polynomiale sur chacun d'entre eux. C'est ce qui est montré sur la figure 7.2. Lors de la modification d'une fonction polynomiale par morceaux, un message de diagnostic est affiché en haut de la liste des morceaux : il permet de se rendre compte si les intervalles choisis pour chacun des morceaux forment une partition correcte de l'intervalle global de définition de la fonction par morceaux. Il est bien entendu possible d'ajouter et de retirer des morceaux à loisir.

7.3 Agents et unités sémantiques

Dans cette section, nous introduisons quelques détails d'implémentation concernant les différents agents. Nous présentons leur mode de communication et de déploiement sur un réseau, ainsi que leurs protocoles de communication.

7.3.1 Enchaînement des actions des agents

Nous commençons par donner une vue d'ensemble du fonctionnement du système, en détaillant bien l'enchaînement de messages échangés entre les différents agents. Lorsque ce sera possible, nous donnerons des exemples concrets du *contenu* des messages échangés. Les détails d'implémentation des agents, ainsi que les protocoles de communication seront abordés ensuite.

Nous avons indiqué plus haut que le modèle KUP permet la séparation nette de deux phases successives : la *fourniture*, puis la *présentation* d'une information, modélisée par une unité sémantique. Nous allons donc décrire ces deux phases l'une après l'autre.

Fourniture d'une unité sémantique Cette étape est relativement simple. Lorsque son service de localisation informe l'agent informateur qu'un agent utilisateur vient de pénétrer dans son espace perceptuel (méthode `locComeNear()`), l'informateur recherche une éventuelle unité sémantique à fournir à l'utilisateur. S'il existe de telles unités sémantiques, l'informateur les communique alors à l'utilisateur grâce au message `giveSU()`. L'utilisateur mémorise les unités sémantiques reçues de façon à pouvoir demander leur présentation lorsqu'il rencontrera un dispositif de présentation.

La figure 7.3 résume ce comportement sous forme d'un diagramme de séquence UML. Ici, l'unité sémantique fournie à l'utilisateur concerne la fermeture de la cafétéria.

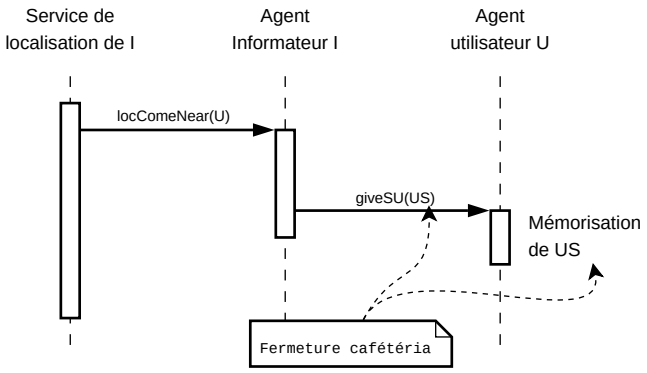


Figure 7.3 : *Fourniture d'une unité sémantique à un agent utilisateur.*

Présentation d'une unité sémantique Il s'agit maintenant d'effectuer la présentation de l'unité sémantique précédemment reçue sur un dispositif de présentation. Lorsqu'un tel dispositif pénètre dans l'espace perceptuel de l'utilisateur humain, le service de localisation de l'agent utilisateur envoie une notification `locComeNear()` à ce dernier. Démarre alors une négociation avec l'agent présentateur correspondant. Celui-ci détermine la modalité à utiliser, et si possible, détermine une instanciation de cette modalité. Ensuite, il envoie une réponse adaptée à l'agent utilisateur.

Celui-ci confirme la présentation avec le message `negCommit()` (il aurait pu l'annuler avec `negCancel()` s'il avait entre-temps reçu une *meilleure* proposition de présentation en provenance d'un autre dispositif). L'agent présentateur demande alors à l'unité sémantique de produire un contenu concret. Pour cela, il lui envoie un message `generate()`, en indiquant en paramètres la modalité instanciée déterminée à l'étape précédente.

En réponse, l'unité sémantique renvoie ce contenu concret généré, dans le *format* correspondant à la modalité choisie (voir section 5.3.2). L'agent présentateur, effectue ensuite la présentation sur le dispositif physique associé : il génère des *signaux électroniques* ou des *commandes* (de bas niveau) à partir du contenu concret.

Ces étapes sont résumées sur la figure 7.4.

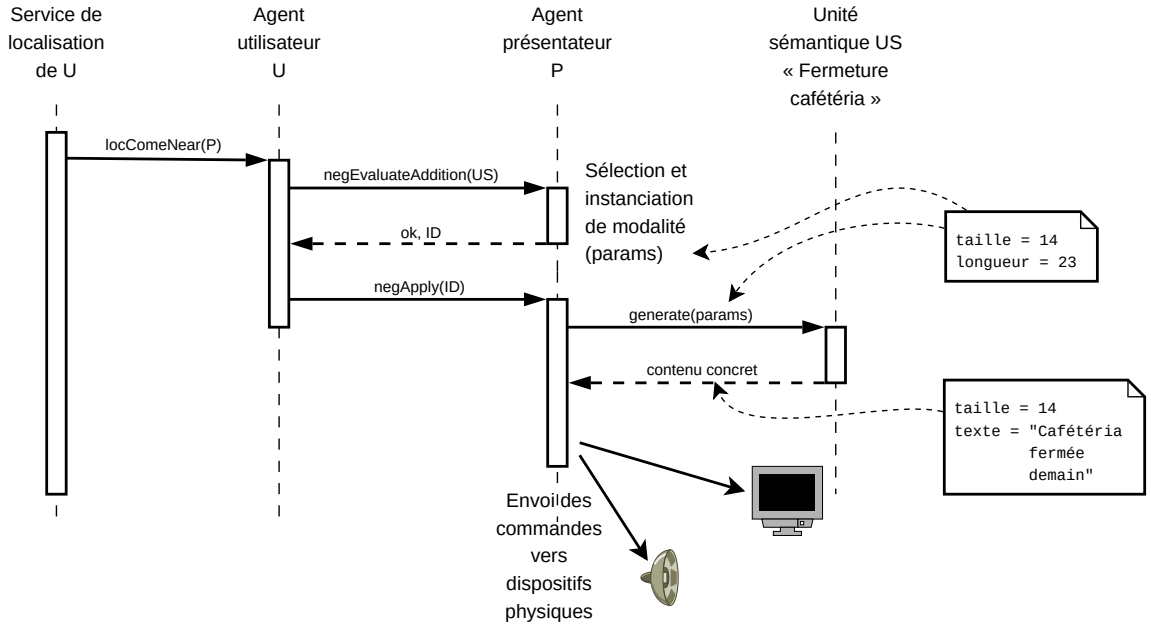


Figure 7.4 : *Présentation d'une unité sémantique pour un utilisateur.*

7.3.2 Hiérarchie de classes

Après avoir donné une vue d'ensemble du fonctionnement de la plate-forme PRIAM, nous allons maintenant décrire plus en détails l'implémentation des agents. Ces derniers se basent sur une interface mère, `IAgent`, qui possède trois interfaces filles, une pour chaque type d'agents (voir fig. 7.5).

Au niveau de l'interface `IAgent` sont situées les méthodes liées aux réactions aux déplacements des entités que modélisent les agents. Ainsi, les agents reçoivent une notification quand :

- un autre agent entre dans leur espace perceptuel (`locComeNear()`) ;
- un autre agent, qui était auparavant dans leur espace perceptuel, en sort (`locMoveAway()`) ;
- la distance entre eux et un agent situé dans leur espace perceptuel change (`locChangeDistance()`).

Ces messages sont envoyés aux agents par l'intermédiaire d'un *service de localisation*, auprès duquel on doit les enregistrer. Le service de localisation fait le lien entre le monde réel et le monde des agents. Il transfert dans le second les *comportements proactifs* détectés dans le premier. Dans la version actuelle de PRIAM, nous avons implémenté deux tels services de localisation : un service de localisation lié au simulateur (voir section 7.4), et un service de localisation basé sur la détection de badges à infrarouges (voir annexe C).

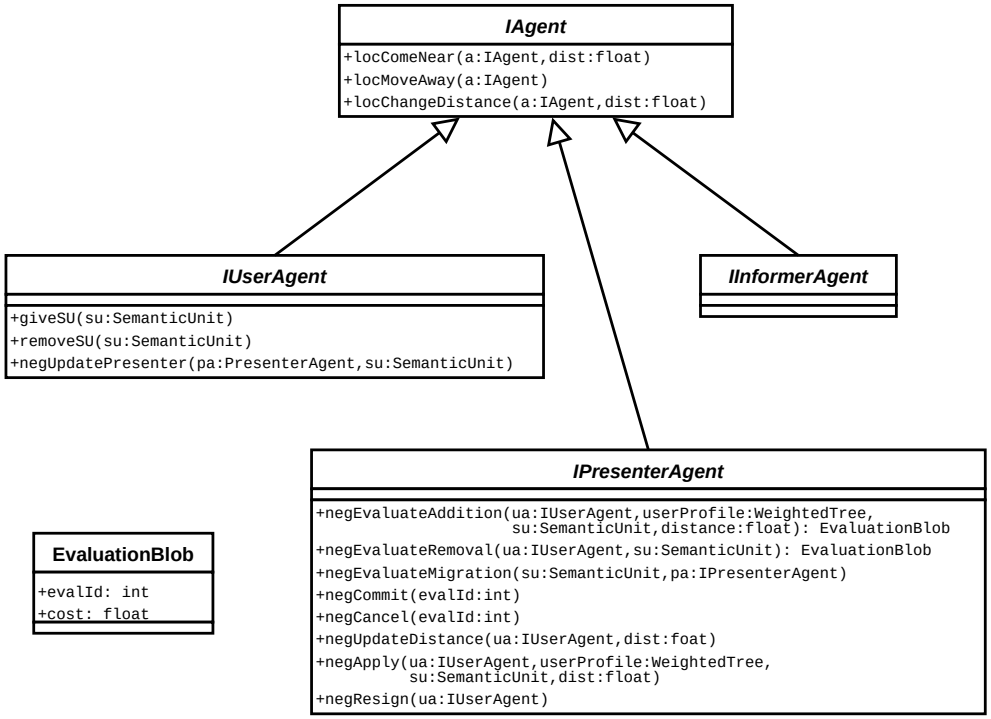


Figure 7.5 : *Diagramme UML des interfaces de base définies pour les agents.*

Comme nous venons de le voir, les noms des méthodes liées au service de localisation commencent par le préfixe `loc`. Il existe une autre sorte de méthodes (déjà évoquées dans la présentation générale), qui sont liées quant à elles aux *négociations* entre agents. Pour cette raison, les noms de ces méthodes commencent par le préfixe `neg`.

Agent informateur L'agent informateur est le plus simple des trois types d'agents envisagés ici. En effet, il n'intervient dans aucune négociation avec d'autres agents. Il se contente de fournir éventuellement des unités sémantiques à des agents utilisateurs lorsque ceux-ci pénètrent dans son espace perceptuel. La fourniture de ces unités sémantiques s'effectue via la méthode `giveSU()` des agents utilisateurs.

Lorsqu'un agent utilisateur quitte l'espace perceptuel d'un agent informateur, ce dernier *peut* lui reprendre une unité sémantique précédemment fournie (*péremption géographique*, voir p. 99). Pour cela, l'agent informateur fait appel à une méthode `removeSU()`.

Agent utilisateur Outre ces méthodes `giveSU()` et `removeSU()`, un agent utilisateur possède une méthode `updatePresenter()`, qui lui ordonne de mettre à jour son dispositif de présentation principal pour une unité sémantique donnée. Cette méthode est appelée par les dispositifs de présentation lors des migrations d'unités sémantiques.

Agent présentateur Tout d'abord, les agents présentateurs possèdent trois méthodes qui permettent à des agents utilisateurs de leur faire évaluer le coût d'opérations élémentaires. Ainsi, on peut demander à un agent présentateur d'évaluer l'ajout (`negEvaluateAddition()`)

ou le retrait (`negEvaluateRemoval()`) d'une unité sémantique. On peut également lui demander d'évaluer le coût de la migration de l'une des unités sémantiques qu'il est en train de présenter vers un autre dispositif (`negEvaluationMigration()`). Ces trois méthodes renvoient le coût correspondant, ainsi qu'un identifiant.

Si l'opération demandée est possible, cet identifiant sert à l'agent appelant pour la confirmer (méthode `negCommit()`) ou pour l'annuler (méthode `negCancel()`).

Si la présentation d'une u.s. est impossible, cela peut vouloir dire que le dispositif de présentation en question est momentanément surchargé. Dans ce cas, il est possible de déposer sa candidature pour la présentation d'une u.s., sachant que les candidats seront « élus » par ordre d'arrivée. La méthode `negApply()` permet ce dépôt de candidature. Lorsqu'une candidature est en cours, il est possible de se désister à l'aide de la méthode `negResign()`, par exemple si l'agent utilisateur a pu faire présenter son u.s. par un autre présentateur.

Enfin, lorsque l'une de ses u.s. est en cours de présentation sur un dispositif donné, un agent utilisateur peut mettre à jour la distance à laquelle il se trouve de ce dispositif en appelant la méthode `negUpdateDistance()`.

Notons que l'intersection des profils de l'utilisateur, du dispositif de présentation et de l'unité sémantique s'effectue au niveau des agents présentateurs, lors de l'évaluation de la présentation d'une u.s. En effet, les agents présentateurs obtiennent ces trois profils :

- ils connaissent le leur propre ;
- les agents utilisateurs leur fournissent le leur en paramètre lors des appels à la méthode `negEvaluateAddition()` ;
- l'unité sémantique est également fournie en paramètre à la méthode `negEvaluateAddition()` ; l'agent présentateur peut alors accéder à son profil à l'aide de la méthode `getProfile()` de cette dernière.

Implémentation Les interfaces que nous venons de décrire sont implémentées sous la forme d'un ensemble d'agents standards dans PRIAM. Ces classes implémentent les algorithmes décrits dans le chapitre 6. Les applications finales peuvent choisir de réutiliser directement ces classes, ou bien d'adapter les comportements des agents en créant des sous-classes.

7.3.3 Unités sémantiques

Les unités sémantiques sont des classes qui dérivent d'une classe de base appelée `SemanticUnit` (voir fig. 7.6), et surchargent deux méthodes :

- `getProfile()` renvoie simplement à l'appelant le profil de l'unité sémantique ;

- `generate()` génère un contenu concret à partir de l'identifiant d'une modalité et d'un ensemble de couples ⟨ attribut, valeur ⟩ qui représentent une instanciation de cette modalité.

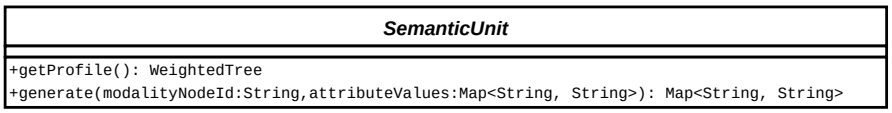


Figure 7.6 : *Diagramme UML d'une unité sémantique.*

Une unité sémantique est un objet sérialisable. Comme dans le cas des fonctions de pondération, le code de la méthode `generate()` doit être sérialisé, car il est caractéristique d'une unité sémantique donnée, et cette dernière n'a aucune raison d'être présente à l'origine ailleurs qu'à proximité des agents informateurs qui la diffusent.

7.3.4 Communications par RMI

Nous avons décidé de baser toutes les communications entre agents sur le mécanisme des RMI[7], intégré à Java. RMI s'apparente à la famille des protocoles d'appel de procédures à distance de type RPC. Certes, RMI introduit une certaine rigidité dans le sens où il impose l'utilisation d'un compilateur spécifique (`rmic`), et à l'utilisation d'un service adapté (`rmiregistry`). Cependant, ses avantages sont nombreux :

- les couches réseau sont en grandes parties masquées par RMI lui-même, donc il n'est pas nécessaire de les manipuler de façon explicite dans les programmes ;

- les envois de messages par le réseau se passent exactement comme des appels de méthodes sur des classes *locales*, ce qui évite d'alourdir inutilement les programmes, et introduit une syntaxe unifiée, très facile à lire ;

- les paramètres des méthodes peuvent être transmis de façon naturelle par le réseau, en utilisant les méthodes de sérialisation standards de Java. Il suffit pour cela que toutes les classes susceptibles de transiter entre machines différentes implémentent l'interface `Serializable`.

- tous les objets reçoivent un identifiant unique sur le réseau, de la forme `//nom_machine/nom_objet_local`, ce qui représentera l'adresse des agents dans la plate-forme PRIAM.

[7] Remote Method Invocation.

7.4 Le simulateur

7.4.1 Introduction – Motivations

Effectuer des expérimentations grandeur nature est une opération assez lourde, tant en termes de matériel à installer que de sujets volontaires à rechercher. De plus, des aspects matériels peuvent se révéler délicats à régler. Par exemple, pour réaliser les expérimentations détaillées au chapitre 8, il a été nécessaire de mettre au point tout le système d'identification par infrarouges présenté dans l'annexe C. En outre, il n'est pas raisonnable de vouloir réaliser une expérience grandeur nature, donc de mobiliser des volontaires, si les mécanismes sous-jacents n'ont pas été sérieusement testés auparavant.

De ce fait, et pour pouvoir tester relativement tôt nos algorithmes et nos implémentations, nous avons décidé de réaliser un simulateur, qui permet de faire fonctionner le système en conditions quasi réelles, et donc permet de réaliser tous les tests nécessaires sans devoir mettre en œuvre les moyens nécessaires à la réalisation d'une expérimentation grandeur nature.

Ce système permet de conduire rapidement des tests sur l'état courant de notre implémentation, sans devoir recourir à des sujets humains. Cependant, notre idée n'est pas de nous *affranchir* d'évaluations avec des sujets humains (que nous présentons au chapitre 8), mais plutôt de détecter et résoudre la majorité des problèmes *avant* de passer à cette phase.

7.4.2 Réalisation

Notre objectif était de réaliser un simulateur qui modifie le moins possible le fonctionnement normal du système. Pour ce faire, nous avons uniquement décidé d'implémenter un service de localisation spécifique, et de faire fonctionner tous les autres composants dans leurs conditions normales d'utilisation. De cette façon, nous limitons les risques de « mauvaises surprises » lors du passage du fonctionnement sur simulateur au fonctionnement en conditions réelles (chapitre 8).

Le simulateur se présente donc sous la forme d'une fenêtre dans laquelle de petites icônes représentent les agents en cours de fonctionnement (voir fig. 7.7). Les agents sont donc « localisés » dans un espace en deux dimensions. Pour se faire, ils sont encapsulés à l'intérieur d'objets spécifiques de type `XYAgent`, qui permettent de mémoriser leurs coordonnées dans l'espace en deux dimensions. Les relations de proximité sont déduites par le service de localisation à partir de l'ensemble des coordonnées des agents.

L'utilisateur peut déplacer les icônes des agents à la souris, ce qui déclenche les réactions correspondantes. Lorsqu'un agent est sélectionné, une frontière (le plus souvent un cercle) apparaît autour de lui, de façon à symboliser son espace perceptuel. En haut à droite de la fenêtre du simulateur, une boîte de propriétés permet de lire et éventuellement de modifier directement les propriétés de l'objet `XYAgent` associé à l'agent sélectionné. Cette fonctionnalité est rendue possible grâce à l'utilisation des mécanismes de *réflexivité* de Java, et par la conception sous forme de *Java bean* de la classe `XYAgent`.

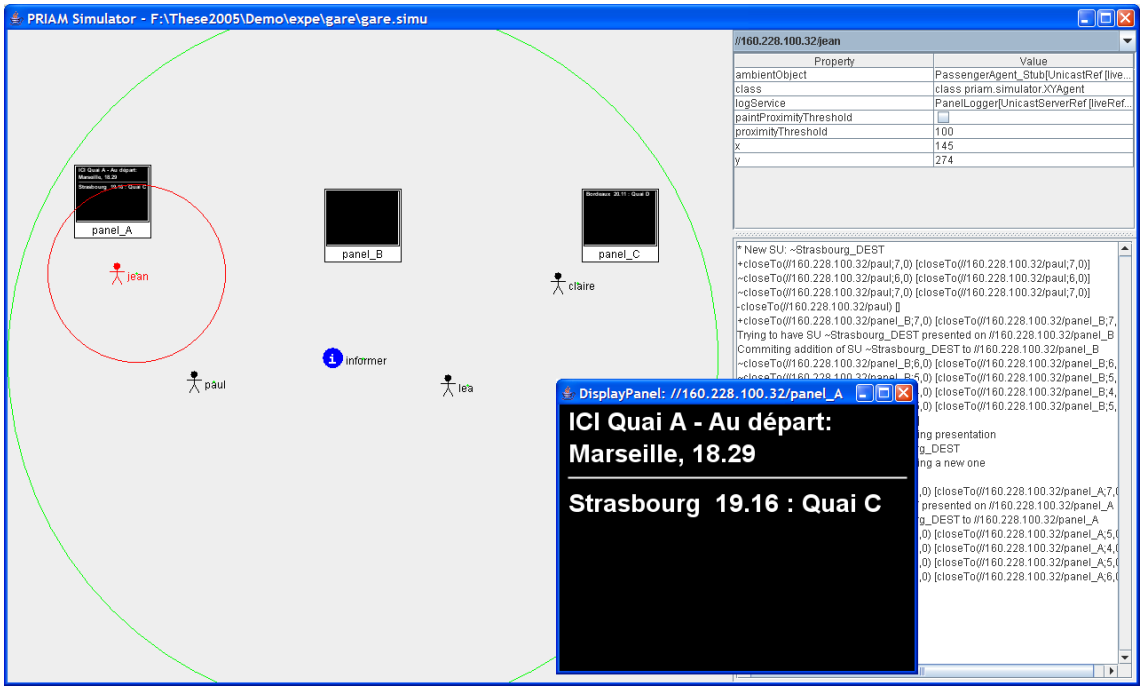


Figure 7.7 : *Interface du simulateur de PRIAM.*

En bas à droite apparaît un journal de messages de l'agent sélectionné. Cela permet à ce dernier d'afficher toutes les informations utiles au suivi du bon déroulement des opérations.

Les agents, s'ils le souhaitent, peuvent ouvrir des fenêtres supplémentaires. Par exemple, les agents présentateurs associés aux écrans ouvrent une fenêtre correspondant à leur affichage (une telle fenêtre est visible sur la figure 7.7). En réalité, cette fenêtre correspond à l'affichage habituel des écrans. En mode simulation, la fenêtre apparaît en taille réduite ; elle apparaîtrait en mode plein écran lors du fonctionnement en conditions réelles.

7.4.3 Format XML de description d'expérimentation

Sur la figure 7.7, nous voyons qu'un certain nombre d'agents sont positionnés à l'intérieur d'une « scène », d'une instance de simulation. Afin de décrire ce type de scènes, nous avons introduit un format XML de description d'expérimentations. Ce format XML permet de spécifier quels sont les agents présents, leurs adresses, leurs paramètres, et pour la simulation, leurs coordonnées dans l'espace en deux dimensions.

Nous avons donc en premier lieu introduit ce format afin de configurer le simulateur. Cependant, comme il permettait de décrire de façon pratique et précise une « configuration » d'un certain nombre d'agents, nous l'avons par la suite également utilisé pour décrire les conditions de fonctionnement des expérimentations grandeur nature décrites au chapitre 8.

Ce format de fichier de description d'expérimentation est décrit plus en détails en annexe B.6.

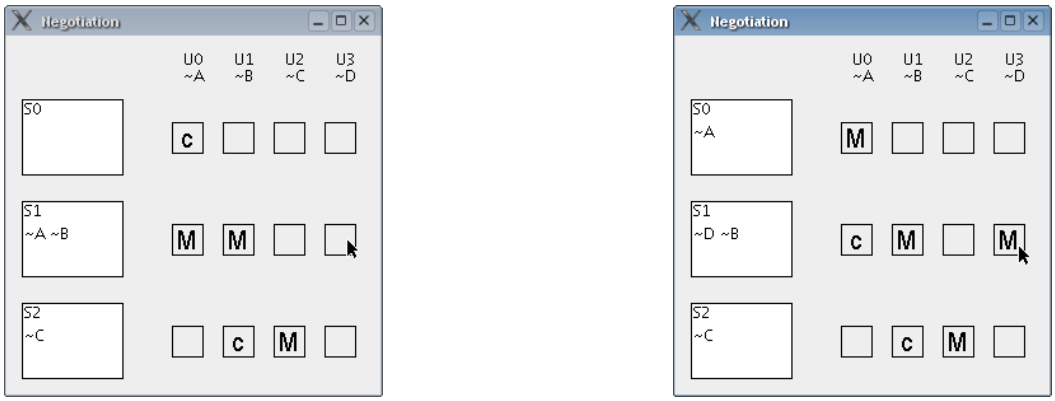
7.5 Simulateur de collaboration

Afin de tester spécifiquement les aspects de collaboration entre dispositifs de présentation (migrations d'unités sémantiques, candidature d'agents utilisateurs, etc.), nous avons également réalisé un deuxième simulateur, qui permet d'étudier ces aspects en particulier. Contrairement au précédent, ce simulateur fait abstraction de la disposition *géographique* des entités, pour ne prendre en compte qu'une notion de proximité binaire des agents.

En particulier, ce simulateur permet de mettre en évidence l'application des contraintes ergonomiques (voir section 6.3.1 page 6.3.1) : complétude (C_1), stabilité (C_2), optimisation (C_3).

La figure 7.8 montre deux copies d'écran successives de ce simulateur. La simulation représentée met en jeu quatre utilisateurs voyants (appelés `U0`, `U1`, `U2` et `U3`) et trois écrans (appelés `S0`, `S1` et `S2`). Les u.s. des quatre utilisateurs sont respectivement `A`, `B`, `C` et `D`. Dans cette simulation, on utilise des écrans spécifiés de telle sorte qu'ils ne peuvent afficher au plus que deux unités sémantiques.

La matrice des 3×4 carrés au centre de la fenêtre du simulateur représente les relations de proximité : si un utilisateur n'est pas proche d'un écran, le carré est vide. Si un écran donné est l'écran *principal* d'un utilisateur, un « M » (*main*) est dessiné dans le carré. Si un écran est proche d'un utilisateur, mais n'est pas pour autant son écran principal, un « c » (*close*) est dessiné dans le carré. Les écrans sont représentés sur la gauche : ils indiquent quelles sont les u.s. affichées.



(a) (b)

Figure 7.8 : *Le simulateur de collaboration propose une interface graphique qui permet de manipuler les relations de proximité.*

Dans un premier temps, considérons la situation initiale représentée sur la figure 7.8a. `U0` et `U1` sont proches de l'écran `S1`, et `S1` est leur écran *principal*. Ainsi, `S1` affiche les u.s. `A` et `B`. `U0` est également proche de l'écran `S0`, `U1` est également proche de l'écran `S2`, mais ce ne sont pas leurs écrans principaux. L'écran principal de l'utilisateur `U2` est `S2`. Pour le moment, `U3` n'a pas d'écran principal.

À un moment donné, l'utilisateur `U3` s'approche de l'écran `S1`. Afin de déclencher cette action dans le simulateur, l'expérimentateur clique sur la case correspondante de la « matrice » (i.e. là où se trouve le pointeur de souris sur la figure 7.8a). Pour satisfaire la contrainte C_1 (complétude), l'écran `S1` devrait afficher l'u.s. `D` en plus de `A` et `B`, mais ceci est impossible, car il ne peut afficher au maximum que deux u.s. En conséquence, `S1` choisit de transgresser la contrainte C_2 (stabilité) en faveur de la contrainte C_1, et de faire migrer l'une des deux u.s. `A` ou `B` vers respectivement l'écran `S0` ou l'écran `S2`. Afin de décider quelle migration effectuer, l'écran `S1` envoie donc des demandes d'évaluation des migrations à `S0` et `S2`, qui lui renvoient les coûts correspondants :

1. si on fait migrer l'u.s. `A` sur `S0`, la charge de ce dernier passera de 0 à 1. Le coût, défini comme dans l'exemple de la section 6.3, vaut $0 + P_{\texttt{U0}}$;
2. si on fait migrer l'u.s. `B` sur `S2`, la charge de ce dernier passera de 1 à 2. Le coût vaut $1^2 - 0^2 + P_{\texttt{U1}} = 1 + P_{\texttt{U1}}$.

Il est raisonnable de supposer que $P_{\texttt{U0}} = P_{\texttt{U1}}$, et donc `S1` choisit donc de faire migrer l'u.s. `A` sur l'écran `S0`. C'est bien ce résultat qui est observable sur la figure 7.8b. Parmi les deux migrations possibles, c'est bien *intuitivement* celle qui provoque le moins de surcharge des écrans qui a été choisie.

Les tests effectués grâce à ce simulateur (du type de l'exemple ci-dessus) nous ont permis de vérifier le comportement des aspects coopératifs de nos algorithmes dans diverses situations. Nous avons ainsi pu vérifier que le cahier des charges est bien suivi : les trois contraintes sont respectées, avec les priorités voulues.

7.6 Conclusion

Au final, les simulateurs s'avèrent constituer un élément essentiel de PRIAM, car ils permettent de vérifier sans devoir recourir à une expérimentation grandeur nature des points aussi variés que :

- le bon fonctionnement des agents ;
- le bon déroulement des interactions entre agents ;
- la correction de leurs profils ;
- etc.

De plus, ces simulateurs nous ont été particulièrement utiles pour la mise au point de la plate-forme elle-même. Ils nous ont permis de tester facilement des situations qu'il aurait été difficile de mettre en œuvre s'il avait fallu réaliser des expériences grandeur nature.

Cependant, nous avons éprouvé le besoin de procéder à une phase d'évaluations de notre travail en conditions quasi réelles, avec des utilisateurs humains, de façon à obtenir une véritable validation, à la fois du fonctionnement de la plate-forme elle-même, mais également des résultats indiqués par le simulateur. Ces expériences grandeur nature sont traitées dans le chapitre suivant. Ce dernier présente également deux démonstrations effectuées sur simulateur, qui permettent de mettre en évidence des aspects de la plate-forme non traités par les expériences.

Chapitre 8

Expérimentations et démonstrations

Nous avons cherché à évaluer la plate-forme PRIAM à l'aide d'un ensemble d'expérimentations et de démonstrations. Dans un premier temps, le simulateur nous a permis de vérifier le bon fonctionnement des algorithmes. Cependant, nous avons dans un second temps cherché à évaluer le gain réel apporté par notre système : nous avons donc réalisé des expérimentations grandeur nature, c'est-à-dire avec des sujets humains. Nous avons mené deux expérimentations :

Recherche d'informations dans une liste : par exemple, liste d'affichage des résultats à un examen scolaire, ou liste d'affichage de vols d'avion. Dans ces cas, plusieurs utilisateurs humains sont censés se trouver à proximité d'un unique dispositif de présentation ;

Recherche de direction dans un environnement public : par exemple, nous avons reproduit un environnement de gare, dans lequel chaque sujet devait se rendre sur le quai où son train était annoncé au départ. Dans ce cas, un utilisateur unique interagit successivement avec plusieurs dispositifs physiques de présentation.

Dans ces expérimentations, nous ne tenons pas compte des différences entre utilisateurs. Ainsi, tous les sujets avaient une acuité visuelle suffisante pour pouvoir lire des informations présentées à distance raisonnable.

Nous n'avons pas mené d'expérimentations grandeur nature sur le caractère multimodal de notre plate-forme : ces aspects ont par contre été testés grâce au simulateur. Nous avons ainsi réalisé deux démonstrations qui mettent en avant les aspects multimodaux du système :

- utilisation de modalités différentes selon le type d'utilisateur (voyant, non-voyant) ;
- valeurs différentes données aux attributs des modalités par le processus d'instanciation.

8.1 Recherche d'informations dans une liste[1]

Dans cette expérience, on affiche une liste d'informations parmi lesquelles chacun des sujets doit rechercher un élément d'information précis qui le concerne. Notre idée est de comparer le temps mis par un utilisateur pour retrouver son information, selon que la liste est statique (liste sur papier ou écran statique), ou bien personnalisée et dynamique, i.e. gérée par la plate-forme PRIAM. Nous souhaitons également mesurer l'influence sur le temps de recherche du nombre de personnes qui se trouvent autour de la zone d'affichage. Nous pressentons en effet que plus il y aura de personnes devant l'écran, plus la recherche des informations risquera d'être difficile pour chacune d'entre elles.

8.1.1 Description générale

À un signal donné, de un à huit sujets s'approchent d'un écran ou d'un panneau d'affichage, et s'emploient à y rechercher une information particulière. Lorsque l'un d'entre eux a identifié son information, il la mémorise, lève la main et s'éloigne tout de suite du panneau. Il écrit alors cette information sur un formulaire papier qui lui a été remis en début d'expérience.

De cette façon, en filmant le déroulement de l'expérience (voir fig. 8.1), nous avons pu facilement mesurer le temps de recherche de chaque utilisateur. Cette quantité est définie comme étant le temps qui sépare l'entrée de l'utilisateur sur la « scène » de l'expérience (i.e. la zone située à proximité du dispositif d'affichage) et le moment où il lève la main.

Figure 8.1 : *Les sujets s'approchent de l'écran puis lèvent la main lorsqu'ils ont trouvé leur information. Image issue du film de l'expérience.*

Les formulaires remplis par les utilisateurs nous ont permis de vérifier l'exactitude des informations trouvées, et ainsi d'éliminer des résultats les expériences dans lesquelles l'utilisateur s'était trompé. Cependant, ces cas d'erreurs étaient très rares, et ne nous ont pas permis d'établir de statistiques intéressantes sur les conditions dans lesquelles ils sont survenus. En

[1] Notons que si cette évaluation ressemble à l'expérience mentionnée dans l'annexe A, ses objectifs sont différents (voir justification page 201).

fait, la majorité des erreurs ont été le fait d'un unique utilisateur, que nous qualifierons de « distrait ».

Dans chaque cas, nous réalisons d'abord une expérience « témoin » dans laquelle les listes d'informations sont statiques. Puis, nous introduisons les listes dynamiques : dans ce cas, les informations affichées par les écrans ne concernent que les utilisateurs situés devant ces derniers. La détection des utilisateurs par le système est réalisée à l'aide du système de badges à infrarouges décrit en annexe C.

Nous avons réalisé deux déclinaisons de cette expérience :

- d'abord, recherche d'un nom et d'une note (nombre entier compris entre 0 et 20) sur un tableau d'affichage des résultats à un examen ;
- ensuite, recherche d'un numéro de vol et d'une porte d'embarquement.

8.1.2 Affichage des résultats d'examen

Dans cette expérience, les utilisateurs doivent chercher la note d'examen d'une personne donnée dans une liste. Dans un premier temps, la recherche s'effectue dans une liste sur papier, qui contient un grand nombre de noms (par exemple, les noms de tous les candidats d'un centre d'examen du baccalauréat). Dans un second temps, la recherche s'effectue sur un écran dynamique.

Les noms sont toujours triés par ordre alphabétique. On fournit un nom aux participants à l'expérience, qui doivent alors rechercher la note correspondante. Chaque participant a recommencé l'exercice plusieurs fois, mais entre chaque passage, les listes étaient changées.

Expérience témoin Nous avons réalisé deux expériences témoins :

1. dans la première, il s'agissait de rechercher une note parmi une liste de 220 noms, ce qui représentait 4 colonnes, comme sur la figure 8.2. Les sujets ont recommencé l'expérience plusieurs fois, en gardant le même nom. Les résultats sont donnés sur le tableau 8.1 ;
2. dans la deuxième (effectuée avec une série de sujets différents), il s'agissait cette fois-ci de rechercher une note parmi une liste de 450 noms, soit huit colonnes. Les sujets ont recommencé l'expérience plusieurs fois, mais le nom à rechercher changeait à chaque fois. Les résultats sont donnés sur le tableau 8.2.

Dans le premier cas, le temps de recherche varie assez peu avec le nombre de personnes en train de chercher une information sur la liste, même si on note une légère tendance à l'augmentation. Cet allongement du temps de recherche paraît normal, car les utilisateurs se gênent mutuellement. Cependant, l'augmentation du temps est modérée car les utilisateurs

Nombre de personnes	Temps moyen (s)	Écart-type	Temps minimum	Temps maximum
1	6,00	0,00	6,00	6,00
2	5,00	0,00	5,00	5,00
3	5,67	0,94	5,00	7,00
4	6,50	1,32	4,00	8,00
5	5,40	1,36	4,00	7,00
6	6,50	0,96	5,00	8,00
7	6,29	2,05	4,00	10,00
8	6,88	1,96	4,00	9,00

Tableau 8.1 : *Temps de recherche d'une note sur un écran statique, série 1.*

Nombre de personnes	Temps moyen (s)	Écart-type	Temps minimum	Temps maximum
1	6,00	0,00	6,00	6,00
2	7,00	0,00	7,00	7,00
3	9,00	2,45	6,00	12,00
4	6,71	1,48	4,00	8,00
5	11,60	6,92	5,00	25,00
6	9,00	1,22	8,00	11,00
7	11,50	6,68	5,00	25,00
8	10,38	4,21	4,00	16,00

Tableau 8.2 : *Temps de recherche d'une note sur un écran statique, série 2.*

ont probablement retenu d'une fois à l'autre la position géographique du nom à rechercher dans la liste[2], ce qui leur a permis d'être plus efficaces.

C'est pourquoi dans la deuxième expérience, les utilisateurs devaient rechercher un nom différent à chaque fois. On note donc une plus grande variabilité des résultats, et une plus forte tendance à l'augmentation du temps avec le nombre d'utilisateurs présents, qui est renforcée par la présence d'un plus grand nombre de noms dans la liste (450 au lieu de 220).

Version augmentée Dans cette expérience, les notes sont affichées dynamiquement sur un écran (voir fig. 8.3). Au départ, l'écran n'affiche rien du tout. Des informations apparaissent et disparaissent au fur et à mesure que des utilisateurs s'approchent ou s'éloignent. Les résultats sont donnés par le tableau 8.3.

[2]Les listes étaient changées à chaque fois, mais globalement, même sur des populations différentes, la position d'un nom donné est toujours à peu près la même dans une liste de noms. C'est pourquoi une personne nommée *Barbier* recherchera d'office son nom en début de liste, tandis qu'une personne nommée *Weber* recherchera d'emblée en fin de liste.

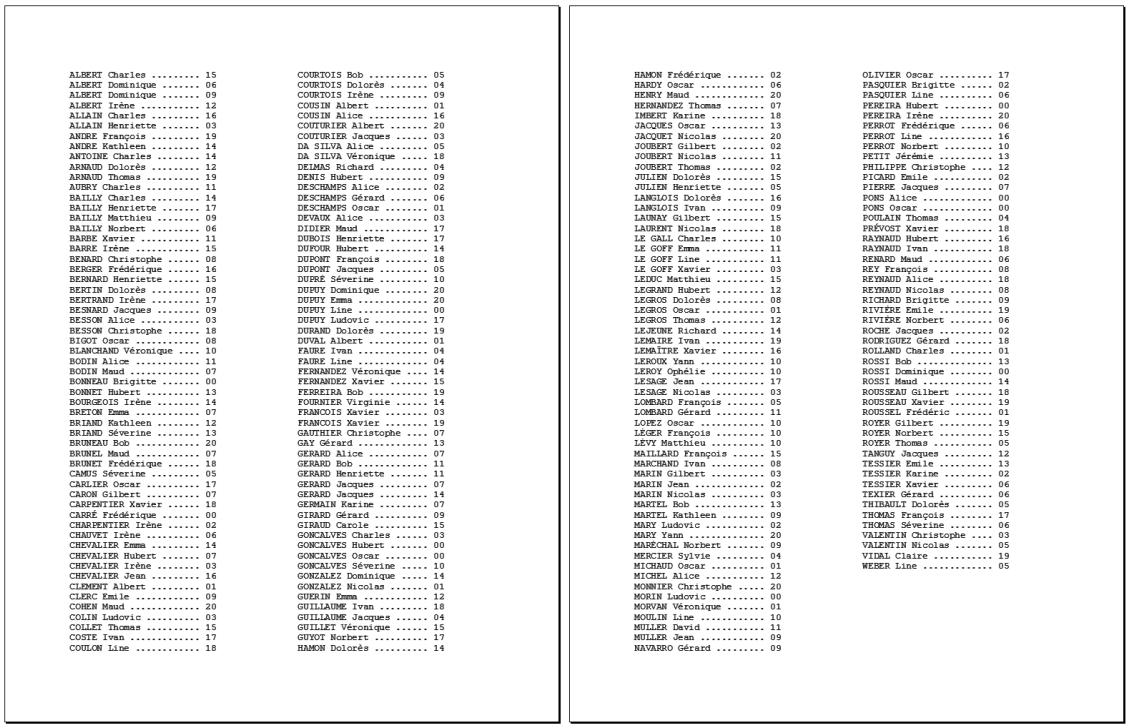


Figure 8.2 : *Liste statique de notes à un examen, sur papier.*

Figure 8.3 : *Affichage dynamique sur un écran des résultats à un examen.*

D'emblée, on remarque que les temps de recherche sont très nettement inférieurs à ceux observés pour la version *statique* de l'expérience, ce qui démontre l'intérêt d'un système *dynamique* et personnalisé. On note également que l'augmentation du temps de recherche en fonction du nombre de personnes présentes simultanément est plus notable qu'avec la version statique. En effet, dans la version statique, le nombre d'informations élémentaires affichées est fixe (220 ou 450 selon l'expérience), tandis qu'ici, il est par définition *égal*[3] au nombre de personnes présentes. Avec l'augmentation du nombre d'informations présentées à l'écran, il est donc logique que le temps de recherche augmente. La figure 8.4 propose une comparaison des résultats dans les trois cas.

[3]Plus précisément, il est inférieur ou égal au nombre de personnes présentes, car il est possible que deux personnes soient intéressées par la même information, par exemple si elles viennent chercher la note de la même personne. Cependant, nous avons construit cette expérience de telle façon que chacun cherche une information *différente*.

Nombre de personnes	Temps moyen (s)	Écart-type	Temps minimum	Temps maximum
1	1,00	0,00	1,00	1,00
2	2,00	0,71	1,00	3,00
3	2,25	1,39	0,00	5,00
4	3,30	1,95	1,00	7,00
5	2,00	1,48	0,00	5,00
6	2,89	1,73	0,00	6,00
7	2,43	1,18	1,00	4,00
8	3,50	2,15	0,00	7,00

Tableau 8.3 : *Temps de recherche d'une note sur un écran dynamique.*

8.1.3 Aérogare

Cette expérience ressemble fortement à la précédente, où les couples ⟨ nom, note ⟩ sont remplacés par des triplets ⟨ vol, heure, porte d'embarquement ⟩. Un écran affiche les informations concernant les vols. Dans un premier temps, il est statique (expérience témoin), puis il devient dynamique (version utilisant PRIAM).

Lors de chaque exercice, on fournit à l'utilisateur un numéro de vol ainsi qu'un horaire de départ. L'utilisateur doit alors trouver sur les écrans la lettre de la porte d'embarquement correspondante, et la noter par écrit. Sur les écrans, les vols sont classés par ordre chronologique (voir fig. 8.5).

Expérience témoin Tout d'abord, les utilisateurs recherchent les informations concernant leur vol parmi un nombre fixe d'informations. Nous avons réalisé deux expériences :

1. recherche d'une information parmi 12 : voir tableau 8.4
2. recherche d'une information parmi 20 (voir fig. 8.5) : voir tableau 8.5.

Les résultats sont comparables à ceux de l'expérience des notes : les temps ont tendance à être plus longs lorsque le nombre de personnes présentes simultanément augmente, sans pour autant que cette tendance soit très franche. On note par contre que les temps de recherche parmi 20 informations sont sensiblement plus longs que les temps de recherche parmi 12 informations.

Version augmentée Comme dans l'expérience des notes, on n'affiche ici que les informations relatives aux utilisateurs situés à proximité de l'écran (voir fig. 8.6). Notons qu'il peut arriver ici que deux utilisateurs cherchent la même information (i.e. ils sont censés prendre le même vol). Dans ce cas, en mode dynamique, l'écran est encore moins chargé que dans l'expérience précédente. Les résultats sont donnés par le tableau 8.6.

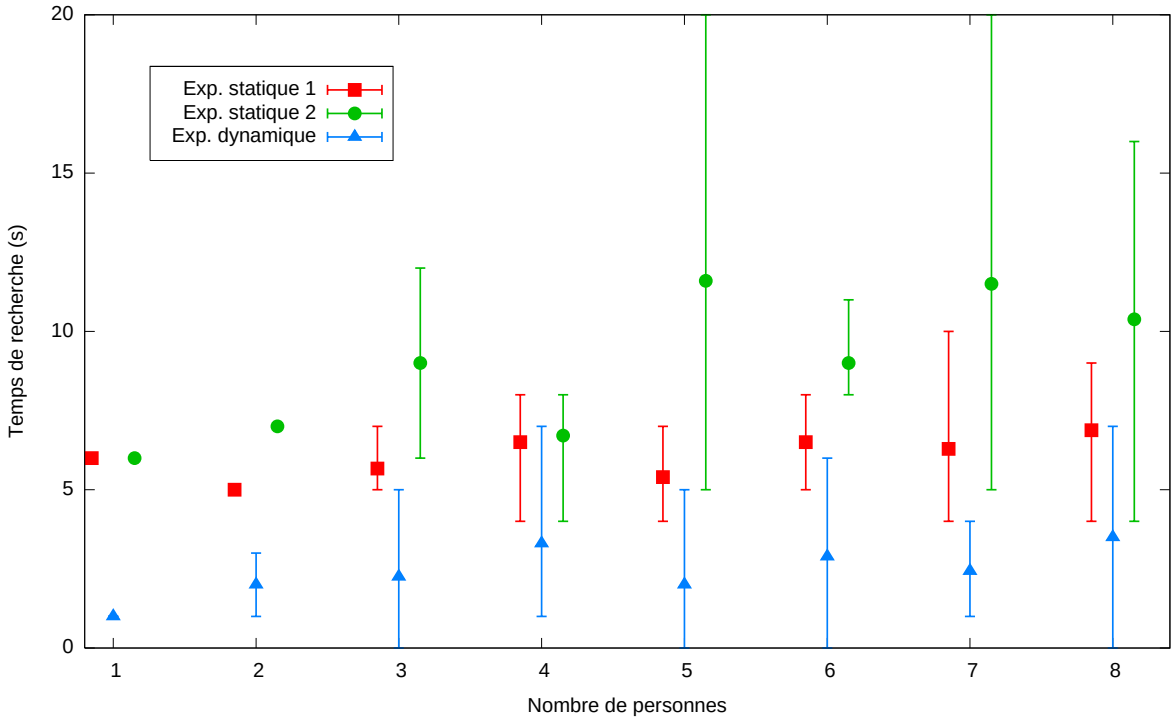


Figure 8.4 : *Comparaison des temps de recherche d'une note en fonction du nombre de personnes présentes simultanément. Pour chaque cas étudié, un segment figure sur le graphique : il indique les temps minimum et maximum. Le point situé sur le segment représente le temps moyen. Pour des raisons de lisibilité, les résultats de l'expérience statique n°1 sont un peu décalés sur la gauche, tandis que ceux de l'expérience statique n°2 sont un peu décalés sur la droite. Les résultats de l'expérience dynamique figurent donc entre les résultats des deux expériences statiques.*

Ces résultats conduisent aux mêmes conclusions que ceux de l'expérience de la section 8.1.2 : la recherche d'une information est bien plus rapide lorsque ne sont présentées que les informations relatives aux utilisateurs situés à proximité immédiate (voir figure 8.7).

8.1.4 Perception subjective des expériences

Afin d'analyser la perception qu'ont eu les participants de ces deux expériences, nous leur avons soumis un questionnaire qui est reproduit en annexe D.

Tout d'abord, la plupart des utilisateurs ont préféré les versions dynamiques des exercices : en général, ils trouvent cela « pratique », « facile », voire « amusant ». La recherche des informations leur semble généralement plus facile, car il y a moins d'items à parcourir, donc moins de *bruit* pour noyer son information d'intérêt. Notons toutefois que la plupart de nos sujets avaient une vingtaine d'années, et du fait, étaient certainement plus réceptifs aux comportements dynamiques que la moyenne de la population. Ainsi, une personne un peu

Nombre de personnes	Temps moyen (s)	Écart-type	Temps minimum	Temps maximum
1	3,00	0,00	3,00	3,00
2	4,00	2,00	2,00	6,00
3	4,00	1,41	3,00	6,00
4	4,63	1,22	2,00	6,00
5	5,80	0,98	4,00	7,00
6	8,67	4,38	4,00	17,00
7	7,67	2,56	5,00	13,00
8	6,88	4,28	4,00	18,00

Tableau 8.4 : *Temps de recherche d'une porte d'embarquement sur un écran statique, série 1.*

Nombre de personnes	Temps moyen (s)	Écart-type	Temps minimum	Temps maximum
1	6,00	0,00	6,00	6,00
2	5,00	1,00	4,00	6,00
3	4,67	0,47	4,00	5,00
4	8,00	2,65	5,00	14,00
5	7,40	3,26	3,00	11,00
6	8,00	4,52	5,00	17,00
7	8,43	2,92	5,00	13,00
8	6,00	2,20	3,00	10,00

Tableau 8.5 : *Temps de recherche d'une porte d'embarquement sur un écran statique, série 2.*

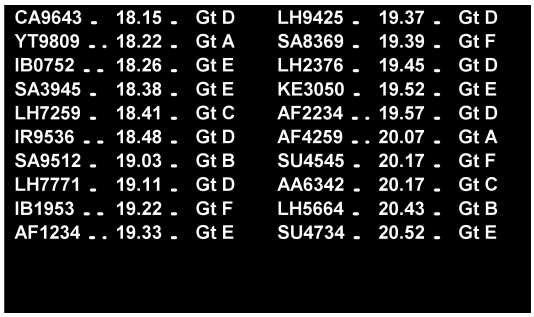


Figure 8.5 : *Affichage statique d'une série de vols.*

plus âgée a définitivement préféré les affichages statiques, tout en se définissant comme étant « linéaire[4] ».

Cependant, certaines personnes ont été gênées par un aspect des affichages dynamiques : le fait que la liste soit périodiquement réordonnée (lors de chaque arrivée ou départ d'utilisateur

[4]Cette personne dit apprécier les recherches et les narrations unidimensionnelles. Par exemple, elle n'aime pas les bandes dessinées pour leur manque de linéarité, qu'elle trouve perturbant.

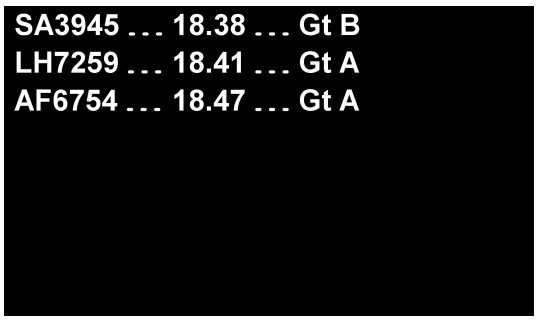


Figure 8.6 : *Affichage dynamique des vols relatifs aux usagers situés à proximité.*

Nombre de personnes	Temps moyen (s)	Écart-type	Temps minimum	Temps maximum
1	1,50	0,50	1,00	2,00
2	2,75	1,92	1,00	6,00
3	2,83	0,69	2,00	4,00
4	3,31	1,49	0,00	6,00
5	3,00	1,73	0,00	6,00
6	2,82	1,85	1,00	7,00
7	3,29	1,94	0,00	7,00
8	4,06	2,34	1,00	10,00

Tableau 8.6 : *Temps de recherche d'une porte d'embarquement sur un écran dynamique.*

à proximité du dispositif d'affichage). En effet, il arrivait qu'une ligne donnée se déplace brusquement sur l'écran (du fait de l'arrivée d'un nouvel utilisateur et du maintien du tri des lignes en permanence), alors même qu'un utilisateur était en train de la lire. Ces fluctuations donnent un effet de *clignotement* qui est assez perturbant pour les utilisateurs. Dans la section 9.2, nous introduisons des ébauches de solution pour résoudre ce problème.

Les méthodes de recherche de l'information mises en œuvre ont été très classiques. Pour rechercher un nom, les personnes ont d'abord essayé de localiser sa position probable sur la liste, et à partir de cet endroit, ils ont effectué une recherche alphabétique linéaire. En ce qui concerne les numéros de vols, la plupart des sujets ont commencé par rechercher l'heure de départ, puis ont vérifié (ou départagé les vols partant à la même heure) à l'aide du numéro de vol.

De ce fait, la plupart des utilisateurs ont trouvé la recherche de numéros de vols plus difficile, et donc plus longue, car elle doit se faire en deux temps : d'abord rechercher l'heure, puis vérifier le numéro de vol. À l'inverse, la recherche de noms est plus rapide, car elle peut se faire en un seul temps. De plus, la *longueur* du nom à rechercher peut faciliter la recherche car il s'agit d'une sorte d'*empreinte* qui permet d'emblée d'éliminer les noms trop dissemblables sans même chercher à les lire. Ceci n'est pas possible pour les numéros de vol, car toutes les données sont *formatées* sur une longueur fixe.

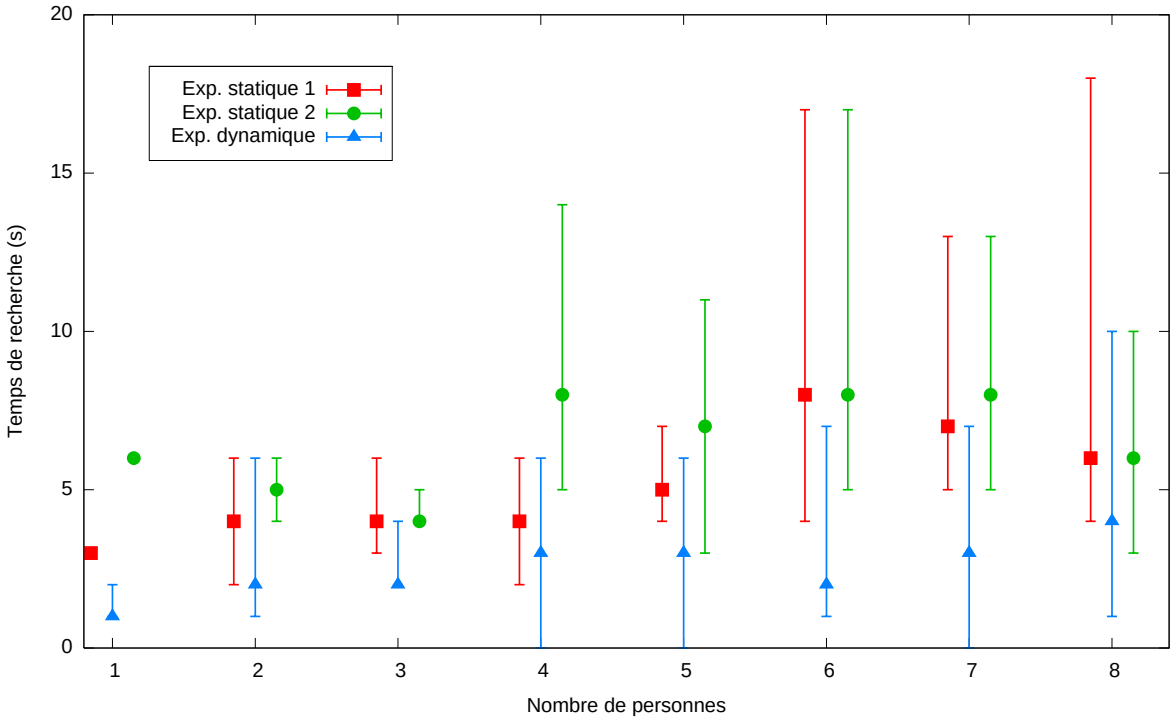


Figure 8.7 : *Comparaison des temps de recherche d'un vol en fonction du nombre de personnes présentes simultanément. Pour chaque cas étudié, un segment figure sur le graphique : il indique les temps minimum et maximum. Le point situé sur le segment représente le temps moyen. Pour des raisons de lisibilité, les résultats de l'expérience statique n°1 sont un peu décalés sur la gauche, tandis que ceux de l'expérience statique n°2 sont un peu décalés sur la droite. Les résultats de l'expérience dynamique figurent donc entre les résultats des deux expériences statiques.*

La plupart des utilisateurs pensent que ce système peut être utile en pratique. Cependant, ils soulignent le fait que ses bénéfices sont visibles uniquement lorsqu'un petit nombre d'utilisateurs se trouvent à proximité d'un écran. En effet, si un grand nombre de personnes se trouvent rassemblées, les écrans afficheront également un grand nombre d'informations, et donc le gain sera nul par rapport à un système statique. Dans ce sens, l'un des sujets nous fait remarquer qu'il faudrait éviter que les passants non intéressés par les informations puissent perturber l'affichage des écrans.

8.1.5 Notes d'implémentation

Nous avons vu jusqu'ici le fonctionnement global de l'expérience de recherche d'information dans une liste, réalisée à l'aide de la plate-forme PRIAM. Dans cette section, nous voyons plus en détails comment cette expérience a été implémentée sur notre plate-forme, notamment en termes d'agents.

Le dispositif de présentation consistait en un simple ordinateur portable à grand écran (17 pouces). Pour des raisons de simplicité, et notamment pour ne pas dépendre d'une connexion

réseau pour la réalisation de l'expérience, tous les agents fonctionnaient sur cet ordinateur portable. Les agents suivants ont été mis en œuvre :

- un agent informateur chargé d'attribuer les notes aux différents utilisateurs. Il est réalisé par une classe assez simple, qui implémente l'interface `IInformerAgent`. Comme tous les autres agents de PRIAM, cet agent dispose d'un constructeur capable de créer une instance à partir d'une série de couples ⟨ attribut, valeur ⟩. Les affectations de notes peuvent lui être transmises par ce moyen, donc au final, les notes peuvent être stockées dans le fichier XML décrivant l'expérience. Cet agent informateur fournit leurs unités sémantiques à tous les agents utilisateurs connus : son espace de rayonnement est donc égal à l'espace d'utilisation du système ;

- les utilisateurs potentiels sont simplement représentés chacun par un agent utilisateur standard de PRIAM. Huit de ces agents sont donc instanciés ;

- un agent présentateur capable d'afficher les notes. Nous avons utilisé un dispositif de présentation adapté à la présentation de données tabulées, appelé `DisplayPanel`.

Nous avons également défini une classe d'unités sémantiques destinée à représenter une note. Ces unités sémantiques sont capables de générer un contenu concret à l'intention d'un dispositif de présentation textuel, par exemple le `DisplayPanel`.

Afin que les agents utilisateurs détectent la proximité de l'écran, nous avons créé un service de localisation adapté à notre système de badges. Ce service est appelé `IRLocalizationService`. Dès que le récepteur à infrarouges détecte un badge à proximité, les agents correspondant à l'écran et à l'utilisateur en question reçoivent une notification de proximité (message `locComeNear`).

Pour qu'un `IRLocalizationService` puisse effectuer la liaison entre les identifiants émis par les badges et les adresses des agents utilisateurs correspondants, un *fichier de correspondances* lui est fourni. Il s'agit d'un fichier XML qui contient en particulier des éléments du type :

```
<mapping id="3" address="_LOCALHOST/bob" />
```

Cette ligne explicite une identité entre le porteur du badge d'identifiant n°3 et l'agent utilisateur d'adresse locale `bob`.

8.2 Recherche de direction

8.2.1 Introduction

Les deux expériences précédentes ont démontré l'apport de notre système à la recherche d'une information dans une liste, y compris par plusieurs utilisateurs simultanément. Nous cherchons maintenant à évaluer en quoi il peut constituer une aide lors de la recherche d'une direction.

Nous prenons comme exemple une configuration de gare classique (voir fig. 8.8). Un hall d'accueil permet d'accéder à un passage souterrain (ou bien une passerelle), qui à son tour donne accès aux différents quais par l'intermédiaire d'escaliers. Dans le hall d'accueil, un panneau d'affichage général indique les horaires et les quais de départ de tous les trains. De plus, au niveau de chaque escalier, un moniteur est situé dans le passage souterrain : il rappelle la liste des trains au départ sur le quai correspondant.

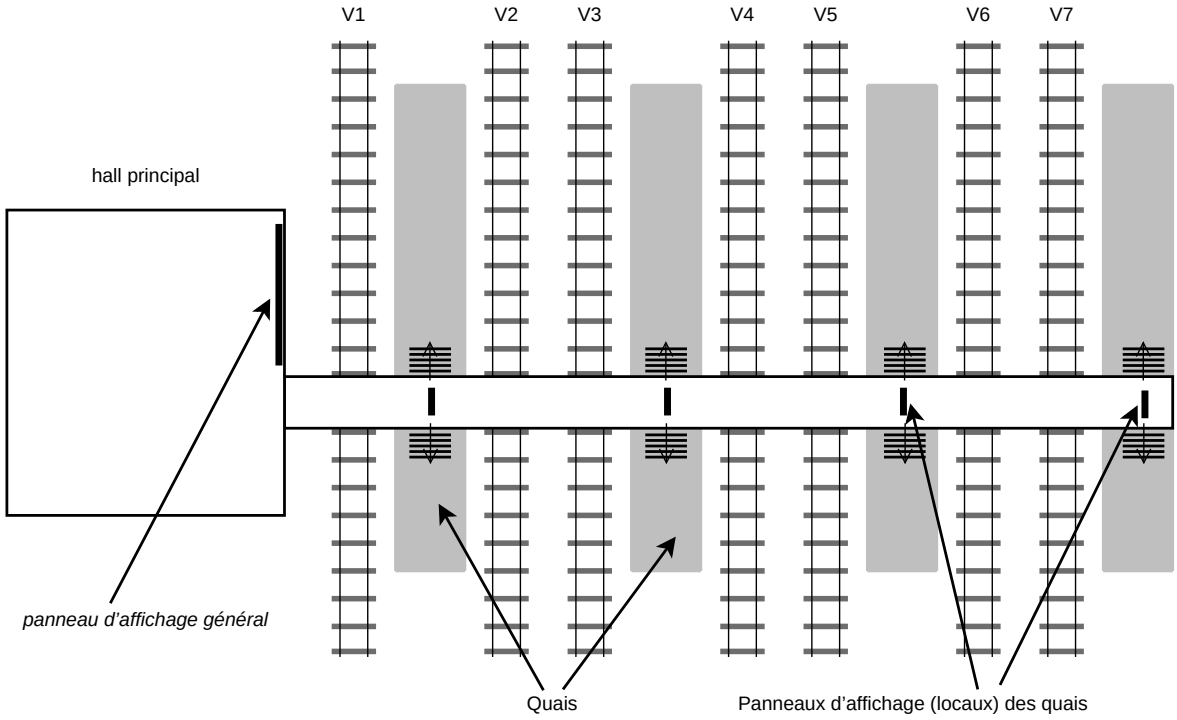


Figure 8.8 : *Disposition classique d'une gare.*

Cette organisation peut paraître assez complète, et de nature à guider parfaitement un voyageur. Par exemple, lorsqu'un usager arrive en gare, il commence par consulter le panneau d'affichage général, qui lui donne son numéro de quai. Il peut alors emprunter le passage souterrain. Au niveau de l'escalier qui mène à son quai, un moniteur lui confirme la destination. Cependant, ce schéma ne tient pas compte des voyageurs en correspondance. En effet, ces derniers descendent d'un train sur l'une quelconque des voies, et doivent se diriger vers une autre voie. Sans information supplémentaire, ils doivent donc :

- soit se rendre dans le hall, consulter le panneau d'affichage général, et alors réemprunter le passage souterrain pour se rendre sur leur quai de destination ;
- soit se diriger dans une destination quelconque dans le passage souterrain, quitte à faire demi-tour pour aller explorer l'autre direction s'ils ne se sont pas dirigés du bon côté. En effet, ils ont dans ce cas en moyenne une chance sur deux de se tromper.

Dans les deux cas, ces stratégies sont sous-optimales, car les usagers sont obligés d'effectuer des déplacements inutiles. Non seulement ces déplacements sont fatigants, notamment

lorsqu'on transporte des bagages, mais ils sont *stressants* psychologiquement si la correspondance est courte.

Il faudrait donc disposer d'un système d'information qui indique d'emblée leur quai de destination aux utilisateurs, sans qu'ils soient obligés d'effectuer des parcours inutiles. Certes, les messages vocaux sur les correspondances, diffusés lors de l'arrivée des trains en gare, sont censés remplir cet office, mais bien souvent ils ne sont pas compris ni même entendus par les voyageurs. Nous proposons donc d'afficher sur les écrans du passage souterrain, *en plus des informations habituelles* (trains au départ du quai correspondant), les informations relatives aux trains des usagers qui s'approchent de ces écrans.

8.2.2 Description générale

Nous avons installé cinq écrans (des ordinateurs portables en réalité) dans un couloir de notre laboratoire, selon la configuration de la figure 8.9. Chacun de ces écrans correspond à un quai, numéroté de A à E. Les usagers peuvent partir de l'une des extrémités du couloir (repères 0 et 2), ou bien d'une position « médiane » (repère 1). Cette position médiane de départ ne se trouve pas précisément au milieu du couloir, mais elle se justifiait de par la configuration intrinsèque des lieux.

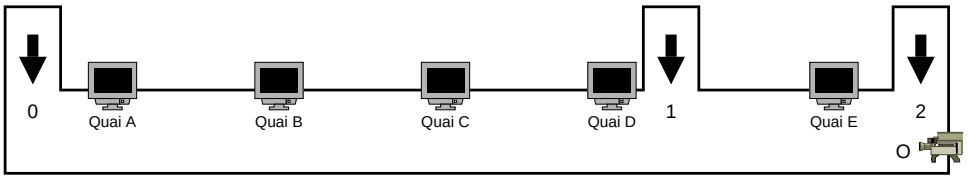


Figure 8.9 : *Installation réalisée pour l'expérience de recherche de direction. Le couloir de notre laboratoire figurait un passage souterrain d'une gare.*

Dans deux cas (écrans statiques qui n'affichent que le train au départ sur leur quai, ou bien écrans dynamiques qui affichent des informations personnalisées), nous voulons étudier les déplacements des utilisateurs pour trouver leur quai, en fonction des différents points de départ possibles. Comme dans les expériences précédentes, nous avons filmé[5] les expérimentations de façon à « segmenter » par la suite les déplacements des utilisateurs (voir fig. 8.10). Lors de cette segmentation, nous avons identifié deux types de déplacements élémentaires :

- avancée du quai Q_1 au quai Q_2. Nous notons ce déplacement $Q_1 \rightarrow Q_2$. Exemple : A $\rightarrow$ B ;
- demi-tour au niveau du quai Q. Nous notons ce déplacement $Q \circlearrowleft$. Exemple : C $\circlearrowleft$.

Ces deux types de déplacements permettent de décrire complètement la trajectoire des utilisateurs. Il nous a semblé bien plus pertinent de réaliser une telle segmentation que de mesurer le temps mis par les utilisateurs pour atteindre leur quai de destination, car ce temps peut

[5]La caméra était située au niveau de la position d'observation notée O sur la figure 8.9.

Figure 8.10 : *Extrait du film de l'expérience de recherche de direction.*

dépendre de la vitesse de déplacement des utilisateurs, ce qui n'est absolument pas un paramètre pertinent pour notre étude. En effet, nous ne voulons pas que les mesures soient faussées par une marche plus ou moins rapide des sujets.

Dans chaque expérience, les utilisateurs devaient chercher le quai du train pour Lyon, en partant de l'un des trois repères 0, 1 et 2. Lorsqu'ils l'avaient trouvé, ils devaient s'arrêter devant l'écran correspondant et lever la main.

Pour l'exploitation des résultats, nous introduisons tout d'abord la notion de *longueur* d'un chemin. La longueur d'un chemin est égale au nombre de déplacements élémentaires sur ce chemin. Nous pouvons alors définir la *longueur relative* du chemin parcouru par un utilisateur comme étant égale au quotient de la longueur ℓ_u du chemin effectivement parcouru par l'utilisateur par la longueur ℓ_o du chemin optimal[6].

Par exemple, supposons que l'utilisateur se rende du repère 1 au quai B. Le chemin optimal est :

$$1 \rightarrow \text{C},\ \text{C} \rightarrow \text{B}$$

On a donc $\ell_o = 2$. Supposons maintenant que l'utilisateur parcoure le chemin suivant :

$$1 \rightarrow \text{C},\ \text{C} \rightarrow \text{D},\ \text{D} \rightarrow \text{E},\ \text{D} \circlearrowleft,\ \text{E} \rightarrow \text{D},\ \text{D} \rightarrow \text{C},\ \text{C} \rightarrow \text{B}$$

Dans ce cas, $\ell_u = 7$. La longueur relative de ce chemin est donc $\frac{\ell_u}{\ell_o} = \frac{7}{2} = 3{,}5$. Cette longueur relative permet de caractériser n'importe quel chemin parcouru par rapport à l'optimum, sans qu'entre en compte l'éloignement entre le point de départ et le point d'arrivée. Nous jugeons donc qu'il s'agit d'un bon critère de comparaison entre expériences différentes.

8.2.3 Expérimentation à un utilisateur

Dans ces expériences, un seul utilisateur à la fois cherche son chemin.

[6] Le *chemin optimal* est celui qui comporte le moins de déplacements élémentaires.

Expérience témoin Dans cette expérience, les écrans affichent uniquement des informations concernant le quai qui leur correspond (voir fig. 8.11). Les résultats sont donnés dans le tableau 8.7. On observe une grande disparité des résultats :

- lorsque l'utilisateur part de l'une des extrémités du couloir (repères 0 ou 2), la longueur relative moyenne est 1, ce qui montre que dans ce cas, les trajets sont optimaux. En effet, il suffit dans ce cas de suivre le couloir dans la seule direction possible, et l'utilisateur finit forcément par arriver à son quai de destination, sans risque de se tromper ;
- par contre, lorsque l'utilisateur part du milieu du couloir (repère 1), la longueur moyenne sur nos expériences a été de 2,75 : les trajets sont loin d'être optimaux, car l'utilisateur peut choisir l'une ou l'autre des deux directions, et donc a une chance sur deux de se tromper.

Figure 8.11 : *Affichage statique d'un train au départ d'un quai donné.*

Version augmentée Dans cette version, les écrans affichent *en plus* les informations concernant les utilisateurs situés à proximité (voir fig. 8.12). Les heures de départ des trains de ces dernières, ainsi que leurs numéros de quais et des flèches indiquant les directions à suivre complètent l'affichage statique de base.

Comme dans l'expérience précédente, les utilisateurs qui partaient des extrémités du couloir (repères 0 et 2) suivaient déjà des trajectoires optimales, nous n'avons pas réitéré ces expériences, et nous sommes concentrés sur les expériences dans lesquels les sujets partent du milieu du couloir (repère 1). Les résultats correspondants sont donnés dans le tableau 8.8.

On note que dans tous les cas, les déplacements sont optimaux. En moyenne, l'utilisation d'un système dynamique a ramené la longueur relative des trajets au départ du repère 1 de 2,75 à 1,00.

8.2.4 Expérimentation à plusieurs utilisateurs

Nous avons également étudié le comportement de cette installation lorsque plusieurs utilisateurs recherchent simultanément leurs quais respectifs.

Sujet	Départ	Dest.	Déplacements	ℓ_u	ℓ_o	Long. rel.
a	2	E	2 → E	1	1	1,0
b	1	E	1 → D, D → C, C → B, B → A, A ↺, A → B, B → C, C → D, D → E	9	2	3,5
c	0	E	0 → A, A → B, B → C, C → D, D → E	5	5	1,0
d	1	E	A → D, D → C, C → B, B → A, A ↺, A → B, B → C, C → D, D → E	9	2	3,5
a	1	B	1 → D, D → E, E ↺, E → D, D → C, C → B	6	3	2,0
b	2	B	2 → E, E → D, D → C, C → B	4	4	1,0
c	1	B	1 → D, D → E, E ↺, E → D, D → C, C → B	6	3	2,0
d	0	B	0 → A, A → B	2	2	1,0
a	0	C	0 → A, A → B, B → C	3	3	1,0
b	0	C	0 → A, A → B, B → C	3	3	1,0
c	2	C	2 → E, E → D, D → C	3	3	1,0
d	2	C	2 → E, E → D, D → C	3	3	1,0

Tableau 8.7 : *Résultats de l'expérience de recherche de direction à l'aide d'écrans statiques. Les lignes horizontales servent à séparer les différentes séries d'expérimentation.*

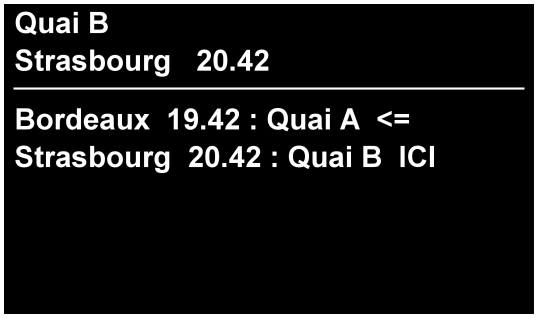


Figure 8.12 : *Affichage d'un train au départ d'un quai donné, complété par des informations dynamiques relatives aux utilisateurs situés à proximité. Cet affichage est à comparer avec celui de la figure 8.11.*

Trois utilisateurs devaient rechercher chacun une direction différente, correspondant à un train au départ sur l'un des cinq quais. Nous avons commencé par réaliser une expérience témoin dans laquelle les écrans étaient statiques, et n'affichaient que la destination du prochain train de leur quai. Les résultats sont donnés dans le tableau 8.9.

Le longueur relative moyenne pour cette expérience est de 2,42, ce qui montre encore une fois que les trajets suivis sont loin d'être optimaux. Ainsi, les utilisateurs b et c sont d'emblée partis dans la mauvaise direction. Nous avons ensuite recommencé l'expérience en mode dynamique. Les résultats sont données dans le tableau 8.10.

Sujet	Départ	Destination	Déplacements	ℓ_u	ℓ_o	Longueur relative
a	1	B	1 → D, D → C, C → B	1	1	1,0
b	1	B	1 → D, D → C, C → B	1	1	1,0
c	1	B	1 → D, D → C, C → B	1	1	1,0
d	1	B	1 → D, D → C, C → B	1	1	1,0
e	1	B	1 → D, D → C, C → B	1	1	1,0
a	1	E	1 → D, D → E	1	1	1,0
b	1	E	1 → D, D → E	1	1	1,0
c	1	E	1 → D, D → E	1	1	1,0
f	1	E	1 → D, D → E	1	1	1,0

Tableau 8.8 : *Résultats de l'expérience de recherche de direction à l'aide d'écrans dynamiques.*

Sujet	Départ	Dest.	Déplacements	ℓ_u	ℓ_o	Long. rel.
a	1	A	1 → D, D → E, E ↻, E → D, D → C, C → B, B → A	7	4	1,75
b	1	B	1 → D, D → C, C → B	3	3	1,00
c	1	E	A → D, D → C, C → B, B → A, A ↻, A → B, B → C, C → D, D → E	9	2	4,50

Tableau 8.9 : *Résultats de l'expérience de recherche de direction par plusieurs utilisateurs à l'aide d'écrans statiques.*

Sujet	Départ	Destination	Déplacements	ℓ_u	ℓ_o	Long. rel.
a	1	A	1 → D, D → C, C → B, B → A	4	4	1,0
b	1	E	1 → D, D → E	2	2	1,0
c	1	B	1 → D, D → C, C → B	3	3	1,0

Tableau 8.10 : *Résultats de l'expérience de recherche de direction par plusieurs utilisateurs à l'aide d'écrans dynamiques.*

Dans ce cas, tous les trajets sont optimaux. Ainsi, même lorsque plusieurs utilisateurs sont présents, l'utilisation de PRIAM pour fournir aux utilisateurs des informations dynamiques et personnalisées permet de faire gagner un temps précieux à ces derniers lorsqu'ils se déplacent.

8.2.5 Notes d'implémentation

Nous avons d'abord testé cette expérience de recherche de direction entièrement en simulation. L'implémentation suivait les mêmes principes que pour l'expérience de recherche d'informations dans une liste. Nous avons utilisé les agents suivants :

- un agent informateur chargé de fournir à chaque utilisateur l'unité sémantique correspondant à sa destination ;
- quelques agents utilisateurs standards de type `UserAgent` : un seul suffit dans le cas où un seul utilisateur parcourt le passage souterrain ; pour l'expérience à trois utilisateurs, nous avons mis en place trois de ces agents ;
- cinq agents présentateurs de type `DisplayScreen`, responsables chacun de l'écran correspondant.

Lors de la réalisation pratique de l'expérience, nous avons utilisé cinq ordinateurs portables pour représenter les cinq écrans, tout en gardant la même architecture. Cependant, afin d'éviter les problèmes liés aux connexions réseau entre les ordinateurs[7], nous avons préféré dupliquer l'agent informateur et les agents utilisateurs *sur chacun des cinq ordinateurs portables*. En résumé, chaque ordinateur contenait donc :

- une copie de l'agent informateur ;
- une copie de tous les agents utilisateurs. Ils utilisaient des `IRLocalizationService` comme expliqué ci-dessus ;
- l'agent présentateur lui correspondant.

En règle générale, une telle duplication des agents pourrait poser problème, car différentes *instances* du même agent pourraient se trouver dans des états différents, ce qui serait incohérent. Cependant, dans le cadre très restreint de notre expérimentation, cela n'a pas perturbé le fonctionnement global du système, car les agents utilisateurs mis en œuvre n'avaient pas d'*états* particuliers.

8.3 Démonstration : aspects multimodaux du système

Le but de cette démonstration est de vérifier le bon fonctionnement de notre système, en ce qui concerne la prise en compte des capacités d'interactions multimodales de chacun des intervenants : utilisateur, dispositif de présentation, et unité sémantique. Nous partons donc d'un environnement dans lequel se trouvent trois utilisateurs, dont deux voyants et un aveugle. Une source d'information diffuse des informations à propos de l'ouverture d'un restaurant. Enfin, deux dispositifs de présentation (un moniteur et un système de synthèse vocale) complètent l'installation (voir fig. 8.13).

Au départ, un utilisateur voyant et l'utilisateur non-voyant passent dans l'espace de rayonnement de la source d'information. Ils reçoivent donc chacun l'unité sémantique relative à l'ouverture du restaurant (voir fig. 8.14).

[7]Bien qu'au début de ce document nous ayons formulé une hypothèse selon laquelle toutes les entités seraient reliées par un réseau sans fil, nous avons préféré ne pas faire dépendre nos expérimentations d'éventuels problèmes liés au réseau.

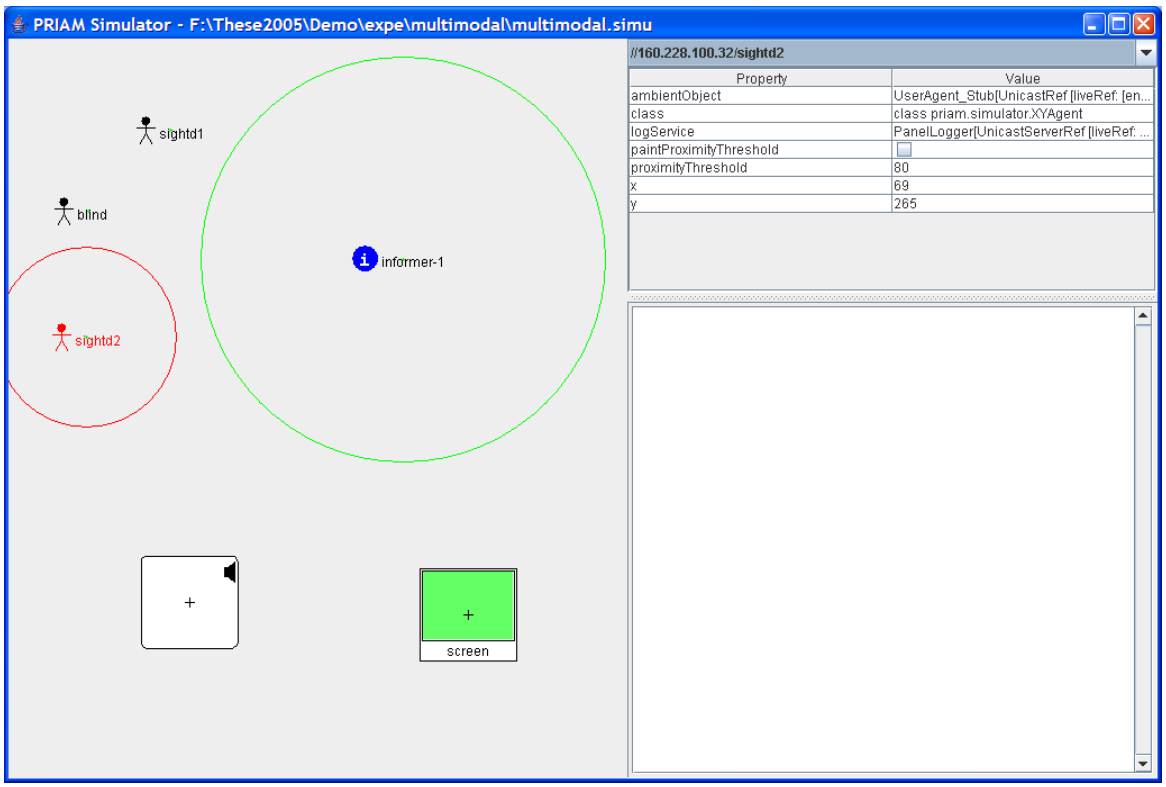


Figure 8.13 : *Vue d'ensemble de la démonstration « multimodale ». Le rectangle blanc où est dessiné un haut-parleur (en bas à gauche) figure le système de synthèse vocale.*

Ensuite, lorsque l'utilisateur voyant s'approche de l'écran, ce dernier lui présente son information (voir fig. 8.15). En effet, les profils multimodaux de l'utilisateur voyant et de l'écran sont compatibles, car la modalité correspondant au texte visuel se trouve dans leur intersection. De plus, cette dernière modalité est supportée par l'unité sémantique, donc la présentation est possible.

Cependant, lorsque l'utilisateur non-voyant s'approche à son tour de l'écran (voir fig. 8.16), ce dernier n'affiche rien du tout, car l'intersection des profils de l'utilisateur et de l'écran est vide. En effet, le premier ne peut percevoir que les modalités auditives et TPK, tandis que le second ne peut effectuer des présentations que selon la modalité visuelle.

En revanche, le dispositif de synthèse vocale est capable de présenter son information aussi bien à un utilisateur voyant (fig. 8.16) qu'à un utilisateur non-voyant (fig. 8.17), car ces derniers peuvent tous deux percevoir les modalités auditives.

Dans cet exemple, nous notons que le fonctionnement multimodal de la plate-forme PRIAM est conforme à ce que nous avons spécifié. En effet, lorsqu'il s'agit d'effectuer la présentation d'une unité sémantique, le système choisit une modalité adaptée à l'utilisateur, en fonction des dispositifs de présentation présents à proximité.

En outre, cet exemple montre que le contenu concret pour une unité sémantique donnée dépend de la modalité selon laquelle elle est présentée. En effet, le contenu généré pour l'unité sémantique relative au restaurant a été clairement différent sur l'écran et sur le système de

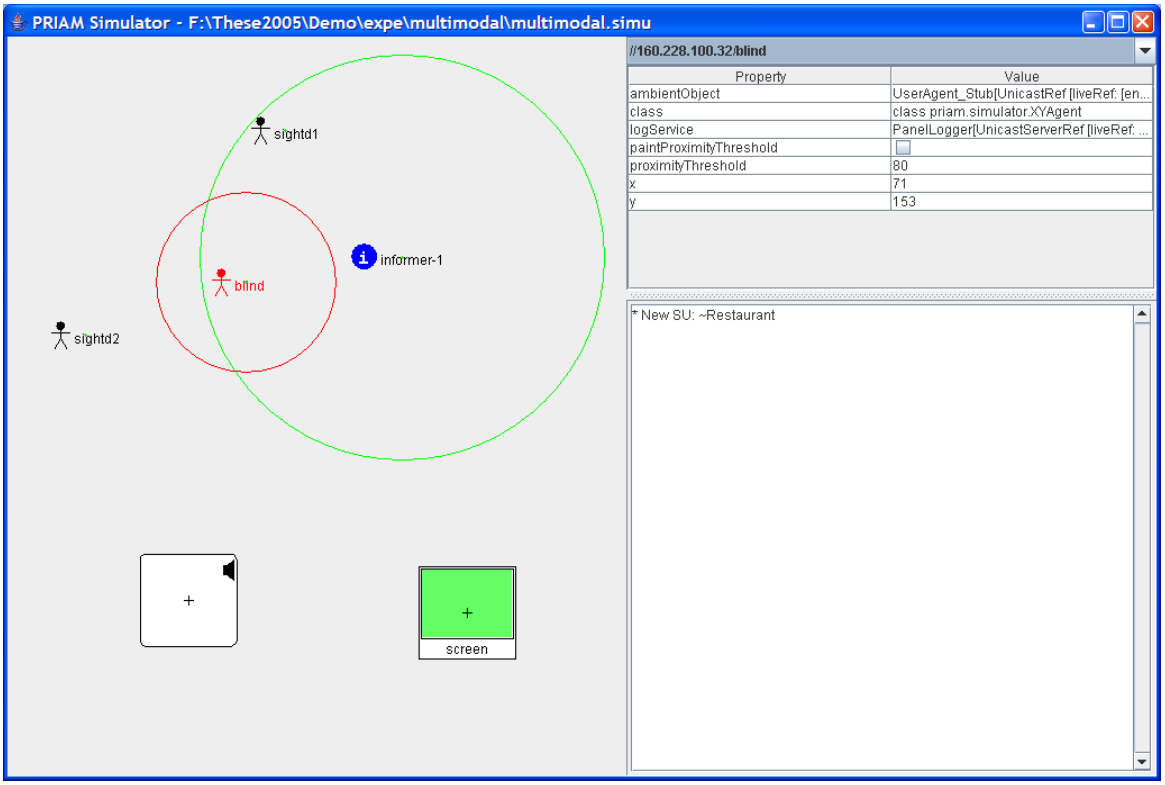


Figure 8.14 : *Les deux utilisateurs reçoivent une unité sémantique.*

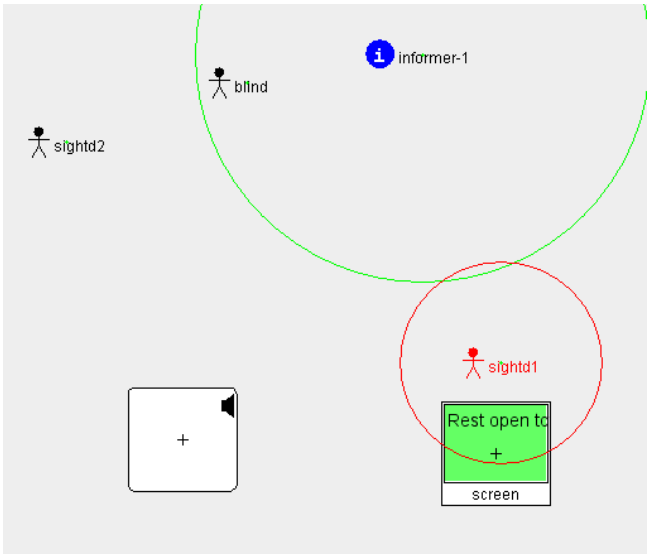


Figure 8.15 : *L'écran présente l'u.s. de l'utilisateur voyant.*

synthèse vocale. Il ne pourrait pas en être autrement, dans la mesure où ce contenu concret est exprimé selon un *format* qui dépend de la modalité choisie, par exemple une chaîne de caractères en UTF-8 pour du texte, ou une image JPEG pour une photographie.

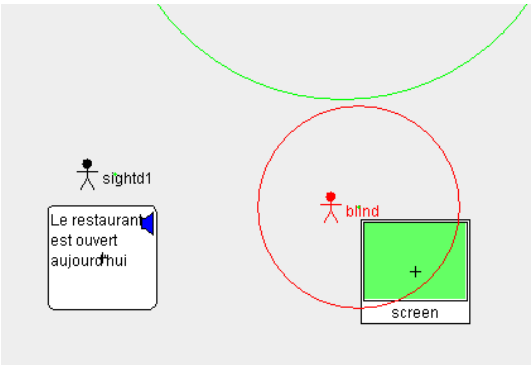


Figure 8.16 : *Présentation et non présentation d'une u.s. en fonction de l'utilisateur.*

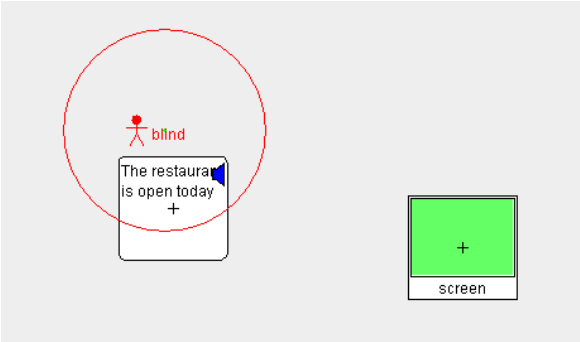


Figure 8.17 : *Présentation auditive d'une u.s. à un utilisateur non-voyant.*

8.4 Démonstration : instanciation

Cette démonstration vise à mettre en évidence les effets du processus d'*instanciation*. Il s'agit d'une variation sur la démonstration précédente. Les entités considérées sont donc les mêmes ; nous nous intéressons en particulier à l'un des deux utilisateurs voyants et à l'écran d'affichage.

Cet utilisateur voyant s'approche de l'écran d'affichage, ce qui engendre une présentation d'une unité sémantique obtenue auparavant. Cependant, en fonction de la distance à laquelle l'utilisateur se trouve de l'écran, les attributs de l'instanciation sont différents. Ainsi, le texte s'affiche en plus gros caractères lorsque l'utilisateur est plus éloigné de l'écran (voir fig. 8.18).

Nous voyons ainsi que le processus d'instanciation consiste à fixer des valeurs pour les attributs d'une modalité, ce qui se répercute directement sur la forme prise par la présentation effectuée sur le dispositif choisi.

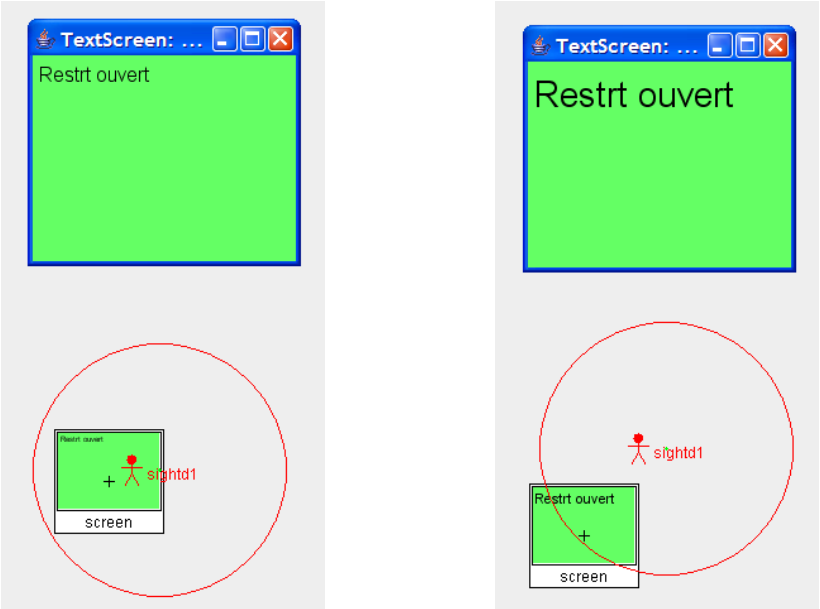


Figure 8.18 : *Lorsque la distance augmente, le processus d'instanciation choisit des valeurs différentes pour les attributs : la taille du texte augmente.*

8.5 Conclusion

Les évaluations que nous avons présentées dans ce chapitre nous montrent que les *concepts* fondamentaux de notre travail permettent de mettre en place des systèmes de présentation d'information qui améliorent le confort des utilisateurs, et donnent à ces derniers les informations dont ils ont besoin de façon plus immédiate.

Cependant, il sera par la suite nécessaire de réaliser un tel système d'information destiné à fonctionner dans un *environnement réel* (par exemple, une gare, un aéroport ou un établissement d'enseignement), et non plus seulement en laboratoire. Cette nouvelle expérimentation se distinguera de la démarche suivie jusqu'ici par deux caractéristiques principales :

- les interfaces concrètes (contenu des écrans, informations diffusées par les haut-parleurs) devront être définies avec un soin particulier, car c'est à travers elles que s'opérera toute la perception que les utilisateurs auront du système. Ainsi, il sera nécessaire de définir précisément la façon dont les textes seront affichés, les transitions lors de l'ajout de nouvelles unités sémantiques, les voix utilisées pour les messages vocaux, etc. ;
- les utilisateurs ne seront plus des volontaires recrutés parmi nos collègues de laboratoires d'informatique spécifiquement pour participer à une expérimentation, mais de vrais usagers, éventuellement pressés, stressés et préoccupés, qui auront un réel besoin des informations distillées par le système.

Chapitre 9

Conclusion et perspectives

Dans cette thèse, nous avons présenté notre contribution aux recherches sur l'informatique contextuelle, et en particulier, à l'utilisation de la multimodalité en sortie dans des environnements d'intelligence ambiante. Dans un premier temps, nous allons résumer les principaux points traités dans cette thèse, avant d'introduire des perspectives pour des travaux futurs.

9.1 Contributions

Le modèle KUP introduit un nouveau paradigme de conception des interfaces homme-machine pour les situations de mobilité. *L'utilisateur est explicitement représenté* dans le modèle, et une entité correspondante existe dans l'implémentation. Cette entité joue un rôle central dans les interactions, car elle élimine les couplages qui existent habituellement entre deux opérations dans les IHM conventionnelles :

- d'une part la *fourniture* d'une information à l'utilisateur par une source d'informations ;
- et d'autre part la *présentation* de cette information sur un dispositif adapté.

Cette distinction s'est avérée nécessaire du fait de la *mobilité* des utilisateurs. En revanche, elle n'est pas nécessaire dans les IHM conventionnelles car dans ces dernières, l'utilisateur est fixe (temporellement et géographiquement) devant l'interface.

Grâce à cette architecture, les utilisateurs de lieux publics peuvent tirer parti de façon *opportuniste* des sources d'informations et des dispositifs de présentation, récoltant des informations au gré de leurs déplacements, pour se les faire présenter ensuite par des dispositifs *adaptés* et situés à proximité.

Notons bien l'emphase mise sur le mot *adapté* : en effet, il ne sert à rien de présenter une information selon une modalité que son destinataire n'est pas à même de percevoir. C'est pourquoi la *multimodalité* est au cœur de notre système, de façon à tenir compte à la fois des capacités des utilisateurs et des dispositifs de présentation. Ceci nous permet de

choisir lors de la présentation de chaque information le *meilleur compromis* entre les capacités et préférences de l'utilisateur, les dispositifs situés à proximité, et les desiderata des *autres* utilisateurs. De cette façon, nous prenons en compte toutes les *différences* qui peuvent exister entre utilisateurs, et en particulier les handicaps sensoriels.

Ce mode de fonctionnement nous permet également de limiter la surcharge des dispositifs de présentation en n'y présentant que les informations utiles à des personnes réellement situées à proximité. Par exemple, les panneaux d'affichage des gares peuvent ainsi devenir *dynamiques*, ce qui améliore le temps de recherche des informations. De plus, notre modèle induit naturellement des comportements *collaboratifs* entre écrans, ce qui permet une judicieuse *répartition de contenu* entre écrans voisins. Notons que ces comportements collaboratifs sont *indirects* : les écrans ne collaborent pas directement entre eux, mais tout passe par l'entité utilisateur, qui joue vraiment un rôle central dans tous les aspects du fonctionnement du système.

Nous avons également introduit une notion de proximité sensorielle entre les différentes entités du modèle KUP (sources d'informations, utilisateurs et dispositifs de présentation). Cette proximité nous a permis de concevoir une architecture complètement décentralisée, à base d'agents qui n'ont qu'une vision *locale* (et donc partielle) du système. Ainsi, notre architecture ne nécessite pas de configuration centrale. Il suffit en effet de démarrer l'exécution de chacun des agents locaux, et ceux-ci sont capables de s'organiser afin de mettre en œuvre les comportements prévus dans le modèle. De même, il est possible à tout instant de modifier l'organisation d'une application donnée (en déplaçant des éléments) sans qu'il soit nécessaire de reconfigurer quoi que ce soit.

Cette architecture a été implémentée en pratique, sous forme de la plate-forme PRIAM. Cette plate-forme est dotée d'un simulateur qui permet de vérifier le bon fonctionnement de toutes les briques logicielles d'une application avant de la déployer grandeur nature. Elle nous a permis de réaliser quelques expérimentations en conditions quasi réelles. Grâce à ces évaluations, nous avons pu d'une part vérifier le bon fonctionnement de notre plate-forme, et d'autre part démontrer qu'une présentation dynamique d'informations personnalisées en environnement mobile permet aux utilisateurs d'obtenir les informations dont ils ont besoin bien plus facilement qu'avec les classiques dispositifs de présentation statiques.

9.2 Perspectives

Ce travail soulève un certain nombre de perspectives, à la fois sous forme d'extensions à ce qui a été modélisé et réalisé, et sous forme de thèmes de recherche à plus grande échéance.

IHM des écrans dynamiques Les personnes qui ont participé à nos expériences d'écrans dynamiques nous ont souvent rapporté qu'elles avaient été perturbées par des modifications trop fréquentes du contenu des écrans (voir section 8.1.4). Deux options sont envisageables pour résoudre ce problème :

1. lorsqu'il est nécessaire de réorganiser l'affichage d'un écran, il faudra prévoir des *transitions* entre l'ancienne et la nouvelle organisation. Par exemple, s'il faut ajouter une nouvelle ligne entre deux lignes précédemment affichées, il est possible de faire défiler très lentement la partie basse de l'écran de façon à ménager l'espace nécessaire, plutôt que de passer brutalement d'un affichage à l'autre. Ou encore, s'il faut supprimer une ligne, il est possible de réduire progressivement sa taille. Nous pourrons réutiliser ici des techniques d'animation validées pour les IHM ;

2. quand il faut afficher un nouvel item d'information, il serait certainement moins perturbant de le rajouter en fin de liste, quitte à ne pas respecter le classement habituel des items. L'attention du nouvel utilisateur pourrait dans ce cas être attirée sur ce nouvel item par un clignotement ou une coloration transitoire. En fait, il s'agit là d'une extension de la *contrainte de stabilité* (voir page 141), non plus seulement au niveau de l'affectation d'une unité sémantique à tel ou tel écran, mais également au niveau de l'organisation d'un écran donné.

Une bonne solution devra probablement mettre en œuvre conjointement ces deux possibilités.

Décomposition du contenu concret Dans notre travail, chaque unité sémantique donne lieu à la génération d'un contenu concret *atomique*, exprimé selon une seule instance de modalité. Cependant, il serait intéressant de proposer une *décomposition* du contenu concret en plus petites parties, chacune pouvant correspondre à une instanciation de modalité différente.

Cela permettrait par exemple la définition d'une sous-modalité « texte en colonnes » de la modalité « texte visuel ». De la sorte, lors de l'instanciation de cette modalité pour la présentation d'une unité sémantique scindée en plusieurs parties, chaque partie se verrait attribuer un numéro de colonne, ainsi que des attributs (couleur, style, fonte) propres.

On pourrait ainsi parler d'instanciation *homogène* (lorsque l'instanciation de la modalité est la même pour toutes les parties du contenu concret) ou *hétérogène* (dans le cas contraire).

Historique des unités sémantiques Dans notre modèle, les agents utilisateurs ne se préoccupent pas de l'*historique* des unités sémantiques reçues. Cependant, ce pourrait être une source d'informations intéressante — et potentiellement exploitable. Ainsi, en effectuant de la fouille de données sur ces unités sémantiques, même périmées, les agents utilisateurs pourraient émettre des suppositions sur l'état actuel du monde physique. Ceci pourrait par exemple se révéler utile en cas de défaillance temporaire des agents présentateurs.

Par exemple, lors de chaque visite dans une entreprise donnée, l'agent d'un utilisateur mémorise les heures d'ouverture de l'accueil pour le jour courant. Si lors d'une visite ultérieure la source d'information est en panne, l'agent utilisateur pourra deviner quelles sont les heures d'ouverture probables de l'accueil, en fonction de ce qu'il aura mémorisé.

Accrochage entre unités sémantiques Il s'agirait d'effectuer des *raisonnements* sur les diverses unités sémantiques reçues par les utilisateurs. Ce point a déjà été abordé dans la section 4.2.1.3, mais n'a pas été pris en compte pour le moment dans notre modèle. C'est un problème à part entière, qui rejoint les thèmes étudiés dans le domaine de la représentation des connaissances et de l'intelligence artificielle. En effet, dans ce que nous avons présenté, les unités sémantiques sont uniquement des *boîtes noires* capables de générer un contenu concret selon diverses modalités.

Par contre, si nous voulons être capables de raisonner sur les unités sémantiques, il est nécessaire d'en donner une description qui soit manipulable par la machine. Pour ce faire, une bonne piste serait certainement d'utiliser les technologies déployées dans le cadre du Web sémantique (RDF [Miller 1998], OWL et langages de requêtes associés), et bien entendu les théories sous-jacentes (logiques du premier et du second ordre, logiques modales, logiques de description [Borgida 1995], etc.).

Priorités entre informations et utilisateurs La question des priorités a déjà été abordée (voir sections 4.2.2.3 et 4.2.1.4). Ce problème n'a pas été soulevé dans le cadre de nos expérimentations, car le nombre d'informations à afficher, ainsi que le nombre d'utilisateurs présents, étaient très faibles. Cependant, dans des environnements réels (par exemple, dans un aéroport), le nombre d'informations et d'utilisateurs sera bien plus grand, et il sera donc nécessaire de *hiérarchiser* la présentation des informations.

De plus, dans ce type d'environnements, un utilisateur disposera potentiellement de plusieurs unités sémantiques différentes à présenter : il s'agira alors de choisir une stratégie de présentation qui soit ergonomique et cohérente. En effet, il est peu probable qu'une présentation simultanée et aléatoire de ces différentes informations sur des dispositifs divers soit une solution viable.

Au contraire, il sera probablement nécessaire d'effectuer des regroupements : faut-il effectuer la présentation de toutes les unités sémantiques du même utilisateur sur le même dispositif ? Faut-il mettre en œuvre une *chorégraphie* (séquencement temporel) pour leur présentation ? Peut-on utiliser plusieurs modalités de façon *complémentaire*[1] ? Pour répondre à ces questions, il sera nécessaire d'effectuer des évaluations précises, et en situation réelle, avec des utilisateurs occupés à une *tâche réelle*.

Évaluations Dans tous les cas, il sera nécessaire de mener des évaluations grandeur nature, par exemple dans une gare, un aéroport ou un centre commercial. Nous n'avons pas pu pour le moment réaliser d'expérimentations d'une telle envergure, car elles nécessitent la mise en place de partenariats avec les opérateurs de ce type de lieux publics, par exemple la SNCF ou Aéroports de Paris, ce qui est relativement long à mettre en œuvre.

[1] cf. propriétés CARE, voir p. 16.

Prise en compte des entrées Le modèle KUP tel que nous l'avons présenté dans ce document a été conçu pour modéliser les interactions multimodales *en sortie*. Ceci se justifie par rapport à notre cahier des charges, selon lequel nous voulions *présenter* des informations pertinentes à des utilisateurs. Cependant, il serait intéressant à plus long terme de *symétriser* le modèle, de façon à ce qu'il puisse tenir compte des *entrées* aussi bien que des sorties et d'étudier l'influence que peuvent avoir les entrées sur les sorties.

Dans ce cas, il sera nécessaire de spécifier les traitements à appliquer aux entrées, entre l'acquisition des données brutes (par exemple, voix numérisée), et leur utilisation par le noyau fonctionnel.

9.3 Conclusion

Dans les années 1980, les ordinateurs ont essaimé à partir des centres de calcul pour s'installer sur tous les bureaux. Nous semblons être sur le seuil d'une révolution similaire, où les ordinateurs essaimeront à partir des bureaux pour s'installer de façon enfouie dans tous nos environnements, et nous aider de façon discrète, opportune et pertinente.

Notre travail s'est inscrit dans cette vision de l'intelligence ambiante, et a visé à démontrer comment cette nouvelle déclinaison de l'informatique pouvait permettre des interactions avec les utilisateurs mobiles. Nous avons également identifié des pistes de recherche qui nous paraissent importantes pour la suite. Les études entreprises dans le cadre de cette thèse nous ont permis d'acquérir les connaissances nécessaires et une vision d'ensemble du domaine pour mener à bien la poursuite de ces travaux de recherche.

Annexes

Annexe A

Temps de recherche d'une information

A.1 Introduction

Nous avons mené une petite expérimentation visant à mesurer le temps de recherche d'une information dans une liste, en fonction du nombre d'informations présentes dans cette liste. Notons que par son protocole, cette expérimentation est quelque peu liée à l'évaluation de la plate-forme PRIAM présentée dans la section 8.1. Cependant, l'objet d'étude n'est pas le même :

1. dans l'expérience de la section 8.1, nous cherchons avant tout à évaluer les gains potentiels de la plate-forme PRIAM pour les utilisateurs, ainsi qu'à vérifier grandeur nature le bon fonctionnement de celle-ci. C'est pourquoi dans cette expérience, nous avons cherché à reproduire une situation quasi réelle dans laquelle plusieurs utilisateurs mobiles doivent rechercher une information ;

2. dans l'expérience présentée ici au contraire, nous cherchons seulement à mesurer le temps mis par un utilisateur pour retrouver une information dans une liste, de façon indépendante de tout contexte (environnement physique, système informatique, conditions dans lesquelles se trouve l'utilisateur, présence d'autres personnes à proximité, etc.).

Cette expérimentation a été réalisée par le Web, à l'aide d'une applet Java, ce qui nous a permis de faire participer un nombre relativement important de personnes. Au total, 42 volontaires y ont pris part. Dans cette expérience, nous avons proposé des exercices très simples aux participants. Il s'agissait de repérer dans une liste un nom ou bien un numéro de vol mémorisé à l'avance, sachant que la liste comportait un nombre variable d'items. Pour signifier qu'ils avaient retrouvé leur information, les volontaires devaient cliquer sur l'item correspondant. Cet exercice était répété plusieurs fois.

Nous avons chronométré le temps mis par les personnes pour retrouver leur information de la façon suivante : le chronomètre était déclenché lors de l'affichage de la liste, et arrêté lors d'un clic sur un item de la liste. Notons que cette méthode de mesure introduit forcément un biais, car elle ne mesure pas réellement le temps t_r mis pour retrouver une information d'intérêt, mais la somme de t_r avec le temps t_c mis par l'utilisateur pour cliquer sur l'item voulu une fois celui-ci repéré.

Cependant, nous estimons :

1. que les fluctuations de t_c doivent être relativement limitées ;
2. que *statistiquement*, ces petites fluctuations de t_c doivent se compenser sur le nombre total d'exercices proposés.

En conséquence, nous considérons en première approximation que t_c est une constante, et donc que le temps mesuré est systématiquement augmenté d'une quantité fixe et inconnue. Comme dans la suite nous nous intéresserons principalement à la *pente* des variations de t_r, nous pouvons considérer que celle-ci est peu différente de la pente des variations du temps mesuré $t_r + t_c$.

Pour pouvoir comparer les mesures effectuées sur les différents utilisateurs entre elles, nous avons *normalisé* toutes les mesures en divisant tous les temps mesurés par le temps moyen pour l'utilisateur en question. Comme la distribution des longueurs de liste est identique pour tous les utilisateurs, nous n'introduisons pas ici de biais supplémentaire. Nous appelons *temps relatifs* les mesures ainsi obtenues.

Nous n'avons pas comptabilisé dans nos résultats les temps correspondant aux réponses fausses. Cependant, les erreurs ont été extrêmement rares, donc nous n'avons pas pu faire de statistiques sur les conditions dans lesquelles elles sont survenues.

A.2 Recherche d'un numéro de vol

Dans un premier type d'exercices, les utilisateurs devaient rechercher un numéro de vol dans une liste. L'interface de l'applet est représentée sur la figure A.1. Nous n'avons pas pris en compte les exercices dans lesquels le numéro à rechercher était tout au début de la liste (précisément, dans le premier quart), car il semble que ces cas soient particulièrement faciles à « résoudre », ce qui aurait pu fausser les résultats.

Les résultats correspondants sont donnés sur le graphe de la figure A.2. Les points sont accompagnés d'un segment vertical qui figure l'écart type. On remarque que la progression est régulière jusqu'à une trentaine de numéros de vol affichés. La progression est beaucoup moins nette ensuite, et surtout les résultats sont beaucoup plus dispersés.

Nous avons effectué une régression linéaire par la méthode des moindres carrés sur les résultats jusqu'à 30 vols. La droite correspondante est représentée sur la figure A.2. Sur cette zone,

Cliquez sur l'item qui vous correspond [YT3595 (16.00)]

KE3797....O....15.33	KE905.....C....16.32
GN1311....L....15.42	MK7574....O....16.41
YT4401....J....15.44	CA7700....B....16.47
EI5170....O....15.47	IB1807....C....16.53
IB5550....I....15.57	BA6489....E....17.09
YT3595....E....16.00	EI2963....D....17.45
IR2178....D....16.15	AF8194....I....17.54
MS2916....N....16.18	LH8771....K....18.03
BA8480....C....16.26	MK5974....N....18.05
IR7234....I....16.27	YT6517....C....18.06
MK2311....C....16.32	MS7593....A....18.25

Figure A.1 : *Applet de recherche d'un vol dans une liste.*

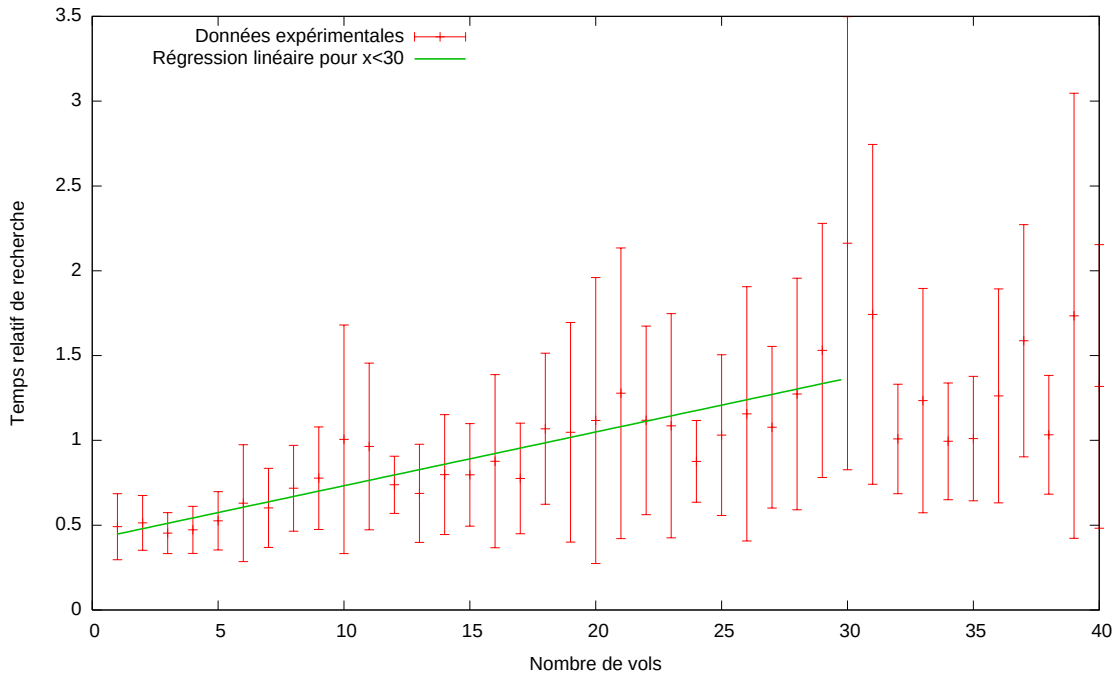


Figure A.2 : *Résultats de l'exercice de recherche d'un numéro de vol.*

cette droite semble correspondre assez bien aux données mesurées ; de plus, l'écart type reste modéré.

Sur la portion ultérieure du graphe (donc pour un nombre de vols supérieur à 30), il n'y a plus de tendance réellement mesurable dans les variations du temps de recherche. Par contre, la dispersion est très grande, et l'écart type a tendance à augmenter considérablement.

Nous estimons donc que pour un nombre assez limité d'informations (jusqu'à 30 vols), l'augmentation du nombre d'items provoque une augmentation linéaire du temps de recherche. Par contre, au-delà d'un seuil qui semble se dessiner aux alentours de 30 vols, le temps de recherche n'augmente pas significativement en moyenne. Par contre, il fluctue énormément : selon la « chance » des utilisateurs, et selon les exercices, ceux-ci peuvent parfois très vite trouver leur information, et d'autres fois mettre très longtemps à trouver leur item d'intérêt.

A.3 Recherche d'un nom

Dans cette deuxième expérience, les volontaires devaient rechercher un nom dans une liste, du type des listes de résultats à des examens. Une copie d'écran de l'applet correspondante est donnée sur la figure A.3. Comme dans l'exercice précédent, nous avons éliminé les cas où le nom à rechercher se trouvait en début ou en fin de liste. Au final, nous n'avons comptabilisé que les résultats où le nom se trouvait entre le premier et le dernier quart (soit les 50 % des noms situés « dans le milieu »).

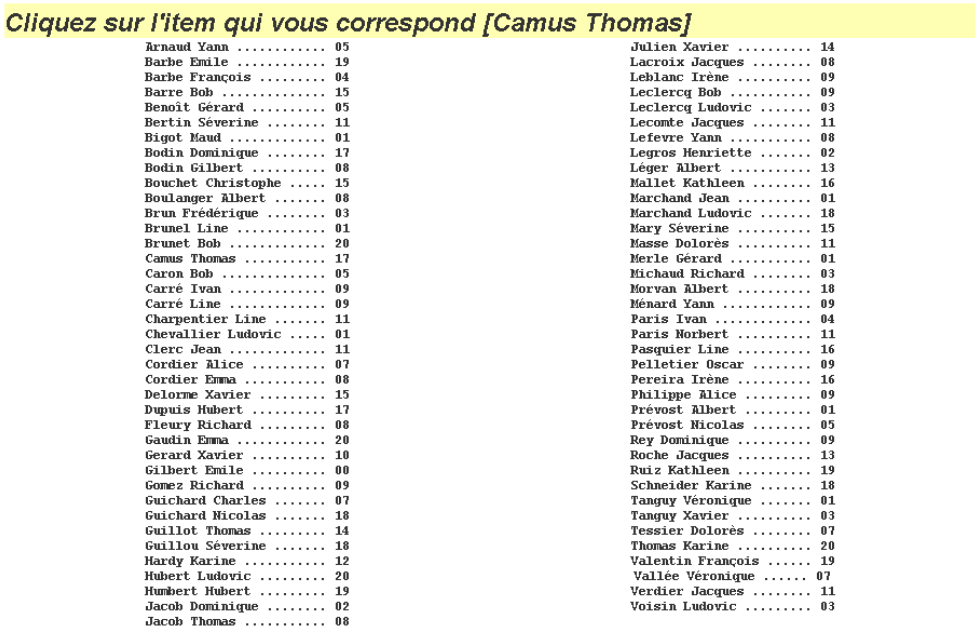


Figure A.3 : *Applet de recherche d'un nom dans une liste.*

La figure A.4 montre les résultats correspondants. Ces résultats vont dans le même sens que ceux de l'expérience précédente : jusqu'à un certain point, le temps de recherche augmente de façon soutenue avec le nombre d'items dans la liste. Puis, la dispersion est assez grande, et les variations du temps de recherche moins nettes.

Le segment situé le plus à gauche est le résultat d'une régression linéaire pour les données allant jusqu'à 60 items présentés. Comme les données situées au-delà de ce seuil nous paraissaient plus régulières que dans l'expérience précédente, nous avons réalisé une deuxième régression linéaire pour celles-ci : on constate que la pente du deuxième segment est bien

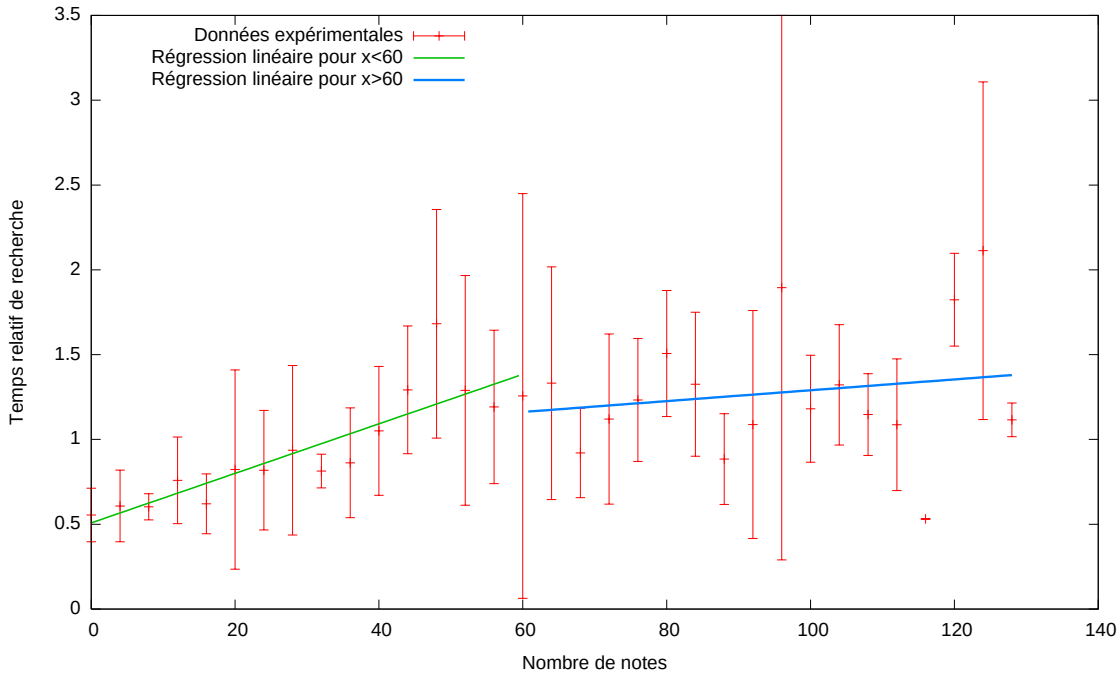


Figure A.4 : *Résultats de l'exercice de recherche d'un nom.*

plus faible que celle du premier. De plus, dans cette zone, l'écart type est assez fluctuant, et peut prendre des valeurs assez fortes.

Cette deuxième expérience confirme donc les résultats de la première, à savoir une franche croissance dans une première zone, et des résultats beaucoup plus variables sur une deuxième.

A.4 Conclusion

D'après ces deux expériences, nous pouvons tirer les conclusions suivantes quant au temps de recherche d'un item dans une liste, en fonction du nombre total d'items :

- *jusqu'à un certain point*, le temps de recherche augmente assez fortement, et régulièrement, avec le nombre d'items ;
- *au-delà de ce seuil*, les temps sont beaucoup plus fluctuants (forte variabilité individuelle), mais en moyenne, les variations du temps avec le nombre d'items sont assez faibles, surtout par comparaison avec les résultats de la première zone. On dénote cependant une tendance à la croissance du temps (ce qui, somme toute, est relativement intuitif).

Annexe B

Extraits de code et détails d'implémentation

Cette annexe fournit des détails supplémentaires, ainsi que des extraits de codes, relatifs à l'implémentation introduite au chapitre 7.

B.1 DTD des taxonomies

```
<!ELEMENT taxonomy ((modality, finiteSet*) | (finiteSet*,modality))>
<!ATTLIST taxonomy name CDATA #REQUIRED>

<!ELEMENT modality (attribute*, modality*)>
<!ATTLIST modality name CDATA #REQUIRED
                   id   ID    #REQUIRED>

<!ELEMENT attribute EMPTY>
<!ATTLIST attribute name    CDATA #REQUIRED
                    type    CDATA #REQUIRED
                    min     CDATA #IMPLIED
                    max     CDATA #IMPLIED
                    step CDATA #IMPLIED>

<!ELEMENT finiteSet (element+)>
<!ATTLIST finiteSet name CDATA #REQUIRED>

<!ELEMENT element EMPTY>
<!ATTLIST element value CDATA #REQUIRED>
```

B.2 Exemple de taxonomie

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE taxonomy SYSTEM "taxonomy.dtd">

<taxonomy name="urn:taxo1">
  <modality name="Modality" id="n0">
    <modality name="Visual" id="n1">
      <modality name="Text" id="n2">
        <attribute name="lang" type="Lang"/>
        <attribute name="size" type="real" min="1" max="10.0" step="2"/>
        <attribute name="length" type="real" min="1.0" max="35.0" step="4"/>
      </modality>
    </modality>
    <modality name="Auditive" id="n3">
      <attribute name="volume" type="real" min="0.0" max="10.0" step="2"/>
      <modality name="SpeechSynthesis" id="n4">
        <attribute name="lang" type="Lang"/>
        <attribute name="length" type="real" min="1.0" max="30.0" step="5"/>
      </modality>
    </modality>
  </modality>
  <finiteSet name="Lang">
    <element value="en"/>
    <element value="fr"/>
  </finiteSet>
</taxonomy>
```

Cette taxonomie d'exemple est représentée sous forme d'un diagramme UML sur la figure B.1.

B.3 DTD des profils

```
<!ELEMENT profile (node*)>
<!ATTLIST profile taxonomy CDATA #REQUIRED>

<!ELEMENT node (attribute*)>
<!ATTLIST node id     ID    #REQUIRED
               weight CDATA #REQUIRED
               weightFunction CDATA #REQUIRED>
```

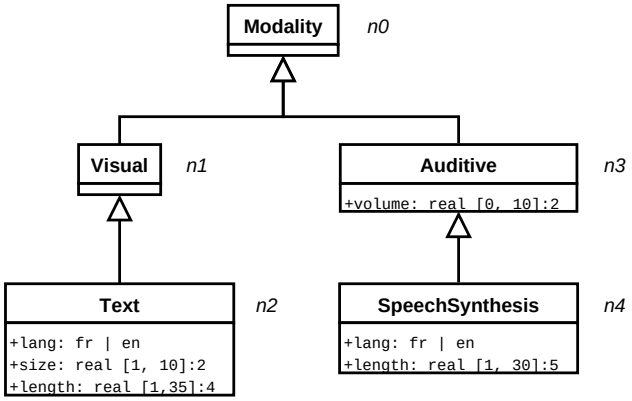


Figure B.1 : *Représentation sous forme d'un diagramme UML de la taxonomie décrite en XML dans la section B.2. Les identifiants des nœuds de l'arbre XML sont reportés sur ce diagramme (n0 à n4).*

B.4 Exemple de profil

Nous donnons ici en exemple profil correspond à un utilisateur voyant. Il se réfère à la taxonomie de l'annexe B.2. Le fichier XML est le suivant :

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE profile SYSTEM "profile.dtd">

<profile taxonomy="urn:taxo1">
  <node id="n3" weight="0.688"
        weightFunction="priam.apps.multimodal_screen.functions.userSighted_n3"/>
  <node id="n1" weight="1.0"
        weightFunction="priam.multimodal.OneFunction"/>
  <node id="n4" weight="1.0"
        weightFunction="priam.apps.multimodal_screen.functions.userSighted_n4"/>
  <node id="n2" weight="1.0"
        weightFunction="priam.apps.multimodal_screen.functions.userSighted_n2"/>
  <node id="n0" weight="1.0"
        weightFunction="priam.multimodal.OneFunction"/>
</profile>
```

Chacun des noms qualifiés de classes indiqués dans les attributs `weightFunction` se réfèrent à une classe Java. Cette classe doit être une sous-classe de `WeightFunction`, et implémenter une fonction de pondération. À titre d'exemple, la classe `userSighted_n4` est donnée ci-dessous :

```
1 package priam.apps.multimodal_screen.functions;
```

```
3  import java.util.Map;

5  import priam.multimodal.WeightFunc;
6  import priam.multimodal.WeightFunction;


9  @WeightFunc({"lang", "length"})
10 public class userSighted_n4 extends WeightFunction {
11    @Override
12    public double apply(Map<String, String> attributeValues, double distance) {
13       double length = Double.parseDouble(attributeValues.get("length"));
14       String lang = attributeValues.get("lang");

16       if("fr".equals(lang)) {
17          if(length<5) return .2;
18          if(length>20) return .6;
19          if(length>=5 && length<=15)
20             return linearFunc(5, .2, 15, 1, length);
21          return linearFunc(15, 1, 20, .6, length);
22       } else {
23          if(length<7) return .1;
24          if(length>25) return 1;
25          return linearFunc(7, .1, 25, 1, length);
26       }
27    }

29    public String toString() {
30       return "userSighted_n4";
31    }
32 }
```

Cette classe se contente de redéfinir la méthode `apply()`, qui calcule la valeur de la fonction en un point donné. Il s'agit d'une fonction de deux variables (i.e. des deux attributs du nœud `n4`, voir figure B.1), qui sont `length` et `lang`. Cette fonction est représentée sur la figure B.2.

B.5 Transmission des fonction de pondération par le réseau

Dans la plate-forme PRIAM, les profils doivent pouvoir être échangés entre agents, donc ils doivent pouvoir transiter par le réseau. Java prévoit un mécanisme pour la transmission des données d'instance par le réseau : la *sérialisation* des objets, d'ailleurs réalisée automatiquement dans les cas simples. Cependant, dans le cas qui nous intéresse, cela ne fonctionnerait pas si l'on n'y prenait pas garde.

En effet, un profil (`WeightedTree`) est composé de `WeightedNode`, qui contiennent chacun une référence à une fonction de pondération. Or, comme nous l'avons vu, une fonction de pondération est représentée par une sous-classe d'une classe abstraite `WeightFunction`. Certes, en

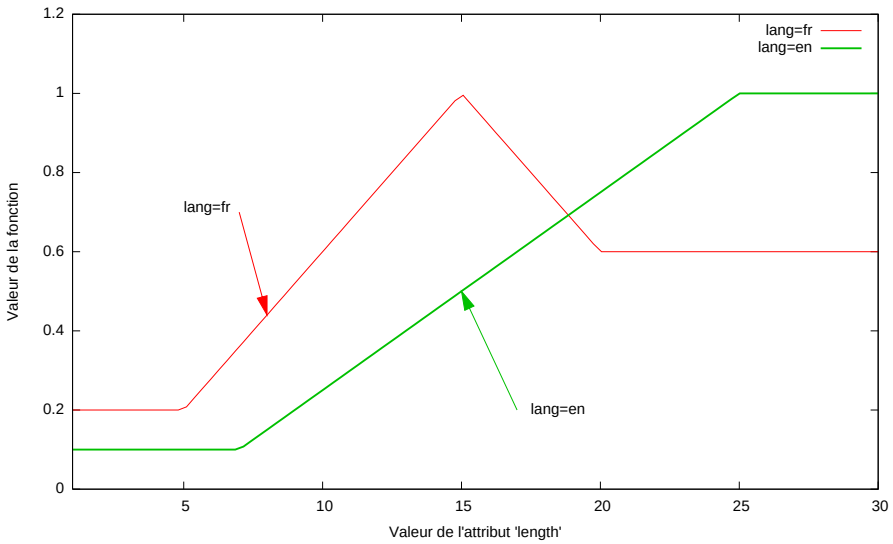


Figure B.2 : *Fonction de pondération du nœud* ***n4****. On a différencié deux cas, selon que la variable* ***lang*** *vaut* ***fr*** *ou* ***en****. Dans chacun de ces deux cas, la fonction se résume alors à une fonction d'une seule variable,* ***length****, qui est tracée ici.*

pratique un `WeightedNode` fera référence à une *instance* de la classe correspondant à la fonction de pondération en question, mais ce qui importe, ce ne sont pas les *données d'instance* (d'ailleurs inexistantes), mais le *code de la classe*, et en particulier, son implémentation de la méthode `apply()`.

En effet, chaque agent pourra posséder ses propres sous-classes de `WeightFunction`, correspondant aux fonctions de pondération de son profil, et que ne posséderont pas les autres agents auxquels il voudra transmettre ce profil. Au final, il est donc nécessaire de *transmettre le code de la classe* correspondant à la fonction de pondération.

Nous avons résolu ce problème en implémentant une sérialisation et une désérialisation personnalisées pour la classe `WeightedNode` : en plus des données de base de l'instance, est sérialisé le *bytecode* de la classe qui implémente la fonction de pondération associée. Lors de la désérialisation, une fois le *bytecode* correspondant reçu, il est utilisé pour recréer une classe, qui est immédiatement instanciée.

L'extrait extrait de code suivant indique la façon dont la classe `WeightedNode` peut sérialiser sa fonction de pondération.

```
1 public class WeightedNode implements Serializable {
2    private transient WeightFunction weightFunction;

4    ...

6    private void writeObject(ObjectOutputStream out) throws IOException {
7       out.defaultWriteObject();
8       out.writeObject(weightFunction.getClass());
9    }
```

```
11     private void readObject(ObjectInputStream in)
12                          throws IOException, ClassNotFoundException {
13        in.defaultReadObject();
14        Class wfClass = (Class)in.readObject();

16        try {
17           Constructor wfConstr = wfClass.getConstructor();
18           weightFunction = (WeightFunction)wfConstr.newInstance();
19        } catch(...) {
20           ...
21        }
22     }
23  }
```

B.6 Format XML de description d'expérimentation

Dans la section 7.4.3, nous avons introduit un format de fichier XML de description d'expérimentation. Cette annexe donne quelques précisions sur ce format de fichier. Un exemple de fichier, ainsi que la DTD correspondante, sont donnés plus loin.

Dans les fichiers de description d'expérimentation, on déclare une série de services de localisation[1], ainsi qu'une série d'agents. À chaque fois, pour indiquer le type précis de service ou d'agent souhaité, on donne le nom qualifié de la classe correspondante.

Pour chaque agent, on indique son service de localisation[2], sa position à l'écran (donc en termes de pixels) lorsqu'il fonctionne sur le simulateur, et surtout un ensemble de couples ⟨ attribut, valeur ⟩. Ces couples permettent de définir de façon précise comment l'agent doit être instancié. Pour cette raison, lorsqu'on implémente un nouvel agent pour la plate-forme PRIAM, on doit *obligatoirement* le munir d'un constructeur qui prend en paramètre un tel ensemble de couples (en pratique, un objet `Map<String, String>`). Ce constructeur est appelé lors de l'instanciation des agents à partir d'un fichier XML de description.

Il peut arriver que l'on veuille spécifier des valeurs différentes de certains de ces paramètres selon que l'expérience décrite fonctionnera en mode simulé ou bien grandeur nature. Par exemple, en simulation nous souhaitons que la fenêtre d'affichage des écrans reste petite ; en conditions réelles par contre, nous voulons qu'elle s'affiche en plein écran. Nous avons donc introduit la notion de *paramètres conditionnels*, qui ne seront pris en compte que dans l'un des deux modes. Ces paramètres sont caractérisés par la présence d'un attribut `only`. Si `only` vaut `simu`, alors il est pris uniquement en compte lors des simulations. Si par contre

[1] Dans l'exemple, nous avons défini un service de localisation de type `AnywhereLoc`. Ce service indique en permanence aux agents dont il est responsable qu'ils sont proches les uns des autres. Ceci nous a par exemple été utile pour faire en sorte que, dans nos expérimentations simples, l'agent informateur fournisse d'emblée une u.s. donnée à tous les agents utilisateurs.

[2] Ce paramètre n'est pris en compte que lors d'un fonctionnement grandeur réelle. En simulation, tous les agents utilisent le service de localisation du simulateur.

il vaut `expe`, il est pris en compte lors des expériences grandeur réelle. S'il est absent, il est pris en compte dans les deux cas.

Pour la simulation, on donne en paramètre un *rayon de perception*[3], qui indique tout simplement le rayon de l'espace perceptuel de l'agent, supposé ici circulaire.

Exemple de description d'expérimentation

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE simulation SYSTEM "simu.dtd">

<simulation taxonomy="taxo_simple.xml">
  <locservice class="priam.localization.ir.IRReceiver" name="IRReceiver">
    <param name="config" value="marks/locservice_mappings.xml"/>
  </locservice>
  <locservice class="priam.localization.UbiquitousLocalizationService"
              name="AnywhereLoc">
    <param name="local0" value="jacques"/>
    <param name="local1" value="sylvie"/>
    <param name="local2" value="david"/>
    <param name="local3" value="carole"/>
    <param name="local4" value="frederic"/>
    <param name="local5" value="virginie"/>
    <param name="local6" value="jeremie"/>
    <param name="local7" value="claire"/>
  </locservice>
  <agent class="priam.agents.UserAgent" name="jacques">
    <param name="profile" value="userSighted.profile"/>
    <position x="50" y="20"/>
    <perception radius="80"/>
    <locservice local="IRReceiver"/>
  </agent>
  <agent class="priam.agents.UserAgent" name="sylvie">
    <param name="profile" value="userSighted.profile"/>
    <position x="50" y="80"/>
    <perception radius="80"/>
    <locservice local="IRReceiver"/>
  </agent>
  <agent class="priam.agents.UserAgent" name="david">
    <param name="profile" value="userSighted.profile"/>
    <position x="50" y="140"/>
```

[3]Grâce à un élément de type `<perception radius="..." />`.

```
    <perception radius="80"/>
    <locservice local="IRReceiver"/>
  </agent>
  <agent class="priam.agents.UserAgent" name="carole">
    <param name="profile" value="userSighted.profile"/>
    <position x="50" y="200"/>
    <perception radius="80"/>
    <locservice local="IRReceiver"/>
  </agent>
  <agent class="priam.agents.UserAgent" name="frederic">
    <param name="profile" value="userSighted.profile"/>
    <position x="50" y="260"/>
    <perception radius="80"/>
    <locservice local="IRReceiver"/>
  </agent>
  <agent class="priam.agents.UserAgent" name="virginie">
    <param name="profile" value="userSighted.profile"/>
    <position x="50" y="320"/>
    <perception radius="80"/>
    <locservice local="IRReceiver"/>
  </agent>
  <agent class="priam.agents.UserAgent" name="jeremie">
    <param name="profile" value="userSighted.profile"/>
    <position x="50" y="380"/>
    <perception radius="80"/>
    <locservice local="IRReceiver"/>
  </agent>
  <agent class="priam.agents.UserAgent" name="claire">
    <param name="profile" value="userSighted.profile"/>
    <position x="50" y="440"/>
    <perception radius="80"/>
    <locservice local="IRReceiver"/>
  </agent>
  <agent class="priam.apps.marks.MarkInformerAgent" name="informer">
    <locservice local="AnywhereLoc"/>
    <param name="su" value="Results"/>
    <param name="suProfile" value="suMark.profile"/>
    <position x="350" y="200"/>
    <perception radius="180"/>
    <param name="%DUPONT Jacques" value="jacques"/>
    <param name="%MERCIER Sylvie" value="sylvie"/>
    <param name="%MULLER David" value="david"/>
    <param name="%GIRAUD Carole" value="carole"/>
```

```
    <param name="%ROUSSEL Frederic" value="frederic"/>
    <param name="%FOURNIER Virginie" value="virginie"/>
    <param name="%PETIT Jeremie" value="jeremie"/>
    <param name="%VIDAL Claire" value="claire"/>
  </agent>
  <agent class="priam.agents.devices.DisplayPanel" name="panel">
    <param name="profile" value="screen.profile"/>
    <param name="modalityNodeId" value="n2"/>
    <param name="top" value="true" only="expe"/>
    <position x="500" y="600"/>
    <perception radius="90"/>
  </agent>
</simulation>
```

DTD des descriptions d'expérimentations

```
<!ELEMENT simulation ((locservice | agent)+)>

<!ATTLIST simulation taxonomy CDATA #REQUIRED>

<!ELEMENT locservice (param*)>

<!ATTLIST locservice class CDATA #IMPLIED
                     name  CDATA #IMPLIED
                     local CDATA #IMPLIED>

<!ELEMENT param EMPTY>

<!ATTLIST param name  CDATA #REQUIRED
                value CDATA #REQUIRED
                only  CDATA #IMPLIED>

<!ELEMENT agent ((param | position | perception | locservice)*)>

<!ATTLIST agent class CDATA #REQUIRED
                name  CDATA #REQUIRED>

<!ELEMENT position EMPTY>

<!ATTLIST position x CDATA #REQUIRED
                   y CDATA #REQUIRED>
```

```
<!ELEMENT perception EMPTY>

<!ATTLIST perception radius CDATA #REQUIRED>
```

Annexe C

Système d'identification de personnes par infrarouges

C.1 Introduction

Afin de permettre l'identification des personnes situées à proximité d'un dispositif de présentation lors de nos expérimentations, nous avions besoin d'un moyen d'identification répondant aux caractéristiques suivantes :

- *fiabilité* : détection rapide et à coup sûr des utilisateurs, ceci sans erreur ;
- *coût* : le plus faible possible ;
- *encombrement réduit* ;
- *compatibilité* maximale avec les équipements informatiques ;
- *facilité de mise en œuvre* ;
- *distance de détection* d'au moins deux mètres.

Notre première idée fut d'utiliser des badges RFID pour identifier les personnes, et d'installer des lecteurs RFID au niveau des dispositifs de présentation. Cependant, les lecteurs RFID disponibles pour une identification à quelques mètres de distance sont à la fois encombrants et onéreux. Nous nous sommes donc intéressés aux systèmes basés sur des badges actifs à infrarouges. Les systèmes à infrarouges peuvent répondre aux exigences énoncées ci-dessus, et présentent une caractéristique intéressante : ils permettent de détecter l'orientation de l'utilisateur, dans le sens où celui-ci n'est détecté que s'il se trouve face au dispositif de présentation. De cette façon, si le détecteur est installé sur un écran, il perçoit l'utilisateur si et seulement si ce dernier peut voir l'écran.

Un certain nombre de systèmes à badges infrarouges existent. Parmi eux, on peut citer :

- l'UbER Badge[1] du MIT[2] MediaLab ;
- l'IR Locator Badge[3] de la société Versus ;
- les badges Eiris de la société Hi-Bryd Solutions.

Le premier système a été créé par et pour des chercheurs, mais il est relativement complexe, et donc assez encombrant et onéreux : ainsi, les badges ne sont pas seulement équipés pour les infrarouges, mais aussi pour les transmissions par radio, la capture du son (microphone), et le stockage de données. Quant aux deux autres systèmes, ils sont conçus pour un usage industriel. Ils semblent être utilisés en majorité dans des hôpitaux, où ils permettent de localiser le personnel et les patients. Cependant, les quantités proposées par leurs fabricants sont d'ordre industriel, et donc non compatibles avec les besoins d'une petite équipe de recherche. Les fabricants n'ont d'ailleurs pas répondu à nos sollicitations.

Nous nous sommes donc tournés vers la réalisation de notre propre système de badges, conçu pour satisfaire nos besoins et le cahier des charges ci-dessus.

C.2 Description générale

L'architecture de notre système est très simple : les dispositifs de présentation sont équipés d'un récepteur infrarouge qui leur permet de détecter les émissions réalisées à proximité.

Les utilisateurs portent un badge, qui émet régulièrement un identifiant par infrarouges. Cet identifiant est associé de façon unique avec l'utilisateur correspondant, ce qui permet la reconnaissance univoque de ce dernier. La figure C.1 récapitule le principe de fonctionnement du système.

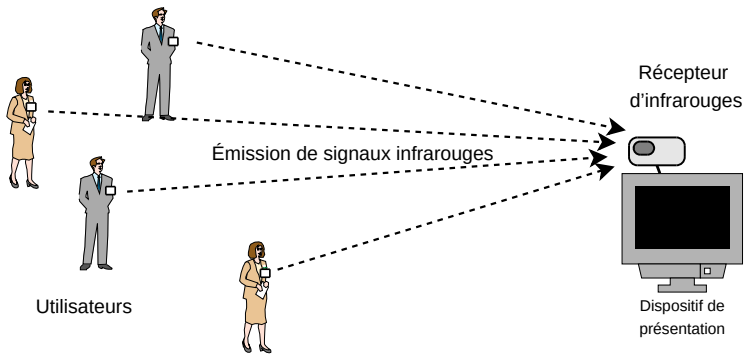


Figure C.1 : *Schéma général de fonctionnement de notre système d'identification par infrarouges.*

[1] Voir `http://www.media.mit.edu/resenv/badge/docs.html`.
[2] Massachussets Institute of Technology.
[3] Voir `http://www.versustech.com/products/badges_and_tags.html`.

C.3 Protocoles

Nos protocoles sont très fortement inspirés des protocoles des télécommandes pour l'électronique grand public, et en particulier de celui utilisé par le matériel de marque Nec. De cette façon, au cours des premières phases de développement, nous avons pu effectuer des tests des différents modules mis en œuvre à l'aide de télécommandes, d'émulateurs de télécommandes et de récepteurs de signaux de télécommandes.

C.3.1 Couche physique (OSI 1)

Au niveau physique, nous utilisons des infrarouges de longueur d'onde fixée à 950 nm. Afin d'assurer une certaine immunité au « bruit infrarouge » ambiant, causé par exemple par les éclairages et les corps chauds, les données ne sont pas transmises telles quelles, mais par modulation d'une porteuse de 38 kHz. Ces caractéristiques nous permettent d'utiliser pour la réception un composant très classique, le TSOP 1738, qui réalise aussi bien la réception des signaux infrarouges à 950 nm que la démodulation de la porteuse de 38 kHz.

Ce canal permet la transmission d'*impulsions* en modulation d'amplitude. Il existe deux sortes d'impulsions, à l'état *haut* ou à l'état *bas*. Une impulsion a une durée égale à 22 périodes de la porteuse à 38 kHz, soit 578 μs. Pendant une impulsion à l'état haut, on émet la porteuse à 38 kHz, tandis que pendant une impulsion à l'état bas, on n'émet rien du tout. L'allure des impulsions est donnée sur la figure C.2.

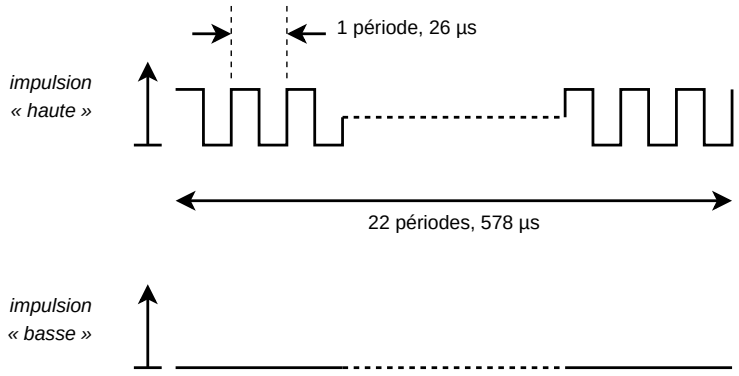


Figure C.2 : *Modulation des impulsions.*

C.3.2 Couche données (OSI 2)

Les données sont organisées en trames. Une trame est composée de quatre éléments :

- un marqueur de début : 8 impulsions hautes, suivies de 4 impulsions basses ;
- la charge utile : la série des n bits à transmettre ;
- les inverses des bits de la charge utile, dans le même ordre (afin de détecter les erreurs) ;

- un marqueur de fin : 4 impulsions hautes.

Les bits sont codés de la façon suivante :

- bit à 0 : une impulsion haute, puis trois impulsions basses ;
- bit à 1 : une impulsion haute, puis une impulsion basse.

Chaque bit de la charge utile est donc transmis deux fois : une fois tel quel, une fois inversé. L'une des deux fois, c'est un 0 qui est transmis ; l'autre fois, c'est un 1. Au total, un bit correspond donc à $4+2=6$ impulsions. Une trame est donc composée de $16+6n$ impulsions, où n est le nombre de bits dans la charge utile.

Pour une utilisation donnée, la charge utile est toujours de taille fixe, donc le nombre n est une constante, connue à la fois des émetteurs et des récepteurs. La figure C.3 donne un exemple de trame pour $n = 3$.

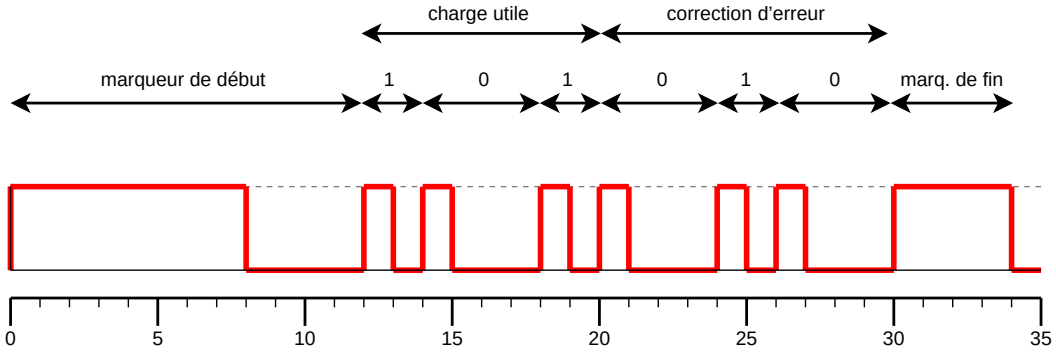


Figure C.3 : *Trame pour $n = 3$. Dans cet exemple, la charge utile est $\underline{101}_2$.*

Détection d'erreur La charge utile est dédoublée de façon à permettre une détection des erreurs. En pratique, une trame reçue sera déclarée valide si les conditions suivantes sont remplies :

- la trame reçue contient en tout $2n$ bits entre les marqueurs de début et de fin ;
- pour tout i compris entre 0 et $n-1$, le bit numéro i est l'inverse du bit numéro $i+n$.

Dans ce cas, la trame est déclarée correcte, et sa charge utile est composée des n premiers bits situés entre les marqueurs de début et de fin.

C.3.3 Couche application (OSI 7)

En pratique, la charge utile des trames est utilisée pour transmettre les identifiants des badges. Lors de nos expérimentations, nous disposions de huit badges : nous avons donc choisi $n = 3$

(comme sur la figure C.3). Dans ce cas, une trame correspond à 34 impulsions, soit une durée de 19,7 ms.

Chaque badge émet donc son identifiant à intervalles réguliers, de façon à être détecté par d'éventuels récepteurs. Ce mode de fonctionnement peut être problématique lorsque plusieurs badges émettent leurs identifiants dans une même zone géographique. En effet, s'il n'est pas très grave qu'une trame donnée soit brouillée par un autre badge (la correction d'erreur joue alors son rôle), il n'est pas tolérable que *toutes* les trames d'un badge soient brouillées, car dans ce cas, ce badge ne pourrait pas être détecté.

Dispositif anticollision Afin de résoudre ce problème, on peut penser à séparer les délais d'émission des badges par une durée *qui dépend du badge*. Ainsi, le badge numéro zéro séparerait ses émissions d'une durée T_0, le badge numéro 1 d'une durée T_1, etc. Cependant, si l'on n'y prend pas garde, il est possible que tout ou partie des émissions des badges soient brouillées lorsque les T_i ont des facteurs communs (voir fig. C.4).

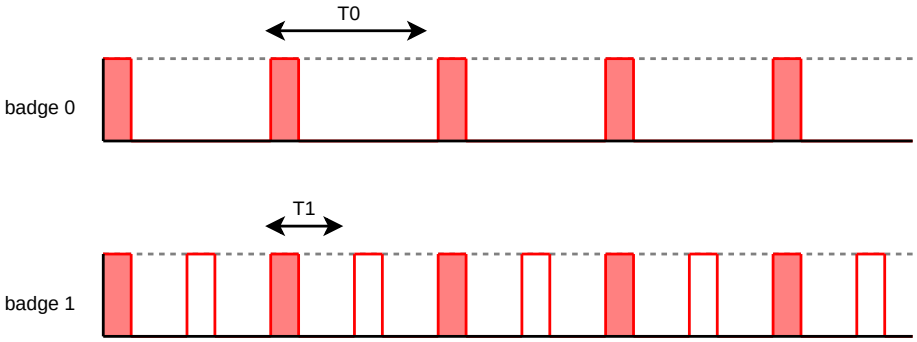


Figure C.4 : *Brouillage mutuel de deux badges, avec $T_0 = 2T_1$. Les émissions brouillées ont leur aire colorée.*

Sur cette figure, la période du badge n°0 est un multiple de la période du badge n°1. Si par malheur les émissions des deux badges sont en phase à un moment donné, elles le restent indéfiniment. Le mal est moindre pour le badge n°1, car seulement une de ses émissions sur deux est brouillée. Par contre, *toutes* les émissions du badge n°0 sont brouillées, ce qui n'est pas tolérable.

Pour éviter cette situation, nous avons décidé que les durées T_i seraient proportionnelles à une série de nombres premiers entre eux. Pour simplifier, les T_i sont tout simplement proportionnels à une série de *nombres premiers*. Ainsi, pour nos huit badges, nous avons fixé une constante t, et les durées T_i sont alors celles données par le tableau C.1.

La constante t a été choisie de sorte que $T_0 \approx 1$ s et $T_7 \approx 2$ s, ce qui représente des délais de détection corrects étant données les applications envisagées.

i (Numéro de badge)	Durée T_i
0	$T_0 = 31t$
1	$T_1 = 37t$
2	$T_2 = 41t$
3	$T_3 = 43t$
4	$T_4 = 47t$
5	$T_5 = 53t$
6	$T_6 = 59t$
7	$T_7 = 61t$

Tableau C.1 : *Durées destinées à éviter les collisions.*

C.4 Réalisation pratique

Les badges et les récepteurs associés ont été conçus et réalisés par nos soins.

C.4.1 Émetteur (badge)

L'émetteur doit générer la porteuse à 38 kHz, la moduler par les données (son identifiant), et enfin utiliser ce signal pour commander un dispositif d'émission d'infrarouges sur 950 nm. Nous avons jugé que le plus simple était d'utiliser un petit microcontrôleur pour la génération des signaux, étant donnée la nature numérique des données et du mode de modulation.

Notre choix s'est porté vers le PIC 12F508 de chez Microchip : dans un boîtier PDIP[4] de 8 broches, il intègre tous les composants nécessaires à son fonctionnement, y compris un oscillateur à 4 MHz, ainsi que 6 entrées/sorties. Ce microcontrôleur exécute un million d'instructions par seconde[5], ce qui est suffisant pour générer la porteuse (à 38 kHz la période est de 26 μs, soit 26 instructions du microcontrôleur).

Les émetteurs infrarouges sont des LED[6] sur 950 nm, de modèle LD 271. Ces LED permettent une émission d'assez forte puissance, avec un angle relativement grand. Nous utilisons deux de ces LED afin d'assurer une émission suffisamment puissante. Le schéma électronique d'un badge est donné sur la figure C.5.

Les émissions sont commandées par la broche GP4 du microcontrôleur, utilisée en sortie. Elle commande la base d'un transistor 2N2222 utilisé en commutation. Dans le circuit émetteur-collecteur de ce dernier se trouvent les deux LED infrarouges, une résistance de ballast et une résistance ajustable. La résistance de ballast R3 est calculée de sorte que le courant dans les LED soit de l'ordre de 100 mA lorsque le potentiomètre est au minimum. Lorsqu'il est au maximum par contre, le courant tombe à moins de 30 mA. Ceci permet dans une certaine

[4]Plastic Dual Inline Package.

[5]Sauf les instructions de saut, qui sont exécutées au rythme de 500 000 par seconde.

[6]En Français DEL, diode électro-luminescente.

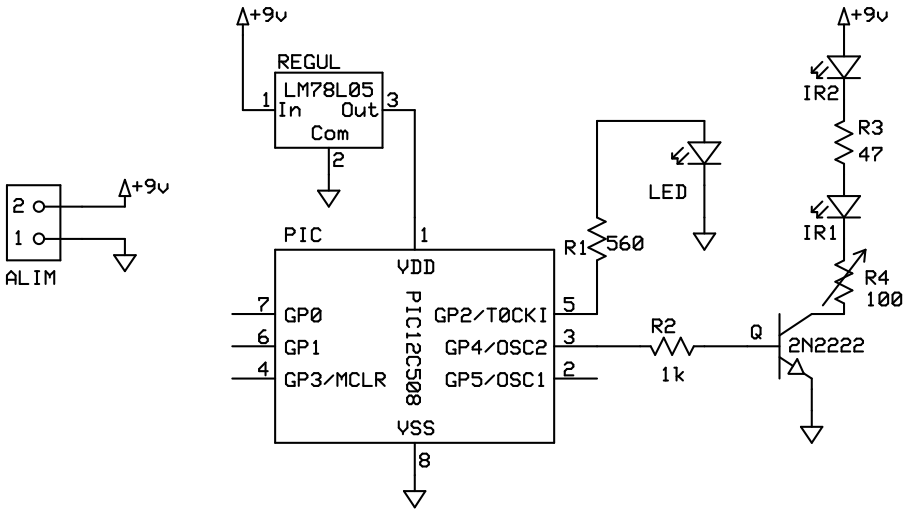


Figure C.5 : *Schéma électronique d'un badge.*

mesure de régler la portée des badges. La LED située sur la sortie GP2 sert de témoin visuel au fonctionnement du badge.

L'énergie est fournie au montage par une pile 9V. Un régulateur 7805 permet de fournir au microcontrôleur les 5V qui lui sont nécessaires. Les LED infrarouges quant à elles sont directement connectées sur le 9V.

Nous avons également intégré à la platine un connecteur ICSP[7] qui permet de reprogrammer les badges sans devoir démonter le microcontrôleur. Ce connecteur n'est pas représenté sur le schéma pour des raisons de lisibilité ; il est simplement relié aux broches correspondantes du PIC 12F508.

Le badge peut s'accrocher à un vêtement grâce à une épingle à nourrice. Les fils de connexion avec la pile sont suffisamment longs pour que cette dernière puisse trouver sa place dans une poche. La photo de la figure C.6 montre la réalisation finale.

C.4.2 Récepteur

Pour la réception, nous utilisons un composant intégré déjà évoqué, le TSOP 1738 : ce composant intègre un phototransistor, ainsi que la logique de démodulation de la porteuse à 38 kHz, avec boucle à verrouillage de phase et contrôle automatique de gain.

Nous avons cherché à connecter le décodeur à un ordinateur de la façon la plus simple possible, et c'est assez naturellement que l'interface USB[8] s'est imposée. En effet, l'USB représente un mode de connexion très fiable, présent sur tous les systèmes récents, et qui de plus permet d'alimenter en énergie les montages électroniques connectés.

[7] In-Circuit Serial Programming.
[8] Universal Serial Bus.

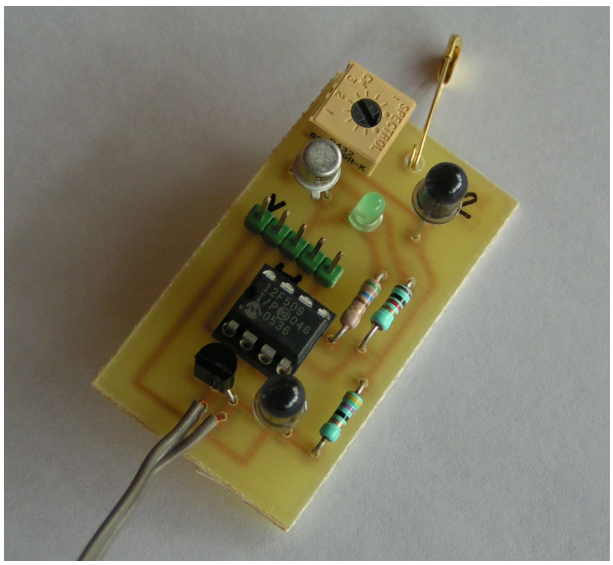


Figure C.6 : *Photo de l'un de nos badges à infrarouges.*

Afin de gérer la connexion USB, nous nous sommes à nouveau tournés vers un microcontrôleur PIC, le 18F2455. De plus, ce microcontrôleur est suffisamment puissant pour pouvoir effectuer lui-même le décodage des trames reçues. Ainsi, aucun traitement n'est effectué sur l'ordinateur : le montage de réception est autonome.

En outre, ce périphérique USB n'a pas besoin de pilote spécifique au niveau du système d'exploitation de l'ordinateur, car il implémente une classe de périphériques normalisée : les émulateurs de port série RS-232. Ainsi, les systèmes d'exploitation reconnaissent en standard le montage, auquel il est alors possible d'accéder par l'intermédiaire d'un port série virtuel. Le récepteur envoie sur ce port série virtuel les identifiants des badges situés à proximité lors de leur réception. Pour tester le fonctionnement du badge, le seul logiciel nécessaire est donc un émulateur de terminal série.

La figure C.7 montre le schéma d'un récepteur. Celui-ci reprend le schéma d'application classique du PIC 18F2455. Le TSOP 1738 est tout simplement connecté sur l'une des broches d'entrée du microcontrôleur. Deux LED permettent de vérifier le bon fonctionnement du montage.

La photo de la figure C.8 présente la réalisation finale. Ce montage se connecte directement sur un ordinateur, via un câble USB standard.

Décodage des trames Étant donnée la simplicité de notre protocole de transmission des trames, le décodage des informations reçues au niveau du récepteur peut s'effectuer à l'aide d'un automate à états fini très simple. De la sorte, le processus de décodage est très robuste. Cet automate à états est implémenté dans le PIC 18F2455 sous forme d'un programme en langage C compilé pour ce microcontrôleur.

La figure C.9 montre l'automate de décodage. Cet automate effectue ses transitions lors d'un changement d'état du signal infrarouge démodulé. Les transitions sont donc étiquetées

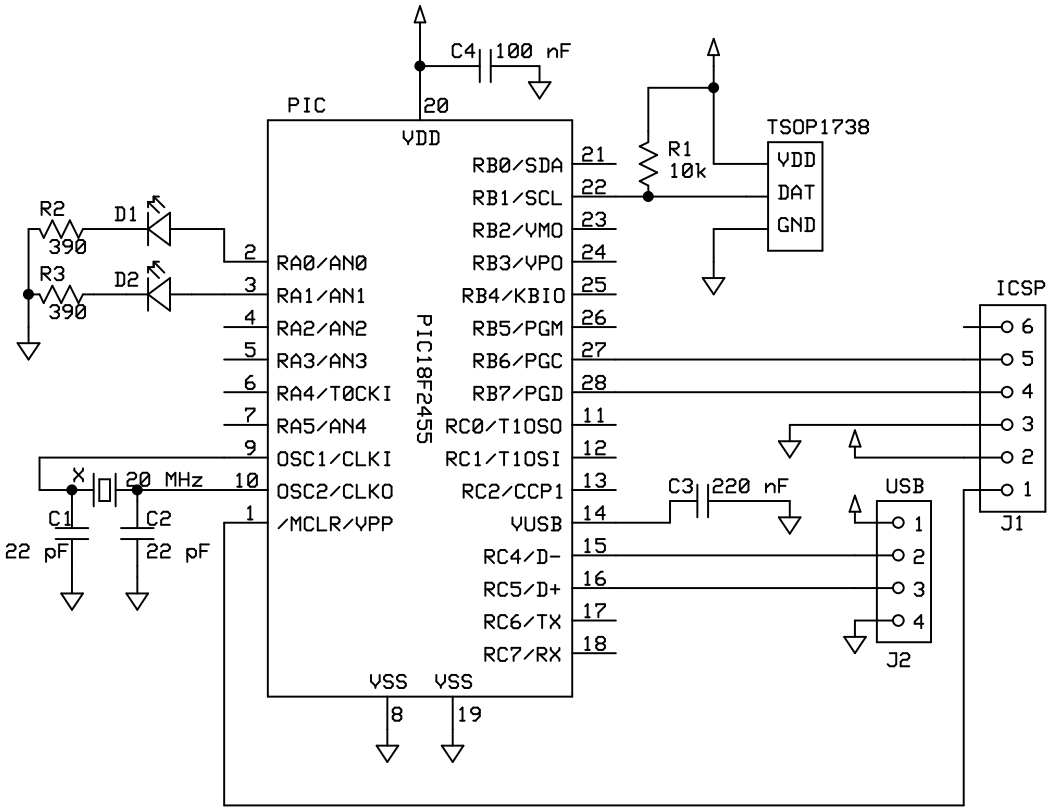


Figure C.7 : *Schéma électronique d'un décodeur.*

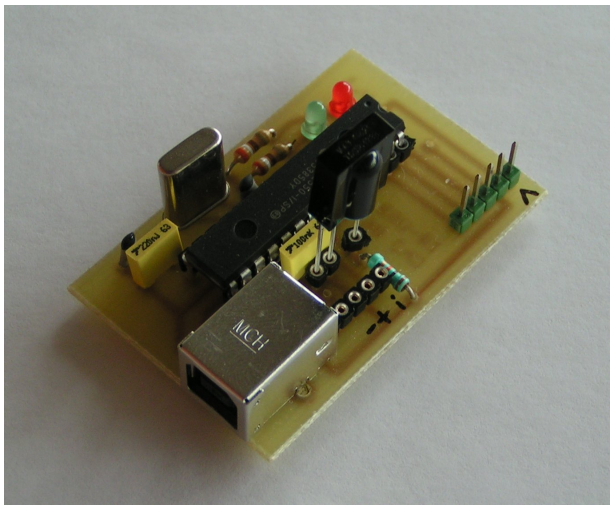


Figure C.8 : *Photo de l'un de nos récepteurs d'infrarouges.*

par le niveau précédent du signal (L pour bas, H pour haut), et le temps (en nombre de d'impulsions, i.e. d'intervalles de 578 μs) pendant lequel le signal est resté à ce niveau. Cet automate peut effectuer deux types d'actions : enregistrer un bit à 0 ou à 1 dans le tampon de réception de trame, ou bien déclencher les traitements de fin de trame (vérification d'erreur, reconstitution de la charge utile, transfert à l'ordinateur par USB).

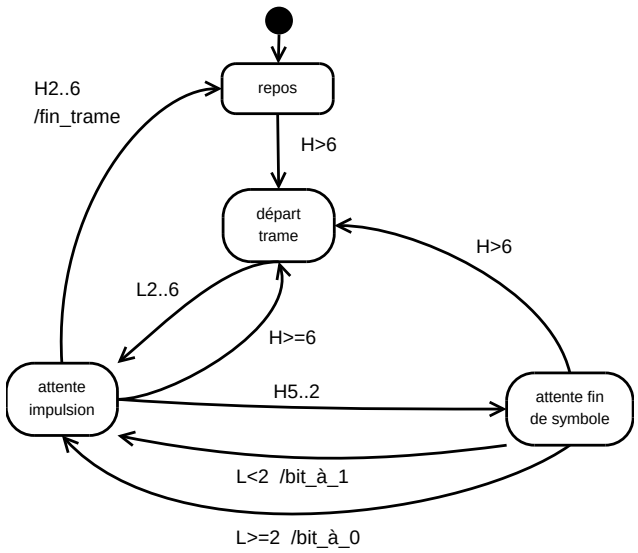


Figure C.9 : *Automate de décodage de trames.*

C.5 Conclusion

Le système présenté ici répond au cahier des charges que nous nous étions fixé :

- *fiabilité* : le système nous a donné entière satisfaction lors de nos évaluations. Ainsi, nous avons pu mener nos expérimentations sans plus nous préoccuper de cet aspect technique ;
- *coût* : le coût total pour la réalisation de huit badges (plus deux prototypes) et quatre récepteurs (plus un prototype) a été de l'ordre de 150 euros, ce qui représente une somme très modique ;
- *encombrement* : les badges sont très légers ; ils mesurent 50 mm × 28 mm. L'élément le plus encombrant et le plus lourd est la pile 9V, mais elle peut se glisser dans la poche[9] ;
- *compatibilité* avec tous les ordinateurs disposant d'un port USB, sur tous les systèmes d'exploitation ;
- *facilité de mise en œuvre* : il suffit de brancher la pile pour mettre en marche un badge. Le décodage des informations se fait dans l'électronique du récepteur, donc il n'est pas nécessaire d'installer de pilote ou de logiciel particulier sur les ordinateurs : le logiciel applicatif suffit ;
- *distance de détection* : selon les conditions et selon l'angle, elle varie de 2 à 5 mètres.

[9] L'encombrement du récepteur était moins critique, mais elle est également très faible : 56 mm × 38 mm.

Annexe D

Questionnaire aux volontaires

À la suite de nos expérimentations portant sur la recherche d'une information en conditions « pseudo-réelles » (voir section 8.1), nous avons demandé aux volontaires de répondre au questionnaire suivant :

- Quelles stratégies de recherche avez-vous appliqué pour retrouver les informations ?
- Y-a-t'il des choses qui vous ont perturbé dans le fonctionnement du système dynamique ? Si oui, lesquelles ?
- Quelle recherche était la plus difficile ? Les résultats, ou bien les vols (dans les deux cas, statique et dynamique) ? Pourquoi ?
- Pensez-vous qu'un tel système soit utile en pratique ? Soit viable à grande échelle ?
- Avez-vous des idées d'amélioration, des suggestions, des critiques ?
- Quelle méthode avez-vous préféré : un affichage statique (traditionnel), ou bien dynamique (avec les badges) ?

Ce questionnaire nous a permis de récolter des avis *subjectifs* sur le ressenti des expériences et de notre système par les utilisateurs.

Table des figures

Liste des tableaux

Index

Les références *principales* sont indiquées en caractères gras.

Nos publications

[Jacquet 2003] Christophe Jacquet, Yacine Bellik et René Farcy. *Description d'architectures en XML en vue de l'utilisation dans un système d'aide au déplacement des aveugles.* In Journée Nationale Image Signal pour le Handicap. J3EA, octobre 2003.

[Jacquet 2004a] Christophe Jacquet. *Modeling Environments for Use in A Location-Aware Locomotion Assistance Device for the Blind.* In Mohamed Kaâniche, éditeur, WCC 2004 Student Forum Proceedings, pages 393–403. Kluwer Academic Publishers, août 2004.

[Jacquet 2004b] Christophe Jacquet, Yacine Bellik et Yolaine Bourda. *A Context-Aware Locomotion Assistance Device for the Blind.* In Klaus Miesenberger, Joachim Klaus, Wolfgang Zagler et Dominique Burger, éditeurs, Computers Helping People with Special Needs, Proceedings of ICCHP 2004, volume 3118 de *Lecture Notes in Computer Science (LNCS)*, pages 438–445. Springer-Verlag, juillet 2004.

[Jacquet 2004c] Christophe Jacquet, Yacine Bellik, René Farcy et Yolaine Bourda. *Aides électroniques pour le déplacement des personnes non-voyantes: vue d'ensemble et perspectives.* In Proceedings of IHM 2004, 16th French-speaking conference on Human Computer Interaction, pages 93–100. ACM Press, septembre 2004.

[Jacquet 2004d] Christophe Jacquet, Yolaine Bourda et Yacine Bellik. *A Context-Aware Locomotion Assistance Device for the Blind.* In Sally Fincher, Panos Markopoulos, David Moore et Roy Ruddle, éditeurs, People and Computers XVIII – Design for Life, pages 315–328. Springer-Verlag (London), septembre 2004.

[Jacquet 2005] Christophe Jacquet, Yolaine Bourda et Yacine Bellik. *An Architecture for Ambient Computing.* In Hani Hagras et Victor Callaghan, éditeurs, The IEE International Workshop on Intelligent Environments, pages 47–54. The Institution of Electrical Engineers, juin 2005.

[Jacquet 2006a] Christophe Jacquet, Yacine Bellik et Yolaine Bourda. *Dynamic Cooperative Information Display in Mobile Environments.* In B. Gabrys, R.J. Howlett, et L.C. Jain, éditeurs, KES2006, 10th International Conference on Knowledge-Based & Intelligent Information & Engineering Systems, volume 4252 de

Lecture Notes in Artificial Intelligence (LNAI), pages 154–161. Springer-Verlag, octobre 2006.

[Jacquet 2006b] Christophe Jacquet, Yacine Bellik et Yolaine Bourda. *Electronic Locomotion Aids for the Blind: Towards More Assistive Systems.* In N. Ichalkaranje, A. Ichalkaranje et L.C. Jain, éditeurs, Intelligent Paradigms for Assistive and Preventive Healthcare, volume 19 de *Studies in Computational Intelligence*, chapitre 5, pages 133–163. Springer-Verlag, juin 2006.

[Jacquet 2006c] Christophe Jacquet, Yacine Bellik et Yolaine Bourda. *KUP, un modèle pour la présentation multimodale et opportuniste d'informations en situation de mobilité.* Ingénierie des Systèmes d'Information, vol. 11, no. 5, pages 115–139, novembre 2006. Adaptation et gestion du contexte.

[Jacquet 2006d] Christophe Jacquet, Yacine Bellik et Yolaine Bourda. *PRIAM : affichage dynamique d'informations par des écrans coopérants en environnement mobile.* In Actes d'Ubimob 2006, conférence francophone ubiquité et mobilité, pages 33–40, Paris, France, septembre 2006.

[Jacquet 2006e] Christophe Jacquet, Yolaine Bourda et Yacine Bellik. *A Component-Based Platform for Accessing Context in Ubiquitous Computing Applications.* Journal of Ubiquitous Computing and Intelligence (JUCI), 2006.

Bibliographie

[Aarts 2003] Emile Aarts et Raf Roovers. *IC design challenges for ambient intelligence.* In Proceedings of the Design, Automation and Test in Europe Conference and Exhibition, pages 2–7. IEEE, 2003.

[Aarts 2004] Emile Aarts. *Ambient intelligence: a multimedia perspective.* IEEE Multimedia, vol. 11, no. 1, pages 12–19, 2004.

[Ahuja 1986] Sudhir Ahuja, Nicholas Carriero et David Gelernter. *Linda and Friends.* IEEE Computer, vol. 19, no. 8, pages 26–34, 1986.

[André 2000] Elisabeth André. *The Generation of Multimedia Presentations.* In R. Dale, H. Moisl et H. Somers, éditeurs, A Handbook of Natural Language Processing, pages 305–327. Marcel Dekker, 2000.

[Anne 2005] Matthieu Anne, James L. Crowley, Vincent Devin et Gilles Privat. *Localisation intra-bâtiment multi-technologies: RFID, Wifi, vision.* In Actes de la conférence Ubimob 2005, pages 29–35, Grenoble, France, mai 2005.

[Appel 1991] Andrew W. Appel et David B. MacQueen. *Standard ML of New Jersey.* In J. Maluszyński et M. Wirsing, éditeurs, Proceedings of the 3rd International Symposium on Programming Language Implementation and Logic Programming, numéro 528 in Lecture Notes in Computer Science, pages 1–13. Springer, 1991.

[Asthana 1994] Abhaya Asthana, Mark Cravatts et Paul Krzyzanowski. *An Indoor Wireless System for Personalized Shopping Assistance.* In IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, California, USA, 1994.

[Bahl 2000] Paramvir Bahl et Venkata N. Padmanabhan. *RADAR: an in-building RF-based user location and tracking system.* In Proceedings of INFOCOM 2000, the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, volume 2, 2000.

[Bardram 2005] Jakob E. Bardram. *The Java Context Awareness Framework (JCAF) - A Service Infrastructure and Programming Framework for Context-Aware Applications.* In Hans-Werner Gellersen, Roy Want et Albrecht Schmidt,

éditeurs, Proceedings of Pervasive 2005, volume 3468 de *Lecture Notes in Computer Science*, pages 98–115, Munich, Germany, 2005. Springer.

[Bass 1992] Len Bass, Ross Faneuf, Reed Little, Niels Mayer, Bob Pellegrino, Scott Reed, Robert Seacord, Sylvia Sheppard et Martha R. Szczur. *A Metamodel for the Runtime Architecture of an Interactive System.* SIGCHI Bulletin, vol. 24, no. 1, pages 32–37, 1992.

[Baudoin 2005] G. Baudoin, O. Venard, G. Uzan, A. Paumier et J. Cesbron. *Le projet RAMPE: système interactif d'information auditive pour la mobilité des personnes aveugles dans les transports publics.* In Actes de la conférence Ubimob 2005, pages 169–176, Grenoble, France, mai 2005.

[Bellik 1992] Yacine Bellik et Daniel Teil. *Définitions terminologiques pour la communication multimodale.* In Proceeding of IHM'92, pages 229–232, novembre 1992.

[Bernstein 1996] David Bernstein et Alain Kornhauser. *An Introduction to Map Matching for Personal Navigation Assistants.* Rapport technique, Princeton University, New Jersey TIDE Center, New Jersey Institute of Technology, août 1996.

[Berrami 2001] Najat Berrami. Étude et définition de mécanismes de conversion transmodaux pour la présentation d'informations. Mémoire de DEA, encadrants Yacine Bellik et Jean-Paul Sansonnet, DEA Sciences Cognitives, Orsay, 2001.

[Betke 1994] Margrit Betke et Leonid Gurvits. *Mobile Robot Localization Using Landmarks.* In IEEE/RSJ/GI International Conference on Intelligent Robots and Systems, 1994.

[Bettstetter 2000] Christian Bettstetter et Christoph Renner. *A Comparison of Service Discovery Protocols and Implementation of the Service Location Protocol.* In Proc. EUNICE Open European Summer School, Twente, Netherlands, septembre 2000.

[Biegel 2004] Gregory Biegel et Vinny Cahill. *A Framework for Developing Mobile, Context-aware Applications.* In Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom 2004), pages 361–365, Orlando, Florida, USA, mars 2004. IEEE Computer Society.

[Blash 1991] B. B. Blash et R. G. Long. *Use or Non-Use of Electronic Travel Aids in the United States.* In The Spanish National Organization of the Blind, éditeur, Sixth International Mobility Conference, pages 49–58. , septembre 1991.

[Bohn 2004] Jürgen Bohn et Friedemann Mattern. *Super-Distributed RFID Tag Infrastructures.* In Proceedings of the 2nd European Symposium on Ambient

Intelligence (EUSAI 2004), numéro 3295 in Lecture Notes in Computer Science (LNCS), pages 1–12, Eindhoven, The Netherlands, novembre 2004. Springer-Verlag.

[Bordegoni 1997] Monica Bordegoni, Giorgio P. Faconti, Steven K. Feiner, Mark T. Maybury, Thomas Rist, S. Ruggieri, Panos E. Trahanias et Michael Wilson. *Standard reference model for intelligent multimedia presentation systems.* Computer Standards and Interfaces, vol. 18, no. 6, pages 477–496, 1997.

[Borenstein 1996] J. Borenstein, H.R. Everett et L. Feng. *Where am I? Sensors and Methods for Mobile Robot Positioning.* Rapport technique, The University of Michigan, avril 1996.

[Borgida 1995] Alexander Borgida. *Description Logics in Data Management.* IEEE Transactions on Knowledge and Data Engineering, vol. 7, no. 5, pages 671–682, octobre 1995.

[Brusey 2003] James Brusey, Christian Floerkemeier, Mark Harrison et Martyn Fletcher. *Reasoning About Uncertainty in Location Identification with RFID.* Workshop on Reasoning with Uncertainty in Robotics at IJCAI, août 2003.

[Bunt 2005] Harry Bunt, Michael Kipp, Mark T. Maybury et Wolfgang Wahlster. *Fusion and Coordination for Multimodal Interactive Information Presentation.* In Oliviero Stock et Massimo Zancanaro, éditeurs, Multimodal Intelligent Information Presentation, volume 27 de *Text, Speech and Language Technologies*, pages 325–340. Kluwer Academic Publishers, Dordrecht, février 2005.

[Carriero 1989] Nicholas Carriero et David Gelernter. *Linda in Context.* Communications of the ACM, vol. 32, no. 4, pages 444–458, 1989.

[Chao 2001] Jason Chao, Yong-qi Chen, Xiaoli Ding Wu Chen, Zhilin Li, Nganying Wong et Meng Yu. *An Experimental Investigation into the Performance of GPS-based Vehicle Positioning in Very Dense Urban Areas.* Journal of Geospatial Engineering, vol. 3, no. 1, pages 59–66, 2001.

[Chen 2000] Guanling Chen et David Kotz. *A Survey of Context-Aware Mobile Computing Research.* Rapport technique TR2000-381, Department of Computer Science, Dartmouth College, novembre 2000.

[Chetan 2004] Shiva Chetan, Jalal Al-Muhtadi, Roy Campbell et M. Dennis Mickunas. *A Middleware for Enabling Personal Ubiquitous Spaces.* In UbiSys '04: System Support for Ubiquitous Computing Workshop at Sixth Annual Conference on Ubiquitous Computing (UbiComp 2004), Nottingham, England, septembre 2004.

[Chetan 2005] Shiva Chetan, Jalal Al-Muhtadi, Roy Campbell et M. Dennis Mickunas. *Mobile Gaia: A Middleware for Ad-hoc Pervasive Computing.* In IEEE

Consumer Communications & Networking Conference (CCNC 2005), Las Vegas, USA, janvier 2005.

[Cohen 1993] Charles J. Cohen et Frank V. Koss. *A Comprehensive Study of Three Object Triangulation.* In Proceedings of the 1993 SPIE Conference on Mobile Robots, Boston, MA, USA, 1993.

[Coutaz 1987] Joëlle Coutaz. *PAC, an Object-Oriented Model for Dialog Design.* In Hans-Jorg Bullinger et Brian Shackel, éditeurs, INTERACT 87 - 2nd IFIP International Conference on Human-Computer Interaction, pages 431–436. North-Holland, septembre 1987.

[Coutaz 1991] Joëlle Coutaz. *Prospects in Software Design with Multi-modal interactive systems.* In Proceedings of SITEF International Symposium on Cognitive Interactions, pages 81–90, Toulouse, France, octobre 1991.

[Coutaz 1995] Joëlle Coutaz, Laurence Nigay, Daniel Salber, Ann Blandford, Jon May et Young M. Richard. *Four easy pieces for assessing the usability of multimodal interaction: the CARE properties.* In Proceedings of INTERACT'95: Fifth IFIP Conference on Human-Computer Interaction, pages 115–120, 1995.

[Coutaz 2002] Joëlle Coutaz et Gaëtan Rey. *Foundations for a Theory of Contextors.* In Computer-Aided Design of User Interfaces III, Proceedings of the Fourth International Conference on Computer-Aided Design of User Interfaces, pages 13–34. Kluwer, mai 2002.

[Daniele 2003] Norbert Daniele, M. Pezzin, S. Derivaz, J. Keignart et Philippe Rouzet. *Principle and Motivations of UWB Technology for High Data Rate WPAN Applications.* In Proceedings of sOc 2003, pages 84–87, Grenoble, France, mai 2003.

[den Os 2003] Els den Os et Lou Boves. *Towards ambient intelligence: Multimodal computers that understand our intentions.* In Proceedings of eChallenges, volume 3, 2003.

[der Hardt 1997] Hans Joachim Von der Hardt. *Contribution au pilotage et à la localisation d'un robot mobile.* Thèse de doctorat, Institut National Polytechnique de Lorraine, Nancy, 1997.

[Dey 2000] Anind K. Dey et Gregory D. Abowd. *Providing architectural support for building context-aware applications.* Thèse de doctorat, Georgia Institute of Technology, 2000.

[Dey 2001] Anind K. Dey, Daniel Salber et Gregory D. Abowd. *A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications.* Human Computer Interaction, vol. 16, no. 2-4, pages 97–166, 2001.

[Dubois 1999] Emmanuel Dubois, Laurence Nigay, Jocelyne Troccaz, Olivier Chavanon et Lionel Carrat. *Classification Space for Augmented Surgery, an Augmented Reality Case Study.* In Proceedings of Interact'99, pages 353–359, 1999.

[Ducatel 2001] K. Ducatel, M. Bogdanowicz, F. Scapolo, J. Leijten et J-C. Burgelman. *Scenarios for Ambient Intelligence in 2010.* Final report, Information Society Technologies Advisory Group (ISTAG), European Commission, février 2001.

[Eissfeller 2004] Bernd Eissfeller, Danilo Gaensch, Swen Müller et Andreas Teuber. *Indoor Positioning Using Wireless LAN Radio Signals.* In Proceedings of ION-GNSS 2004, pages 1936–1947, Long Beach, California, USA, septembre 2004.

[Fitzmaurice 1993] George W. Fitzmaurice. *Situated information spaces and spatially aware palmtop computers.* Communications of the ACM, vol. 36, no. 7, pages 39–49, 1993.

[Floerkemeier 2004] Christian Floerkemeier et Matthias Lampe. *Issues with RFID usage in ubiquitous computing applications.* In Alois Ferscha et Friedemann Mattern, éditeurs, Pervasive Computing: Second International Conference, PERVASIVE 2004, numéro 3001 in LNCS, pages 188–193, Linz/Vienna, Austria, avril 2004. Springer-Verlag.

[Fontana 2003] Robert J. Fontana, Edward Richley et JoAnn Barney. *Commercialization of an ultra wideband precision asset location system.* In Proceedings of IEEE Conference on Ultra Wideband Systems and Technologies, pages 369–373, 2003.

[Fox 1998] Dieter Fox, Wolfram Burgard et Sebastian Thrun. *Active Markov Localization for Mobile Robots.* Robotics and Autonomous Systems, vol. 25, pages 195–207, 1998.

[Friedewald 2005] Michael Friedewald, Olivier Da Costa, Yves Punie, Petteri Alahuhta et SirkkaHeinonen. *Perspectives of ambient intelligence in the home environment.* Telematics and Informatics, vol. 22, no. 3, pages 221–238, 2005.

[Fusiello 1997] Andrea Fusiello et Bruno Caprile. *Synthesis of Indoor Maps in Presence of Uncertainty.* Robotics and Autonomous Systems, vol. 22, no. 2, pages 103–114, novembre 1997.

[Gellersen 2004] Hans Gellersen, Gerd Kortuem, Albrecht Schmidt et Michael Beigl. *Physical Prototyping with Smart-Its.* IEEE Pervasive Computing, vol. 3, no. 3, pages 74–82, 2004.

[Gray 2001] Philip D. Gray et Daniel Salber. *Modelling and Using Sensed Context Information in the Design of Interactive Applications.* In Proceedings of the

8th IFIP International Conference on Engineering for Human-Computer Interaction, pages 317–336. Springer-Verlag, 2001.

[Greenhalgh 2002] Chris Greenhalgh. *EQUIP: a Software Platform for Distributed Interactive Systems.* Technical Report, Mixed Reality Laboratory, University of Nottingham, 2002.

[Hagras 2004] Hani Hagras, Victor Callaghan, Martin Colley, Graham Clarke, Anthony Pounds-Cornish et Hakan Duman. *Creating an Ambient-Intelligence Environment Using Embedded Agents.* IEEE Intelligent Systems, vol. 19, no. 6, pages 12–20, 2004.

[Hähnel 2004] D. Hähnel, Wolfram Burgard, Dieter Fox, K. Fishkin et M. Philipose. *Mapping and Localization with RFID Technology.* In Proc. of the IEEE International Conference on Robotics and Automation (ICRA), pages 1015–1020, 2004.

[Harter 1994] Andy Harter et Andy Hopper. *A Distributed Location System for the Active Office.* IEEE Network, vol. 8, no. 1, pages 62–70, janvier 1994.

[Harter 1999] Andy Harter, Andy Hopper, Pete Steggles, Andy Ward et Paul Webster. *The anatomy of a context-aware application.* In MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking, pages 59–68, Seattle, Washington, USA, 1999. ACM Press.

[Hartmann 1998] William M. Hartmann. Signals, Sound and Sensation. Springer, 1998.

[Hightower 2001] Jeffrey Hightower et Gaetano Borriello. *Location Systems for Ubiquitous Computing.* IEEE Computer, vol. 34, no. 8, pages 57–66, 2001.

[Hightower 2002] Jeffrey Hightower, Barry Brumitt et Gaetano Borriello. *The location stack: A layered model for location in ubiquitous computing.* In Proceedings of the 4th IEEE Workshop on mobile Computing Systems & Applications (WMCSA 2002), pages 22–28, Callicoon, New-York, USA, juin 2002. IEEE Computer Society Press.

[Hong 2002] Jason I. Hong. *The context fabric: an infrastructure for context-aware computing.* In CHI '02: CHI '02 extended abstracts on Human factors in computing systems, pages 554–555, Minneapolis, Minnesota, USA, 2002. ACM Press.

[Hull 1997] Richard Hull, Philip Neaves et James Bedford-Roberts. *Towards Situated Computing.* In ISWC '97, page 146, Washington, DC, USA, 1997. IEEE Comp. Soc.

[Jansson 1991] G. Jansson. *The Functions of Present and Future Electronic Travel Aids for Visually Impaired Children and Adults.* In The Spanish National Organization of the Blind, éditeur, Proceedings of the Sixth International Mobility Conference, pages 59–64, septembre 1991.

[Jensfelt 2001] Patric Jensfelt et Steen Kristensen. *Active Global Localisation for a Mobile Robot Using Multiple Hypothesis Tracking.* IEEE Transactions on Robotics and Automation, vol. 17, no. 5, pages 748–760, octobre 2001.

[Johanson 2002] Brad Johanson, Armando Fox et Terry Winograd. *The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms.* IEEE Pervasive Computing, vol. 1, no. 2, pages 67–74, 2002.

[Judd 2003] Glenn Judd et Peter Steenkiste. *Providing Contextual Information to Pervasive Computing Applications.* In PERCOM '03: Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, pages 133–142, Washington, DC, USA, 2003. IEEE Computer Society.

[Kavanagh 2001] Kevin T. Kavanagh. *Evaluation of hearing handicaps and presbyacusis using World Wide Web-based calculators.* Journal of the American Academy of Audiology, vol. 12, no. 10, pages 497–505, novembre 2001.

[Kindberg 2001] Tim Kindberg et John Barton. *A Web-based nomadic computing system.* Computer Networks (Amsterdam, Netherlands: 1999), vol. 35, no. 4, pages 443–456, 2001.

[Kindberg 2002] Tim Kindberg et Armando Fox. *System Software for Ubiquitous Computing.* IEEE Pervasive Computing, vol. 1, no. 1, pages 70–81, 2002.

[Kitazawa 2000] Kay Kitazawa, Yusuke Konishi et Ryosuke Shibasaki. *A Method of Map Matching For Personal Positioning Systems.* In Proceedings of the Asian Conference on Remote Sensing (ACRS 2000), volume 2, pages 726–731, Taipei, décembre 2000.

[Konishi 2001] Yusuke Konishi et Ryosuke Shibasaki. *Development of an autonomous personal positioning system.* In Proceedings of Asia GIS 2001, juin 2001.

[Krasner 1988] Glenn E. Krasner et Stephen T. Pope. *A cookbook for using the model-view controller user interface paradigm in Smalltalk-80.* Journal of Object Oriented Programming, vol. 1, no. 3, pages 26–49, 1988.

[Krumm 2003] John Krumm, Gerry Cermak et Eric Horvitz. *RightSPOT: A Novel Sense of Location for a Smart Personal Object.* In Anind K. Dey, Albrecht Schmidt et Joseph F. McCarthy, éditeurs, UbiComp 2003: Ubiquitous Computing, 5th International Conference, Seattle, WA, USA, October 12-15, 2003, Proceedings, volume 2864 de *Lecture Notes in Computer Science*, pages 36–43. Springer, 2003.

[Ladetto 2000] Quentin Ladetto. *On foot navigation: continuous step calibration using both complementary recursive prediction and adaptive Kalman filtering.* In ION GPS 2000, Salt Lake City, Utah, USA, septembre 2000.

[Lallement 1999] Alex Lallement. *Localisation d'un robot mobile par coopération entre vision monoculaire et télémétrie laser.* Thèse de doctorat, INPL-CRAN, juillet 1999.

[LaMarca 2005] Anthony LaMarca, Yatin Chawathe, Sunny Consolvo, Jeffrey Hightower, Ian Smith, James Scott, Tim Sohn, James Howard, Jeff Hughes, Fred Potter, Jason Tabert, Pauline Powledge, Gaetano Borriello et Bill Schilit. *Place Lab: Device Positioning Using Radio Beacons in the Wild.* In Proceedings of the 3rd International Conference on Pervasive Computing (Pervasive 2005), volume 3468 de *LNCS*. Springer-Verlag, mai 2005.

[Livatino 1994] Salvatore Livatino et Claus B. Madsen. *Automatic Selection of Visual Landmark for Mobile Robot Navigation.* In Computer Vision and Mobile Robotics Workshop, 1994.

[Long 1996] Sue Long, Rob Kooper, Gregory D. Abowd et Christopher G. Atkeson. *Rapid Prototyping of Mobile Context-Aware Applications: The Cyberguide Case Study.* In Mobile Computing and Networking, pages 97–107, 1996.

[Madsen 1997] Claus B. Madsen, Claus S. Andersen et Jens S. Sørensen. *A Robustness Analysis of Triangulation-based Robot Self-positioning.* In Proceedings of the International Symposium on Intelligent Robotic Systems, Stockholm, Sweden, 1997.

[Mailler 2004] Roger Mailler et Victor Lesser. *Solving Distributed Constraint Optimization Problems Using Cooperative Mediation.* In Proceedings of AAMAS 2004, pages 438–445. IEEE Computer Society, 2004.

[Mansoux 2005] Benoît Mansoux, Laurence Nigay et Jocelyne Troccaz. *The Mini-Screen: an Innovative Device for Computer Assisted Surgery Systems.* Studies in Health Technology and Informatics, vol. 111, pages 314–320, 2005.

[Mäntyjärvi 2001] Jani Mäntyjärvi, Johan Himberg et Tapio Seppänen. *Recognizing human motion with multiple acceleration sensors.* In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, volume 2, pages 747–752. IEEE Press, 2001.

[Marucci 2002] Luisa Marucci et Fabio Paternò. *Supporting Adaptivity to Heterogeneous Platforms through User Models.* In Proceedings of the 4th International Symposium on Mobile Human-Computer Interaction, pages 409–413. Springer-Verlag, 2002.

[Meier 2003] René Meier et Vinny Cahill. *Exploiting Proximity in Event-Based Middleware for Collaborative Mobile Applications.* In Jean-Bernard Stefani, Isabelle M. Demeure et Daniel Hagimont, éditeurs, Proceedings of the 4^{th} IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS'03), volume 2893 de *Lecture Notes in Computer*

Science, pages 285–296, Heidelberg, Germany, novembre 2003. Springer-Verlag.

[Merle 2002] Philippe Merle. *CORBA Component Model Tutorial*. In OMG Meeting, Yokohama, Japan, avril 2002.

[Milgram 1994] Paul Milgram, Haruo Takemura, Akira Utsumi et Fumio Kishino. *Augmented Reality: A Class of Displays on the Reality-Virtuality Continuum*. In Proceedings of Telemanipulator and Telepresence Technologies, pages 282–292, 1994.

[Miller 1998] Eric Miller. *An Introduction to the Resource Description Framework*. D-Lib Magazine, mai 1998.

[Mizoguchi 2001] Riichiro Mizoguchi. *Ontological Engineering: Foundation of the next generation knowledge processing*. In N.Zhong*et al.*, éditeurs, WI2001, volume 2198 de *Lecture Notes in Artificial Intelligence*, pages 44–57. Springer-Verlag, 2001.

[Ni 2004] Lionel M. Ni, Yunhao Liu, Yiu Cho Lau et Abhishek P. Patil. *LANDMARC: indoor location sensing using active RFID*. Wireless Networks, vol. 10, no. 6, pages 701–710, 2004.

[Nigay 1995] Laurence Nigay et Joëlle Coutaz. *A generic platform for addressing the multimodal challenge*. In Proceedings of the SIGCHI conference on Human factors in computing systems, pages 98–105, Denver, Colorado, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[Nigay 1997] Laurence Nigay et Joëlle Coutaz. *Multifeature Systems: The CARE Properties and Their Impact on Software Design*. In John Lee, éditeur, Intelligence and Multimodality in Multimedia Interfaces. AAAI Press, 1997.

[Nigay 2001] Laurence Nigay, Emmanuel Dubois et Jocelyne Troccaz. *Compatibility and Continuity in Augmented Reality Systems*. In I3 Spring Days Workshop, Continuity in Future Computing Systems, Porto, Portugal, avril 2001.

[Orr 2000] Robert J. Orr et Gregory D. Abowd. *The Smart Floor: a Mechanism for Natural User Identification and Tracking*. In Proceedings of CHI 2000, the Conference on Human Factors in Computing Systems, pages 275–276. ACM Press, avril 2000.

[Pascoe 1997] Jason Pascoe. *The stick-e note architecture: extending the interface beyond the user*. In IUI '97: Proceedings of the 2nd international conference on Intelligent user interfaces, pages 261–264, Orlando, Florida, USA, 1997. ACM Press.

[Pfaff 1985] Günther E. Pfaff, éditeur. User Interface Management Systems: Proceedings of the Seeheim Workshop. Springer, 1985.

[Philip 2002] Ogunro Philip. *Car Navigational Systems: Futuristic Systems.* In 2nd Annual CM316 Conference on Multimedia Systems, Southampton University, UK, 2002. Southampton University.

[Pireaux 1964] Henry Pireaux. Dictionnaire général d'acoustique et d'électro-acoustique. Eyrolles, 1964.

[Podevin 2002] Aurore Podevin. Accès aux formules mathématiques par des personnes non voyantes : étude et définition d'une méthode adaptée. Mémoire de DEA, encadrants Christine Porquet (ENSI Caen), Yacine Bellik et Pierre Lorenzon (LIMSI), Anne Nicolle (Université de Caen), DEA Intelligence Artificielle et Algorithmique, Caen, 2002.

[Randell 2000] Cliff Randell et Henk Muller. *Context Awareness by Analysing Accelerometer Data.* In Blair MacIntyre et Bob Iannucci, éditeurs, The Fourth International Symposium on Wearable Computers, pages 175–176. IEEE Computer Society, octobre 2000.

[Rekleitis 2004] Ioannis M. Rekleitis. *A Particle Filter Tutorial for Mobile Robot Localization.* Rapport technique TR-CIM-04-02, Centre for Intelligent Machines, McGill University, Montreal, Québec, Canada, 2004.

[Rey 2002] Gaëtan Rey et Joëlle Coutaz. *Le contexteur : une abstraction logicielle pour la réalisation de systèmes interactifs sensibles au contexte.* In Proceedings of the 14th French-speaking conference on Human-computer interaction (Conférence Francophone sur l'Interaction Homme-Machine), pages 105–112, Poitiers, France, 2002. ACM Press.

[Rey 2004] Gaëtan Rey et Joëlle Coutaz. *Le Contexteur : Capture et distribution dynamique d'information contextuelle.* In Actes de la conférence Ubimob 04, pages 331–338, Nice, France, juin 2004.

[Ricquebourg 2006] Vincent Ricquebourg, David Menga, Laurent Delahoche, Bruno Marhic, David Durand et Christophe Logé. *Architecture de perception de contexte orientée service pour l'habitat communicant.* In Actes de la conférence Ubimob 2006, pages 65–72, Paris, France, septembre 2006.

[Rist 2005] Thomas Rist. *Supporting Mobile Users Through Adaptive Information Presentation.* In Oliviero Stock et Massimo Zancanaro, éditeurs, Multimodal Intelligent Information Presentation, volume 27 de *Text, Speech and Language Technologies*, chapitre 6, pages 113–141. Kluwer Academic Publishers, Dordrecht, février 2005.

[Römer 2004] Kay Römer, Thomas Schoch, Friedemann Mattern et Thomas Dübendorfer. *Smart identification frameworks for ubiquitous computing applications.* Wireless Networks, vol. 10, no. 6, pages 689–700, 2004.

[Roth 2003] Jörg Roth. *Flexible Positioning for Location-based Services.* In IADIS International Conference e-Society, volume I, pages 296–304, Lisbonne, Portugal, juin 2003. IADIS Press.

[Roudaut 2006] Anne Roudaut et Joëlle Coutaz. *Méta-IHM ou comment contrôler l'espace interactif ambiant.* In Actes de la conférence Ubimob 2006, pages 73–80, Paris, France, septembre 2006.

[Rousseau 2003] Cyril Rousseau. Multimodalité en sortie : étude d un modèle pour une interaction Homme-Machine dynamique et contextuelle. Mémoire de DEA, encadrants Yacine Bellik et Jean-Paul Sansonnet, DEA Information, Interaction, Intelligence, Orsay, 2003.

[Rousseau 2005] Cyril Rousseau, Yacine Bellik et Frédéric Vernier. *WWHT: Un modèle conceptuel pour la présentation multimodale d'information.* In Proceedings of 17th French-speaking conference on Human Computer Interaction (IHM 2005), pages 27–30, Toulouse, France, septembre 2005.

[Samama 2005] Nel Samama, Alexandre Vervisch-Picois et Marc Francois. *La localisation en intérieur à l'aide de répéteurs GPS: vers un système de positionnement universel ?* In Actes de la conférence Ubimob 2005, pages 21–28, Grenoble, France, mai 2005.

[Scharstein 1999] Daniel Scharstein et Amy J. Briggs. *Real-time Recognition of Self-similar Landmarks.* In Workshop on Perception for Mobile Agents, pages 74–81, juin 1999.

[Schilit 1993] Bill Noah Schilit, Norman Adams, Rich Gold, Michael Tso et Roy Want. *The ParcTab Mobile Computing System.* In Proceedings Fourth Workshop on Workstation Operating Systems (WWOS-IV), pages 34–39, Napa, California, octobre 1993. IEEE.

[Schilit 1994] Bill Schilit, Norman Adams et Roy Want. *Context-Aware Computing Applications.* In IEEE Workshop on Mobile Computing Systems and Applications, pages 85–90, Santa Cruz, CA, US, 1994.

[Schilit 1995] Bill Schilit. *System Architecture for Context-Aware Mobile Computing.* Thèse de doctorat, Columbia University, 1995.

[Schmidt 1999] Albrecht Schmidt, Kofi Asante Aidoo, Antti Takaluoma, Urpo Tuomela, Kristof Van Laerhoven et Walter Van de Velde. *Advanced Interaction in Context.* In HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing, pages 89–101, London, UK, 1999. Springer-Verlag.

[Shadbolt 2003] Nigel Shadbolt. *Ambient Intelligence.* IEEE Intelligent Systems, vol. 18, no. 4, pages 2–3, 2003.

[Souchon 2002] Nathalie Souchon, Quentin Limbourg et Jean Vanderdonckt. *Task Modelling in Multiple Contexts of Use*. In Proceedings of the 9th International Workshop on Interactive Systems. Design, Specification, and Verification, pages 59–73. Springer-Verlag, 2002.

[Spoor 1967] A. Spoor. *Presbycusis values in relation to noise induced hearing loss*. Int. Audiology, vol. 6, pages 48–57, 1967.

[Streitz 2003] Norbert A. Streitz, Carsten Röcker, Thorsten Prante, Richard Stenzel et Daniel van Alphen. *Situated Interaction with Ambient Information: Facilitating Awareness and Communication in Ubiquitous Work Environments*. In HCI International 2003, juin 2003.

[Strietzel 1999] Roland Strietzel. *A Rendezvous and Docking Sensor with CCD Camera*. In EEC '99, 1999.

[Sutton 1979] D. W. Sutton et G. J. Robinson. *Age effects and hearing. A comparative analysis of published threshold data*. Audiology, vol. 18, pages 320–334, 1979.

[Teil 2000] Daniel Teil et Yacine Bellik. *Multimodal Interaction Interface using Voice and Gesture*. In M. M. Taylor, F. Néel et D. G. Bouwhuis, éditeurs, The Structure of Multimodal Dialogue II, chapitre 19, pages 349–366. John Benjamins Publishing, 2000.

[Thevenin 1999] David Thevenin et Joëlle Coutaz. *Plasticity of User Interfaces: Framework and Research Agenda*. In Interact '99, volume 1, pages 110–117. IOS Press, 1999.

[van Loenen 2003] Evert J. van Loenen. *On the Role of Graspable Objects in the Ambient Intelligence Paradigm*. In Smart Objects Conference, pages 3–7, Grenoble, France, mai 2003.

[Varile 1996] Giovanni Varile, Antonio Zampolli, Ronald A. Cole, Joseph Mariani, Hans Uszkoreit, Annie Zaenen et Victor Zue, éditeurs. Survey of the State of the Art in Human Language Technology, chapitre 4, pages 131–153. Cambridge University Press, 1996.

[Voelker 1994] Geoffrey M. Voelker et Brian N. Bershad. *Mobisaic: an information system for a mobile wireless computing environment*. In Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications, pages 185–190, 1994.

[Vogel 2004] Daniel Vogel et Ravin Balakrishnan. *Interactive public ambient displays: transitioning from implicit to explicit, public to personal, interaction with multiple users*. In UIST '04, pages 137–146, Santa Fe, NM, USA, 2004. ACM Press.

[Volpe 1995] Richard Volpe, Todd Litwin et Larry Matthies. *Mobile Robot Localization by Remote Viewing of a Colored Cylinder.* In Proceedings of the International Conference on Intelligent Robots and Systems (IROS '95), Pittsburgh, USA, 1995.

[Wang 2004] Xiaohang Wang, Jin Song Dong, ChungYau Chin, SankaRavipriya Hettiarachchi et Daqing Zhang. *Semantic Space: An Infrastructure for Smart Spaces.* IEEE Pervasive Computing, vol. 03, no. 3, pages 32–39, 2004.

[Want 1992] Roy Want, Andy Hopper, Veronica Falc ao et Jonathan Gibbons. *The Active Badge Location System.* ACM Transactions on Information Systems, vol. 10, no. 1, pages 91–102, 1992.

[Weber 2003] Werner Weber. *Ambient intelligence: industrial research on a visionary concept.* In Proceedings of the 2003 international symposium on Low power electronics and design, pages 247–251. ACM Press, 2003.

[Weiser 1993] Mark Weiser. *Some computer science issues in ubiquitous computing.* Communications of the ACM, vol. 36, no. 7, pages 75–84, 1993.

[White 2000] Christopher E. White, David Bernstein et Alain L. Kornhauser. *Some map matching algorithms for personal navigation assistants.* Transportation Research Part C: Emerging Technologies, vol. 8, pages 91–108, février 2000.

[Yoshimura 2003] Tetsuhiko Yoshimura et Naoto Hasegawa. *Development of a highly accurate navigation system for forestry vehicles and workers.* In High Tech Forest Operations for Mountainous Terrain (Austro 2003), Schlaegl, Austria, octobre 2003.

[Zock 2002] Michael Zock et Gérard Sabah. *La génération automatique de textes.* In M. Fayol, éditeur, Production du langage, pages 263–285. Hermès, 2002.

Résumé

Ce travail se place dans le domaine de l'interaction homme-machine et en particulier dans celui de la multimodalité et des systèmes ambiants. Il vise à concevoir un modèle théorique et une plate-forme pour la spécification et l'implémentation de services d'assistance à des utilisateurs mobiles. Nous introduisons le modèle KUP, dans lequel le noyau fonctionnel du système, les utilisateurs et les dispositifs de présentation (écrans, haut-parleurs, etc.) sont représentés par des entités logiques. Ce modèle permet un découplage spatial et temporel entre d'une part la fourniture d'une information par le noyau fonctionnel à l'entité utilisateur, et d'autre part, la présentation de cette information par un dispositif adéquat, sur demande de l'entité utilisateur. Ces deux phases sont opportunistes : elles surviennent au gré des déplacements des utilisateurs.

Lorsqu'un utilisateur se trouve à proximité de dispositifs de présentation, il faut déterminer quel dispositif et quelle modalité utiliser. Un premier algorithme, conçu de façon incrémentale, permet de choisir le dispositif tout en respectant trois contraintes d'utilisabilité : complétude, stabilité, optimisation de l'espace. Un second algorithme permet de sélectionner et d'instancier les modalités, en essayant de donner satisfaction aux utilisateurs concernés.

Le modèle KUP et les algorithmes correspondants ont été implémentés dans la plate-forme PRIAM (PRésentation des Informations dans l'AMbiant), qui a permis de réaliser des évaluations en laboratoire. Celles-ci ont montré que des systèmes d'affichage dynamiques permettent à des utilisateurs de retrouver leurs informations bien plus rapidement que des affichages statiques.

Abstract

This research work takes place in the domain of human-computer interaction, and particularly multimodal interaction and ambient intelligence. It aims at specifying a theoretical model and a platform for the design and implementation of mobile users assistance systems. We introduce the KUP model, in which the system's functional core, the users and the presentation devices (screens, loudspeakers, etc.) are represented by logical entities. This model is original because it imposes no spatial nor temporal coupling between information providing by the functional core to the user entity on the one hand, and information presentation by a suitable device on the other hand. Both phases are opportunistic: they happen fortuitously, as (physical) users move around.

When a user is located at proximity of a number of presentation devices, the system must determine which device and which modality shall be used to convey information. First, an incremental algorithm is responsible for choosing a device while abiding by three usability constraints: completeness, stability and display space optimization. Second, a tree-based algorithm selects and instantiates a modality while satisfying users' preferences.

The KUP model and the algorithms have been implemented in the PRIAM platform (PResentation of Information in AMbient Intelligence), which has enabled us to carry out evaluations in mock-up environments. The evaluations have shown that dynamic display systems enable users to look up their information far more quickly than static displays.