Conception et modelisation logicielles des systemes interactifs : application aux interfaces multimodales

Laurence Nigay

THESE

présentée par

Laurence Nigay

pour obtenir le titre de DOCTEUR
de l'Université Joseph Fourier - Grenoble 1
(arrêté ministériel du 5 juillet 1984 et du 30 mars 1992)
spécialité : INFORMATIQUE

♦♦♦♦♦♦♦

Conception et modélisation logicielles des systèmes interactifs :

application aux interfaces multimodales

♦♦♦♦♦♦♦

Date de soutenance : 28 janvier 1994

Composition du jury :

Président :	M. Yves Chiaramella
Rapporteurs :	M. Michel Beaudouin-Lafon M. Marc Nanard
	M. Jean Caelen Mme. Joëlle Coutaz M. Michael Harrison

THÈSE PRÉPARÉE
AU SEIN DU LABORATOIRE DE GÉNIE INFORMATIQUE - IMAG
A L'UNIVERSITÉ JOSEPH FOURIER - GRENOBLE 1

Au souvenir de mon père,

Remerciements

Cette thèse revient à mes parents.

Je tiens aussi à remercier :

Joëlle Coutaz, Professeur à l'Université Joseph Fourier, qui m'a donné mon premier emploi après mon retour d'Australie, qui m'a accueillie dans son équipe, qui m'a encadrée tout au long de ce travail et qui m'a fait découvrir un domaine de recherche si passionnant ! Pour tout cela je la remercie très vivement et sûrement plus !!!

Yves Chiaramella, Professeur à l'Université Joseph Fourier et directeur du Laboratoire de Génie Informatique pour son intérêt, ses remarques, et pour avoir accepté de présider le jury,

Michel Beaudouin-Lafon, Professeur à l'Université de Paris XI, et Marc Nanard, Professeur au Conservatoire des arts et métiers Paris, qui ont bien voulu être rapporteurs de ce document. Je les remercie très sincèrement pour leur intérêt, leur enthousiasme et leurs critiques constructives qui ont permis d'améliorer ce travail.

Jean Caelen, Directeur de Recherche CNRS à l'Institut de la Communication Parlée, qui a contribué à ce travail et a accepté d'être membre du jury,

Michael Harrison, Professeur à l'Université de York, qui a bien voulu être membre de ce jury et qui par sa lecture attentive a apporté beaucoup à ce travail. Partenaire du projet européen Esprit BRA Amodeus depuis 1989, je le remercie très vivement.

Alexander Rudnicky, "senior scientist" à Carnegie Mellon University à Pittsburgh (E.U.), pour m'avoir accueillie dans son équipe à deux reprises et pour être à l'origine du système MATIS,

Tous les membres du projet européen Esprit BRA Amodeus avec qui j'ai un très grand plaisir de travailler,

Les membres du groupe IFIP qui m'ont fait l'honneur de m'accueillir dans leur groupe si sympathique,

George pour son enthousiasme communicatif au travail, pour les échanges d'idées à la pause, pour sa précieuse aide dans ce travail,

Tous les membres ou ex-membres de cette équipe Interface Homme-Machine qui ont contribué à créer ce cadre de travail si spécial, si dynamique et si amical ! Sans ordre particulier : Annie, Bérangère, Gilles, Arno, Sandrine, Nathalie, Christine, Véronique, Francis, Sébastien, les générations d'étudiants en DESS ou ENSIMAG, les étudiants qui restent comme François, Jean-Pascal,

Un remerciement spécial à Daniel qui partage son bureau avec moi et qui supporte "The River" un million de fois par jour,

Tous les membres du laboratoire pour l'ambiance de travail, pour leur soutien amical,

L'ensemble des personnes qui, par leur patience, leur encouragement et leur aide, ont contribué à l'aboutissement de ce document :

En tout premier, Pascal pour les heures perdues à m'attendre, pour ses encouragements, pour sa compréhension et son support permanent,

Annie, pour m'avoir aidé à préparer la soutenance, pour y avoir participé, et pour, fait extraordinaire, être en France à chaque événement universitaire important,

Nathalie pour être revenue à temps en France et pour la carte postale du "caillou" pendant la rédaction de ce document, Paul, Nadine, Bertrand, Marie-Luce, Stéphane, Philippe (qui a un drôle d'accent), Bala, Mumu, Philippe (qui à tour de rôle et sans préavis peut être un agent PAC, une photocopieuse etc.), Claude, Eric, Marie, Valérie, Liliane, Nicole, Véronique, Vonvon et Odile, Hélène, Hervé ...

Les amis de Pittsburgh, Philippe, Alain, David, Jean-Christophe,

Et surtout toute la grande famille,

mon oncle, mon parrain Fred et ma marraine Odette qui ont eu la patience de m'écouter, les frères et belles-sœurs, Henri, François, Pierre (appelé aussi Pi.....), Sylviane ou Simone, Françoise,

et sans oublier les plus joyeux et actifs : Pierre, Sophie et Marie.

Résumé

Cette thèse s'inscrit dans le domaine de l'ingénierie des Interfaces Homme-Machine. Elle a pour thème la conception et la réalisation des systèmes interactifs. Parmi ces systèmes, nous étudions les interfaces utilisateur qui intègrent les aspects innovants de la technologie actuelle en communication homme-machine : les systèmes à caractéristiques multiples.

L'approche adoptée s'appuie sur une analyse conceptuelle des systèmes à caractéristiques multiples en prenant en compte à la fois les activités du système et de l'utilisateur. Cette étude nous conduit à distinguer des étapes dans les mécanismes d'abstraction et de concrétisation des informations échangées entre l'utilisateur et le système que nous organisons en un canevas : l'espace Pipe-lines. De l'espace Pipe-lines, canevas intégrateur des activités de l'utilisateur et du système, nous dérivons une méthode de raisonnement et de classification des systèmes en fonction de leurs utilisations : la méthode UOM.

En adoptant comme fondement l'espace Pipe-lines, nous identifions les besoins pour la réalisation logicielle. Nous proposons un modèle d'architecture logicielle, PAC-Amodeus qui répond à ces besoins. Ce modèle est complété d'un guide d'application constitué d'un ensemble de règles heuristiques que nous avons concrétisées par un système expert PAC-Expert. En symbiose avec PAC-Amodeus, nous proposons un moteur de fusion des informations spécifiées par l'utilisateur. De part ses critères et sa technique de représentation des informations, le moteur est générique. PAC-Amodeus corrélé au moteur de fusion constitue une plate-forme pour la réalisation logicielle de systèmes à caractéristiques multiples.

Mots-clefs : Système interactif, Interface homme-machine, noyau fonctionnel, espace de conception, espace de classification, modèle d'architecture logicielle, squelette d'application, langage d'interaction, dispositif physique, fusion, fission.

Abstract

This thesis focuses on the software engineering design and implementation of Human-Computer Interfaces. We concentrate on multifeature systems which integrate new techniques as multiple communication means.

We examine user, as well as system activities. We therefore assign various steps in the abstraction and concretisation mechanisms for information exchange between the user and the system. The Pipe-lines design space organises these steps within a framework that integrates both user and system activities. On the basis of Pipe-lines we develop a classification scheme which helps evaluating the usability of such systems: The UOM scheme (Usage, Option and Multiplicity).

A software engineering perspective based on Pipe-lines is then adopted: To do so we identified software requirements from Pipe-lines for defining the PAC-Amodeus software architecture model. The exploitation of PAC-Amodeus urged us to develop a methodology expressed as a set of heuristic rules that helps defining the software agents. These rules led to the development of a software engineering tool - an expert system called PAC-Expert. In relation to PAC-Amodeus, a mechanism is provided that combines information specified by the user through different input devices and interaction languages. The fusion mechanism is domain-independent and forms the structural skeleton upon which multifeature systems can be built.

Key-words : Interactive system, human-computer interaction, functional core, design space, classification space, software architecture model, application framework, interaction language, I/O device, fusion, fission.

Plan de la thèse

Préambule

Chapitre I **Introduction générale**

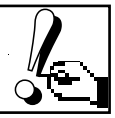
PARTIE 1 ESPACE PROBLEME

Chapitre II Espaces de conception

Chapitre III Le modèle Pipe-lines

Chapitre IV U.O.M., une méthode de classification des systèmes interactifs

PARTIE 2 ESPACE SOLUTION



Chapitre V Interface Homme-Machine :
cycle de développement et outils

Chapitre VI PAC-Amodeus : notre modèle de référence

Chapitre VII PAC-Amodeus :
intégration des langages et des dispositifs

Chapitre VIII Du modèle à la réalité :
les systèmes NoteBook et MATIS

Chapitre IX **Conclusion générale**

Annexes

Annexe A : Scénario d'utilisation de MATIS

Annexe B : Spécification de MATIS en langage UAN

Annexe C : Langage naturel dans le contexte de NoteBook

Annexe D : Langage naturel dans le contexte de MATIS

Références bibliographiques

Glossaire

Table des figures

Table des matières

Préambule

En guise d'introduction avant d'exposer nos travaux, nous souhaitons citer un paragraphe d'un article intitulé "Au lieu de l'enrichir l'informatisation appauvrit l'humanité" :

"..., une machine à écrire dotée d'un système sophistiqué de reconnaissance de la parole constituera probablement la première application de cette nouvelle technologie. Quelle chance pour la secrétaire, qui pourra ainsi se consacrer à d'autres tâches ! La libération des tâches routinières sera-t-elle vraiment source de créativité ?, demande Elektronik, qui constate par ailleurs que le comportement des êtres humains pendant leurs loisirs se distingue par une passivité croissante.
Les systèmes informatiques destinés à des utilisateurs faiblement qualifiés présentent de plus en plus, outre une convivialité poussée, des fonctions intelligentes de support à la décision. Mais, avec ce transfert de la compétence de décision, l'homme perdra aussi la conscience de sa responsabilité. Les ordinateurs seraient bien mieux employés en tant qu'outils "aidant à penser", et il vaudrait mieux pousser vers une informatisation plus humaine, plutôt que d'accélérer continuellement les développements technologiques. "

(découvert à la cafétéria de l'IMAG)

Chapitre I

Introduction

"Rabelais et Cervantès fondèrent sur l'invention de l'imprimerie, il y a cinq siècles, le grand élan de la Renaissance : ils ébranlèrent l'ordre moral. Sous les bouleversements des techniques, la société du Moyen Age vit se lézarder ses structures sociales et ses schémas mentaux. Ce choc devait engendrer la Renaissance... Il fallut alors près d'un siècle pour que la réflexion forge une nouvelle vision du monde... Un siècle de l'ère de l'imprimerie c'est, sans doute, dix ans à celle des ordinateurs."

- Jean-Jacques Servan-Schreiber -

Le sujet

Nos travaux de recherche ont trait à la conception et à la modélisation logicielles des systèmes interactifs. Parmi ces systèmes, nous étudions les interfaces utilisateur qui intègrent les aspects innovants de la technologie actuelle en communication homme-machine. La figure 1.1 rappelle, s'il en était besoin, le rôle d'une interface dans un système informatique : un intermédiaire matériel et logiciel entre l'utilisateur et le noyau fonctionnel. Nous entendons par noyau fonctionnel, le logiciel modeleur des concepts de la tâche informatisée.

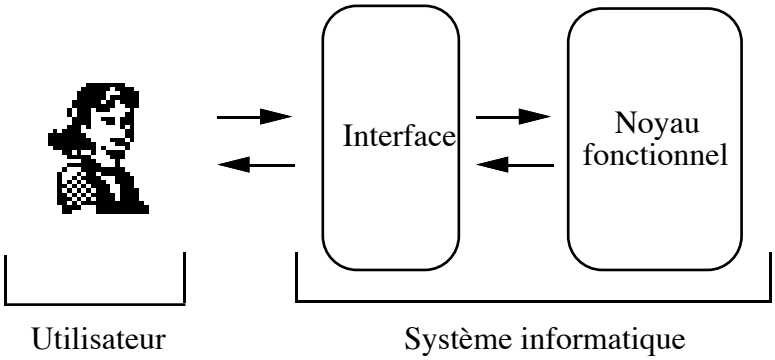


Figure 1.1 : Un système informatique et son utilisateur.

L'innovation dans les interfaces tient à la disponibilité sur les stations de travail usuelles, de mécanismes d'interaction sophistiqués de reconnaissance et de synthèse : parole, vision par ordinateur, génération automatique de textes et d'images réalistes, vidéo, etc. L'innovation tient également à la possibilité pour l'utilisateur et le système de combiner ces différentes formes de techniques en fonction de la tâche, des préférences ou des intentions communicationnelles.

La multiplicité des possibilités peut se voir comme un facteur de souplesse avec, sa compagne immédiate, la complexité. Il convient alors de s'interroger sur l'utilité et l'utilisabilité de ces nouvelles interfaces, deux aspects qui relèvent de la psychologie et de l'ergonomie cognitives. En psychologie, on visera à développer ou à étendre des théories existantes pour étudier, c'est-à-dire prédire ou expliquer, les performances sensori-motrices et cognitives de l'utilisateur mis en situation d'interaction innovante. Les travaux actuels dans ce domaine restent prospectifs [Barnard 93] et font souvent appel à l'approche expérimentale du Magicien d'Oz [Mignot 93, Coutaz 93f].

Bien que la psychologie ou les règles d'usage ergonomiques ne puissent, à l'heure présente, fournir des réponses opérationnelles au cas des nouvelles interfaces, les recommandations générales ou les critères qui nous viennent des interfaces graphiques restent

valides. En tant qu'informaticien, nous devons continuer à les respecter, voire les étendre au cas des nouvelles interfaces.

Nos objectifs se situent dans ce contexte.

Les objectifs

Notre étude sur la conception et la modélisation logicielles des systèmes interactifs vise un double objectif :

1) comprendre et structurer l'espace problème des interfaces nouvelles en des termes qui soient communs aux deux disciplines essentielles de la communication homme-machine : la psychologie cognitive et l'ingénierie des logiciels. Le modèle visé doit permettre aux concepteurs ergonomes et psychologues, tout comme les concepteurs de logiciel, de raisonner sur le domaine des nouvelles interfaces ;

2) fournir un espace solution à la fois conceptuel et opérationnel, en accord avec les besoins de l'espace problème. Bien que nous visions l'intégration des recommandations de l'ergonomie cognitive, le modèle solution s'adressera, pour l'essentiel, aux besoins de l'ingénierie des logiciels. Nous voulons un modèle conceptuel, c'est-à-dire indépendant d'une technologie particulière, mais aussi opérationnel, c'est-à-dire applicable par heuristique au cas de la conception logicielle d'une interface utilisateur donnée.

Ces objectifs se justifient ainsi :

La communication homme-machine est un domaine évolutif tiraillé entre la pratique créative de la technique et l'approche conceptuelle mesurée de la théorie. Pratique et théorie sont les deux approches complémentaires qui servent de piliers aux démarches scientifiques. Nous constatons aujourd'hui une explosion de techniques de communication ouvrant des voies nouvelles dont on ne cerne pas les limites. Prenons pour preuve le flou d'une terminologie mal assise qui fléchit au gré de la mode : interfaces multimédia, interfaces multimodales. Sait-on ce que ces termes recouvrent?

Il convient dans ces conditions d'organiser le savoir expérimental dans des cadres aussi rigoureux que possibles. De là, un espace problème structurant permet de raisonner, comparer, évaluer les interfaces existantes ou à concevoir. Nous estimons que ce premier objectif est un point contributif important au domaine.

Notre deuxième objectif vient enrichir les modèles d'architectures logicielles centrés jusqu'ici sur les interfaces graphiques. Avec l'arrivée sur le marché des boîtes à outils multimédia, ou des systèmes de reconnaissance de la parole et de l'écriture, nous nous devons de les situer dans les architectures traditionnelles. La disponibilité simultanée et l'usage combiné de ces nouvelles techniques accroissent nécessairement la complexité de la mise en œuvre. Si les générateurs d'interface évolués dispensent parfois le concepteur de définir l'architecture globale de l'interface, il n'échappe pas à une telle spécification dans le cas des interfaces nouvelles. Il lui faut alors un guide général mais facilement transférable au cas à traiter. Ce deuxième objectif est, à notre sens, une contribution utile à l'ingénierie des logiciels.

Organisation du mémoire

La décomposition en deux parties de ce mémoire reflète notre démarche de recherche : espace problème et espace solution.

La première partie définit l'espace problème. Elle comprend trois chapitres.

Le Chapitre II fait le point sur la terminologie et sur les référentiels applicables à la conception d'interfaces avancées. Nous constatons que la terminologie est confuse et dépendante des objectifs. Nous convenons d'éliminer les termes multimédia et multimodal et de parler de systèmes à caractéristiques multiples. Les référentiels quant à eux, sont soit limités par leur portée (entrées ou sorties, par exemple), soit généraux (couverture à la fois des entrées et des sorties) mais bornés par un point de vue restrictif (technologie versus utilisateur). Ce bilan nous conduira à combler les insuffisances de la situation avec le modèle Pipe-lines.

Le Chapitre III fournit une description détaillée du modèle Pipe-lines. Ce modèle explicite les traitements effectués par l'utilisateur et par le système à partir de leurs points de contact : les langages d'interaction et les dispositifs physiques d'entrée/sortie. Il permet donc de raisonner sur un système selon les deux points de vue complémentaires de la conception des interfaces : utilisation et réalisation.

Au chapitre IV, nous proposons UOM, une méthode de classification des systèmes qui s'appuie sur les notions de langages et de dispositifs du modèle Pipe-lines. UOM se décompose en trois volets selon que l'on considère l'Usage, le caractère Optionnel ou la Multiplicité des langages d'interaction et des dispositifs physiques d'un système. L'étude de la multiplicité des langages et des dispositifs conduit à une classification intemporelle, statique. L'analyse des options et de l'usage offre une perspective dynamique instantanée. L'étude des options met l'accent sur les possibilités de choix de langage et de dispositif à un instant donné ; l'usage caractérise les combinaisons possibles de ces choix. UOM perpétue la complémentarité

système/utilisateur : la capacité de choix concerne aussi bien l'utilisateur que le système. Nous illustrons l'utilisation d'UOM au moyen de systèmes à caractéristiques multiples réels.

Après avoir identifié les propriétés des systèmes en termes de langages et de dispositifs, **la deuxième partie de ce mémoire définit l'espace solution**. Il s'agit là de solution à la conception logicielle d'interfaces à caractéristiques multiples. L'exposé de l'espace solution comprend quatre chapitres.

Le Chapitre V explore le support pratique de l'activité de réalisation des interfaces par une revue des outils logiciels d'aide au développement. L'objectif est de situer, dans cet ensemble, nos travaux sur les architectures logicielles et de montrer leur utilité et usage dans le processus de développement d'un système.

Le Chapitre VI présente PAC-Amodeus, notre contribution à la modélisation des architectures logicielles. PAC-Amodeus, qui se veut à la fois pratique et conceptuel, préconise la décomposition fonctionnelle d'ARCH et intègre les capacités d'affinement et de prise en compte du parallélisme de PAC. PAC-Amodeus définit des niveaux d'abstraction, indique le principe du support des dialogues à plusieurs fils, montre les points d'ancrage avec les boîtes à outils et le noyau fonctionnel. Mais, parce qu'il est conceptuel, PAC-Amodeus fournit peu d'indication sur la nature et le rôle des agents du Contrôleur de Dialogue qu'il convient de définir pour chaque cas de système. Afin de rendre PAC-Amodeus accessible, nous fournissons un ensemble de règles heuristiques d'aide au processus d'identification de ces agents. Ces règles constituent le cœur de PAC-Expert, un système expert d'aide à la conception d'architecture globale.

Avec le chapitre VII, nous voulons montrer comment l'usage combiné des langages et des dispositifs propre aux systèmes à caractéristiques multiples s'inscrit dans une architecture logicielle et comment PAC-Amodeus en permet et facilite la prise en compte. L'un des phénomènes caractérisant de cet usage est la fusion d'information. Nous décrivons en détail notre moteur de fusion et précisons son lieu d'intégration dans PAC-Amodeus. Nous identifions trois catégories de fusion puis proposons les stratégies, les critères et les représentations de données sous-jacentes au processus de fusion. Nous étayons la discussion par des systèmes existants.

Le Chapitre VIII présente NoteBook et MATIS, deux systèmes à caractéristiques multiples. Ces systèmes sont analysés selon les volets de la méthode UOM. Leur architecture illustre de manière concrète la validité de PAC-Amodeus.

En conclusion, nous soulignons les points contributifs de nos travaux et nous développons les perspectives de travaux futurs.

Quatre annexes sont fournies en fin du mémoire. L'Annexe A présente un scénario d'utilisation de MATIS ; l'Annexe B contient une partie de la spécification formelle de MATIS avec le langage UAN (User Action Notation) ; l'Annexe C et l'Annexe D contiennent une partie de la description du langage naturel dans le contexte des systèmes NoteBook et MATIS.

Un glossaire des termes importants est ensuite fourni.

Nous avons réalisé deux systèmes à caractéristiques multiples qui nous servent d'exemples dans l'exposé de nos travaux. On trouvera ci-dessous une description rapide de l'interface utilisateur de ces deux systèmes.

NoteBook et MATIS, deux exemples de référence

NoteBook et MATIS ont été développés sur machine NeXT en utilisant Interface Builder [Webster 89] pour la partie graphique et Sphinx [Lunati 91], un système de reconnaissance de la parole continue et multilocuteur. Sphinx a été développé à Carnegie Mellon University par l'équipe d'Alexander Rudnicky que nous remercions pour son accueil et son aide.

Ces deux systèmes offrent une interface graphique accessible via la souris qui représente l'univers de l'action par une métaphore du monde réel. Ils permettent de communiquer :

- en langue naturelle écrite ou orale en utilisant respectivement le clavier ou le microphone,
- en langage de commande écrit en saisissant la commande au clavier.

NoteBook [Nigay 91a] est un bloc-notes électronique qui permet de créer, d'éditer, de feuilleter et de détruire des notes de texte. La figure 1.2 présente la fenêtre principale. Pour toutes les commandes, l'utilisateur a le choix du langage et du dispositif. Par exemple, la destruction d'une note dans le bloc-notes peut être spécifiée :

- en sélectionnant le bouton "Remove note" avec la souris (manipulation directe),
- en parlant en langue naturelle : "*Remove the current note*",
- en saisissant la phrase au clavier : "Remove the current note".

NoteBook ne privilégie aucun langage d'interaction : le choix est laissé à l'utilisateur. Une exception à cette règle concerne le contenu d'une note. Celui-ci doit être saisi au clavier afin de limiter la taille du dictionnaire géré par Sphinx. D'autre part, NoteBook offre la possibilité de

combiner la parole avec des gestes de désignation spécifiés avec la souris. Ainsi l'insertion d'une note peut se faire :

- en énonçant la phrase "*Insert a note between this note and this one*",
- tout en sélectionnant avec la souris deux notes (dans la partie "title" de la fenêtre présentée à la figure 1.2).

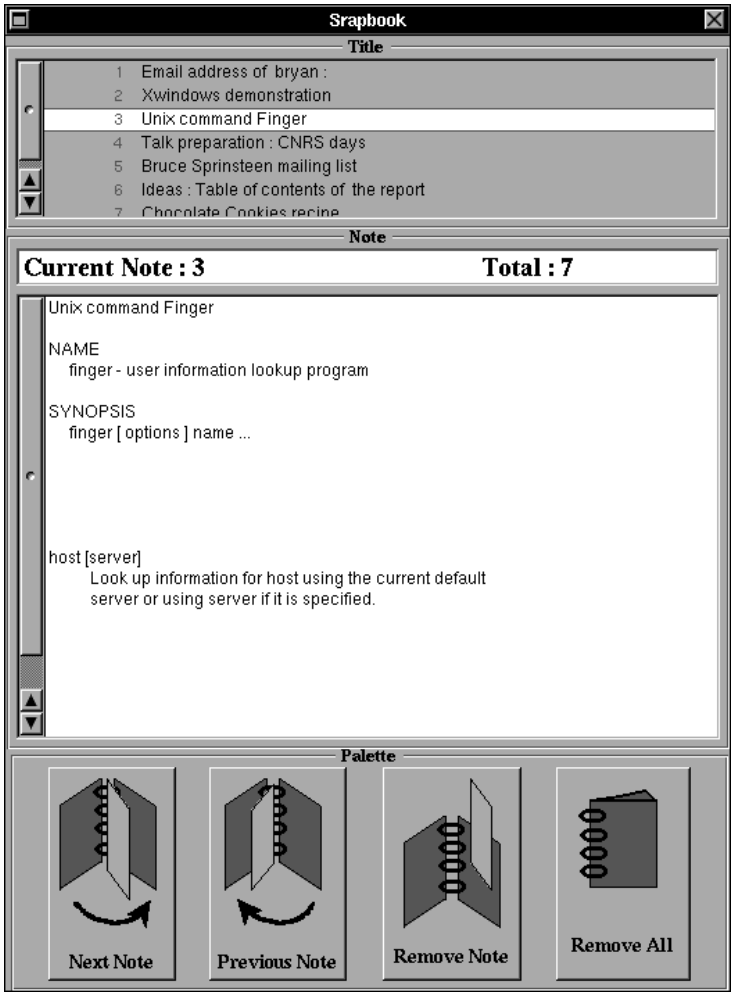


Figure 1.2 : La fenêtre principale de NoteBook.

NoteBook offre un dialogue orienté vers la manipulation d'objets, ces objets étant les notes d'un cahier. La langue naturelle permet de créer et de manipuler ces objets en les nommant.

Au contraire notre deuxième réalisation, MATIS [Nigay 93c] (Multimodal Airline Travel Information System) est un système d'information sur les transports aériens qui offre un dialogue finalisé restreint à un domaine sémantique délimité pour lequel l'utilisateur utilise la langue naturelle restreinte : le domaine couvre toutes les informations sur des transports aériens. Ce dialogue est de type question/réponse simple [Bourguet 92] où les demandes et les

informations transmises sur un vol sont exprimées en langue naturelle mais où l'enchaînement des requêtes et de négociation n'existent pas. MATIS fournit, en réponse à des requêtes de l'utilisateur, des informations sur les vols entre deux villes. La base de données sous-jacente contient actuellement neuf villes américaines, neuf compagnies aériennes et des informations sur chaque vol telles que le numéro du vol, les repas servis à bord etc. La figure 1.3 présente une copie d'écran de l'interface graphique de MATIS .

• La fenêtre "Office Manager" (en haut et au milieu de l'écran) met en évidence les systèmes connectées (c.-à-d., MATIS) au système de reconnaissance de la parole. Cette fenêtre affiche aussi la dernière phrase reconnue par le système de reconnaissance de la parole, ou les messages d'erreur en cas d'échec de la reconnaissance. C'est aussi la zone d'édition des phrases écrites en langage naturel. A la figure 1.3, la dernière phrase reconnue est “USAIR FLIGHT FROM PITTSBURGH TO BOSTON SERVING A MEAL”.

• Le système offre un ensemble d'outils ("Requests Tools") pour la spécification d'une requête par manipulation directe, tel que la liste des villes. L'usage de MATIS a montré que ces outils étaient aussi utilisés comme compléments à la parole, l'utilisateur ne sachant pas quelles étaient les informations disponibles dans la base de données.

• Des outils d'utilité générale sont aussi fournis : un bloc-notes pour rassembler les informations intéressantes sur les vols, un historique des requêtes qui permet en particulier de réitérer une requête via les opérations Copier/Coller.

• La fenêtre "Results of request", au centre de l'écran, affiche sous forme de table le résultat d'une requête soumise à la base de données. Les informations contenues dans la table peuvent être sélectionnées et copiées dans une autre fenêtre, comme celle du bloc-notes.

• La fenêtre "Request" contient la requête en cours de spécification. Comme les champs d'une requête sont nombreux, nous les avons divisés en deux catégories selon leurs fréquences d'utilisation. Les champs obligatoires ou très utilisés sont affichés par défaut à l'ouverture d'une nouvelle requête (c.-à-d., From, To, Dep. Time, Arr. Time). L'utilisateur peut ajouter d'autres champs dans le formulaire par manipulation directe en utilisant les outils de la fenêtre "Requests Tools" ou en les spécifiant en langage naturel (écrit ou oral). Ainsi, comme le montre la figure 1.3, l'utilisateur a énoncé "USAIR FLIGHT FROM PITTSBURGH TO BOSTON

SERVING A MEAL". La phrase reconnue a provoqué l'ajout automatique de deux champs intitulés "Airline" et "Meal".

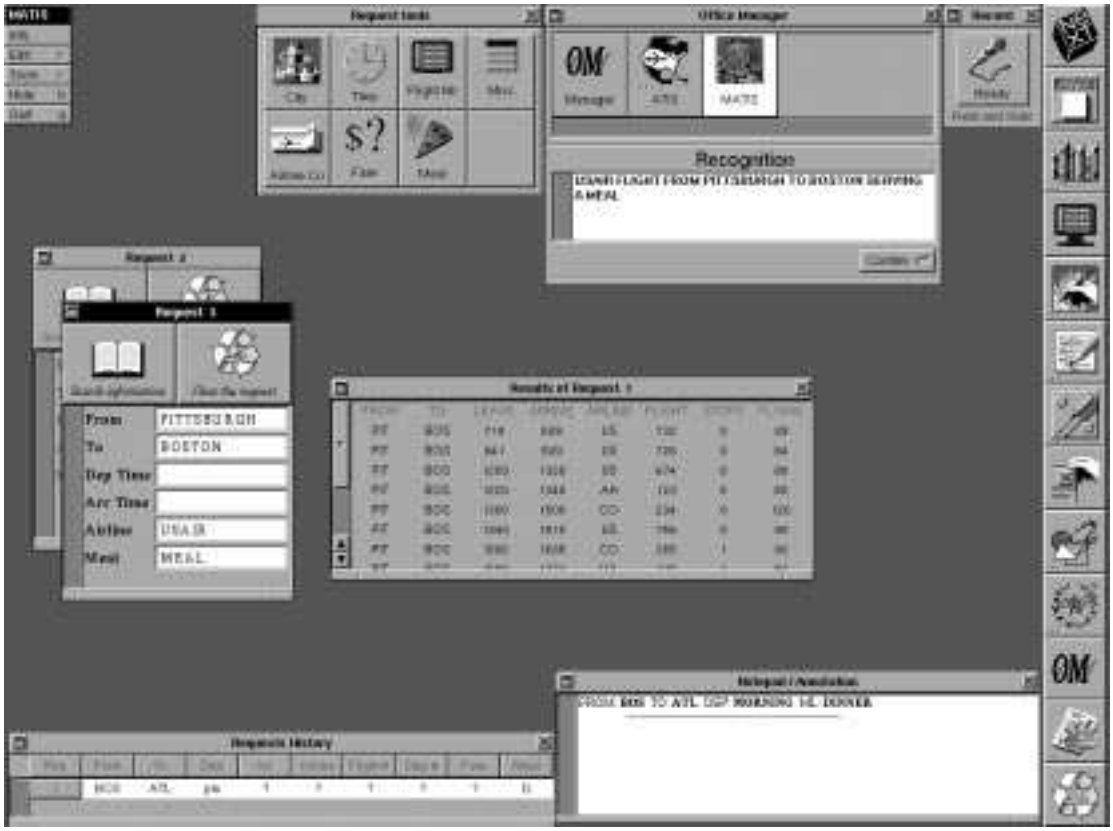


Figure 1.3 : Copie d'écran du système MATIS.

Une requête vide est affichée par défaut au lancement de MATIS. MATIS offre un dialogue à plusieurs fils d'activité : il est possible de commencer une nouvelle requête sans achever la requête courante. Cette possibilité est illustrée à la figure 1.3 où deux requêtes sont en cours de spécification. Pour rendre active une requête, il suffit de sélectionner la fenêtre correspondante. Celle-ci est alors affichée au premier plan et définit le nouveau contexte de l'interaction.

L'utilisateur exprime des requêtes au système afin de planifier son voyage. Pour cela :

• L'utilisateur peut utiliser la souris et spécifier sa requête par manipulation directe.

• L'utilisateur peut utiliser le clavier et spécifier sa requête en langue naturelle ou en remplissant un formulaire.

• L'utilisateur peut utiliser la parole et dicter sa requête. Par exemple : "*I would like to know the Usair flights from Pittsburgh to Boston*".

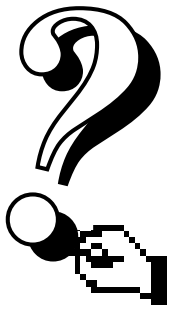
• L'utilisateur peut utiliser les trois formes précédentes de communication en les combinant. Par exemple :

<Parole : "*I would like to know the Usair flights from this city*">
+ <Sélection avec la souris d'une ville>
+ <Parole : "*arriving at*">
+ <Saisie d'un horaire en utilisant le clavier>

Ces deux réalisations, fondées sur des dialogues de types différents, ont permis une étude sur l'intégration des modalités (mécanisme de fusion des événements, références multimodales) ainsi que sur l'architecture logicielle de tels systèmes.

PARTIE 1

ESPACE PROBLÈME



"Il n'y a rien de simple dans la nature, il n'y a que du simplifié"
- Gaston Bachelard -

Chapitre II

Espaces de conception

"*Technology is not a substitute for human values, but the servant for human values.*"
- William Paley - Founder of CBS

1. INTRODUCTION : UN TERRAIN VAGUE

L'explosion de la technologie communicante et l'absence de théorie prédictive sur son utilité nécessitent de "faire une pause et reprendre sa respiration" [Takebayashi 92]. Cette avancée brutale se manifeste notamment par une prolifération de termes employés le plus souvent sans rigueur, en général les uns pour les autres.

Le message formulé à l'occasion d'un colloque est symptomatique du malaise actuel d'une communauté contrainte par l'absence d'un référentiel commun : "J'ai le perpétuel sentiment que nous ne communiquons pas effectivement simplement parce que nous n'avons pas de vocabulaire commun consensuel. Prenez le titre de ce workshop, par exemple. Combien de participants ou participants potentiels comprennent-ils la subtile distinction, s'il y en a une, entre multimédia et multimodal?[1]" (Edwards 90, CHI'90 workshop on Multi-media and Multi-modal Interface Design [Blattner 90]).

Ce chapitre vise à répondre à l'appel d'Edwards en faisant le point sur la terminologie et sur les référentiels applicables à la conception d'interfaces avancées. Un bilan analytique nous conduira à combler les insuffisances de la situation par le modèle Pipe-lines que nous présentons au chapitre III.

2. TERMINOLOGIE

Parce que les acteurs de la communication, l'homme et la machine, posent des problèmes différents, les notions de multimédia et de multimodalité sont appréhendées selon des perspectives distinctes, l'une centrée sur l'individu, l'autre sur la technologie. L'ambiguité des expressions repose pour l'essentiel sur cette dualité. La revue qui suit se devra de préciser l'angle adopté.

Toutefois, la diversité des points de vue ne justifie que partiellement l'absence de consensus. En vérité, et notre analyse le démontre, les désaccords qui se manifestent au sein d'une même perspective traduisent une mauvaise maîtrise des concepts. Notre première

---

1 Dans le texte anglais original, "I have a constant nagging feeling that we are not communicating effectively, simply because we have no common, agreed vocabulary. Take the title of this workshop for instance. How many participants and would-be participants understand the subtle distinction between multimedia and multimodal, if there is one?"

contribution sera de choisir et proposer un ensemble de termes qui couvrent raisonnablement notre domaine de réflexion.

Avant de présenter les taxonomies relevées dans la littérature, il convient de considérer les définitions du sens commun.

2.1. Terminologie et le sens commun

Sur le plan linguistique, les termes “multimédia” et “multimodalité” ont en commun le préfixe “multi” qui traduit le nombre. Les radicaux “média” et “modalité” les distinguent. Si l’on se réfère aux dictionnaires de langue française [Hachette 1991, Larousse 1986] :

> • un *média* est un “procédé technique permettant la distribution, la diffusion ou la communication des œuvres de l’esprit écrites, sonores, ou visuelles (depuis la presse imprimée jusqu’à l’ordinateur ....)”. Appliquée à notre domaine, cette définition couvre le système informatique dans son tout. La tâche de conception d’un tel système nécessite une analyse plus fine de la notion de média.
>
> • la *modalité* a plusieurs acceptions selon le domaine de référence. Nous retenons deux domaines en relation directe avec la communication homme-machine : la psychologie et la linguistique. En psychologie, une modalité désigne l’une des “grandes catégories de qualités sensorielles (vision, audition, olfaction, etc.)”. En linguistique, la modalité permet “d’exprimer les diverses manières dont le contenu est envisagé” par le locuteur. Ces manières se manifestent par des marqueurs linguistiques (mode grammatical, adverbe, interjection) auxquels il faut ajouter le rôle important de l’intonation de la voix ou l’attitude corporelle. Ainsi pour un même contenu (neige-tomber), les expressions “il faut que la neige tombe”, “Ah, si la neige tombait!”, “la neige tombera”, traduisent, par leurs modalités, des engagements différents du locuteur vis-à-vis du contenu informationnel. Nous constatons que ces deux définitions, les catégories sensorielles et les marqueurs de statut linguistiques, sont deux ingrédients valides de la communication homme-machine. Il faudra être capable de les distinguer sans confusion ou de les encapsuler en une définition commune.

Corrélé à *modalité*, nous trouvons le *mode*. En philosophie, le mode est la manière d’être d’une substance (un objet, un être, une entité) et la modalité est la propriété que possède la substance d’avoir des modes. En limitant la discussion à l’action, l’un des constituants essentiels de la communication homme-machine, le mode est la manière dont se fait l’action et la

modalité en désigne la forme. Notons que la manière de faire influence la forme et que la forme permet de déterminer la manière! Il n'est donc pas surprenant que les termes mode et modalité soient l'un et l'autre employés pour qualifier une action. Enfin, l'adjectif modal relatif à la fois aux substantifs mode et modalité souligne l'ambiguïté du terme multimodal : lorsque nous caractérisons une interface de multimodale, s'agit-il de multi-*modalité* ou de multi-*mode*?

Ayant rappelé les définitions de référence académique, nous sommes en mesure de les infléchir au cas de la communication homme-machine.

2.2. Terminologie et communication homme-machine

Une revue de la littérature nous a permis de relever trois groupes de termes directeurs : média, modalité, multimédia et multimodal. Dans chacun de ces groupes, les auteurs se démarquent par le point de vue adopté (utilisateur versus technologie) et le niveau d'abstraction. Si possible, nous expliciterons ces différences de perspective.

2.2.1. Média

Une majorité des auteurs s'accordent à penser qu'un *média* est un support technique de l'information. Exprimée ainsi, cette définition est cohérente avec l'acception du sens commun. Mais chaque auteur augmente ce pivot commun de précisions personnelles qui montrent la diversité des points de vue.

Pour certains, un média est un dispositif physique capteur ou effecteur d'un système informatique. Une souris est un capteur, l'écran est un effecteur. Ainsi défini, un média est une spécialisation de la classe "support technique". Mais pour Blattner, un média n'a pas nécessairement une incarnation physique [Blattner 90]. En tant que véhicule d'information, il peut se voir réalisé par du logiciel. Dans ces conditions, un message électronique est un média.

On observe ici une différence de point de vue qui tient à la différence des niveaux d'abstraction. Les uns s'en tiennent au support matériel, les autres voient dans le média toute forme logique capable de véhiculer de l'information. Nous verrons en 4.2 comment la théorie des dispositifs de Mackinlay, Card et Robertson concilie ces deux points de vue. Si elles diffèrent par le degré de conceptualisation, toutes ces définitions relèvent du même parti : celui de la technologie.

Au point de vue technique on oppose des définitions qui reposent sur les capacités de perception ou de compréhension de l'homme. Alty, par exemple, voit dans "média" un système

représentationnel [Alty 91, Frohlich 91]. Un texte en langage naturel, un graphe sont des systèmes représentationnels. Arens dans son système de présentation automatique d'information utilise également média dans ce sens [Arens 93].

D'autres définitions lient le média, non pas comme ci-dessus aux capacités d'interprétation de l'homme, mais à ses capacités sensorielles. Ainsi, pour Bernsen un média est un ensemble de qualités perceptives couplé à l'équipement sensoriel humain nécessaire à la perception de ces qualités [Bernsen 93]. Par exemple les qualités visuelle et graphique nécessitent la vision, les qualités du son, l'audition. Ce couplage définit un média.

M.L. Bourguet adopte un point vue technique mais néanmoins proche de cette idée de perception : le média est un dispositif physique auquel il convient d'associer une substance ou matériau non analysé [Bourguet 92]. Ici, "dispositif physique" sert à traduire la capacité matérielle à véhiculer de l'information et la "substance" (au sens de Hjemslev) sert à caractériser l'information du point de vue sensoriel. En ce sens, le couple <dispositif = clavier ou crayon, substance = écrit>) caractérise un média.

2.2.2. Modalité

Comme pour "média", la plupart des auteurs partent d'un pivot commun selon lequel une modalité, au sens établi en psychologie, désigne une catégorie de qualités sensorielles. Mais comme pour "média", ils divergent en fonction des objectifs.

Les concepteurs d'interfaces (ergonomes et psychologues) associent la modalité à la capacité de compréhension du sujet humain. Ceux-ci, comme Bernsen dans sa théorie des modalités pures (voir paragraphe 5), y voient un système représentationnel de l'information. Modalité représentationnelle et média au sens d'Alty et d'Arens recouvrent le même concept. Pour les concepteurs de logiciel, la modalité est liée au contenu, à la nature des informations que le système véhicule [Blattner 90, Bourguet 92].

2.2.3. Multimédia et multimodal

La diversité des définitions des termes média et modalité voire leur recouvrement, laisse entrevoir la difficulté d'identifier une définition consensuelle pour multimédia et multimodal. Nous retiendrons que les objectifs justifient les différences. A l'évidence, ces termes sont tiraillés entre d'une part, les préoccupations des psychologues et des ergonomes qui visent à développer des théories explicatives ou prédictives sur les interfaces nouvelles, et d'autres part, les considérations des informaticiens qui visent à définir des modèles d'architecture logicielle et

des outils pour la réalisation de telles interfaces. Nous tentons ici une présentation synthétique de la littérature.

Pour les psychologues et les ergonomes, la multimodalité d'un système tient au fait que la machine sollicite les capacités multi-sensorielles de l'utilisateur. Multimédia couvre la diversité des supports techniques de ce système. Pour certains informaticiens, multimodal est lié à la capacité du système à extraire du sens des informations échangées tandis que multimédia dénote l'incapacité à le faire. Nous verrons avec l'espace MSM présenté en 7, une définition plus précise de cette distinction. Pour d'autres informaticiens, un système multimédia se caractérise en entrée, par l'utilisation simultanée de différents dispositifs physiques, par la gestion en parallèle des événements qui en découlent, et par la construction d'événements de haut niveau d'abstraction à partir d'événements de plus bas niveau. Il se caractérise en sortie, par différents canaux en parallèle (texte, vidéo etc.) [Duce 90]. Pour d'autres encore, tel Buxton [Buxton 90], multimédia ou multimodal impliquent : plusieurs modalités sensorielles, plusieurs canaux appartenant à une ou plusieurs modalités (par exemple les deux oreilles sont deux canaux appartenant à la même modalités l'ouïe), plusieurs tâches en parallèle, plusieurs utilisateurs effectuant la même tâche en parallèle!

Nous venons de proposer une revue de la terminologie de notre sujet d'étude. Il convient maintenant d'en relier les termes en un espace problème utile à la conception d'interfaces homme-machine. Dans cet objectif, nous analysons les propositions de la littérature en adoptant une démarche que nous précisons dans la section qui suit.

3. NOTRE DÉMARCHE D'ANALYSE : INTERFACE D'ENTRÉE ET INTERFACE DE SORTIE

La plupart des espaces de conception que nous allons étudier dissocient l'interface d'entrée de l'interface de sortie. D'autres ne considèrent qu'une seule des deux dimensions. Si l'on adopte le parti de la technologie, ces notions se définissent ainsi :

- L'interface d'entrée est constituée des moyens de communication artificiels mis à la disposition de l'utilisateur pour modifier l'état du système.

- L'interface de sortie couvre les moyens de communication qui rendent l'état du système perceptible à l'utilisateur.

Pour Card, ces deux interfaces, qui permettent un échange bidirectionnel d'information, sont les ingrédients incontournables de l'interaction. L'information spécifiée par l'utilisateur

traverse l'interface d'entrée en subissant des transformations pour aboutir à une représentation interne au système. En sens inverse, l'information interne traverse l'interface de sortie en subissant des transformations pour aboutir à une représentation perceptible. L'information perçue est alors traitée par l'utilisateur selon un processus similaire d'acquisition-traitement-action que Card, Newell et Moran ont modélisé schématiquement avec leur Modèle du Processeur Humain [Card 83].

La distinction entre interface d'entrée et interface de sortie se retrouve dans la théorie de l'action de Norman [Norman 86]. Cette théorie, qui centre l'analyse sur l'utilisateur, met en évidence les efforts mentaux et physiques que met en jeu un sujet en situation d'interaction avec un système informatique. Comme le montre métaphoriquement la figure 2.1, ces efforts traduisent la distance entre les représentations mentales et compétences physiques du sujet et les représentations et dispositifs d'entrée/sortie du système informatique. Le gouffre de l'exécution, qui désigne la distance mentale que le sujet doit parcourir pour passer des buts à l'exécution d'actions, tient aux propriétés de l'interface d'entrée. Le gouffre de l'évaluation, qui représente le parcours inverse, est lié à l'interface de sortie.

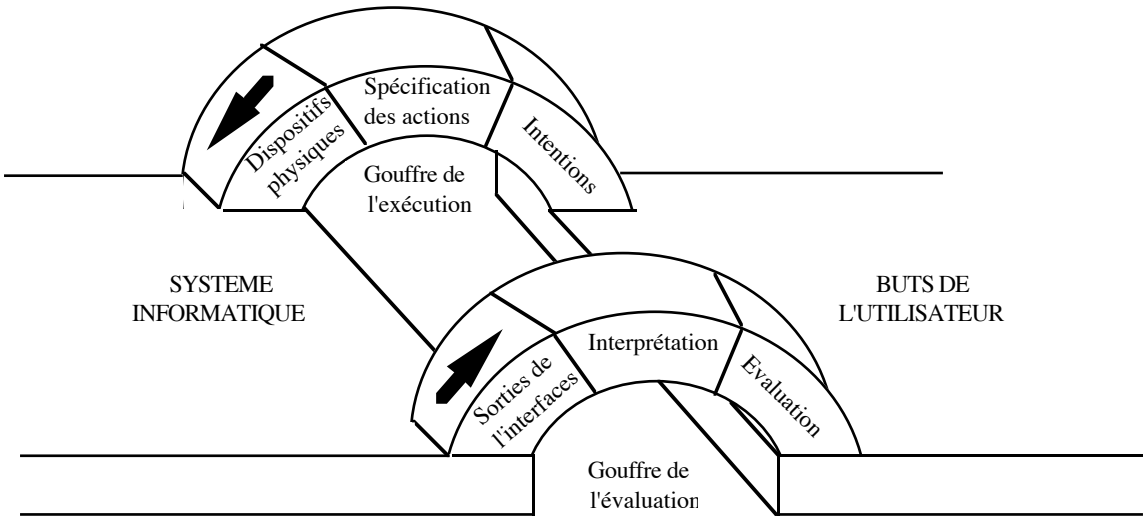


Figure 2.1 : La théorie de l'action de Norman : gouffres de l'exécution et de l'évaluation et interfaces d'entrée et de sortie.

Cette distinction entre interface d'entrée et interface de sortie se conçoit à des fins pédagogiques ou analytiques. Le schéma cyclique :

1) interface d'entrée : acquisition de données,

2) traitement interne : changement d'état,

3) interface de sortie : expression perceptible du changement d'état,

4) retour à l'étape 1,

est une vue réductrice du fonctionnement souhaitable d'un système. Les interfaces d'entrée et de sortie ne peuvent être conçues de manière indépendante car les données reçues par l'une peuvent influencer les résultats de l'autre avant même que la boucle ait été exécutée dans sa totalité. Par exemple, le principe du "retour d'information immédiat" prôné en ergonomie, nécessite souvent l'entrelacement des traitements de l'interface d'entrée avec ceux de l'interface de sortie. De même, le parallélisme ne peut se satisfaire d'un schéma pipe-line séquentiel. Nous reviendrons sur ces problèmes au chapitre suivant. Le cycle ci-dessus a donc valeur pédagogique de cadre général. La même observation s'applique aux modèles de Norman et de Card.

Dans les sections qui suivent, nous présentons les espaces de conception en fonction de leur portée. Ici, la portée se définit par l'interface considérée (d'entrée ou de sortie) et le niveau d'abstraction envisagé (dispositifs physiques ou traitements de haut niveau).

4. LES ESPACES DE CONCEPTION DES INTERFACES D'ENTRÉE

Les premières tentatives de taxonomies concernent les dispositifs physiques d'entrée. Dans cette lignée, nous trouvons les espaces de Buxton [Buxton 83], de Card, Mackinlay et al. [Card 90, Mackinlay 90] et de Foley [Foley 84] qui correspondent chacun à des niveaux d'abstraction ou d'affinement différents.

4.1. La taxonomie des dispositifs physiques selon Buxton

Selon Buxton, un *dispositif d'entrée* est un transducteur de propriétés physiques que l'on peut caractériser dans un espace à trois dimensions. Comme le montre la figure 2.2,

- le premier axe de l'espace de classification dénote les propriétés que le dispositif est capable de capter. Ces propriétés sont au nombre de trois : la position, la pression, le mouvement ;

- le second axe correspond au nombre de dimensions captées pour chaque propriété. Par exemple, pour la souris, la position est déterminée par deux valeurs prises dans un système de coordonnées. Pour le manche à balai, la position est un triplet ;

• le troisième axe distingue le type direct ou indirect de capture. Par exemple, pour un écran tactile ou une caméra, la mesure est directe alors que la souris implique un intermédiaire mécanique.

Property sensed		Number of Dimensions			Sensing type
		1	2	3	
	Position	Rotary Pot Sliding Pot	Tablet Light Pen Joystick Touch Tablet Touch Screen	3D Joystick	M T
	Motion	Continuous Rotary Pod Treadmill Thumbwheel Tasa Ferinstat	Trackball Tasa X-Y Pad	Trackball	M T
	Pressure	Torque Sensing Pressure Pad	Isometric Joystick		T

Figure 2.2 : Dispositif physique vu comme transducteur de trois propriétés physiques (d'après [Buxton 83]).

Les deux premiers axes de cette taxonomie définissent les degrés de liberté du transducteur dans l'espace. Ils expriment la capacité du dispositif à traduire les actions de l'utilisateur en une forme exploitable par le système informatique. La richesse du dispositif est alors fonction du nombre de propriétés physiques captées et pour chaque propriété, le nombre de dimensions spatiales. Le type (direct ou indirect) de la capture, qui traduit le degré d'engagement physique de l'utilisateur, est une notion intéressante qu'il convient d'associer aux propriétés des interfaces à manipulation directe [Hutchins 86].

Toutefois, la taxonomie de Buxton ne concerne que les dispositifs continus. Les dispositifs à fonctionnement discret comme les boutons crantés ne sont pas représentés. De plus, ne sont considérés que les dispositifs actionnables par le geste. En particulier, le microphone et la caméra ne sont pas évoqués. Ces derniers constituent des cas intéressants car leur frontière en tant que système d'acquisition, est mal cernée. Par exemple, toute caméra numérique se caractérise par la résolution de l'image captée et la fréquence d'échantillonnage. Si cette définition est nécessaire au concepteur d'un logiciel de vision par ordinateur, elle n'a pas

d'intérêt direct en conception d'interface. Un niveau d'abstraction pertinent est la détection des contours et leur mouvement comme dans le "videoplace" de Krueger [Krueger 91], ou mieux encore, la position (x, y, z) de chaque doigt comme il faudrait le faire pour le "digital desk" d'EuroPARC [Wellner 93]. J. Crowley traduit l'existence de niveaux d'abstraction avec la notion de "capteur logique" [Crowley 94]. Une argumentation similaire vaut pour le microphone qui peut numériser le signal sonore d'une phrase, fournir un mot ou une suite de mots, modéliser ou non la prosodie. La taxonomie de Mackinlay, Card et Robertson permettent d'exprimer la variabilité des niveaux d'abstraction pour un même dispositif physique.

4.2. L'espace de conception des dispositifs physiques de Mackinlay, Card et Robertson

Mackinlay, Card et Robertson partent d'un constat similaire au notre concernant l'incomplétude de la taxonomie de Buxton. Ils observent aussi que l'assemblage de dispositifs élémentaires en unités de contrôle plus complexes est une dimension importante en conception. L'objectif est alors de définir une théorie qui systématise la connaissance et le raisonnement sur les dispositifs d'entrée.

La théorie de Mackinlay, Card et Robertson se résume ainsi : le domaine est l'ensemble des dispositifs d'entrée. Chaque *dispositif* est défini par un sextuplet <M, In, Out, R, S, W>, et les opérateurs de la théorie expriment la possibilité d'assembler des dispositifs. Le sextuplet caractérise un dispositif d'entrée élémentaire. Il contient les attributs suivants :

• **M** est un opérateur de manipulation ("Manipulation operator") appliqué par l'utilisateur. Il désigne une propriété physique captable par le dispositif. A la différence de la taxonomie de Buxton qui comporte trois propriétés (la position, le mouvement et la pression), une propriété chez Mackinlay et ses coauteurs est une position ou une force. Comme le montre la figure 2.3, une propriété peut être une donnée absolue ou relative, linéaire ou circulaire. Et chaque propriété est une valeur trivaluée notée (x, y, z). Chez Buxton, la dimensionalité d'une propriété n'est pas contrainte à 3 mais variable.

On observe que la modélisation du mouvement est moins précise chez Buxton que chez Mackinlay qui le voit comme une position relative linéaire ou circulaire. Par exemple, chez Buxton, la "trackball" est un transducteur de mouvement bi-valué (x,y) au même titre que la souris. Chez Mackinlay, le mouvement de la trackball est

décrit par une position relative circulaire alors que celui d'une souris est caractérisé par une position relative linéaire.

	Linear	**Rotary**
Position **Absolute** **Relative**	Position **P** Movement **dP**	Rotation **R** DeltaRotation **dR**
Force **Absolute** **Relative**	Force **F** DeltaForce **dF**	Torque **T** DeltaTorque **dT**

Figure 2.3 : Les propriétés physiques captées par les dispositifs d'entrée (d'après Mackinlay, Card, Robertson, figure 5 page 154 de l'article [Mackinlay 90]).

On notera P l'opérateur de position linéaire absolue, T l'opérateur de torsion absolue, R l'opérateur de position angulaire absolue. Les opérateurs de mouvement (modélisés par des valeurs relatives), sont notés dP, dT, dR respectivement. Ces six opérateurs constituent, pour Mackinlay et ses coauteurs, l'ensemble des manipulations primitives qu'un sujet peut appliquer à un dispositif d'entrée.

• **In** est le domaine des valeurs d'entrée possibles de M. Par exemple, pour une caméra, ce sera une matrice mxn de pixels. Pour un bouton rotatif de sélection de la ventilation dans une voiture, In est l'ensemble des réels compris entre 0 et 180 degrés.

• **Out** est le domaine des valeurs de sortie possibles du dispositif. Si le bouton de ventilation comporte 5 positions (cas d'un bouton cranté), Out est alors les 5 valeurs angulaires correspondantes.

• **R**, ou fonction de résolution ("Resolution function"), définit la correspondance entre les domaines In et Out. Dans le cas de la caméra, R est égale à la fonction Identité (les domaines de valeurs In et Out sont identiques). Dans le cas du bouton de ventilation, elle exprime les conditions de passage entre les angles du mouvement effectué par l'utilisateur et les crans (angles en sortie) imposés par le dispositif.

• **S** dénote l'état actuel du dispositif ("state"). Il comprend la valeur *In* de l'entrée prise dans In à l'instant présent t, la valeur *Out* prise dans Out et un état interne. Nous verrons ci-dessous comment S peut être utilisé dans la description du fonctionnement du dispositif.

• **W** désigne le fonctionnement externe et interne du dispositif physique ("Works"). Il permet de modéliser des relations intéressantes entre In et Out en fonction de l'état. Par exemple, pour un joystick de torsion qui revient à la position de repos quand on le relâche, on peut écrire : *Si (relâche = vrai) alors i = 0* où *relâche* est une variable booléenne qui représente la maintient en torsion par l'utilisateur et *i*, la valeur d'entrée. Un autre exemple est la sensibilité sur le domaine In : certains dispositifs permettent la capture de valeurs sans percevoir les valeurs intermédiaires entre deux valeurs captées successives. La tablette est dans ce cas : il y a des gestes humains que le dispositif ne peut percevoir. Pour le bouton rotatif, tout geste se traduit par un opérateur de manipulation avec capture.

A titre d'exemple, le bouton de ventilation se modélise ainsi :

BoutonVentilation =

< M = Rz, (il s'agit d'une rotation autour d'un axe vertical)
In = [0, 180]
Out = <0, 45, 90, 135, 180>
S = s
R = [0, 45) -> 0
[45, 90) -> 45
etc.
>

Le sextuplet <M, In, Out, R, S, W> dénote un dispositif primitif : il décrit la prise en compte d'une manipulation humaine prise dans l'ensemble de base {P, dP, T, dT, R, dR}. Mackinlay et ses coauteurs proposent des opérateurs de composition pour modéliser des dispositifs plus complexes : la fusion ("merge"), la connexion et la disposition ("layout").

• Formellement, la fusion est le produit cartésien de deux dispositifs sur les domaines d'entrée des dispositifs. Par exemple, la position d'une souris est le produit cartésien des deux dispositifs élémentaires captant respectivement l'ordonnée et l'abscisse.

• La connexion traduit la composition en cascade de dispositifs : la sortie de l'un sert d'entrée à d'autres. Par exemple, le bouton du ventilateur est connecté au dispositif qui agira sur la vitesse de rotation du ventilateur : sa sortie (l'angle du cran) sert d'entrée au ventilateur (dont la fonction de résolution met en correspondance la valeur d'angle d'entrée avec une vitesse de rotation).

• La disposition traduit une composition spatiale comme les trois boutons de la souris sont liés spatialement aux dispositifs de position (x,y).

Ces opérateurs de composition sont complétés par un mécanisme d'encapsulation classique qui permet de définir des "dispositifs génériques" c'est-à-dire des classes. Par exemple, l'expression simplifiée <Pz, [min, max] -> [R(min), R(max)]> définit une classe de boutons pressoir qui peut être incarnée en plusieurs exemplaires spécifiques en remplaçant les paramètres formels de l'expression par des paramètres effectifs. Par exemple, un bouton pressoir de souris devient <Pz, [0, max] -> [relevé, enfoncé]>. A son tour, la souris du Sun[2] peut se voir comme l'encapsulation de la fusion de deux dispositifs de position linéaire, trois boutons reliés par une opération de disposition avec la fusion des positions. Si l'on poursuit le raisonnement, la souris est liée au curseur de l'écran (qui est un dispositif) par un opérateur de connexion. L'encapsulation de cette connexion constitue un nouveau dispositif : le pointeur.

Par encapsulations successives, il est possible de définir des dispositifs abstraits qui permettent de passer progressivement et élégamment du monde physique au monde logiciel. En particulier, la caméra peut être modélisée comme un dispositif primitif :

Camera = <

M	: Ø	-- l'utilisateur n'applique pas d'opérateurs de manipulation
In	: <MxN, profondeur>	-- tableau de pixels et leur profondeur
Out	: <MxN, profondeur>	
S	: mxn	-- la scène actuelle en terme de pixels
R	: I	-- fonction identité
W	; {} >	

Connectée à des dispositifs abstraits eux-même composés, la caméra constitue un système de vision par ordinateur. De la même manière, il est possible de représenter les services et le fonctionnement pertinent d'un système de reconnaissance de la parole. Plus généralement, un système interactif peut être vu comme un dispositif composé d'une société de dispositifs plus simples. Nous verrons au chapitre VI que les modèles d'architecture logiciels multiagent répondent au même mécanisme. Comme les dispositifs de Mackinlay, ces agents ont un état, un fonctionnement, des domaines d'entrée et de sortie [Duke 92, Nigay 93].

Pour résumer, Mackinlay et ses coauteurs systématisent la description des dispositifs d'entrée au moyen d'une algèbre simplifiée. L'avantage est double : élargir le champ de conception à la prospection de nouveaux dispositifs et ouvrir la voie à l'évaluation du pouvoir d'expression des dispositifs en relation avec la tâche. La démonstration présentée par Mackinlay, Card et Robertson, dans leur article ne concerne que les dispositifs actionnables

---

2 Sun est une marque déposée de Sun Microsystems, Inc.

explicitement par l'utilisateur. Elle ne tient pas directement pour les dispositifs actionnables implicitement tels la caméra ou le microphone. Cette théorie avec la notion de domaines de valeurs, contribue cependant à attirer l'attention du concepteur. Par exemple, dans le cas du "digital desk", il convient que le système de vision par ordinateur, partant de l'ensemble In de pixels, soit capable en sortie (domaine Out) de modéliser les doigts et la main en accord avec le type de tâches envisagées. La taxonomie de Foley, plus ancienne, part de cette notion de tâche.

4.3. La taxonomie des tâches graphiques selon J. Foley

A l'origine de nombreux travaux comme le nouveau modèle d'interaction de B. Myers [Myers 90], la taxonomie de Foley adopte la tâche comme point d'ancrage. Pour classer un dispositif, Foley propose de le mettre en relation avec les tâches graphiques qu'il permet d'accomplir. Ces tâches sont au nombre de cinq : sélection, position, orientation, chemin, quantificateur et texte qui peuvent, selon les dispositifs être accomplies de manière directe ou indirecte. La figure 2.4 illustre l'approche de Foley.

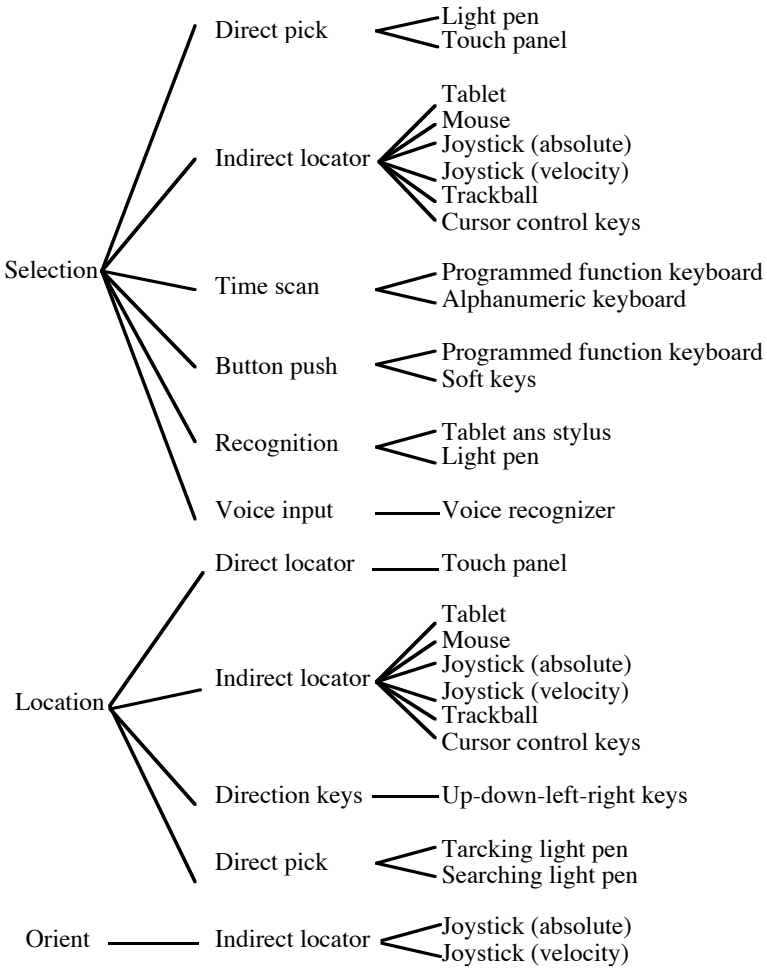


Figure 2.4 : La taxonomie de Foley dirigée par les tâches graphiques [Foley 84].
(Schéma adapté d'après la figure 1 de l'article [Mackinlay 90].)

Contrairement aux taxonomies précédentes qui organisent les dispositifs en fonction des propriétés physiques qu'ils sont capables de modéliser, Foley dirige la classification par la tâche. On peut y voir comme avantage, des recommandations prescriptives sur le choix des dispositifs en fonction des tâches à accomplir. Toutefois, l'espace des tâches envisagé concerne le domaine graphique uniquement. Comme pour les auteurs précédents, les tâches possibles avec la caméra ou le microphone ne sont pas couvertes.

La classification de Foley rend difficile la comparaison des dispositifs entre eux : comme le montre la figure 2.4, un dispositif, qui peut se retrouver dans plusieurs sous-arbres, n'a pas une localisation unique dans l'espace de conception. Sur ce point, les propositions de Buxton et Mackinlay sont plus utiles. Enfin, aucune preuve n'est avancée quant au recouvrement de tous les dispositifs physiques par l'espace proposé. La théorie de Mackinlay, qui se veut générative, doit pouvoir répondre à ce critère.

A l'inverse des interfaces d'entrée, peu d'efforts se sont portés sur la classification des interfaces de sortie. La taxonomie sur les modalités pures d'Ole Bernsen est une exception. Nous la présentons dans la section qui suit.

5. UN ESPACE DE CONCEPTION DES INTERFACES DE SORTIE

En accord avec la métaphore des distances sémantiques et articulatoires de la théorie de Norman, Bernsen s'interroge sur le problème de la représentation optimale des informations que le système et l'utilisateur sont amenés à échanger au cours de leur tâche commune. Dans cet espace problème, Bernsen centre son étude sur la représentation des expressions de sortie qu'il appelle *modalités représentationnelles* [Bernsen 93]. Un graphe, un texte, une icône sont de telles modalités. Il s'agit de représentations produites par le système à destination de l'utilisateur. Ici, modalité doit se comprendre comme signifiant. Il ne s'agit pas des capacités sensorielles humaines telles qu'on l'entend en psychologie. L'objectif est de définir une théorie qui permette au concepteur d'identifier les modalités (les signifiants) qui conviennent à la représentation des concepts du système (les signifiés). Le schéma ci-dessous en décrit les principes.

"Ce qui doit être représenté" -> "principes de correspondance" -> "représentations"

Tout comme la taxonomie de Mackinlay, Card et Robertson, la classification de Bernsen se veut "générative" : elle s'appuie sur une classe de modalités élémentaires dites "modalités pures" combinables en représentations plus complexes. Au sextuplet <M, In, Out, R, S, W>

des dispositifs physiques du paragraphe 4.2, correspond chez Bernsen le doublet <média, profil> :

> • le *média* désigne le support d'expression en relation avec les capacités sensorielles humaines. Il prend sa valeur dans l'ensemble {graphique, son, toucher}. Chacune de ces valeurs - graphique, son, toucher - est caractérisée par un ensemble de qualités perceptuelles particulières (respectivement, visuelles, auditives, et tactiles) que Bernsen appelle *canaux d'information*. Par exemple, la couleur et la texture sont des canaux. Dans un tableau, les lignes et les colonnes sont aussi des canaux. En informatique, nous parlerions d'attributs. Le média ainsi défini correspond à ce que Hemjslev appelle la *substance de l'expression* ou matérialité perceptible (par opposition à la *substance du contenu* qui désigne le concept, le signifié).
>
> • le *profil* regroupe un ensemble de propriétés qui peuvent être de nature :
> - statique ou dynamique,
> - linguistique ou non-linguistique.
> - analogique ou non-analogique,
> - arbitraire ou non-arbitraire.
>
> Par exemple, la modalité langue naturelle écrite a pour média "graphique", et pour profil le quadruplet <linguistique, non analogique, non arbitraire, statique>. Le langage parlé a pour média d'expression le son et présente le même profil que la langue naturelle écrite à l'exception de la dimension temporelle. Nous allons décrire ce que recouvre les qualificatifs d'un profil.

Le caractère *dynamique* ou *statique* d'une modalité repose sur la présence de la dimension temporelle dans la représentation. Une *modalité linguistique* désigne un langage, c'est-à-dire un système structuré de signes remplissant une fonction de communication. Par exemple, la langue parlée et écrite, le langage gestuel, sont des modalités représentationnelles linguistiques : tous s'appuient sur l'acquis d'un lexique et d'une syntaxe et tous, pour remplir leur fonction de communication, recouvrent une sémantique en situation (ou pragmatique).

Une *modalité analogique* entretient un rapport de ressemblance avec la réalité : elle fonctionne par ressemblance à la chose signifiée. Par exemple, le dessin d'une maison est une représentation analogique par rapport à la maison qu'il représente. On parle aussi de représentation iconique ou isomorphique. Il convient de noter que les icônes que l'on conçoit en IHM graphique ne sont pas toujours des représentations analogiques. Une icône est qualifiée *d'abstraite* lorsque le dessin qui la constitue n'est pas analogique mais contient un indice sémantique (par exemple, une flèche pour désigner une direction) ; elle est arbitraire lorsqu'elle ne présente aucun indice sémantique sur le signifié. Une icône qui opère par analogie est

nommée *icône représentationnelle*. Les *icônes semi-abstraites* résultent de la composition d'éléments abstraits et représentationnels

Une *modalité arbitraire* fonctionne en dehors d'un système conventionnel. A l'inverse, les *modalités non arbitraires*, tels les langages, reposent, pour remplir leur rôle, sur l'existence d'un système sémantique appris. Il convient cependant de noter qu'à un niveau fin, le signe linguistique, unité constitutive du langage, se définit selon Saussure, par l'association arbitraire d'une forme (sonore ou graphique) et d'un contenu conceptuel. Par exemple, l'association entre la forme graphique "loup" ou le son [lu] et l'animal sauvage qu'ils évoquent n'est pas logiquement motivée. En vérité, l'association est-elle arbitraire ou bien ne sommes-nous pas en mesure de reconstituer le chemin qui a conduit à cette association? Il semble aussi que la part d'arbitraire dans le lexique n'est pas aussi systématique que cela : les onomatopées lexicalisées comme "clic, vlan" ou les noms d'oiseaux constitués par imitation de leurs cris n'ont pas été formés arbitrairement. A notre sens, ces termes ne sont pas arbitraires et revêtent en plus un caractère analogique.

En combinant les caractéristiques du profil d'une modalité et les valeurs possibles pour le média de restitution, on obtient 48 modalités pures de base : 2 (linguistique, non linguistique) x 2 (analogique, non analogique) x 2 (arbitraire, non arbitraire) x 2 (dynamique, statique) x 3 (graphique, son, toucher). De ces combinaisons, il faut éliminer les incohérences : les modalités analogiques et linguistiques ne sont pas arbitraires et les médias son et toucher ne peuvent être statiques. Bernsen obtient ainsi 28 modalités pures qu'il illustre par des types représentationnels connus. Par exemple, la modalité pure "langage écrit analogue statique" est illustrée par les hiéroglyphes, la modalité "son réel", tel le bruit d'une porte qui s'ouvre, est une modalité à profil <non linguistique, analogique, non arbitraire, dynamique>.

Les dimensions du profil d'une modalité ont des propriétés que l'on peut relier aux capacités perceptuelles et cognitives humaines. La "dynamicité" confère un caractère éphémère qu'il convient d'exploiter avec prudence en fonction du contexte situationnel. Par exemple, Palmiter constate que les représentations graphiques animées, admises pour faciliter l'apprentissage initial, ne promeuvent pas nécessairement la rétention de l'information à plus long terme [Palmiter 91] : des sujets ayant reçu un enseignement par démonstration animée des services "auteur" d'HyperCard se sont révélés plus performants (en terme de temps d'exécution de tâche) que les sujets instruits par écrit. Huit jours plus tard, la supériorité des performances est inversée résultant, pense-t-on, d'un traitement superficiel de l'information dans le cas de l'apprentissage par illustration animée.

Au caractère éphémère des modalités dynamiques, il convient de relier les notions d'inévitable et de prolongation introduites par Sellen et ses coauteurs. Sellen [Sellen 92]

caractérise les retours d'information selon trois critères : éphémère/non-éphémère, évitable/inévitable, prolongé/non-prolongé (“sustained/unsustained”). Par exemple, un bip émis à l’affichage d’une fenêtre est éphémère (car dynamique), inévitable (dans des conditions normales) et non-prolongé (une fois émis, il n’est pas répété). Il est prolongé par le système s’il est répété jusqu’à ce que l’action attendue soit réalisée par l'utilisateur (cas de l’imprimante qui signale l’absence de papier). Ces trois dimensions, sont, à notre sens, d’utiles affinements de la dimension statique/dynamique.

Les représentations analogiques, qui imitent ou reproduisent la réalité, induisent certainement moins d’efforts cognitifs que les représentations non analogiques. De même, les représentations arbitraires impliquent de la part du récipient l’effort d’apprendre une nouvelle convention. Il faut insister sur le fait que le concepteur peut choisir d’utiliser une modalité analogique mais adopter une correspondance arbitraire avec le concept à représenter. Par exemple, il pourrait décider de représenter l’état de transfert d’un fichier par le grincement d’une porte! A l’inverse, Gaver note à propos du SonicFinder, que l’association d’un son analogique ne convient pas nécessairement à décrire le changement d’état d’objets métaphoriques [Gaver 89]. Par exemple, l’une des premières versions du SonicFinder associait des bruits de fenêtre du monde réel aux changements d’état des fenêtres de l’écran. L’expérience a montré que des sons analogiques mais impropres aux fenêtres physiques (par exemple, “schuss”), étaient mieux appropriés.

Les représentations linguistiques sont focalisées mais manquent de spécificité [Bernsen 93]. A l’inverse, les représentations analogiques sont spécifiques mais manquent de focalisation. On entend par *spécificité* la capacité d’une représentation à limiter le champ interprétationnel. La *focalisation* propre aux modalités linguistiques met en lumière le sujet du discours, l’intention communicationnelle de l’auteur. Par exemple, le texte “un angle de 60º” est plus focalisé que la représentation graphique d’un angle de 60º. L’observateur du dessin est amené à s’interroger sur le trait important qu’il convient de relever : est-ce l’existence de l’angle qui compte ou bien est-ce la valeur 60º? La modalité linguistique “un angle de 60º” ne laisse aucun doute. Elle est focalisée mais elle est moins spécifique : elle laisse de côté les détails telles que l’épaisseur du trait, l’orientation de l’angle dans l’espace, etc. La complémentarité “focalisation-spécificité” des modalités linguistique et analogique conduit à les combiner en des représentations multimodales. Les cartes annotées de texte ou les textes illustrés de représentations analogiques sont des exemples courants. On trouvera dans [Nielsen 93] une revue intéressante d’études sur la combinaison de la langue naturelle écrite et des représentations graphiques.

Pour résumer, la résolution du problème de la correspondance entre signifiants et signifiés repose sur l’existence d’une taxonomie solide qui offre des possibilités d’analyse. La

théorie de Mackinlay, Card et Robertson vise cet objectif pour les dispositifs d'entrée. Bernsen poursuit un but similaire pour les sorties. Si la théorie des premiers est dirigée par la technologie (les dispositifs d'entrée sont modélisés comme des transducteurs de propriétés physiques), la taxonomie de Bernsen est davantage liée aux capacités cognitives humaines avec les notions de profil et de média.

Nous venons de présenter des taxonomies qui concernent soit les entrées soit les sorties. Nous considérons maintenant des espaces de conception qui couvrent à la fois l'interface d'entrée et l'interface de sortie. L'espace de Frohlich, centré sur l'utilisateur, est présenté en section 6. Les classifications de la section 7 proposent une vue complémentaire orientée système.

6. L'ESPACE DE CONCEPTION DE DAVID FROHLICH

D. Frohlich adopte comme point de départ le canevas du Modèle du Processeur Humain [Card 93] présenté brièvement au paragraphe 3. Pour le besoin de notre analyse, ce modèle se résume ainsi : 1) l'interaction est le fruit d'un échange bidirectionnel d'informations à travers deux interfaces distinctes, l'une d'entrée, l'autre de sortie ; 2) du côté humain comme du côté informatique, l'information subit des transformations selon le cycle "acquisition-traitement-action". Partant de ce canevas, Frohlich introduit les éléments de son espace de classification [Frohlich 91].

6.1. Présentation des dimensions de l'espace

Dans son analyse, Frohlich adopte la distinction entre entrée et sortie. Il en résulte deux espaces de conception illustrés par les schémas des figures 2.5 et 2.6. On y distingue quatre éléments structurants : les notions de mode, de canal, de média et de style.

6.1.1 Modes

D'après Frohlich, les *modes* d'une interface se définissent comme "des états dans lesquels différentes actions de l'utilisateur peuvent avoir le même effet[3]." [Frohlich 91, p. 58]. Par exemple, la destruction d'un fichier peut s'exprimer par la saisie de la commande "Détruis <nom du fichier>" ou se faire en glissant l'icône du fichier dans la corbeille. Le premier procédé relève d'une activité langagière, la seconde d'un engagement dans l'action physique. Pour

3 Dans le texte anglais original, on lit : "interface modes are defined as states across which different user actions can have the same effects."

l'utilisateur, et pour Frohlich, langage et action sont deux modes distincts pour lesquels les méthodes diffèrent.

La notion de mode recouvre chez Frohlich les deux métaphores essentielles de l'interaction homme-machine présentées par Hutchins et ses coauteurs dans [Hutchins 86]. Il distingue le *mode action* qui correspond à la métaphore du monde modèle ("model world metaphor") et le *mode langage* pour désigner la métaphore conversationnelle ("conversation metaphor").

Une interface qui répond au *mode action*, ou métaphore du monde modèle, présente à l'utilisateur un monde sur lequel il lui est possible d'agir directement. Il n'y a pas d'intermédiaire entre l'utilisateur et le monde présenté. L'interface *est* le monde et ce monde change d'état sous l'effet des manipulations de l'utilisateur. Lorsque le principe de cette métaphore est appliqué avec soin, l'utilisateur est conduit à assimiler le monde perçu aux objets du domaine de la tâche. La métaphore du bureau d'Apple est l'une des premières applications réussies de ce principe. B. Laurel qualifie d'interfaces "à la première personne" ces systèmes qui offrent le sentiment d'engagement direct dans l'action sans relai intermédiaire [Laurel 86].

Dans le mode langage ou métaphore conversationnelle, l'interface est un support linguistique qui permet au système et à l'utilisateur de converser à propos du monde supposé des objets de la tâche. Les objets du domaine ne sont plus représentés explicitement et ne sont manipulables que par un intermédiaire : l'interface. L'utilisateur décrit les actions à appliquer sur les objets. Il ne les réalise pas directement lui-même. Il le fait réaliser. Pour marquer la présence d'un intermédiaire entre l'utilisateur et les objets de la tâche, B. Laurel parle d'interfaces "à la deuxième personne" [Laurel 86]. Si l'on reprend la terminologie des actes de langage d'Austin, le mode langage conduit l'utilisateur à "faire-savoir" au système ou à "faire-faire" via l'interface [Austin 62]. Inversement, dans le mode action, l'utilisateur "fait".

La plupart des systèmes informatiques offrent, en vérité, un curieux mélange de ces interfaces à la première et à la deuxième personne. Par exemple, les éditeurs de texte adoptent tous l'action pour ce qui est de la saisie de texte mais ils imposent le conversationnel pour l'activation des commandes. Certes, les menus, les formulaires et les boutons, font l'objet de manipulations directes mais ils servent d'intermédiaires entre l'utilisateur et leurs signifiés, concepts du domaine, qui sont alors manipulés par indirection et selon des conventions langagières. Il convient donc d'être prudent sur la qualification "interface à manipulation directe". Il faut savoir que la présence d'objets de présentation comme les menus et les formulaires, ne suffit pas à garantir l'action directe sur les concepts du domaine.

Cette vision de la notion de mode contraste avec la définition communément admise en informatique : selon le *mode*, une même suite d'actions utilisateur peut conduire à des effets différents [Thimbleby 90]. D'après Frohlich, le mode informatique est un concept commode qui permet au système de résoudre les ambiguités d'interprétation. Nous disons que le mode, du point de vue système, décrit une situation ou un *contexte interactionnel* qui permet au système d'orienter son interprétation des données acquises. Le mode de Frohlich, centré sur l'utilisateur, met en relief des propriétés de "redondance" au sens où l'utilisateur a plusieurs façons de communiquer au système un objectif donné. Nous verrons au chapitre IV, que d'autres propriétés intéressantes, telle la combinaison, caractérisent utilement l'usage des "modalités". La redondance de Frohlich, que nous appelons "équivalence", ne recouvre qu'un aspect dans l'espace du possible.

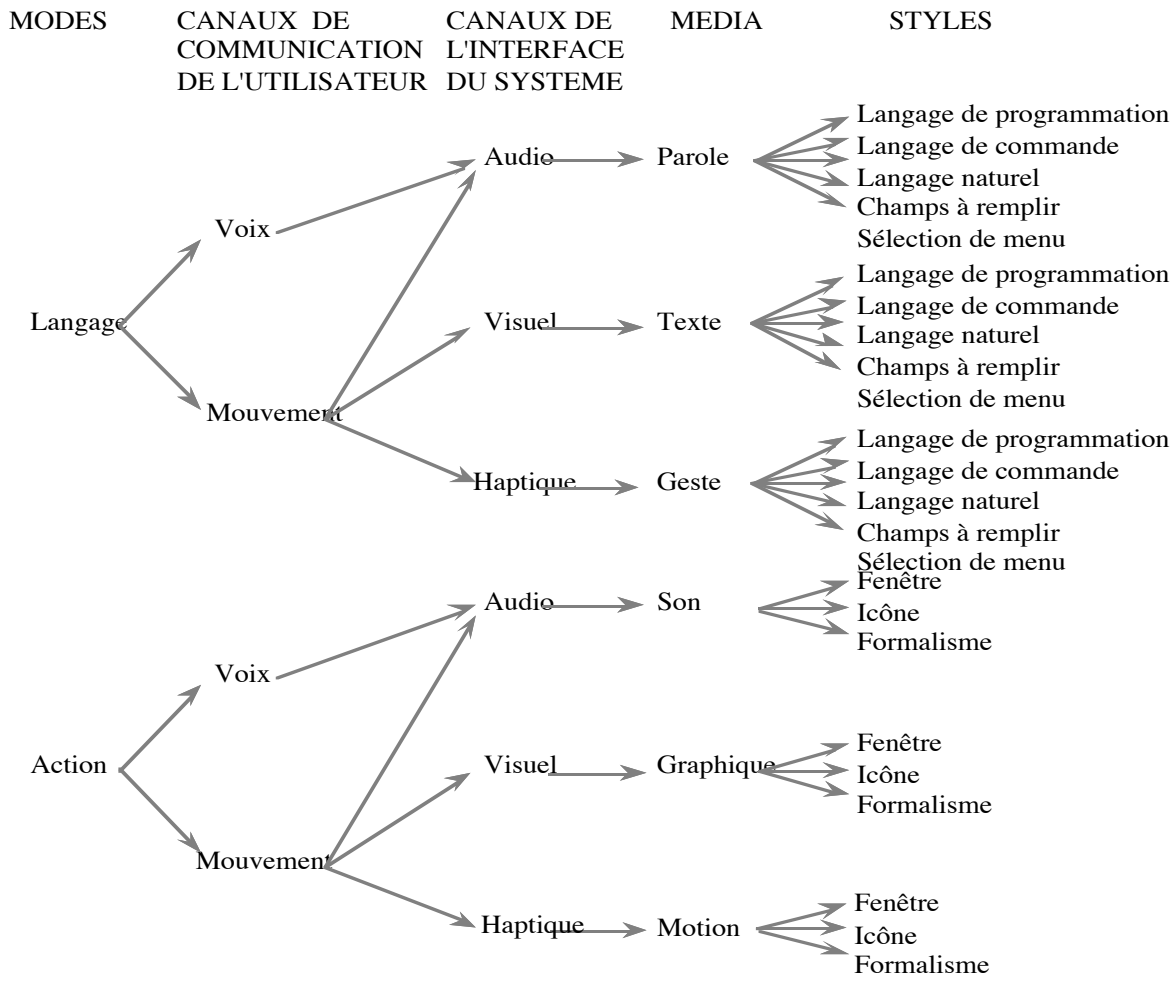


Figure 2.5 : L'espace de conception des interfaces en entrée de Frohlich.□

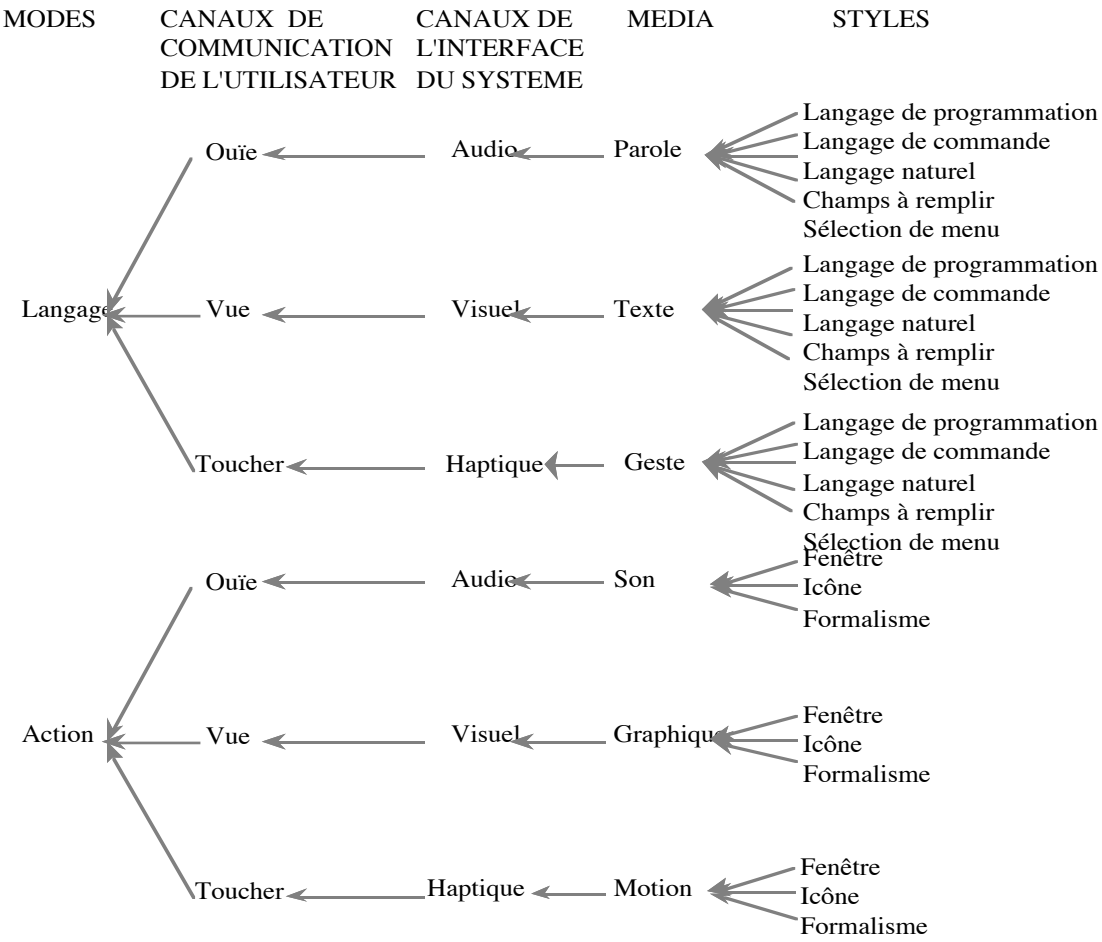


Figure 2.6 : L'espace de conception des interfaces en sortie de Frohlich.

6.1.2. Canaux

Frohlich définit un *canal* "comme une interface qui opère une transformation d'énergie[4]" [Frohlich 91, pp. 58]. Il adopte une position anthropomorphique en mettant en relation les capteurs et effecteurs qui définissent les canaux de communication humains telles l'ouïe ou la vue, avec les dispositifs physiques qui correspondent aux canaux de l'interface du système. A chaque canal correspond un dispositif chargé de la transformation d'énergie. Par exemple, pour le canal "voix humaine", les dispositifs sont les cordes vocales. Pour le canal de l'interface "audio", le dispositif est un microphone. Dans le premier cas, l'énergie est de nature électrochimique. Elle est électrique pour le microphone.

D'après la figure 2.5, les mode langage et action en entrée font appel aux canaux de communication humains voix et mouvement. La voix humaine est captée par le canal audio du

4 Dans le texte original : "a channel is defined as an interface across which there is a transformation of energy"

système tandis que le mouvement peut l'être par les trois canaux système audio, visuel et haptique. En sortie, il y a correspondance biunivoque entre canal système et canal humain.

6.1.3. Média

Un *média* est "un système représentationnel qui permet l'échange d'information[5]" [Frohlich 91, pp. 58]. Comme le montrent les figures 2.5 et 2.6, Frohlich distingue six médias : la parole, le texte, le geste qu'il rattache au mode langage ; le son, le graphique et la motion attribués au mode action. La parole recouvre toute forme de vocalisation à fonction communicationnelle y compris les silences. Le texte recouvre la trace visible d'un mouvement communicationnel tandis que le geste désigne un mouvement communicationnel sans trace visible. Le son est toute forme audible autre que la parole, le graphique toute forme de trace visible autre que le texte et la motion tout mouvement autre que le geste!

Les définitions que Frohlich adopte pour ses six catégories de médias sont certainement discutables. L'intérêt qui se dégage de son espace vient de la mise en relation entre les notions de mode, de canal et de média. En particulier, le média pris comme système représentationnel, est fixé par le couple <canal, mode>. Par exemple, la distinction entre un geste et une motion ne réside pas tant dans la nature des muscles mis en jeu, mais dans la fonction intentionnelle, c'est-à-dire, pour revenir au modèle de Frohlich, au mode langage ou action. De même, ayant choisi le canal système visuel, le seul média d'entrée possible en mode langage est le texte alors que pour le mode action, ce média ne peut être que graphique. Notons qu'en sortie, le choix du média implique le choix du canal système.

6.1.4. Styles d'interface

Dans son espace de conception, Frohlich définit un *style d'interface* comme "une classe reconnue de méthodes qui rend possible l'activité d'interaction[6]" [Frohlich 91, pp. 59]. Il distingue deux classes de méthodes correspondant chacune aux modes langage et action et qu'il note respectivement "style langage" et "style action". Les styles langage permettent de produire des expressions alors que les styles action permettent la production d'événements. L'auteur ne précise pas ce qu'il entend par expression et événement. Nous croyons deviner une différence de granularité dans les unités d'interaction entre le système et l'utilisateur. Nous étudierons les conséquences de cette distinction en termes d'architecture au chapitre VI. Dans le cadre de cette

---

5 Dans le texte original en anglais, "a medium is defined as a representational system for the exchange of information".

6 Dans le texte original anglais, "interface style is defined as a recognised class of methods for supporting interface activity".

classification, nous estimons que les notions d'expression et d'événement ne sont pas clarifiantes.

Les styles langage recouvrent les langages de programmation, les langages de commande, la langue naturelle, les champs de formulaires que l'on remplit et les menus :

- Le langage de commande définit le langage formel que l'utilisateur emploie pour instruire le système.
- Le langage de programmation correspond à un langage de commande autorisant des extensions par la possibilité de définir des procédures ou des méta-commandes.
- Le langage naturel est un langage de commande, sous-ensemble bien défini d'une langue naturelle telle que le Français, etc.

Les styles action incluent :

- Les environnements fenêtrés qui structurent l'espace de travail par groupes fonctionnels.
- Les interfaces iconiques qui s'appuient sur une métaphore du monde réel.
- Les formalismes graphiques qui correspondent à une interaction non-métaphorique et où le sens de l'activité est défini par le logiciel d'application.

Cette classification est une reprise de celle de Baecker et Buxton [Baecker 87]. Par rapport aux taxonomies de Mackinlay et de Bernsen, son manque de généricité nous donne le droit de nous interroger sur sa complétude.

6.1.5. Liens entre les dimensions de l'espace

Pour résumer, nous explicitons les liens entre les cinq dimensions de l'espace de Frohlich : mode, canaux humain et système, média, et style.

1) Un média est défini par le couplage d'un mode et d'un canal système.

2) A un média correspond un style adéquat. Par exemple, au média parole est associé le langage naturel comme style approprié.

3) A un mode d'interaction correspond une et une seule classe de styles.

Jusqu'ici, nous avons décrit les éléments structurants de l'espace de Frohlich et émis quelques jugements de valeur sur chacune de ces dimensions. Dans la discussion qui suit, nous analysons l'apport et les faiblesses de cet espace en le considérant dans son ensemble.

6.2. Discussion

6.2.1. Intérêt

L'espace de Frohlich peut être utilisé a posteriori pour analyser le résultat d'une conception ou a priori pour guider les choix de conception.

Pour tout système existant, il est possible d'instancier le modèle général appliqué au cas particulier du système à étudier. Par exemple, l'interface du Finder d'Apple, qui s'appuie sur la métaphore du bureau, recouvre les deux modes action et langage en entrée comme en sortie. En entrée, les actions de manipulation directe sur les icônes de fichier se caractérisent par le quintuplet :

- mode = action,
- canal humain = mouvement,
- canal système = haptique,
- média = motion,
- style = iconique.

Certaines fonctions, telle l'ouverture de fichier, sont disponibles de manière équivalente mais dans le mode langage via les menus, les formulaires et les raccourcis clavier. Plus précisément, le raccourci clavier se caractérise par :

- mode = langage,
- canal humain = mouvement,
- canal système = haptique,
- média = geste (le mouvement ne laisse pas de trace),
- style = langage de commande.

et les formulaires par :

- mode = langage,
- canal humain = mouvement,
- canal système = visuel,
- média = texte,
- style = remplissage de formulaire.

Un raisonnement similaire peut être appliqué à l'interface de sortie. Par exemple, une fenêtre de dialogue qui affiche un message d'erreur se voit caractérisée par :

- style = langage naturel,
- média = texte,

- canal système = visuel,
- canal humain = vue,
- mode = langage.

et l'accompagnement sonore d'une fenêtre de dialogue se définit par :

- style = formalisme (qui signifie “erreur” dans le logiciel),
- média = son,
- canal système = audio,
- canal humain = ouïe,
- mode = action.

Cet exercice démontre qu’il convient d’être précis lorsque l’on vise à évaluer l’utilisabilité d’un système en fonction de ses caractéristiques. Comme le note Frohlich, on avance peut être un peu hâtivement que la manipulation directe, comparée au mode langage, augmente l’utilisabilité du système. Le Finder d’Apple, que l’on s’accorde à dire “facile d’emploi parce que relevant de la manipulation directe”, montre à l’analyse, que son interface accorde une large part au mode langage. Son utilisabilité ne provient donc pas uniquement du mode action mais d’une combinaison adaptée des différents modes.

Le second apport de l’espace de Frohlich est son utilisation pendant la conception d'un système. L'espace définit des niveaux d'abstraction auxquels correspondent des décisions de conception. Les liens entre les niveaux définissent des implications entre les décisions. Par exemple, dans une approche descendante, le choix d'un mode implique un nombre limité de styles. Inversement, dans une approche ascendante, le choix d'un style implique un mode. Cette structuration de l’espace de conception en niveaux d’abstraction n’est pas sans rappeler l’approche en couches dont CLG s’est fait le pionnier [Moran 81].

En résumé, l’espace de Frohlich permet de caractériser de façon concise aussi bien les interfaces d’entrée que les interfaces de sortie de systèmes existants. En conception, il permet de situer différentes décisions et d'expliciter leurs liens. De manière générale, il fixe une terminologie organisée dans un cadre. Ainsi, introduisant les termes mode et média, il permet de distinguer un système multimode d'un système multimédia. Mais à ces avantages correspondent aussi des limitations.

6.2.2. Limitations

L’espace de Frohlich ne couvre pas les dispositifs d’entrée/sortie dont les propriétés, nous l’avons vu, ont des incidences sur l’utilisabilité du système. Centré sur l’utilisateur, cet espace ne permet pas de considérer les problèmes liés à la mise en œuvre des systèmes.

La notion de canal encapsule, chez Frohlich, un ensemble de capteurs et d'effecteurs sans préciser leur nature. Puisque chaque dimension du modèle de Frohlich peut être vue comme un niveau d'abstraction, nous suggérons d'injecter dans l'interface d'entrée les taxonomies des dispositifs physiques de Buxton ou de Mackinlay comme une dimension intermédiaire entre les canaux de communication de l'utilisateur et les canaux système. Nous adopterons le principe de cet affinement dans notre propre taxonomie présentée au chapitre III.

Pour les sorties, il est possible d'expliciter le lien entre la taxonomie des modalités pures de Bernsen et les dimensions média et style de Frohlich. Dans la terminologie de Bernsen, une modalité représentationnelle se définit par le couple <média, profil> où "média", qui désigne le support d'expression graphique, son, toucher, peut être mis en corrrespondance avec la notion de média de Frohlich. Le profil permet de qualifier une modalité représentationnelle. Chez Frohlich, cette notion de modalité est couverte par celle de style qui peut être alors caractérisé dans la taxonomie de Bernsen. Ainsi le "style langage de programmation" produit par le système selon le "média parole" est une modalité au sens de Bernsen dont le média est le son et le profil "linguistique, non analogique, non arbitraire, dynamique".

L'espace de Frohlich ne permet pas de relier les décisions de conception à leurs conséquences en terme de réalisation. Il n'explicite pas non plus le parallélisme qui peut exister dans l'utilisation des canaux de communication qu'ils soient humains ou système. Cette dimension nous semble importante à considérer aussi bien pour l'utilisateur du système que pour le réalisateur :

- Pour l'utilisateur du système : peut-il parler tout en sélectionnant des objets avec la souris? Comment les modes langage et action sont-ils disponibles? Le sont-ils simultanément ou de manière exclusive? Autant de questions qui ont trait à l'usage temporel des canaux humains et à la résolution de tâche.

- Pour la réalisation du système : faut-il considérer des traitements de données en parallèle? Le faut-il en entrée uniquement ou bien en sortie? Autant de questions en relation avec les traitements temporels du système depuis les canaux système jusqu'aux services internes de plus haut niveau.

En résumé, cet espace a le mérite d'être le premier, à notre connaissance, à avoir défini une taxonomie qui recouvre à la fois les interfaces d'entrée et de sortie. S'il constitue un cadre de pensée général, il ne permet pas de différencier les systèmes en fonction des possibilités d'utilisation offertes à l'utilisateur et en fonction des problèmes de réalisation. Ces aspects vont être abordés avec l'espace MSM auquel nous avons contribué.

7. L'ESPACE MSM

Avec le concept de *système Multi-Sensori-Moteur*, nous optons pour une perspective ouverte qui s'affranchit des querelles de définition à propos du "multimodal" et du "multimédia". Nous proposons un espace de référence qui caractérise un système en des termes utiles au concepteur de logiciel : à ces caractéristiques correspondent des implications sur les solutions logicielles.

Après un exposé des dimensions de l'espace en 7.1, nous mettons en évidence les points contributifs de MSM en 7.2 à savoir : 1) un cadre précis pour une définition de la multimodalité et du multimédia, 2) une amélioration des concepts esquissés par l'espace IHMM'91 [IHMM 91]. Nous concluons la discussion par les points faibles de MSM qui justifient, aux chapitres III et IV, la présentation de nos travaux personnels sur l'analyse de l'espace problème

7.1. Les dimensions de MSM

On trouvera une description complète de MSM dans [Coutaz 93c] et [Nigay 93a]. Nous reprenons ici les lignes directrices. Comme le montre la figure 2.7, l'espace MSM comprend 6 dimensions.

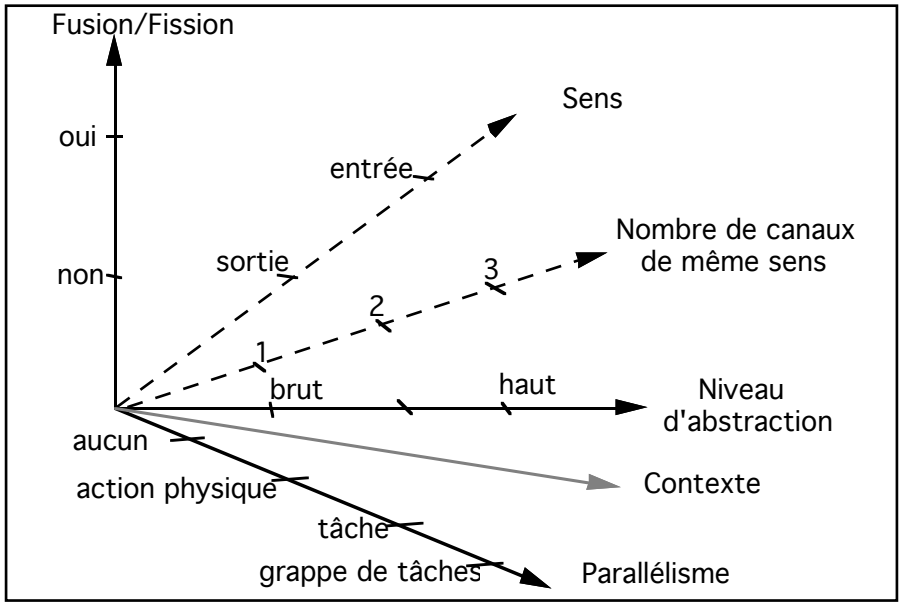


Figure 2.7 : Le référentiel MSM : un espace de référence pour système multi-sensori-moteur.

Les deux premières dimensions ont trait à la notion de canal de communication : le nombre et le sens des canaux permis pour un système donné. Ces notions font l'objet du

paragraphe 7.1.1. Les autres dimensions caractérisent le degré de sophistication cognitive des fonctions d'interprétation et de restitution du système : niveaux d'abstraction, contexte, fusion et fission des informations échangées, granularité du parallélisme. Ces dimensions sont présentées au paragraphe 7.1.2.

7.1.1. Canal de communication

Un canal de communication d'entrée (inversement, de sortie) regroupe des dispositifs physiques d'entrée ou capteurs (inversement, de sortie ou effecteurs) capables de recevoir (d'émettre) des informations de types donnés sous le contrôle d'une capacité computationnelle ou processus. Inspirée du modèle psychologique de la communication tel ICS [Barnard 87], cette définition met l'accent, non pas sur l'aspect liaison, mais sur les intervenants de l'acte de communication : la source et le récipient.

Notre définition s'applique aussi bien aux canaux humains qu'aux canaux digitaux. Par exemple, le canal visuel humain comprend plusieurs capteurs, les rétines, et un sous-système local visuel qui traduit les relations spatio-temporelles des photons en une représentation interne acceptable pour les "sous-systèmes représentationnels" internes [Barnard 87]. De même, le processus serveur X Window gère plusieurs dispositifs d'entrée, la souris et le clavier, et traduit les signaux d'interruption en une forme logique, l'événement, compréhensible par les processus internes clients. On trouvera dans [Coutaz 93a] une description détaillée de la correspondance entre les canaux de communication humains et les sous-systèmes sensori-moteurs d'ICS.

7.1.2. Fonctions d'interprétation et de restitution

L'information acquise par les canaux d'entrée digitaux est transformée par les processus internes du système. Cette suite d'opérations modélise la *fonction d'interprétation*. Dans l'autre sens, l'information interne (c.-à-d. l'état interne du système) est transformée jusqu'à ce qu'elle soit acceptable par les canaux de sortie du système. Cette suite d'opérations constitue la *fonction de restitution.*

A chaque canal d'entrée, nous associons une fonction d'interprétation qui prolonge au cœur du système, les capacités de calcul du canal. De même, à chaque canal de sortie, nous associons une fonction de restitution. Interprétation et restitution font intervenir quatre ingrédients en intime relation : niveaux d'abstraction, contexte, fusion et fission, parallélisme.

7.1.2.1. Niveaux d'abstraction

La notion de niveau d'abstraction exprime le degré de transformations subies par les informations reçues ou émises sur les canaux. Elle couvre également l'éventail des

représentations que gère le système depuis les informations brutes (les signaux) jusqu'aux représentations symboliques (le sens). Une fonction d'interprétation se caractérise par son *pouvoir d'abstraction* des événements reçus du canal d'entrée auquel elle est associée. Une fonction de restitution se qualifie par son *pouvoir de matérialisation* depuis les représentations internes jusqu'aux représentations de bas niveau de son canal de sortie. En raison de la correspondance bi-univoque que nous avons établie entre un canal et une fonction, tout canal peut être caractérisé par son niveau d'abstraction. Pour un système donné, certains canaux peuvent être très performants en abstraction d'autres pas.

7.1.2.2. Contexte

La capacité d'une fonction à abstraire ou à matérialiser peut dépendre de *variables contextuelles* ou *contexte*. Nous définirons de manière plus complète cette notion au chapitre IV. Pour l'instant, nous conviendrons qu'un contexte comprend un ensemble de paramètres d'état utilisés par les processus internes pour contrôler l'interprétation ou la restitution. Il agit comme un filtre cognitif. Par exemple, dans les systèmes actuels, nous observons deux contextes d'interprétation pertinents : les commandes et les concepts du domaine. Les informations liées aux commandes font l'objet d'une interprétation à un haut niveau d'abstraction tandis que les informations ayant trait aux concepts du domaine, sont souvent laissées inchangées. Par exemple, dans les systèmes de traitement de texte, les saisies au clavier ou à la souris qui correspondent aux commandes sont "comprises" par le système tandis que ces mêmes saisies à destination du contenu des documents ne sont pas interprétées.

Ainsi, le pouvoir d'abstraction ou de matérialisation d'un système est propre à chaque canal, mais pour un canal donné, ce pouvoir dépend du contexte.

7.1.2.3. Fusion et fission d'information

La *fusion* est la combinaison de plusieurs unités d'information pour former de nouvelles unités. La *fission* correspond au processus inverse. L'une et l'autre traduisent deux activités importantes des processus d'interprétation et de restitution. Nous reviendrons fréquemment sur ces deux points dans la suite de ce mémoire.

En interprétation, *la fusion* peut intervenir à un bas niveau d'abstraction entre des informations pouvant provenir de plusieurs canaux d'entrée. Elle peut aussi s'effectuer à un plus haut niveau d'abstraction pour des informations issues de différents contextes. Par exemple, le paradigme du "mets ça ici" nécessite la fusion de l'événement parole reçu via le canal du microphone avec les événements souris émanant certainement d'un autre canal. Mieux connue est la fusion de clics souris répartis sur une palette et une zone de dessin pour dessiner une figure géométrique. Ces informations issues du même canal transitent selon des contextes

différents pour être regroupés à un haut niveau d'abstraction [Nigay 91a]. Nous verrons au chapitre VI que les agents d'une architecture logicielle multi-agent constituent de tels contextes.

En restitution, *la fusion* intervient également à plusieurs niveaux d'abstraction. Au niveau le plus haut, elle a lieu dans l'adaptation des informations du noyau fonctionnel aux besoins conceptuels de l'interface [Coutaz 91]. (Nous verrons au chapitre VI des exemples d'adaptation.) Au niveau le plus bas, elle se manifeste par exemple sous forme d'incrustations (vidéo et graphique).

La *fission* en interprétation traduit le besoin de décomposer une information issue d'un canal ou d'un contexte pour franchir un niveau d'abstraction. Par exemple, l'acte de parole "dessine un cercle dans une nouvelle fenêtre" fait référence à deux domaines de discours : les figures géométriques ("dessine un cercle") et l'interface homme-machine ("nouvelle fenêtre). Cette phrase dont le sens a pu être identifié le long d'un canal unique doit être décomposée en deux primitives de haut niveau du système : "créer fenêtre" et "créer cercle" dans la nouvelle fenêtre.

La *fission* en restitution revêt plusieurs formes. La plus courante est la représentation multiple d'un même concept sur un canal donné. Par exemple, le concept de température est restitué sous forme d'un thermomètre gradué ou d'un réel. On dit que les deux représentations sont équivalentes. La représentation multiple peut aussi s'effectuer en coréférence sur des canaux distincts tel le message oral "attention à cette température" accompagné de l'affichage en rouge du thermomètre à surveiller. Dans ce cas, nous parlons de complémentarité. Equivalence et complémentarité seront présentées de manière formelle au chapitre IV.

7.1.2.4. Parallélisme

Nous limitons la discussion au parallélisme perceptible à l'interface. Nous en étudierons les conséquences sur l'architecture logicielle au chapitre VI. Au niveau de l'interface, le parallélisme se manifeste à trois niveaux de granularité : action physique, tâche élémentaire, grappe de tâches.

Au niveau physique, le parallélisme en entrée autorise l'utilisateur à agir simultanément sur plusieurs dispositifs d'entrée. Si ces dispositifs sont répartis sur différents canaux, alors plusieurs canaux d'entrée sont sollicités en parallèle comme dans l'exemple du "mets ça ici". En sortie, parallélisme signifie que plusieurs primitives de sortie peuvent être produites simultanément sur un même canal (comme en animation graphique) ou sur plusieurs canaux (comme pour l'exemple "attention à cette température"). Notons que la fusion et la fission à bas niveau d'abstraction dépendent de l'existence du parallélisme au niveau physique.

On appelle *tâche élémentaire*, toute tâche que l'utilisateur peut accomplir avec le système via une et une seule commande. Au niveau tâche élémentaire, le parallélisme en entrée autorise l'utilisateur à construire plusieurs commandes de manière concurrente : en parallélisme vrai si le parallélisme est géré au niveau physique ou, en l'absence de parallélisme physique, de manière entrelacée (dialogue à plusieurs fils [Coutaz 87]). En sortie, une tâche élémentaire recouvre l'ensemble des primitives de sortie pour exprimer un changement d'état du système.

Nous appelons *grappe de tâches*, un ensemble de tâches élémentaires permettant à l'utilisateur d'accomplir une tâche composée. Au niveau grappe de tâches, le parallélisme exprime les possibilités d'entrelacement entre des espaces de travail. Par exemple, pour un robot mobile de surveillance, les tâches de l'utilisateur peuvent être structurées en trois espaces : description de la géographie, spécification de missions, et contrôle d'exécution de mission. Les relations entre ces trois espaces peuvent avoir une incidence sur l'utilisation temporelle des canaux de communication et donc sur les fonctions d'interprétation et de restitution du système.

7.2 Contributions de MSM

MSM contribue à clarifier l'espace problème en le structurant de dimensions qu'il convient de considérer lors de la conception logicielle. Il offre aussi un cadre pour la définition des termes "multimodal" et "multimédia" et précise les concepts encore flous de IHMM'91. Nous reprenons ces deux points ci-dessous.

7.2.1. Interfaces multimédia et interfaces multimodales

Avec la dualité "homme/système" de la communication homme-machine, nous devons préciser la perspective adoptée pour la définition des termes multimédia et multimodal : MSM, centré sur les caractéristiques des systèmes, offre une définition dirigée par la technique.

La dimension "nombre de canaux de même sens" permet de distinguer le multi- du mono-. Etre *multi*modal ou *multi*média pour un sens donné s (s=entrée ou s=sortie) signifie disposer d'au moins 2 canaux de sens s. En conséquence, un système qui disposerait d'un canal en entrée et d'un canal de sortie aurait deux canaux à offrir à l'utilisateur mais ceux-ci étant de sens opposé, ce système ne saurait être ni *multi*modal ni *multi*média. Cette définition s'énonce ainsi de manière plus formelle. Soient :

- C_e, l'ensemble des canaux d'entrée du système,
- C_s, l'ensemble des canaux de sortie du système,
- N_e et N_s les cardinaux respectifs de C_e et C_s.

Dans ces conditions, le système est :

- multi- en entrée si $N_e \geq 2$,
- multi- en sortie si $N_s \geq 2$.

Il convient de retenir que le caractère multi- d'un système tient au caractère multi- de son espace de canaux et ceci pour chaque sens, entrée et sortie. En particulier, il est fréquent que les systèmes actuels soient mono- en entrée (le clavier et la souris sont gérés par un même service système de bas niveau) et multi- en sortie (l'écran et les hauts-parleurs étant gérés par des services système distincts). Nous admettons que notre distinction entre multi- et mono-, centrée sur la technique logicielle, dépend étroitement de la plate-forme d'accueil. S'il est vrai que la plate-forme fait la différence pour l'implémenteur, nous proposons au chapitre suivant de nous affranchir de cette dépendance en remplaçant la notion de canal par celle de dispositif physique.

En supposant qu'un système soit multi- au sens de la règle sur les canaux, la dimension "niveau d'abstraction" de l'espace MSM, va nous permettre de choisir entre le caractère -modal ou -média. Soient :

- C_e, l'ensemble des canaux d'entrée du système,
- C_s, l'ensemble des canaux de sortie du système,
- n_e, le nombre de canaux de C_e dont la fonction d'interprétation a un fort pouvoir d'abstraction,
- n_s, le nombre de canaux de C_s dont la fonction d'interprétation a un fort pouvoir d'abstraction,

alors, le système est :

- multimodal en entrée si $n_e \geq 2$, sinon il est multimédia en entrée,
- multimodal en sortie si $n_s \geq 2$, sinon il est multimédia en sortie.

La figure 2.8 traduit graphiquement les conditions nécessaires et suffisantes pour qu'un système soit, du point de vue technique, multimédia ou multimodal.

S'il était besoin de simplifier, la distinction technique entre le multimédia et le multimodal repose pour l'essentiel sur les capacités d'abstraction des canaux de communication.

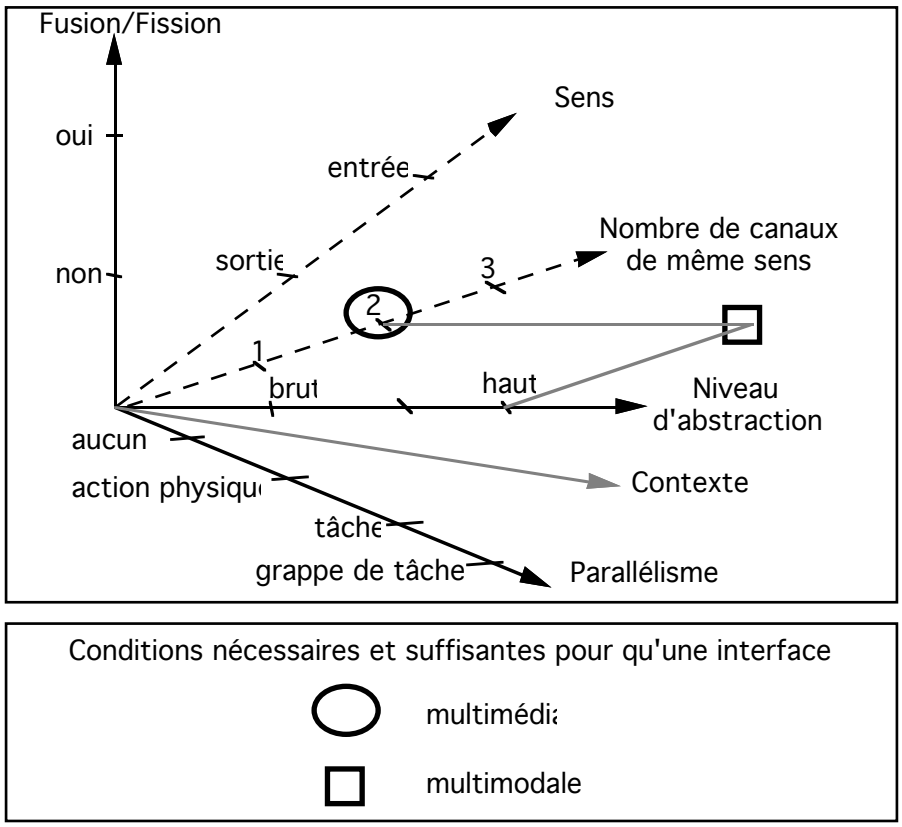


Figure 2.8 : Interfaces multimédia et multimodale au sein du référentiel MSM.

7.2.2. Lien avec le référentiel IHMM'91

Le référentiel IHMM'91 est le résultat d'un travail de groupe auquel nous avons participé lors des journées IHMM'91. Le mérite de cette analyse est d'avoir esquissé un tout premier cadre de réflexion. La figure 2.9 rapporte notre interprétation de ce travail. Trois dimensions y sont représentées : les niveaux d'abstraction, la fusion et l'usage des modalités. Les axes niveaux d'asbtraction et fusion correspondent aux axes de même nom dans MSM. L'usage des modalités se rapproche de la notion de parallélisme dans MSM.

Concernant le niveau d'abstraction, deux extrêmes ont été retenus : capacité du système à extraire du "sens" et son opposé, "pas de sens". L'axe fusion couvre la possibilité d'associer plusieurs sources d'information. L'absence de fusion s'appelle indépendance et la présence combinaison. L'usage des modalités exprime la disponibilité temporelle des modalités. Du point de vue de l'utilisateur, une modalité correspond à un canal sensori-moteur. Du point de vue du système, à une modalité est associé un ensemble de types d'information pouvant être captés par un canal humain. Par exemple, la vidéo et le graphique constituent deux types d'information pouvant être perçus par le canal visuel humain. L'usage de plusieurs de ces canaux pour communiquer avec un ordinateur peut être parallèle (simultané) ou séquentiel.

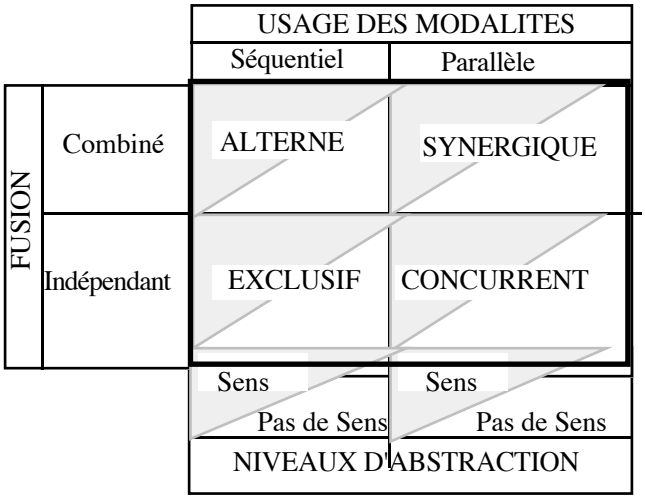


Figure 2.9 : Espace de référence pour système interactif à caractéristiques multiples.

A partir de ce référentiel, le groupe a convenu de définir quatre classes d'interfaces multimodales (sachant que "multimodal" implique la valeur "sens" pour l'axe "niveaux d'abstraction") : les interfaces multimodales alternées, synergiques, exclusives et concurrentes :

- alterné signifie qu'il y a fusion mais que les modalités doivent être utilisées en séquence,
- synergique implique fusion et usage simulatné de plusieurs modalités,
- exclusif est le plus pauvre des cas : absence de fusion et usage en séquence des modalités,
- concurrent correspond à l'usage simultané de plusieurs modalités mais sans q'il y ait fusion.

Cette présentation succinte du référentiel IHMM'91 permet néanmoins de constater qu'il s'obtient par projection de l'espace MSM en ayant fixé la direction et le nombre des canaux étudiés. Nous observons aussi que ce premier référentiel ne considérait pas le phénomène de fission ni la notion de contexte. Le parallélisme ne fait pas de distinction explicite entre les niveaux de granularité (action physique, tâche élémentaire, etc.). En particulier, les interfaces de type "alterné" et "synergique" supposent implicitement la fusion des informations au niveau des actions physiques. Les interfaces de type "exclusif" et "concurrent" sous-entendent, quant à elles, l'absence de fusion au niveau bas seulement mais nécessitent une fusion à d'autres niveaux d'abstraction de façon à rendre possible l'élaboration des commandes (que ce soit de manière séquentielle ou non). Ces points seront précisés au chapitre IV où nous donnerons une définition plus précise des termes alterné, synergique, etc.

Malgré son imprécision, nous avons vu dans le référentiel IHMM'91 des éléments utiles à la classification des systèmes. Nous rapportons ci-dessous le principe de nos idées que nous appliquerons à nouveau au chapitre IV sur un référentiel que nous estimons mieux assis.

7.2.3. Méthode de classification

Toute classification s'appuie sur un ensemble de traits. Un trait t_i est caractérisé par un poids p_i qui traduit une importance estimée du trait et une localisation l_i dans un espace. La définition formelle d'un trait s'exprime par le doublet :

$$t_i = (p_i , l_i)$$

La situation G d'un système dans un espace de référence correspond au centre de gravité de ses traits, ce qui donne :

$$\mathbf{G} = \frac{1}{\Sigma_p} \times \sum_i l_i \times p_i \qquad \Sigma_p = \sum_i p_i \qquad (1)$$

Nous avons choisi comme traits pertinents l'ensemble des commandes offertes par le système. Comme le montre la figure 2.10, nous avons défini quatre valeurs de poids qui traduisent la fréquence d'utilisation supposée des commandes.

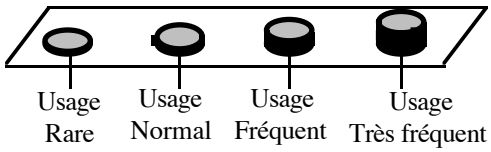


Figure 2.10 : Quatre valeurs de poids traduisant la fréquence d'utilisation supposée des commandes.

A titre d'exemple, considérons les commandes de NoteBook [Nigay 91a] réalisé sur NeXT avec Interface Builder [Webster 89] et le système Sphinx qui assure une reconnaissance multilocuteur de la parole continue [Lunati 91]. NoteBook permet de créer, d'éditer, de feuilleter et de détruire des notes de texte. Pour insérer une note entre deux autres, l'utilisateur dit "insérer une note" tout en désignant avec la souris le lieu d'insertion dans la liste de notes. Cette commande, qui implique la fusion parallèle de paroles et de gestes, est synergique et est utilisée fréquemment. Elle est représentée dans la figure 2.11 au moyen du symbole <1>. La commande d'édition de note (notée <2> dans la figure 2.11) est très fréquente et est exclusive : une seule modalité est accessible (frappe au clavier) et aucune autre commande ne peut être émise en parallèle. Il en est de même pour les commandes de type "tourner les pages" (symbole <3>) qui s'effectuent soit à la souris, soit à la voix, et qui n'autorisent pas d'autres commandes en parallèle. La dernière commande (symbole <4>), détruire le contenu du calepin, est rarement utilisée mais est également exclusive. Au total, la position G de NoteBook calculée au moyen

de la formule (1) se rapproche plutôt de celle d'un système de type exclusif que de celle d'un système synergique.

Un calcul similaire pour VoicePaint [Gourdol 91] révèle un système essentiellement synergique. VoicePaint est un éditeur de dessin que nous avons développé sur Macintosh. Il offre à l'utilisateur la possibilité de dessiner avec la souris tout en modifiant à la voix les caractéristiques du pinceau (peinture, épaisseur, luminosité). Les commandes de dessin étant les plus fréquentes, la position de VoicePaint est proche du point noté Synergique.

NoteBook et VoicePaint offrent uniquement des exemples de commandes synergiques et exclusives. Nous trouvons des exemples de commandes concurrentes dans le système VoiceFinder qui ajoute des commandes vocales au bureau électronique sur Macintosh. En effet, avec le système VoiceFinder, il est possible de dire "Vider la corbeille" tout en ouvrant un fichier en le sélectionnant avec la souris. Il s'agit là de deux commandes indépendantes exprimées en parallèle au moyen de modalités différentes.

Le système MMI2 [Wilson 91] supporte des commandes alternées. Dans ce système, la langue naturelle écrite est la modalité dominante[7]. Mais il est possible de définir des expressions déictiques dans une phrase écrite, telle que "ce site", en la faisant suivre d'une sélection avec la souris. Les modalités sont alors combinées mais les expressions sont acquises séquentiellement (avec MMI2, la phrase écrite puis la sélection faite avec la souris).

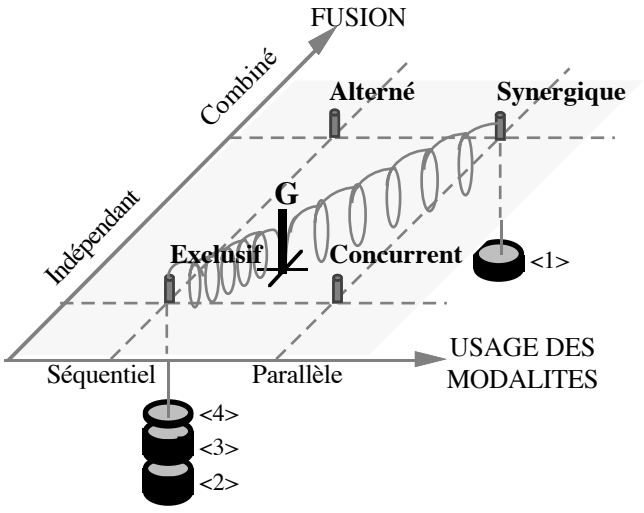


Figure 2.11 : Méthode de classification illustrée avec NoteBook.

7 Au chapitre VII, nos donnerons une définition précise de la notion de dominance.

Nous venons de démontrer une première utilité d'un espace de référence : qualifier un système. Une seconde retombée est la possibilité d'analyser l'utilisabilité du système au regard de ses modalités. L'utilisabilité est le potentiel à exploiter au maximum les services d'un système donné [Vainio-Larsson 90]. Celle-ci peut être évaluée en considérant le langage de commandes permis par le système, comme il est proposé dans [Vainio-Larsson 90]. Dès lors que le langage de commandes est considéré correct et approprié au domaine de la tâche, l'utilisabilité d'un système peut être évaluée au niveau des modalités offertes pour exprimer ces commandes. Par exemple, à l'utilisation du système réel, nous pourrions observer qu'une commande a un poids faible en position "synergique" et un poids fort en position "exclusif". Ceci pourrait signifier que les modalités à fusionner pour cette commande sont incompatibles ou encore que cette commande s'exprime plus naturellement au moyen d'une seule modalité.

8. CONCLUSION

Dans ce chapitre, nous avons présenté une revue de la terminologie et des cadres susceptibles de nous aider à structurer l'espace problème des interfaces nouvelles. Nous constatons que la terminologie est confuse et dépendante des objectifs. Les cadres quant à eux, sont soit limités par leur portée (entrées ou sorties, par exemple), soit généraux (couverture à la fois des entrées et des sorties) mais bornés par un point de vue restrictif (technologie versus utilisateur).

De ce bouillon conceptuel, nous retenons les éléments communs suivants : dans la communication, que l'on adopte le point de vue système ou utilisateur, il convient de distinguer le signifié du signifiant (selon l'école de Saussure), le représenté du représentant (selon l'expression de Bernsen) ou encore, selon notre approche, le concept de son expression. Signifié et représentant recouvrent l'un et l'autre notre idée de contenu conceptuel ou concept. Le concept concerne aussi bien les modèles mentaux humains que les représentations logicielles internes d'un système. Un concept est représenté, c'est-à-dire rendu perceptible, par un signifiant.

Un signifiant est une manifestation observable ou expression qui, selon Hemjslev, peut s'envisager comme une substance ou comme une forme, c'est-à-dire aussi bien sous sa matérialité que sous l'aspect de la structuration de ce matériau. La structuration définit les règles de dépendance qu'entretiennent les unités constitutives d'une expression. Nous pensons à langage d'interaction. La matérialité est liée aux mécanismes de création ou de réception d'énergie. Nous pensons aux dispositifs physiques d'entrée sortie qui en sont les producteurs ou les récepteurs.

En synthèse, nous prenons comme point de départ la dualité concept/expression. Pour le cas précis de la communication homme-machine, nous conviendrons qu'une expression s'étudie selon deux perspectives : sa structure qui dépend d'un langage d'interaction et sa substance qui dépend du dispositif physique support. Langages d'interaction et dispositifs physiques définissent les points de contact entre l'utilisateur et le système informatique. Ces notions nous permettent donc d'envisager la définition d'un cadre conceptuel qui recouvre à la fois les perspectives utilisateur et système. Sa description fait l'objet du chapitre suivant.

Dans le reste du mémoire, nous éliminons délibérémment les termes, média, modalité, mode. Lorsqu'ils seront utilisés, multimodal et multimédia devront être compris au sens technique formel de MSM. Nous parlerons désormais de langage d'interaction, de dispositif physique, et de système à caractéristiques multiples.

Chapitre III

Le modèle "Pipe-Lines"

"*What is needed is a common framework for describing the design space of interfaces...* Without *such a framework, it will remain difficult to interpret the progress already made in exploring this design space...*"
- David M. Frohlich -

1. INTRODUCTION

Nous venons de constater que les espaces de conception actuels sont trop restrictifs. Les uns, telles les propositions de Mackinlay et de Bernsen, sont limités par leur portée ; les autres, bien que couvrant à la fois les entrées et les sorties, sont centrés sur un point de vue étroit. En particulier, l'espace de Frohlich adopte une perspective utilisateur alors que MSM est régi par des considérations système. Avec notre modèle Pipe-Lines, l'objectif est de couvrir à la fois les entrées/sorties et les points de vue système et utilisateur. Ce modèle nous conduira aux affinements M2LD et O2LD présentés au chapitre suivant.

Le modèle Pipe-Lines est motivé par un double jeu de considérations indissociables. D'une part, nous voulons caractériser un système en termes de choix offerts à l'utilisateur : c'est la perspective utilisateur. Un espace de choix étant fixé, il faut ensuite identifier les requis logiciels correspondants : c'est la perspective du concepteur de logiciel. Les figures 3.1 et 3.2 illustrent ces deux points de vue complémentaires.

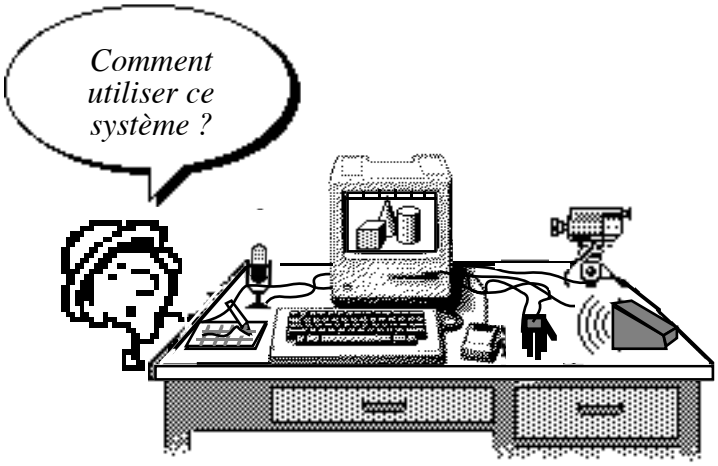


Figure 3.1 : Perspective utilisateur : "Comment utiliser ce système ?"

Considérant le point de vue utilisateur, à une commande par exemple, correspond, pour un système donné, un ensemble fini d'énoncés. La connaissance des énoncés possibles répond à la question qu'un utilisateur pourrait se poser (figure 3.1) : "Comment utiliser ce système ?" En sens inverse, l'interface du système définit l'ensemble des présentations possibles pour un concept donné. La perception de ces présentations peut solliciter plusieurs canaux sensoriels humains. La diversité des énoncés et des canaux perceptifs définissent un espace de choix caractérisants. Ainsi comparer deux systèmes A et B revient, selon le modèle Pipe-Lines, à répondre à la question : "en terme de communication, le système A offre-t-il plus de possibilités à l'utilisateur que le système B?"

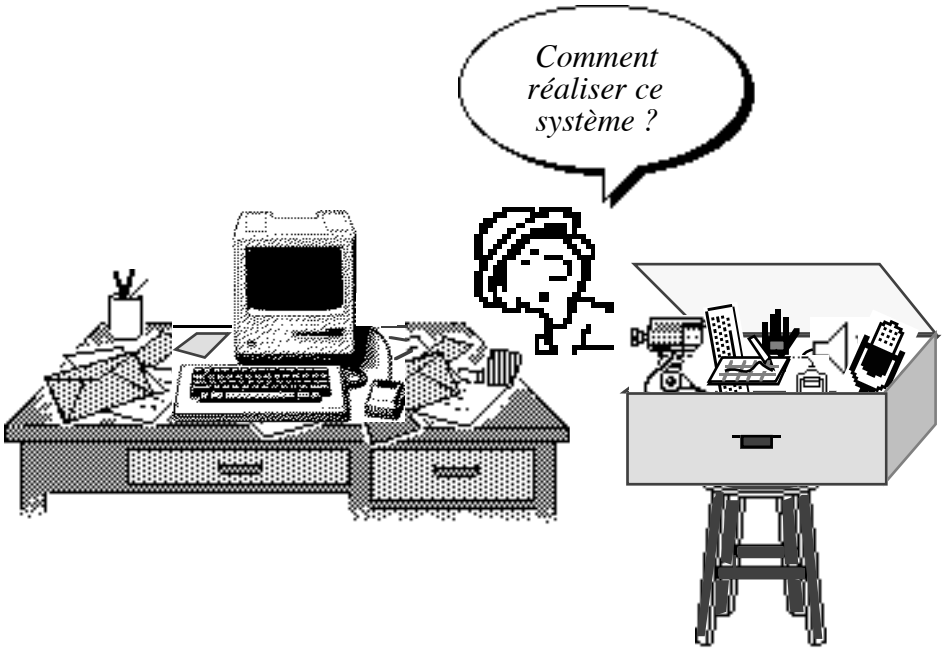


Figure 3.2 : Point de vue du concepteur : "Comment réaliser ce système ?"

Du point de vue du concepteur et du réalisateur de logiciel, la richesse de l'espace de choix offerts à l'utilisateur n'est pas sans conséquence. Et la question illustrée dans la figure 3.2, "Comment réaliser ce système ?" est légitime. MSM aborde ce point avec les notions de fonctions d'interprétation et de restitution. L'espace Pipe-Lines affine ces fonctions en étapes pertinentes qui se traduisent ensuite dans les architectures logicielles. On obtient ainsi un pont direct entre le modèle conceptuel Pipe-Lines et la conception d'une architecture idoine. Ces aspects seront traités aux chapitres VI et VII.

Le modèle Pipe-Lines repose, comme la théorie de Norman et le Modèle du Processeur Humain, sur le principe des flux d'information entre les acteurs communicants. A ce titre, il distingue deux plans d'analyse : le premier pour les entrées, le second pour les sorties. Ces plans font l'objet de la section 2. Alors que Norman détaille les activités mentales de l'utilisateur sous forme d'étapes et esquisse les traitements du système, le modèle Pipe-Lines offre une analyse complémentaire : les activités mentales y sont esquissées et les traitements informatiques y sont détaillés et organisés en étapes. Chaque étape est modélisée par une fonction aux paramètres spécifiques. Le rôle de chacune est présenté dans la section 3. La notion de contexte, déjà introduite de manière informelle par MSM, est l'un des ingrédients directeur des opérations des fonctions d'interprétation et de restitution. Nous en présentons une description plus formelle et complète dans la section 4. En 5, nous montrons comment le flux central pipe-line est en réalité détourné par des chemins complexes qui traduisent l'activité des dispositifs physiques, les retours d'information, le parallélisme et les phénomènes de fusion et de fission.

Nous étayerons la discussion au moyen d'exemples tirés le plus souvent de deux logiciels que nous avons développés : MATIS et NoteBook.

2. LES DEUX PLANS DU MODÈLE PIPE-LINES

Considérant un système en cours d'utilisation, il est possible d'identifier et d'analyser les traitements effectués par l'utilisateur et le système lorsqu'il y a transfert d'informations. Cette étude peut se faire en prenant :

- comme point de départ l'utilisateur du système : c'est le premier plan que nous présentons au paragraphe 2.1. Nous étudions les différentes étapes de traduction et de traitement ainsi que les entités intervenantes dans le passage des intentions de l'utilisateur aux effets produits par le système. Ce plan correspond au transfert d'informations de l'utilisateur vers le système ou interface en entrée ;

- comme point de départ le système : c'est le deuxième plan introduit au paragraphe 2.2. En partant des effets d'une action système, nous explicitons les traitements et les entités qui interviennent jusqu'aux états perceptibles. Ce deuxième plan correspond au transfert d'informations du système vers l'utilisateur ou interface en sortie.

Comme le soulignent Norman et bien d'autres, ces deux plans sont fortement corrélés. Au paragraphe 2.3 nous en montrons la dépendance.

2.1. Premier plan : des intentions de l'utilisateur aux effets produits par le système

L'utilisateur traduit une intention en actions physiques notées AP. Une fois exécutée, une action ou un ensemble d'actions peut être acquis puis interprété par le système pour former une unité informationnelle. A partir d'une unité informationnelle, le système déduit une action système AS dont l'exécution, par le système, provoque un effet. Dans l'exemple de la figure 3.3, une intention utilisateur Int_a se traduit par deux actions physiques, $Act\text{-}phys_i$ et $Act\text{-}phys_j$. $Act\text{-}phys_i$ intervient dans la construction de l'unité informationnelle $Unit\text{-}Info_u$ qui conduit à l'exécution de deux actions système aux effets respectifs Eff_α et Eff_β.

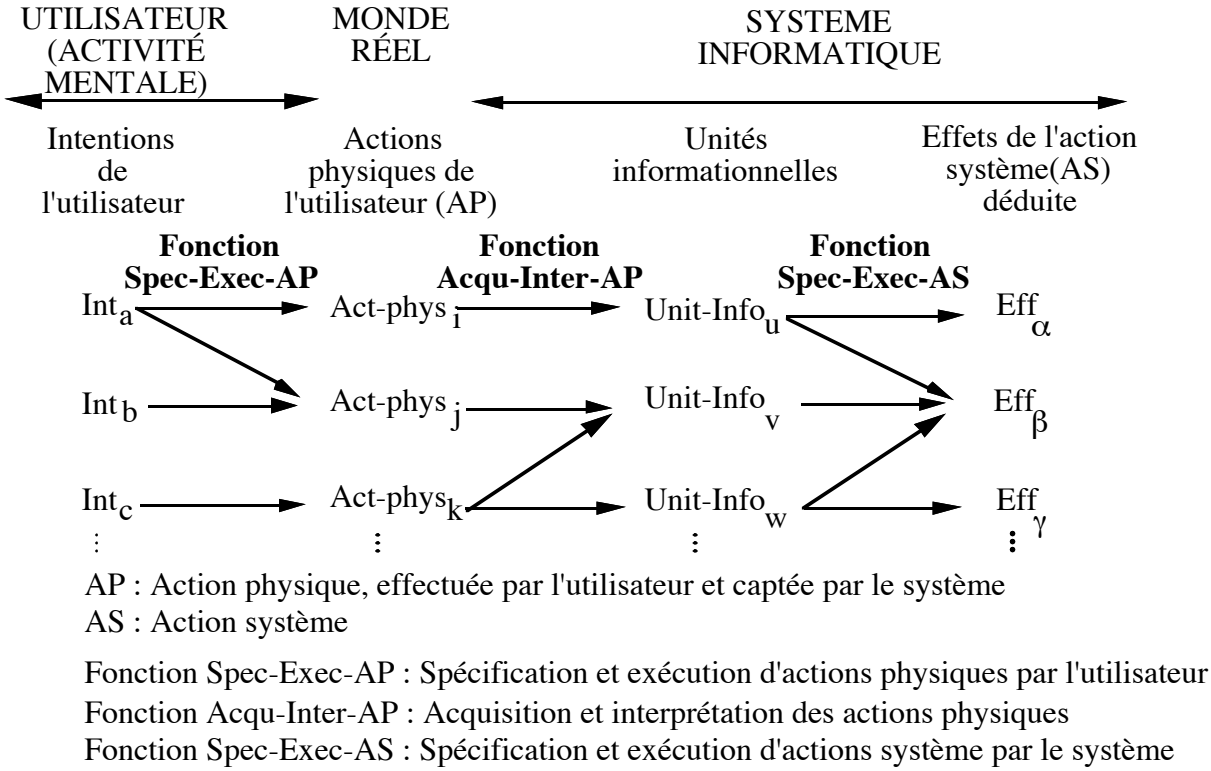


Figure 3.3 : Le premier plan du modèle Pipe-Lines : des intentions de l'utilisateur aux effets obtenus par le système.

Une *unité informationnelle* est une unité conceptuelle : elle regroupe un ensemble indivisible d'informations qui, dans le système, modélise un concept du domaine de la tâche. Parce qu'elle fait sens, une unité informationnelle est qualifiée de "représentation du niveau d'abstraction le plus haut". Une *action système* est une unité de traitement atomique : elle est indivisible. Elle ne peut donc être interrompue. Un *effet* est la différence entre l'état interne initial (c.-à-d. l'état interne avant l'exécution de l'action) et l'état interne final (c.-à-d. l'état interne après l'exécution de l'action). La figure 3.4 illustre de manière schématique le concept d'effet en relation avec les notions d'état et d'action système.

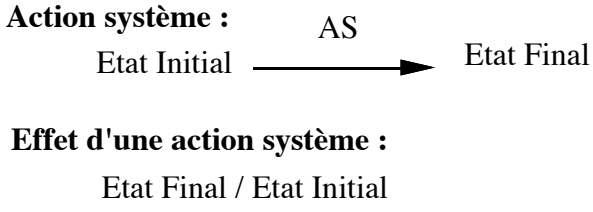


Figure 3.4 : L'exécution d'une action système et son effet en terme de changements d'état.

Les intentions, les actions, les unités informationnelles et les effets modélisent les catégories d'informations que manipulent l'utilisateur et le système dans leurs activités. Dans notre modèle, les activités du premier plan sont structurées sous forme de trois fonctions reliées en pipe-line :

- Spec-Exec-AP traduit une activité mentale. Elle consiste à spécifier une suite d'actions physiques correspondant à une intention donnée ;

- Acqu-Inter-AP et Spec-Exec-AS sont effectuées par le système. La première acquiert les actions physiques produites par l'utilisateur, les interprète pour constituer, lorsque c'est possible, des unités informationnelles. A partir d'unités informationnelles, la seconde spécifie une action système et en assure l'exécution.

La nature des paramètres impliqués dans le fonctionnement de ces fonctions est détaillée au paragraphe 3. Pour l'instant, nous illustrons les activités de ce premier plan, "des intentions aux effets", par l'exemple de la figure 3.5. Ici, l'utilisateur a pour intention de créer une nouvelle ligne dans un texte. En termes d'actions physiques, trois possibilités s'offrent à lui : la commande vocale *Insère une ligne après la ligne numéro 3* (action Act-phys$_i$), déplacer le curseur sur la ligne précédente et insérer un retour chariot (action Act-phys$_j$), ou bien accompagner l'acte vocal *insère une ligne ici* d'un clic souris (action Act-phys$_k$).

Dans l'exemple, les actions physiques Act-phys$_i$ et Act-phys$_k$, qui s'appuient sur le vocal, donnent naissance à une même unité informationnelle Unit-Info$_u$: *Insérer une nouvelle ligne* tandis que Act-phys$_j$ conduit à Unit-Info$_v$: *Insérer un retour chariot*. Deux effets sont associés à Unit-Info$_v$ selon l'état interne du système :

- Si l'état interne est "insertion", la fonction Spec-Exec-AS associe l'action système AS = <Ajout d'une ligne> dont l'exécution a pour effet <Nouvelle ligne dans le texte et point d'insertion modifié>.

- Si l'état interne est "réécriture", Spec-Exec-AS associe l'action système AS = <Changement de la position du point d'insertion> dont l'exécution a pour effet <point d'insertion modifié>.

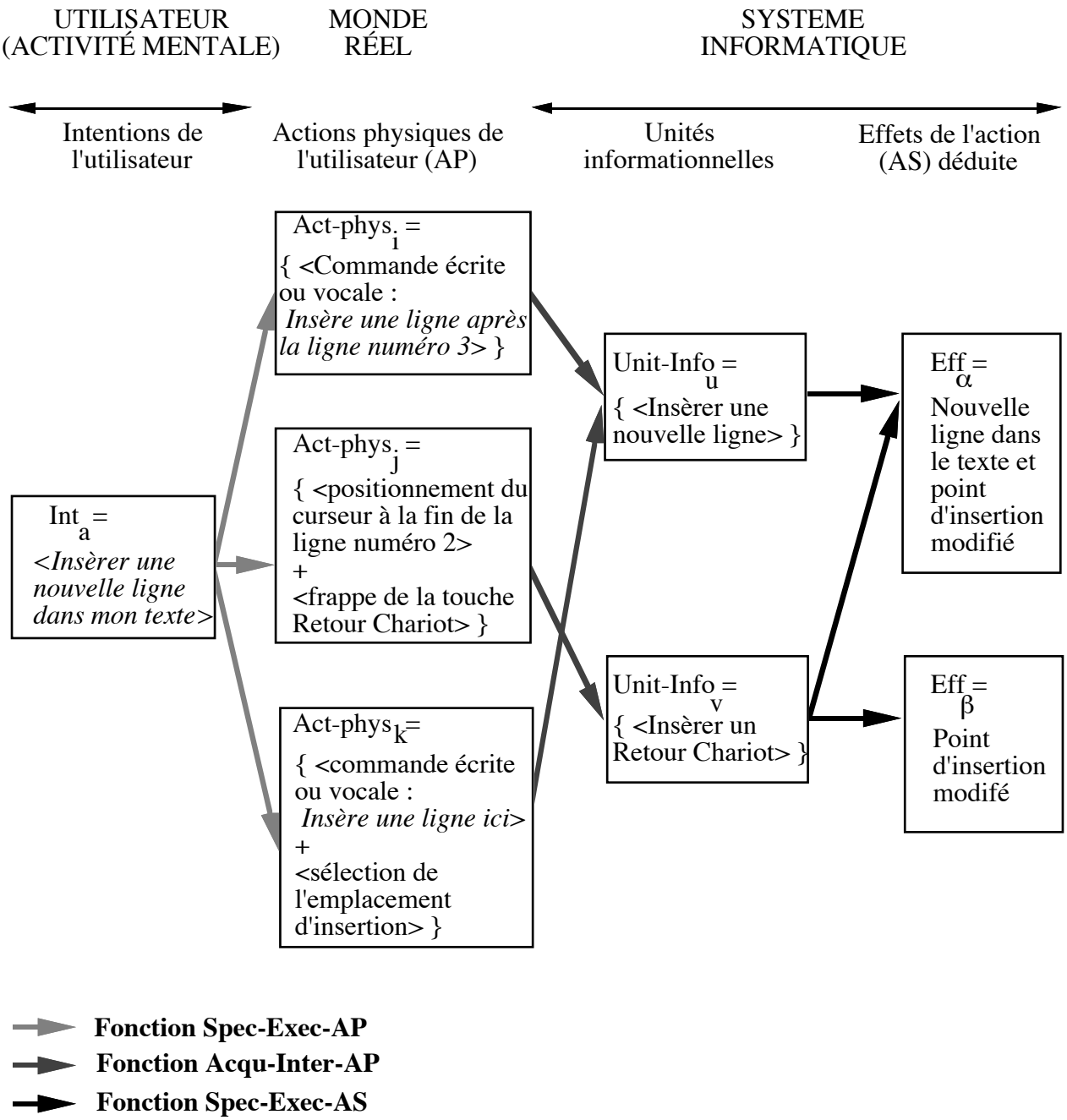
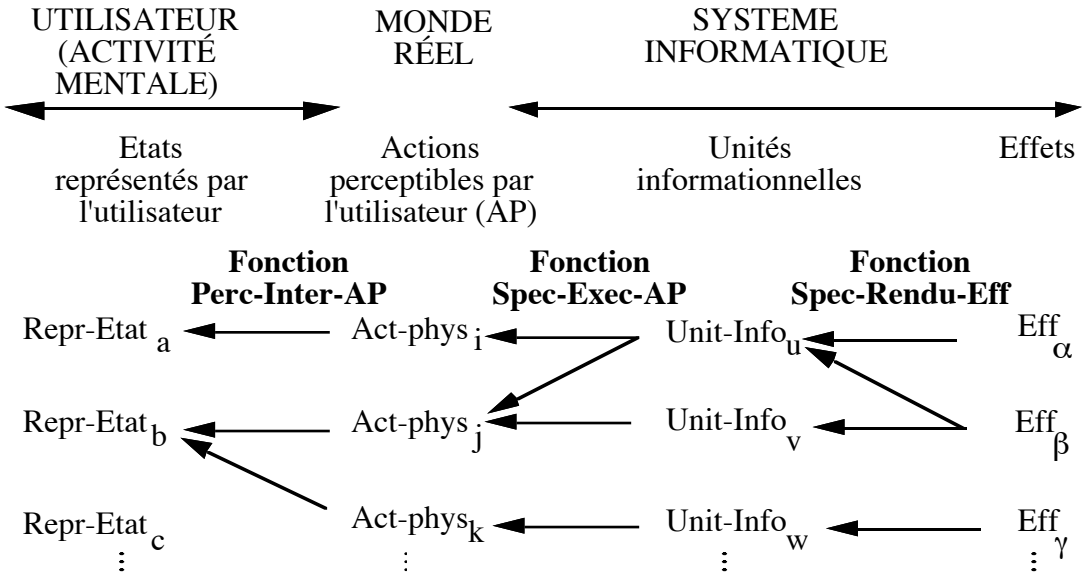


Figure 3.5 : Illustration des activités et des informations du plan “des intentions aux effets” : cas de l'insertion d'une nouvelle ligne dans un éditeur de texte.

Notre premier plan ou interface en entrée, explicite les phases qui permettent de passer d'une intention de l'utilisateur à l'effet d'une action système. Le deuxième plan, qui fait l’objet du paragraphe suivant, montre le cheminement inverse : comment l’effet se traduit-il en éléments perceptibles interprétables par l’utilisateur?

2.2. Deuxième plan : des effets aux états mentaux

Dans le deuxième plan, l'activité du système consiste à exprimer le changement de son état interne. Partant d'un effet, il choisit la ou les unités informationnelles aptes à représenter l'effet en fonction de la situation. Cet ensemble d'unités informationnelles est traduit en actions physiques réalisées cette fois-ci par le système. L'utilisateur est alors en mesure de percevoir puis d'interpréter ces actions pour élaborer une représentation mentale de l'état et/ou du fonctionnement du système. Dans l'exemple de la figure 3.6, l'effet Eff_{α} est exprimé par l'unité informationnelle $Unit\text{-}Info_u$ qui, à son tour, conduit le système à produire les actions physiques $Act\text{-}phys_i$ et $Act\text{-}phys_j$. La perception et l'interprétation de $Act\text{-}phys_i$ par l'utilisateur conduit à une représentation $Repr\text{-}Etat_a$ de l'effet Eff_{α}.



AP : Action physique, produite par le système et perçue par l'utilisateur
AS : Action système

Fonction Perc-Inter-AP : Perception et interprétion des actions physiques
Fonction Spec-Exec-AP : Spécification et exécution d'actions physiques
Fonction Spec-Rendu-Eff : Spécification du rendu de l'effet

Figure 3.6 : Le deuxième plan du modèle Pipe-Lines : des effets système aux états mentaux utilisateur.

Comme pour le premier plan, les activités du deuxième plan sont structurées sous forme de trois fonctions reliées en pipe-line :

- Spec-Rendu-Eff et Spec-Exec-AP sont effectuées par le système. La première détermine les unités informationnelles adaptées au rendu de l'effet. La seconde

consiste à spécifier la suite d'actions physiques correspondant au rendu perceptible de l'unité informationnelle et en assure l'exécution.

- Perc-Inter-AP traduit l'activité mentale de perception et d'interprétation des actions physiques système en une représentation mentale de l'état et/ou du fonctionnement du système.

En reprenant l'exemple de la figure 3.5, à l'effet Effβ=<emplacement du point d'insertion modifié>, la fonction Spec-Rendu-Eff fait correspondre l'unité informationnelle {<localisation point insertion = l>}. La valeur l désigne un emplacement dans le système de coordonnées abstrait du document. La fonction Spec-Exec-AP traduit le concept de point d'insertion en un objet de présentation physique appelé curseur dont le positionnement p dans l'écran est déduit de l. Le curseur sous forme de trait vertical clignotant en l'emplacement p, est le résultat de l'exécution des actions physiques par Spec-Exec-AP.

Il faut noter que l'effet n'est pas nécessairement la conséquence d'une action système déclenchée par l'utilisateur. L'application peut en effet être génératrice d'actions système dont l'exécution provoque un effet à rendre perceptible. C'est le cas des systèmes asynchrones (qualifiés également de dynamiques) tels les systèmes de contrôle de processus [Coutaz 91]. Nous en donnons un exemple à la figure 3.7. Nous considérons que la température d'un fluide est passée de 40 à 60 degrés Celsius et que l'utilisateur doit être averti de ce changement.

Le concept utile à la tâche de surveillance a trait à la température. Comme le montre la figure 3.7, les unités informationnelles possibles sont :

- la nouvelle température,
- la variation de température.

L'unité informationnelle une fois fixée par la fonction Spec-Rendu-Eff, le système détermine son rendu perceptible en termes d'actions physiques (fonction Spec-Exec-AP). Pour chacune des unités informationnelles, les actions physiques envisagées sont soit la génération d'un message vocal, soit l'affichage à l'écran d'un thermomètre, soit les deux. L'utilisateur est alors en mesure de percevoir et d'interpréter les actions physiques (fonction Perc-Inter-AP). De cette interprétation, il déduit le nouvel état du système.

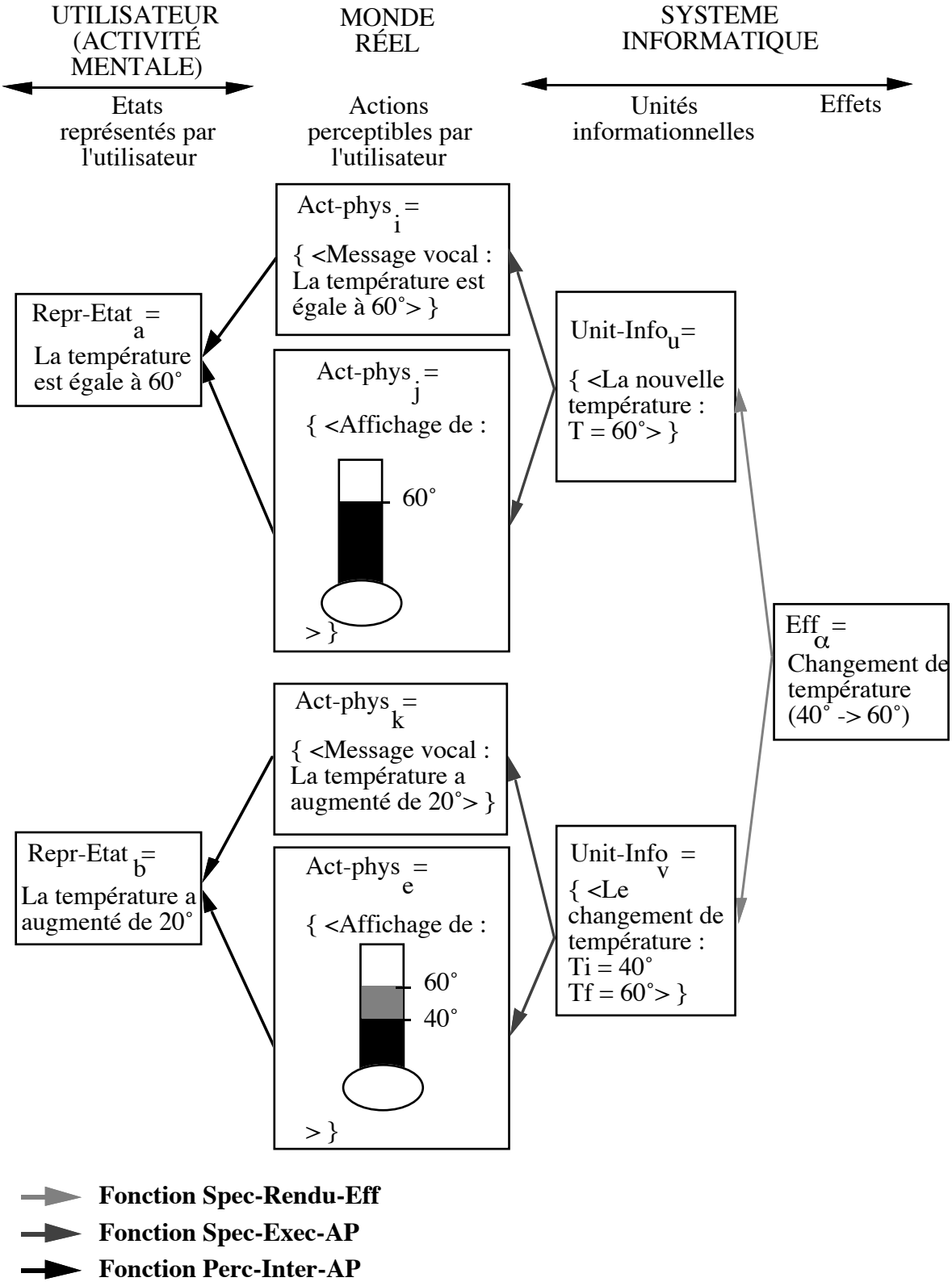


Figure 3.7 : Illustration des activités et des informations du plan “des effets aux états représentés” Le changement de température dans un système de contrôle de processus industriel.

Nous venons de présenter les plans du modèle Pipe-Lines de manière indépendante. Nous les mettons en relation au paragraphe suivant.

2.3. Relations entre les deux plans du modèle Pipe-Lines

La figure 3.8 rend explicites les relations entre les deux plans du modèle Pipe-Lines. Chacun d'entre eux est structuré en trois fonctions qui manipulent des données de même niveau d'abstraction. Ce constat conduit à grouper les fonctions en couple de fonctions symétriques. Par exemple, <Acqu-Inter-AP (premier plan), Spec-Exec-AP (deuxième plan)> représente un tel couple : la première fonction associe à une action physique utilisateur une unité informationnelle ; réciproquement, la deuxième fonction associe à une unité informationnelle une action physique système. La même observation tient pour <Spec-Exec-AS (premier plan), Spec-Rendu-Eff (deuxième plan)> et <Spec-Exec-AP (premier plan), Perc-Inter-AP (deuxième plan)>.

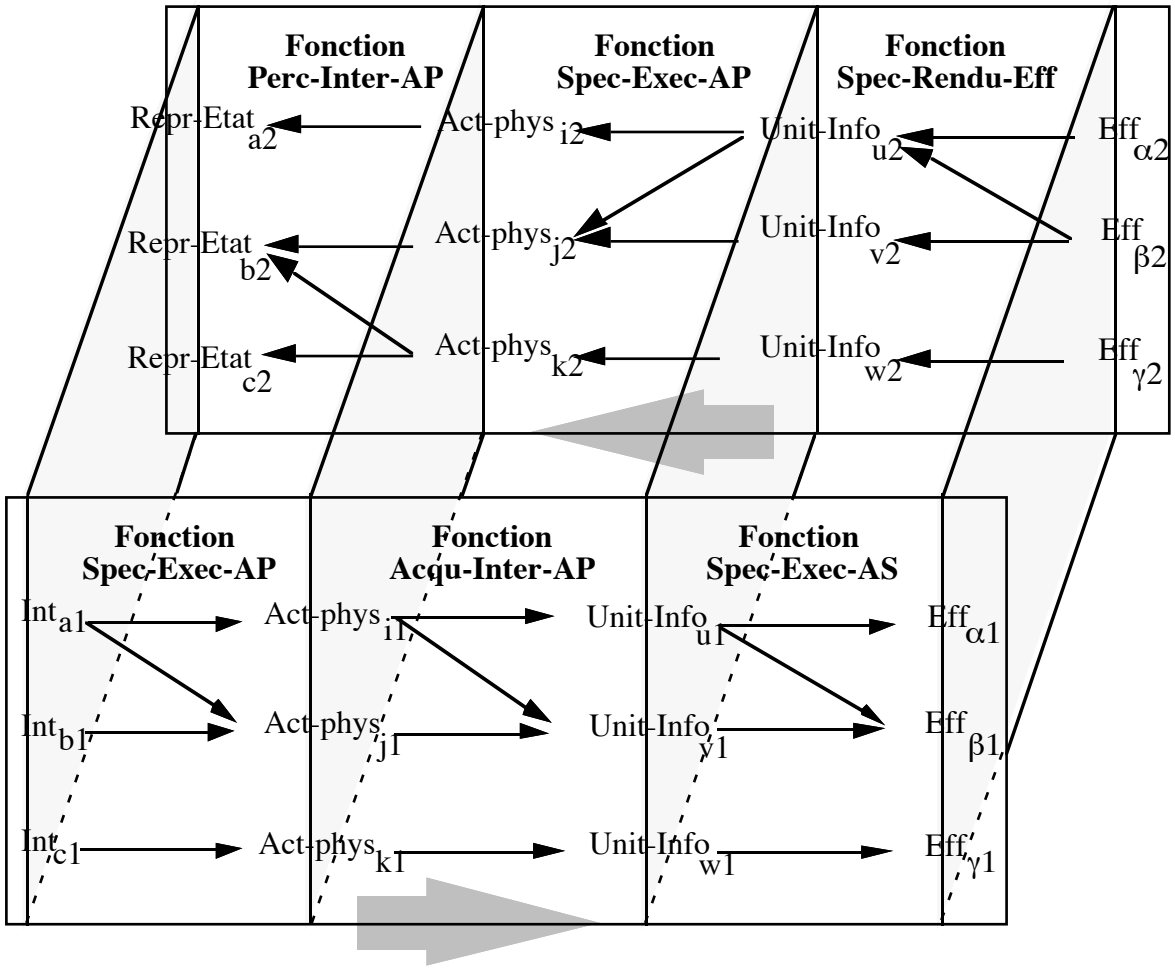


Figure 3.8 : Les deux plans parallèles de notre espace, interface en entrée et interface en sortie.

Les couples de fonctions symétriques explicitent un premier lien entre les plans qu'illustre la figure 3.9. Les ensembles de données communes ou de même niveau d'abstraction, matérialisés par des plans perpendiculaires aux plans des interfaces d'entrée et de sortie de la figure 3.8 illustrent un deuxième lien. Les couples de fonctions symétriques et les données communes montrent la corrélation entre les entrées et les sorties. Nous reviendrons sur ce point au paragraphe 5 qui détaille le flux d'informations entre les plans. Pour l'instant, nous devons identifier les éléments intervenant dans le fonctionnement des fonctions du modèle Pipe-Lines.

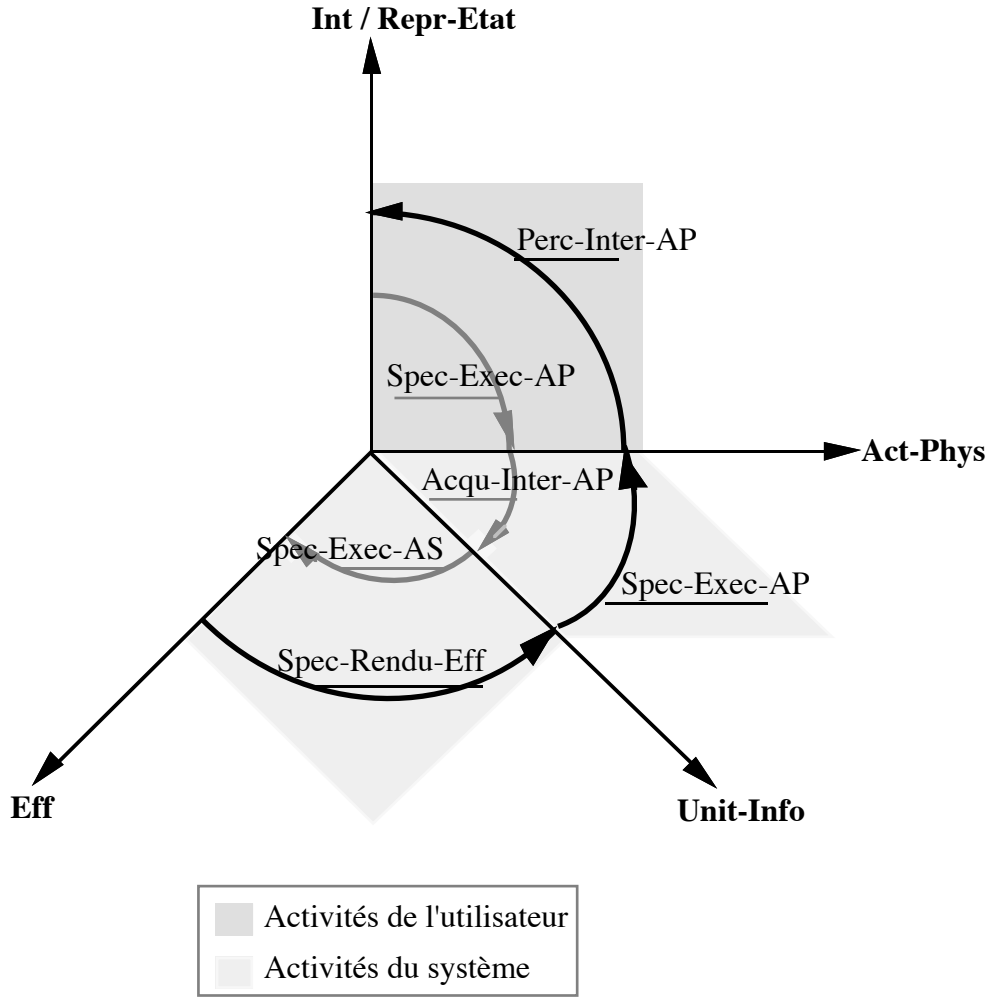


Figure 3.9 : La réciprocité des fonctions des deux plans. Chaque axe correspond à un ensemble de données et chaque fonction est représentée par une rotation d'un axe sur un autre. Ainsi une fonction et sa réciproque sont représentées par deux rotations : l'une d'un axe A vers un axe B et l'autre inverse de l'axe B vers l'axe A.

3. LES FONCTIONS DES PLANS DU MODÈLE PIPE-LINES ET LEURS PARAMÈTRES

Ayant présenté le canevas et le rôle des composants du modèle Pipe-Lines, nous devons étudier les paramètres directeurs des fonctions. Comme le montre la figure 3.10, ces fonctions peuvent être regroupées par couple selon leur relation de symétrie. A chacun de ces couples correspondent des paramètres et notamment le langage d'interaction et le dispositif physique.

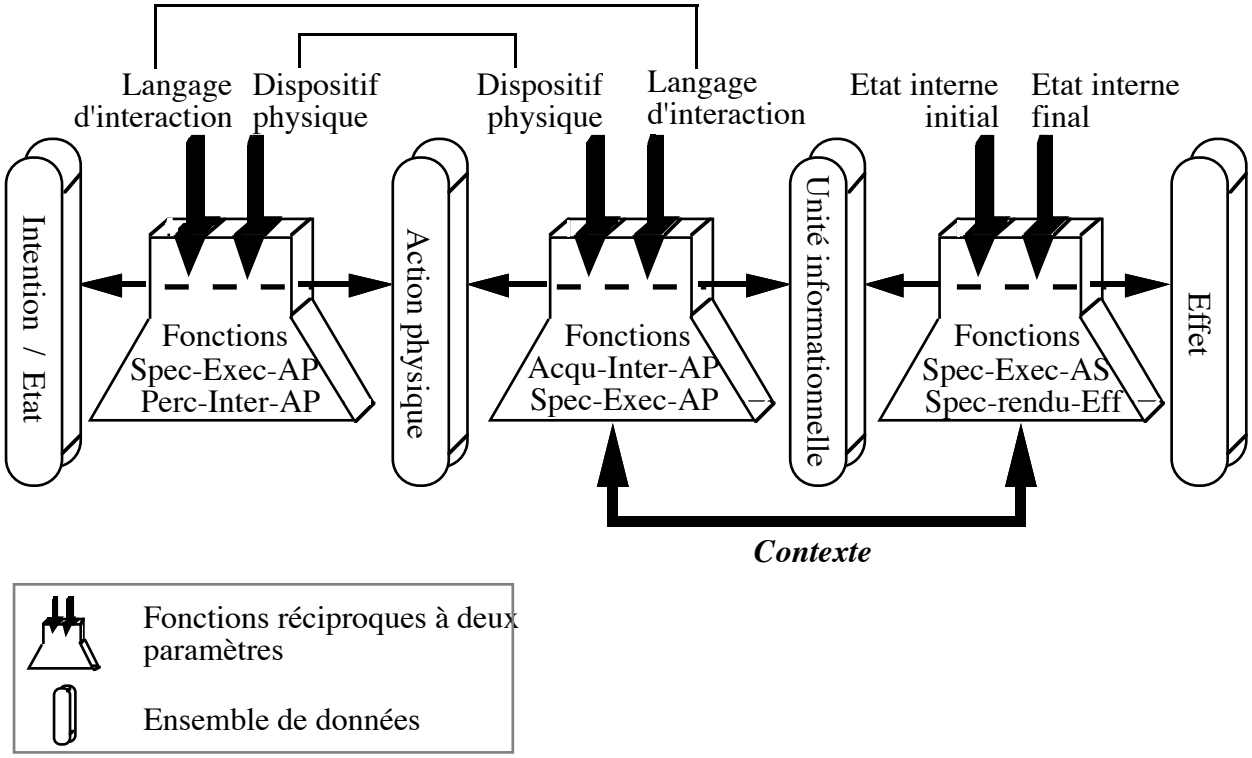


Figure 3.10 : Les couples de fonctions réciproques et leurs paramètres.

3.1. Entre représentation mentale et actions physiques (Spec-Exec-AP et sa réciproque (Perc-Inter-AP)

Les activités mentales de production et d'interprétation d'une expression sont, nous l'avons vu, modélisées respectivement par les fonctions Spec-Exec-AP et Perc-Inter-AP. A leur tour, ces fonctions se décomposent en deux étapes marquée chacune par un paramètre de traitement. Pour Spec-Exec-AP, la première étape est dirigée par le choix du langage d'interaction, la seconde par le choix du dispositif physique. Pour Perc-Inter-AP, la première phase, dite de perception, est concernée par les dispositifs physiques tandis que la seconde, l'interprétation, s'appuie sur la connaissance d'un langage d'interaction.

Nous appelons *langage d'interaction* un système conventionnel structuré de signes qui assure une fonction de communication entre un système informatique et un utilisateur. De manière plus formelle, un langage d'interaction se définit par un vocabulaire d'éléments terminaux et une grammaire. Un *dispositif physique* est un transducteur de propriétés physiques : un dispositif d'entrée capte des mouvements de l'environnement (et notamment les actions de l'utilisateur) ; un dispositif de sortie produit des mouvements perceptibles par l'environnement (et notamment par l'utilisateur).

Le langage d'interaction et le dispositif physique forment les deux facettes complémentaires d'une expression : le langage définit la structure c'est-à-dire les relations de dépendance qu'entretiennent les constituants de l'expression ; le dispositif définit la réalité physique observable. Si l'on adopte la terminologie de Hejmslev, le langage d'interaction définit la forme d'une expression tandis que le dispositif physique en définit la substance. Pour Norman, la puissance d'expression[8] d'un langage d'interaction est corrélée aux distances sémantiques d'exécution et d'évaluation tandis que les propriétés d'un dispositif physique sont liées aux distances articulatoires. Rappelons que chez Frohlich, un langage d'interaction est appelé style et que les styles sont regroupés en deux modes : les modes langage et action. Notons que notre notion de dispositif d'entrée est identique à celle de Mackinlay ; celle de dispositif de sortie couvre la notion de média chez Bernsen.

Pour illustrer la fonction Spec-Exec-AP, nous considérons un exemple tiré de notre système MATIS. L'intention de l'utilisateur est de spécifier une requête concernant les vols de Pittsburgh à Boston. Une étape concerne le choix du langage d'interaction : langage naturel[9], manipulation directe de la liste des villes, ou encore remplissage d'un formulaire de requête impliquant un vocabulaire particulier (par exemple, Pittsburgh s'écrit PIT). Une autre étape est le choix des dispositifs d'entrée. En particulier, une expression en langage naturel, telle "*flights from Pittsburgh to Boston*", peut être écrite au moyen du clavier ou énoncée oralement par l'intermédiaire du microphone. La figure 3.11 résume les options possibles.

---

8 La puissance d'expression répond aux deux questions : "*Is it possible to say what one wants to say in this language?*" et "*Can the things of interest be said concisely?*" [Norman 86 page 100].

9 Nous appelons "langage naturel" toute langue pseudo-naturelle : une langue naturelle contrainte par une grammaire déterministe sans support pragmatique.

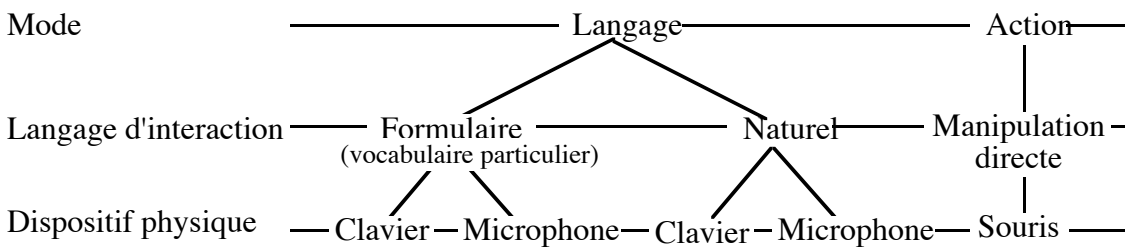


Figure 3.11 : MATIS et les options possibles pour spécifier un vol de Pittsburgh à Boston.

Pour des raisons analytiques, nous venons de présenter la fonction Spec-Exec-AP comme une suite de choix ordonnés : choix du langage puis choix du dispositif physique. L'activité mentale ne suit pas un canevas aussi rigide. Elle adopte au contraire des schémas opportunistes dirigés par la compétence ou par des facteurs contextuels. Par exemple, dans un environnement bruité où la robustesse du système de reconnaissance de la parole est mise en péril, on évitera les énoncés oraux. Un virtuose du clavier aura choisi en premier le clavier avant de déterminer son langage. Notons enfin que ces choix ne résultent pas nécessairement d'un plan conscient. Comme l'indique Rasmussen, il convient de distinguer trois niveaux de comportement cognitif : le niveau expert (ou skill-level) de type stimuli-réponse, le niveau des procédés (ou rule-based level) qui associe de manière consciente une procédure prête à l'emploi à une situation-problème, le niveau du savoir (ou knowledge-based) mis en jeu lorsqu'aucune procédure ne répond à la situation [Rasmussen 86].

La fonction Perc-Inter-AP se décompose elle aussi en deux phases. La première phase correspond à la perception des actions physiques produites par le système. Elle consiste donc à capter l'information brute et à la représenter mentalement. Le système de représentation est lié à la substance de l'expression. Et cette dernière est liée aux canaux sensoriels humains. Barnard [Barnard 93] identifie, dans son modèle cognitif humain, ICS (Interactive Cognitive Subsystems), trois sous-systèmes périphériques dédiés à la fonction de perception : le sous-système acoustique (AC), le sous-système visuel et le sous-système corporel (Body State). Chacun d'eux capte l'information et la transforme en un code mental correspondant à un système de représentation. La deuxième phase de la fonction Perc-Inter-AP consiste à déterminer le sens des informations captées et codées. Un exemple illustrant cette phase d'interprétation dans les sous-systèmes centraux (morpholexical, objet, propositionnel et implicationnel) est présenté dans [Barnard 93].

En résumé :

Les fonctions Spec-Exec-AP et Perc-Inter-AP traduisent **des activités mentales** et impliquent :

• le choix **du/des langages d'interaction**,

• le choix **du/des dispositifs physiques** (déterminant les actions physiques à effectuer ou à percevoir).

Nous venons d'analyser les paramètres des fonctions qui relèvent de l'activité de l'utilisateur. Nous étudions maintenant le cas des fonctions à la charge du système.

3.2. Entre actions utilisateur et unités informationnelles (Acqu-Inter-AP et sa réciproque Spec-Exec-AP)

La fonction Acqu-Inter-AP, qui permet au système de passer des actions utilisateur aux unités informationnelles, comprend une phase d'acquisition et une phase d'interprétation. L'acquisition des actions est assurée par un mécanisme de capture et de représentation de propriétés physiques observables : les dispositifs physiques d'entrée que l'utilisateur sollicite (de manière explicite ou non). Les dispositifs sollicités constituent le paramètre premier de Acqu-Inter-AP. Dans MSM, ils correspondent à la notion de canal numérique. Sans viser la comparaison anthropocentrique mais à titre pédagogique, les représentations internes de la phase d'acquisition sont au système informatique ce que les représentations des sous-systèmes périphériques sont au modèle ICS.

En phase d'interprétation, Acqu-Inter-AP cherche à associer une signification aux données produites par le processus d'acquisition. Acqu-Inter-AP s'appuie pour cela sur un deuxième paramètre : le langage d'interaction de l'utilisateur. L'identification de ce langage peut être complexe. Cette tâche est le plus souvent simplifiée en la déléguant à l'utilisateur. Par exemple, les bornes interactives imposent comme première tâche de spécifier le langage naturel de communication. Dans MATIS, la fenêtre active identifie sans ambiguïté le langage d'interaction : il existe une fenêtre de saisie des expressions en langage naturel ou des fenêtres de formulaires de requête. Mais la désignation de la fenêtre active revient à l'utilisateur.

Le langage d'interaction connu, Acqu-Inter-AP est en mesure de traduire les données de la phase d'acquisition en unités informationnelles. L'exemple qui suit, tiré de notre système NoteBook, illustre le rôle du langage d'interaction sur la construction des unités

informationnelles. Nous augmentons NoteBook en sorte que l'utilisateur puisse dicter le contenu d'une note. Nous nous plaçons dans la situation où l'utilisateur vient de produire oralement l'énoncé : "*Insert a new note*". NoteBook doit alors déterminer :

- (1) si la phrase est une commande de création de note,
- (2) ou si la phrase est un élément du contenu d'une note.

Ce choix revient à déterminer le langage d'interaction. Comme le montre la figure 3.12, l'arbre syntaxique d'interprétation et l'unité informationnelle qui en résulte dépendent de ce langage. Selon le langage d'interaction, les unités linguistiques de l'énoncé sont comprises comme les éléments d'une commande de création de note (arbre syntaxique 1), ou au contraire sont interprétées comme un tout indivisible pour former le paramètre "contenu" d'une commande d'ajout de texte dans la note courante (arbre syntaxique 2).

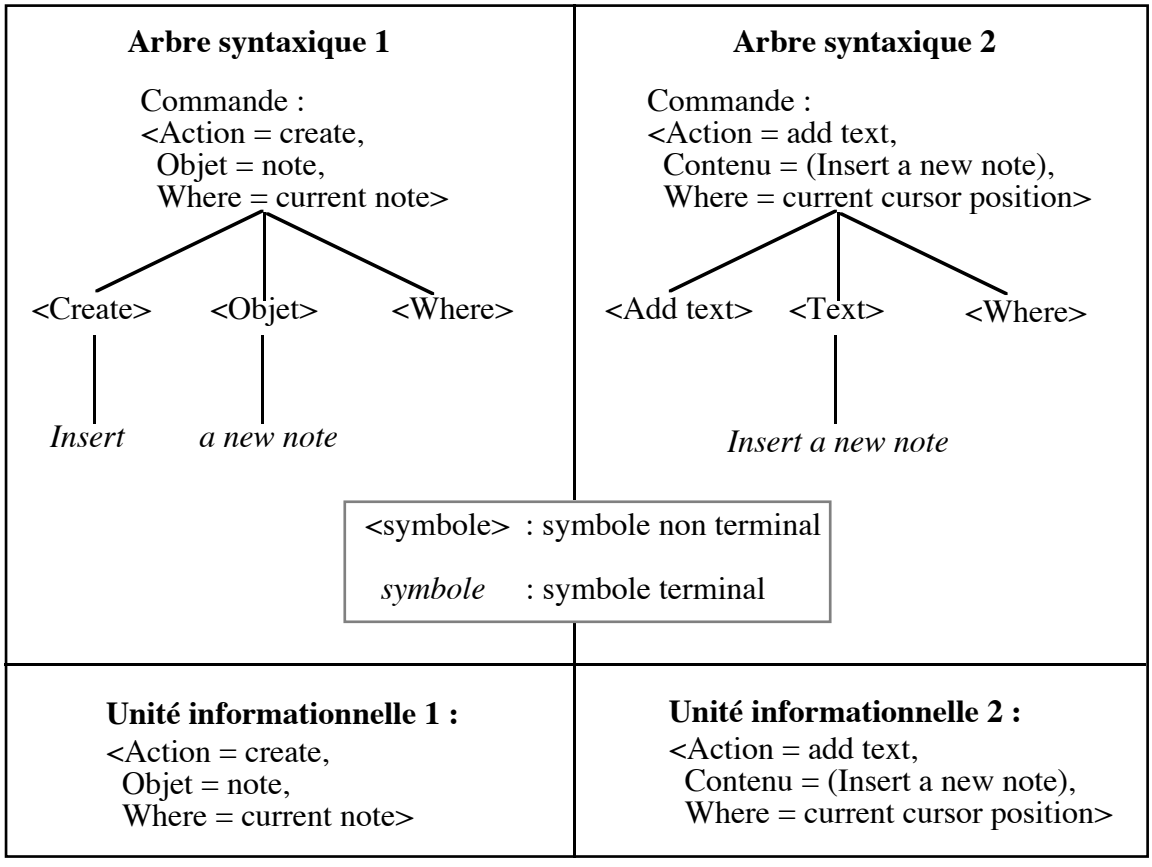


Figure 3.12 : Acquisition et interprétation de la phrase dictée "*Insert a new note*" dans NoteBook augmenté.

Dans l'autre sens, la fonction Spec-Exec-AP a la charge de rendre perceptible des unités informationnelles. Comme sa fonction symétrique, elle admet en paramètres un (ou plusieurs)

langage(s) d'interaction et des dispositifs physiques de sortie. L'association <langage, dispositif> correspond à la notion de modalités représentationnelles de Bernsen. Le choix du langage d'interaction définit la première phase de Spec-Exec-AP. Le choix du dispositif et son pilotage, correspond à la seconde étape.

Les choix des langages d'interaction et des dispositifs de sortie font intervenir au moins trois critères : les propriétés intrinsèques des modalités représentationnelles, le contenu sémantique de l'unité informationnelle, le contexte. Nous avons présenté au chapitre II une taxonomie des modalités de sortie que Bernsen caractérise en termes de profil et de média. Nous avons à cette occasion fait une rapide revue des propriétés intrinsèques que couvre le profil d'une modalité représentationnelle et qui correspond à notre notion de langage d'interaction de sortie.

Concernant le contenu sémantique de l'unité informationnelle, il faut citer les travaux de J. Brooke et K. Ducan [Brooke 80] qui montrent que si l'information à présenter correspond à des relations entre concepts, il est préférable de choisir une représentation graphique ; inversement, si l'information est un concept isolé, une représentation textuelle semble mieux appropriée. Si l'information à présenter correspond à un cas d'alerte, alors la restitution ne devra être ni éphémère ni inévitable.

COMET [Feiner 91] est un exemple de système qui à partir du contenu sémantique d'une unité informationnelle, en l'occurrence une solution pour réparer une radio, génère automatiquement une représentation graphique en trois dimensions accompagnée d'un texte pour expliquer les manipulations sur la radio.

Le contexte fera l'objet du paragraphe 4. Citons pour l'instant les préférences de l'utilisateur ou l'intention communicationnelle, deux constituants du contexte, comme point de départ à la génération automatique de présentations multimodales. Par exemple, le système expert de Gargan [Gargan 88] fait intervenir les préférences de l'utilisateur. Le système WIP [André 93], les travaux de Arens et Hovy [Arens 93] ou le projet Intuitive [Sutcliffe 93] s'appuient sur la notion d'intention qu'ils modélisent avec RST (Rhetorical Structure Theory) [Mann 88]. Basé sur

RST permet d'exprimer des relations entre les "constituants du discours" (c'est-à-dire, dans notre terminologie, entre les unités informationnelles). Par exemple, Arens et ses coauteurs proposent les relations d'élaboration et d'équivalence. L'élaboration relie un concept de base à des informations (concepts) qui permettent d'en savoir plus sur le concept noyau. L'équivalence vaut entre deux descriptions équivalentes d'un même concept. Tous ces systèmes fonctionnent selon notre modèle en deux étapes : choix des unités informationnelles puis choix

des restitutions physiques. Et chacune de ces phases vise à satisfaire à la fois puissance d'expression, efficacité et adaptativité.

Au-delà des critères de puissance et d'efficacité, les choix adoptés en sortie doivent être cohérents avec ceux offerts pour les entrées. Il est bien connu par exemple que le langage naturel adopté par les utilisateurs est le reflet du langage naturel de sortie du système [Salber 93]. En conséquence, si le système s'exprime de manière sophistiquée, l'utilisateur s'attend à disposer du même pouvoir d'expression en entrée. Si tel n'est pas le cas, il devra gérer deux modèles de représentation de l'interface avec à la clef une surcharge cognitive. De même, dans l'exemple de la figure 3.7, si l'utilisateur modifie la température par manipulation directe du thermomètre, alors il convient que le changement d'état se traduise par l'affichage de la nouvelle température sur le thermomètre. Il serait extravagant d'utiliser un autre langage et support tel effacer le thermomètre et produire le message vocal *la nouvelle température est 20°*.

En résumé :

Les fonctions Acqu-Inter-AP et Spec-Exec-AP correspondent à **des traitements effectués par le système** et impliquent :

- le choix **du/des langages d'interaction**,
- le choix **du/des dispositifs physiques**

Les deux paramètres de la fonction Spec-Exec-AP impliquent ceux de la fonction Perc-Inter-AP exposé au paragraphe précédent.

3.3. Entre unités informationnelles et effets (Spec-Exec-AS et sa réciproque Spec-Rendu-Eff)

Dans sa première phase, Spec-Exec-AS détermine la ou les actions système en fonction des deux paramètres d'entrée : unité informationnelle et contexte. La deuxième phase consiste en l'exécution de cette ou de ces action(s) avec, pour conséquence, un changement d'état interne.

Les concepts d'action système et d'état interne servent de fondements aux modèles formels de description d'interfaces [Harrison 90]. Ces modèles, tel que PIE [Dix 91], ont pour objectif la vérification automatique de propriétés. Dans [Abowd 92], nous organisons ces

propriétés en facteurs et critères et les situons dans le cadre général de l'utilisabilité. Par exemple, la robustesse, qui participe à l'utilisabilité d'un système, inclut les propriétés d'accessibilité et de réparabilité.

A titre illustratif, dans PIE, la phase d'exécution d'une action système se décrit par la fonction *doit*. *doit* permet de spécifier formellement les propriétés d'accessibilité et de réparabilité. Elle est définie sur l'ensemble des états E et l'ensemble des actions P :

$$doit : E \times P \rightarrow E$$

• Un système vérifie la propriété d'*accessibilité* si à partir de n'importe quel état, il est possible d'atteindre n'importe quel autre état. Ceci se traduit formellement avec le modèle PIE par :

$\forall e, e' \in E \bullet (\exists p \in P \bullet doit(e,p) = e')$

• La propriété de *réparabilité* est un cas particulier de la précédente. En effet c'est le cas où l'on souhaite atteindre l'état que l'on avait atteint avant l'exécution de la dernière action. La situation est simple lorsque cela est possible par une seule action système notée défaire[10], ce qui se traduit formellement par :

$\forall p \in P \bullet doit(e, p \cap \text{défaire}) = e$ avec $\cap$ qui signifie "suivi de".

Si l'on note e_X l'état atteint à partir de l'état e et de l'action x, $e_X = doit(e,x)$, nous obtenons :

$doit(e_X, \text{défaire}) = doit(e, x \cap \text{défaire}) = e$

Si l'on est dans l'état courant e, l'action défaire est alors égale à :

$doit(e, \text{défaire}) = doit(e_X, \text{défaire} \cap \text{défaire}) = e_X$

sachant que $e = doit(e_X, \text{défaire})$

La dernière équation formalise le fait que la propriété est récursive et s'applique aussi à l'action défaire.

Réciproquement la fonction Spec-Rendu-Eff associe une unité informationnelle à présenter à l'utilisateur, selon l'effet de l'exécution d'une action système. La première phase consiste à déterminer l'ensemble des concepts concernés par l'action système à partir des états internes initial et final. Ensuite parmi ces concepts, le système définit le contenu sémantique de l'unité informationnelle. Ce contenu est éventuellement enrichi d'informations contextuelles.

---

10 Notons que c'est un exemple simple de description formelle de la propriété Défaire, et que celle-ci peut être beaucoup plus complexe.

En résumé :

Les fonctions Spec-Exec-AS et Spec-Rendu-Eff correspondent à **des traitements effectués par le système** et impliquent :

- le choix **du/des actions système** pour l'interface en entrée,
- le choix **du/des concepts à présenter** pour l'interface en sortie.

4. LE CONTEXTE : VERS UN MÉCANISME D'ABSTRACTION ET DE CONCRÉTISATION CONTEXTUEL

Le modèle Pipe-Lines fait intervenir la notion de contexte comme paramètre directeur des fonctions dont le système a la charge (se reporter aux couples <Acqu-Inter-AP, Spec-Exec-AP> et <Spec-Exec-AS, Spec-Rendu-Eff> de la figure 3.10). L'objet de ce paragraphe est de montrer comment le contexte et ses perspectives sur l'environnement, l'utilisateur, l'interface, le dialogue et le noyau fonctionnel, agissent sur le comportement des fonctions système.

4.1. Définition du contexte

D'une manière générale, la notion de contexte recouvre un ensemble d'informations (ou conditions) dans lequel il est possible de situer (ou d'interpréter) une information donnée. Cet ensemble dépend des objectifs de l'acteur. Par exemple, l'économiste, analysant un événement social, fera appel au contexte "circonstances de l'événement" pour en étudier l'impact sur la stabilité de la bourse ; Dans le cas d'un énoncé, le contexte est l'ouvrage dans lequel l'énoncé se situe pour en tirer une signification.

Dans le cas qui nous intéresse, l'acteur est le système et ce système a pour objet le traitement d'informations telles les actions physiques, les unités informationnelles et les effets. Dans ce cadre, la notion de contexte est ainsi définie :

Pour un système donné, le contexte est un ensemble d'informations maintenu par ce système et indispensable aux traitements des informations échangées avec l'environnement.

Parce que les traitements d'un système diffèrent (les phases des fonctions du modèle Pipe-Lines en sont un exemple), les informations contenues dans un contexte diffèrent en fonction des besoins. La figure 3.13 montre les dimensions structurantes des informations contextuelles : le sujet, la variabilité et la temporalité.

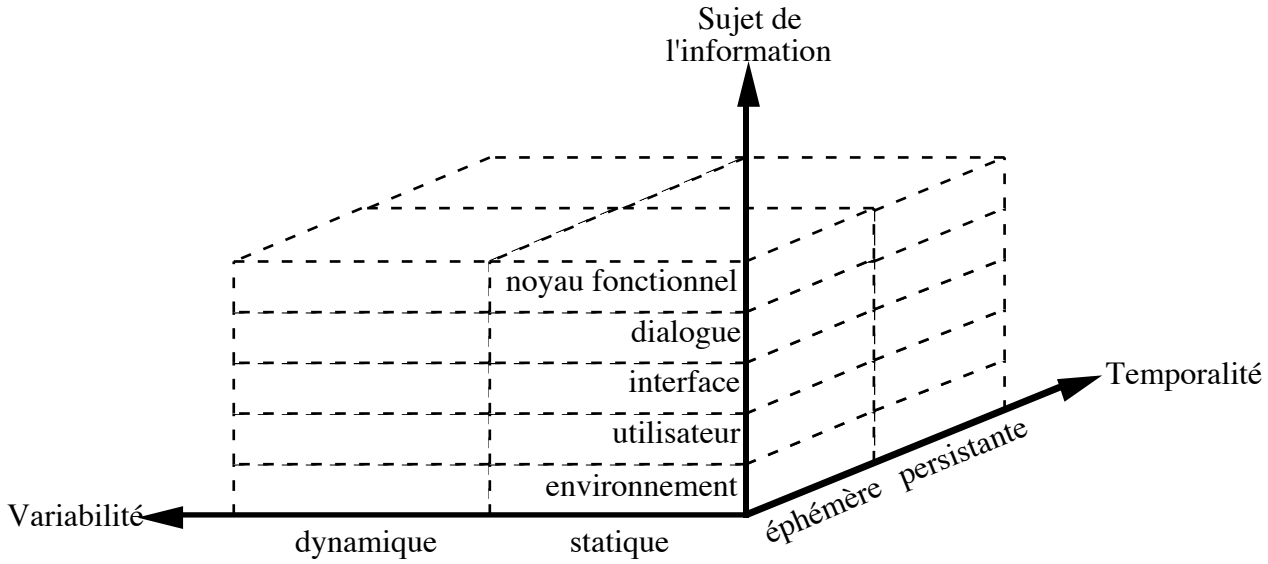


Figure 3.13 : Axes de classification des informations contenues dans le contexte.

La *variabilité* exprime la capacité d'une information contextuelle à changer au cours des traitements. Les valeurs, statique et dynamique, définissent deux types de contexte :

- un contexte statique dont le système possède une description immuable,

- un contexte dynamique qui varie au cours de l'utilisation du système. Il correspond au niveau de description d'une interface que Kuutti appelle "situation d'utilisation" [Kuutti 93][11].

La notion de contexte dynamique nous permet de préciser le concept d'état interne :

L'état interne d'un système à l'instant t est la valeur des informations contenues dans le contexte dynamique du système à cet instant. C'est la photographie du contexte dynamique pratiquée à l'instant t.

L'axe de la *temporalité* permet de caractériser la durée de vie des informations à l'intérieur du contexte. Certaines sont éphémères, d'autres sont permanentes.

11 Le niveau *situation d'utilisation* (en anglais, *Use situation*) définit l'interface en cours d'utilisation dont Kuutti souligne l'importance : les interfaces émergent quand le système est utilisé (*"interfaces emerge when the system is used"*).

Le *sujet* dénote un thème que modélise un ensemble d'informations contextuelles. Nous avons identifié cinq sujets possibles : l'environnement, l'utilisateur, l'interface, le dialogue et le noyau fonctionnel. Nous étudions maintenant le rôle du contexte selon ces cinq perspectives.

4.2. Le contexte et son rôle

Dans l'analyse qui suit, nous montrons l'impact des informations contextuelles sur le comportement des fonctions de l'espace Pipe-Lines. L'objectif est d'identifier des problèmes pertinents pour la conception de modèles d'architecture logicielle. Nous ne visons pas l'exhaustivité.

4.2.1. Contexte et environnement

L'environnement désigne des conditions extérieures susceptibles d'agir sur le fonctionnement d'un système. Considérant le système constitué du couple <artefact informatique, utilisateur>, les conditions de travail de l'acteur humain est une composante pertinente en interface homme-machine.

L'utilisateur et son environnement constituent un sujet d'étude approfondie dans les phases préliminaires de conception des interfaces [Barthet 88]. Par exemple, on éliminera les interfaces vocales pour un système dont l'utilisation est prévue en milieu bruité. Si l'étude sur l'environnement de travail est effectuée avec attention en phase de conception, ces conditions, en tant que variables explicitement maintenues, disparaissent du système livré. Au mieux, des paramètres d'installation, informations statiques, permettront de configurer le système.

La modélisation dynamique de l'environnement de travail est une dimension sous-estimée dans les études sur les capacités d'adaptation automatique des systèmes. Par exemple, le système pourrait éliminer ses restitutions vocales dès que le niveau sonore ambiant dépasse le seuil de détectabilité de l'opérateur humain. De manière générale, les informations dynamiques sur l'environnement de travail de l'utilisateur permettraient une meilleure intégration du monde réel et du monde informatique.

4.2.2. Contexte et utilisateur

individuelles. A cette fin, il contient des hypothèses (ou informations) sur les connaissances de l'utilisateur, ses attitudes, ses stratégies, ses objectifs, etc. Il convient de distinguer les informations statiques (comme les attributs physiques), des informations dynamiques.

Certaines sont éphémères comme les hypothèses auxquelles le système devra renoncer, d'autres sont permanentes.

Selon la fonction assignée au système, l'accent est mis sur une composante précise de l'utilisateur. Par exemple, s'il s'agit d'un jeu, le modèle que le système devra maintenir sur son adversaire humain sera centré sur les stratégies. Dans un système de consultation, l'objectif de l'utilisateur devient un facteur important. Pour un didacticiel, les connaissances de l'apprenant sont essentielles.

D'un modèle à l'autre, les techniques de représentation diffèrent. En Intelligence Artificielle, les connaissances sur l'utilisateur sont représentées par des prédicats tels que " croit", "sait", "n'est pas certain que", etc. ; les objectifs par "veut", "veut savoir si", "n'a pas de préférence pour", etc. [Kobsa 89]. MMI2 est un exemple de système multimodal véhiculant ce type d'informations [Kuijpers 92].

La technique des stéréotypes constitue chez certains systèmes, le fondement d'un raisonnement inférentiel par défaut. Dans Grundy, par exemple, un stéréotype encapsule un ensemble d'assertions sur un utilisateur canonique [Rich 89]. Le stéréotype, information statique, sert de valeur initiale par défaut au modèle de l'utilisateur réel. Celui-ci, structure dynamique, est mis à jour au cours de l'utilisation du système par un mécanisme de raisonnement inférentiel qui maintient, pour chaque assertion du modèle, un facteur de confiance et une justification. Par exemple, l'attribut *sexe* de l'utilisateur est *féminin* avec un facteur de confiance élevé parce que le prénom de l'utilisateur inscrit dans */etc/passwd* est *Laurence*.

En enseignement assisté par ordinateur, on retrouve une approche similaire où le modèle canonique est celui d'un étudiant expert dans le domaine. Le système raisonne alors par recouvrement, par différence ou par perturbation [Kass 89]. La technique du recouvrement, la plus simple des trois, fait l'hypothèse que la connaissance de l'apprenant est un sous-ensemble de celle de l'expert. Cette approche est acceptable pour des cas simples où il s'agit d'enseigner des faits (connaissance factuelle). Elle est mal adaptée au cas de la connaissance procédurale car incapable de considérer des stratégies autres que celles de l'expert. Les modèles différentiels et par pertubation visent à améliorer ces restrictions.

Le modèle de l'utilisateur est indispensable aux systèmes à vocation experte. Sans lui, la production de réponses ou de suggestions idoines ne pourrait avoir lieu. Dans les systèmes usuels qui laissent à l'utilisateur la maîtrise du domaine, le modèle de l'utilisateur est implicite ou, s'il est explicite, prend généralement la forme d'une liste de préférences ajustables manuellement par l'utilisateur. Le système n'est pas adaptatif mais adaptable. Et pourtant, sans puiser dans les techniques sophistiquées de l'Intelligence Artificielle, il est possible de détecter

comme dans Eager les actions répétitives [Cypher 91]. Avec les insignes actives[12], capables de localiser les employés d'une entreprise, il devient possible de télécharger automatiquement l'environnement de l'utilisateur lorsqu'il se présente à une station de travail [Pountain 93].

Avec un système de vision par ordinateur capable d'observer la posture de l'utilisateur, l'économiseur d'écran saurait être actif à bon escient évitant des intrusions disruptives lorsque l'utilisateur réfléchit un peu longuement sur les informations affichées [Beaudouin-Lafon 93]. Ou encore, pour l'unité informationnelle correspondant à la commande <Grossir les caractères du texte>, l'action système pourrait être différente selon la position de l'utilisateur (fonction Spec-Exec-AS) :

- <Ajouter +2 à la taille courante des caractères> si l'utilisateur est assis devant son ordinateur,

- <Ajouter +6 à la taille courante des caractères> si l'utilisateur est debout devant son ordinateur.

En résumé, le rôle d'un modèle de l'utilisateur est de permettre au système de tenir compte de la variabilité inter- et intra-individuelle. Mais la nature de la variabilité dépend de la fonction première de chaque système. Nous avons vu une première différence entre les systèmes à vocation experte et les systèmes outils. Dans le premier cas, l'adaptation est nécessairement assurée par le système. Dans le second cas, le modèle de l'utilisateur est le plus souvent implicite ou se présente comme une liste finie de paramètres modifiables par l'utilisateur uniquement. Le système est rarement adaptatif.

Si l'adaptativité apparaît comme la version noble du traitement de la variabilité, il convient de s'interroger sur l'opportunité des changements : une modification inattendue est nécessairement intrusive.

4.2.3. Contexte et interface

Le contexte sur l'interface englobe les informations qui définissent l'état de l'interaction. Le mode courant, l'historique des unités informationnelles en font partie.

Le mode (informatique) au sens où nous l'avons introduit au chapitre II, désigne on le rappelle, une information dynamique qui modifie le sens de ce que l'utilisateur fait ou perçoit

12 Insigne active : de l'anglais "*active badge*".

[13]. Il a un impact direct sur la sélection de l'action système à exécuter. A titre illustratif, nous reprenons l'exemple de l'éditeur de texte de la figure 3.5. Ce système gère deux modes d'édition : la réécriture et l'insertion. En mode réécriture, à l'unité informationnelle [Insérer retour chariot], le système associe l'action système [Positionnement du point d'insertion à la ligne suivante]. En mode insertion, l'action système est [Ajout d'une nouvelle ligne et positionnement du retour chariot au début de cette nouvelle ligne]. Un raisonnement similaire tient pour l'identité du langage d'interaction courant qui dirige l'interprétation des actions physiques utilisateur.

L'historique de l'interaction est, comme le mode, une information dynamique qui a trait à l'interface. Il influence la détermination de l'action système en se référant au passé. Ce passé s'exprime en terme d'unités informationnelles qui, par corrélation, déterminent une action système. Nous reviendrons sur cette mise en relation d'unités informationnelles lorsque nous présenterons au chapitre VII notre mécanisme générique de fusion des unités informationnelles. Pour l'instant, nous en illustrons le principe avec le système MATIS. Supposons que l'unité informationnelle courante véhicule l'information [To Boston]. Les unités informationnelles déjà traitées sont maintenues dans le contexte "dynamique-interface". Si la nouvelle unité est logiquement cohérente avec une autre unité, celles-ci seront combinées et l'action système deviendra [Compléter la requête : Destination = Boston]. Dans le cas contraire, l'action système consistera à [Créer une nouvelle requête : Destination = Boston].

Le contexte sur l'interface est plus largement dynamique que statique. Les variables système comme TERM dans les fichiers de type .login sont des exemples d'éléments statiques. Nous trouvons dans MATIS d'autres informations statiques comme le nombre maximum de fenêtres ouvertes à un instant donné. La limite étant atteinte, MATIS exécute une action système de fermeture de la fenêtre la plus ancienne.

4.2.4. Contexte et dialogue

En interface homme-machine, la notion de dialogue est mal cernée. Au sens général, un dialogue désigne une conversation entre plusieurs entités communicantes sur un sujet donné ; il désigne aussi le contenu de cette conversation. En interface homme-machine, ce terme couvre le phénomène conversationnel plutôt que le contenu des échanges. C'est pourquoi, dialogue et interaction sont souvent employés de manière interchangeable. Il est intéressant d'observer qu'en recherche sur la communication en langage naturel, on parle volontiers de dialogue. A

---

13 "The mode is the variable information in the computer system affecting the meaning of what the user sees and does" [Thimbleby 90 page 228]. Cette définition est reprise dans [Blattner 92 page xxiv].

notre sens, le dialogue traduit un phénomène d'échange tandis que l'interaction exprime des changements d'état chez les acteurs du dialogue et ces changements sont les conséquences du dialogue.

Les modèles de dialogue se répartissent en deux classes [Normand 92] :

• Les modèles à événements, issus de la programmation parallèle, décrivent l'interaction de façon asynchrone : l'utilisateur est source d'événements qui sont captés par le système par l'intermédiaire des traitants d'événements. L'état courant du dialogue maintenu par le contexte se traduit par l'ensemble des traitants d'événements qui peuvent être rendus actifs à un instant donné.

• Les modèles de dialogue conversationnels sont fondés sur une perspective linguistique. L'interaction est ici considérée comme une conversation fondée sur un langage. Il existe des modèles de dialogue conversationnels statiques et dynamiques [Bourguet 93]. Le système ICPplan [Bourguet 93] est un exemple de système intégrant un modèle du dialogue dynamique ; l'état courant du dialogue est situé dans un espace à deux dimensions définis par deux axes, l'axe régissant et l'axe incident issus du modèle de Luzzati [Luzzati 89]. L'axe régissant symbolise la recherche d'un accord : il traduit l'avancement du dialogue. L'axe incident traduit une volonté de clarification ou de réparation. Comme il est expliqué dans [Bourguet 93], un dialogue dont la position est forcée sur l'axe régissant acquiert de la robustesse mais fige l'interaction. Au contraire un dialogue dont la position descend selon l'axe incident, gagne en flexibilité mais est plus complexe.

Considérant une unité informationnelle comme un acte de langage ayant valeur illocutoire, le contexte au sujet du dialogue (modèle de dialogue conversationnel) prend toute son importance. Nous illustrons l'influence du dialogue sur la détermination de l'unité informationnelle à rendre perceptible (fonction Spec-Rendu-Eff) : pour cela, nous considérons l'exemple de modification de la température de la figure 3.7 et nous supposons que l'utilisateur a spécifié une augmentation de la température sans préciser exactement la nouvelle température. Nous étudions les unités informationnelles possibles selon la stratégie de dialogue adoptée [Bourguet 92]. La stratégie de dialogue est une information maintenue dans le contexte dynamique sur le dialogue. Elle peut être réactive, directive, coopérative, négociée ou intentionnelle :

• Stratégie réactive : le système complète l'unité informationnelle avec une valeur par défaut. Dans notre exemple, l'action système déterminée par Spec-Exec-AS sera par exemple : [Augmenter de 20° la température]. La stratégie réactive influence la

fonction Spec-Exec-AS (interface en entrée), au contraire des autres stratégies qui influencent la fonction Spec-Rendu-Eff (interface en sortie).

• Stratégie directive : l'interface guide l'utilisateur vers une spécification complète de ses objectifs. Dans notre exemple, l'interface demandera à l'utilisateur de compléter sa requête. L'unité informationnelle véhiculera alors cette question :
- "Quelle température voulez-vous atteindre ?"
- "De combien de degrés voulez-vous augmenter la température ?"

• Stratégie coopérative : l'interface aide l'utilisateur à compléter la tâche qu'il vient de commencer en lui soumettant les choix possibles. Par exemple, pour l'augmentation de la température, ces choix seront contenus dans l'unité informationnelle :
- "Voulez-vous atteindre une température de 60°, 80° ou 100° ?"
- "Voulez-vous augmenter la température de 20°, 40° ou 60° ?"

• Stratégie négociée : l'interface propose des solutions pour les informations manquantes. Par exemple l'unité informationnelle contiendra :
- "Voulez-vous atteindre une température de 60° ?"
- "Voulez-vous augmenter la température de 20° ?"

• Stratégie intentionnelle : l'interface, guidée par les intentions de l'utilisateur, infère les informations manquantes. Dans ce cas, le modèle de dialogue s'appuie sur le modèle de l'utilisateur. Dans l'exemple, si lors des interactions précédentes, l'utilisateur a toujours augmenté de la température 10 en 10 degrés, l'unité informationnelle correspondra alors à la demande :
"Est-ce bien de 10° que vous voulez augmenter la température?"

4.2.5. Contexte et noyau fonctionnel

La dernière case du contexte concerne le noyau fonctionnel. Elle regroupe les informations qui définissent l'état courant du noyau fonctionnel et les tâches qu'il permet de réaliser. Le modèle de la tâche décrit les tâches à mettre en œuvre, leurs structurations en sous-tâches et leurs relations ou enchaînements. La partie dynamique du contexte concernant la tâche définit l'historique des tâches antérieures et celle en cours de spécification. Ceci permet de situer l'interaction dans la décomposition hiérarchique des tâches décrite par le modèle statique de la tâche.

Nous illustrons l'influence de la case "noyau fonctionnel" du contexte avec le système NoteBook : comme au paragraphe 3.2, nous considérons que l'utilisateur a la possibilité de dicter le contenu d'une note et qu'il vient d'articuler la phrase "*Insert a new note*". Nous avons montré que la fonction Acqu-Inter-AP choisit le langage d'interaction pour l'analyse : ce choix consiste à déterminer si la phrase est une commande ou le contenu d'une note (Figure 3.12). Nous montrons ici que la position courante dans l'arbre des tâches permet d'influencer le choix du langage. La figure 3.14 présente une partie de l'arbre des tâches. Si la position courante est au point (1) alors la phrase correspond à une commande. Au contraire si la position est au point (2) alors la phrase est le contenu d'une note. Ainsi la position courante de l'interaction dans l'arbre des tâches influence la fonction Acqu-Inter-AP dans sa phase d'interprétation.

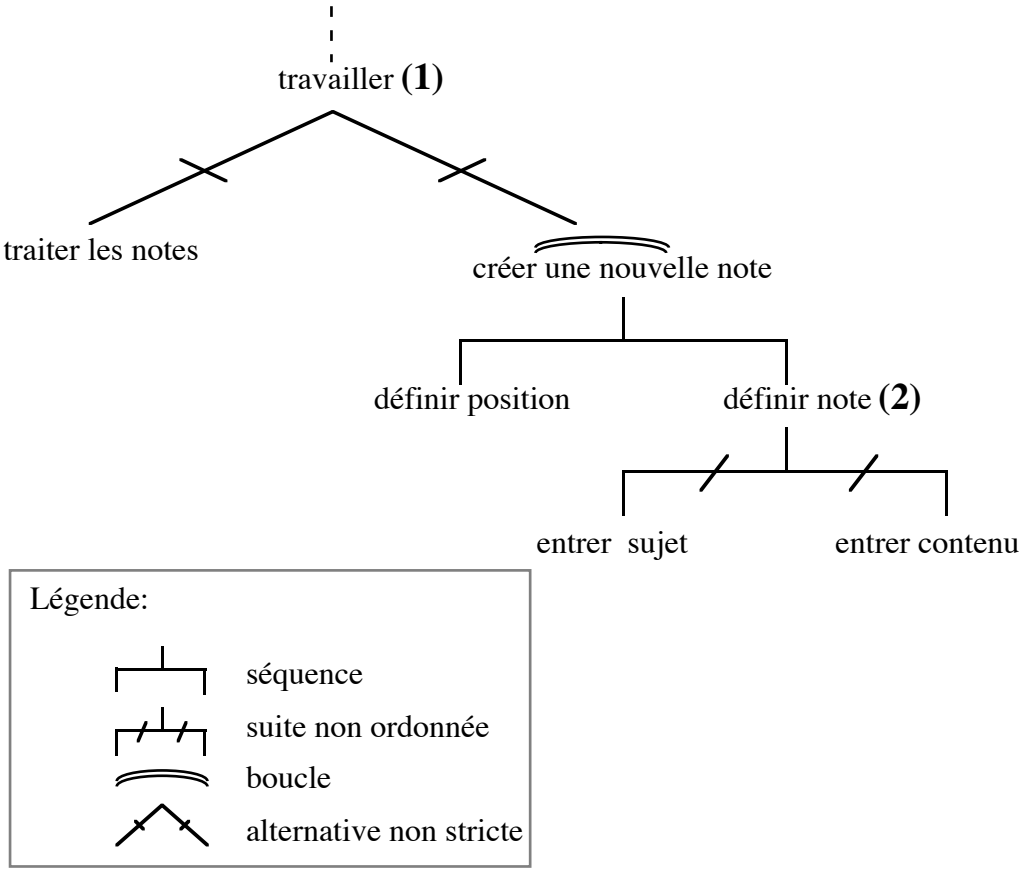


Figure 3.14 : Une partie de l'arbre des tâches de NoteBook.

5. LE FLOT DES INFORMATIONS

Pour simplifier l'exposé, nous avons jusqu'ici considéré que le flot des informations suivait un cheminement séquentiel et symétrique allant des intentions aux effets, puis des effets aux représentations mentales. L'objet de ce paragraphe est de montrer d'autres cheminements, sortes de courts-circuits entre les plans d'entrée et de sortie. (On retrouve un phénomène semblable dans le modèle de Rasmussen sur les comportements cognitifs.) Du côté système, ces raccourcis se manifestent à quatre niveaux d'abstraction. Le premier a trait aux dispositifs d'entrée/sortie, les trois autres concernent les retours d'information. Deux phénomènes viennent se greffer sur les cheminements et les retours d'information : le parallélisme et la fusion/fission. Nous montrons comment le modèle Pipe-Lines affine ces notions issues de MSM.

5.1. Activité des dispositifs physiques

A chaque dispositif physique est associé un processus de traitement mais ce dernier peut être à l'écoute ou indisponible. Nous disons :

• qu'un dispositif est actif s'il est prêt à capter de l'information, son processus d'acquisition étant à l'écoute.

• qu'un dispositif est inactif s'il n'est pas prêt à capter de l'information, son processus d'acquisition étant indisponible.

Si l'utilisateur manipule un dispositif physique inactif, ses actions ne sont pas captées donc non traitées par le système. Cette situation est schématisée dans le plan des entrées par la figure 3.15.

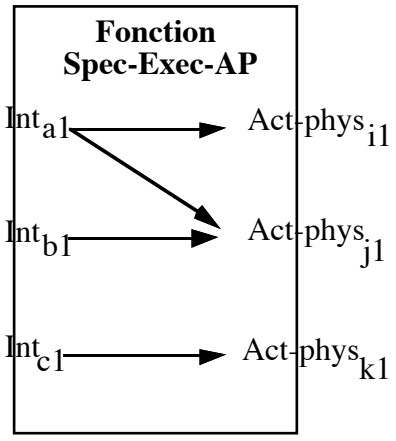


Figure 3.15 : Manipulation de dispositifs physiques d'entrée inactifs. Les informations en provenance de l'utilisateur ne cheminent pas plus avant.

L'inactivation, pour l'utilisateur, d'un dispositif physique a quatre causes possibles selon la nature de la préemption :

Cas 1- Le noyau fonctionnel fait préemption sur les dispositifs. Ce cas, illustré par le schéma de gauche de la figure 3.16, est celui des systèmes de démonstration comme le Guided Tour de NeXTStep [Tschumy 90]. Les mouvements du curseur associés à la souris sont programmés ou enregistrés au préalable. L'utilisateur doit se cantonner dans le rôle d'observateur : ses manipulations de la souris sont ignorées.

Cas 2- Un autre utilisateur fait préemption. Ce cas, illustré dans la figure 3.16, peut se présenter dans un collecticiel comme Timbuktu™ [Farallon 87]. Par l'intermédiaire d'une connexion réseau, ce système permet de visualiser l'écran et les curseurs de la souris et du clavier d'un utilisateur compère. Il est alors possible de voir se déplacer ses propres fenêtres et curseurs suite aux actions du compère distant et cela sans pouvoir agir à notre tour sur nos propres dispositifs locaux[14].

Cas 3- L'usage d'un dispositif physique rend inactif un autre dispositif. Cette situation peut se présenter lorsqu'un seul processus d'acquisition doit gérer plusieurs dispositifs physiques conceptuellement incompatibles, par exemple deux claviers. L'utilisation de l'un rend l'autre inactif.

Cas 4- L'activité permanente du dispositif physique rend difficile l'utilisation du système. Ce cas, illustré par le schéma de droite de la figure 3.16, est celui des dispositifs, tel le microphone et la caméra, qui ne nécessitent pas une manipulation explicite de la part de l'utilisateur. En particulier, un microphone en écoute permanente conduit le système à interpréter tous les signaux sonores et notamment les discussions avec les collègues de bureau. Il est clair que dans ces conditions, le risque est grand de devoir répondre aux messages d'incompréhension du système ou pire de défaire une action système non désirée. Office Manager [Lunati 91], l'interface du système de reconnaissance de la parole Sphinx, propose plusieurs modes

---

14 Le système Timbuktu offre plusieurs modes d'utilisation. Le mode observateur, ici décrit, permet une simple visualisation de l'écran d'un autre utilisateur ; les dispositifs physiques locaux sont sans effet sur l'application distante. En mode contrôle, les dispositifs physiques locaux permettent d'agir sur l'application distante. Au niveau physique, la gestion des conflits est résolue en appliquant le principe du "permier arrivé, premier servi". Ainsi si deux utilisateurs cherchent à déplacer le curseur "en même temps", c'est le permier (au niveau des interruptions) qui a droit de préemption. Pour les autres niveaux d'abstraction, il n'y a pas de gestion des conflits d'accès. Cette tâche revient aux utilisateurs.

fonctionnement. Trois d'entre eux permettent à l'utilisateur de spécifier quand le microphone est actif : le processus d'acquisition lié au microphone est par défaut inactif et rendu actif à la demande de l'utilisateur (par sélection d'un bouton). Dans le dernier mode de fonctionnement, le microphone est en écoute permanente.

La figure 3.17 regroupe les quatre cas de figure.

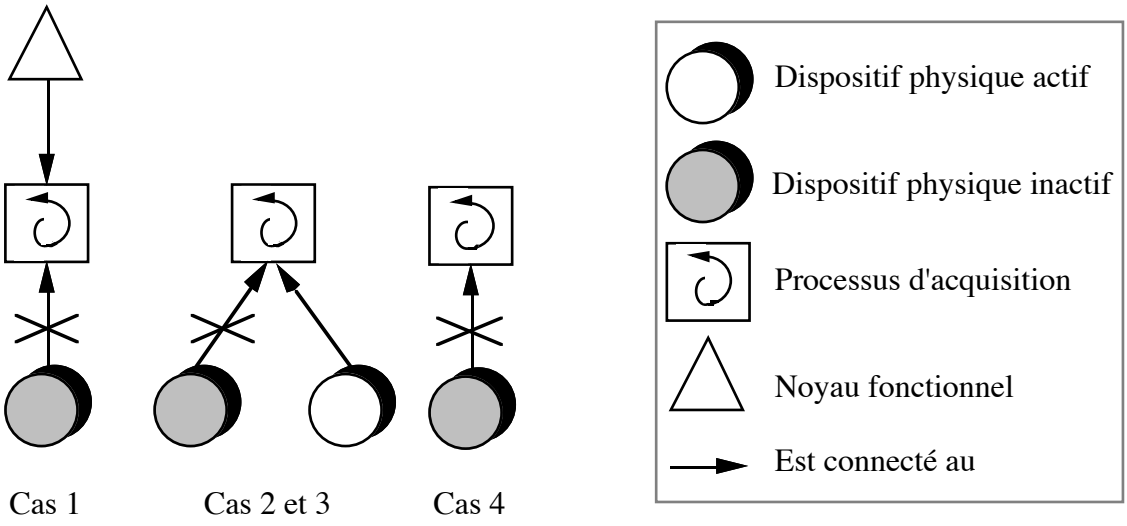


Figure 3.16 : Inactivité d'un dispositif physique.

5.2. Les retours d'informations

Un retour d'information se manifeste sous forme d'actions physiques système. Le modèle Pipe-Lines permet de dégager quatre niveaux de retours d'informations allant des retours sensori-moteur à la sémantique du domaine.

5.2.1. Retours d'information sensori-moteur

Un retour d'information sensori-moteur est une réaction de dispositif physique sans intervention logicielle. Il traduit une propriété inhérente au dispositif. La sonorité ou la dureté mécanique des touches du clavier, le poids et le roulement de la souris sont des exemples de retours d'information sensori-moteur. Dans la figure 3.17, nous les représentons par une flèche dans le plan qui conduit directement des actions physiques utilisateur aux actions physiques système.

Bien que non contrôlés par le logiciel, les retours d'information sensori-moteur ont leur part dans l'interaction homme-machine. Par exemple, un clavier bruyant peut nuire aux efforts

de concentration d'un collègue de bureau[15]. Un clic souris insonore rend les tâches de sélection plus difficiles [Jambon 94]. La littérature abonde d'études sur les propriétés intrinsèques des dispositifs physiques [Card 83].

5.2.2. Retours d'information de bas niveau

Un retour d'information est de bas niveau lorsque l'interprétation des actions utilisateur conduit Acqu-Inter-AP à transmettre directement une information à sa fonction symétrique Spec-Exec-AP qui, à son tour, produit une action physique système. Ce cheminement des informations est présenté dans la figure 3.17. La mise en évidence d'une icône après sa sélection ou un bip sonore est souvent un retour d'information de bas niveau.

Noter qu'à la différence des retours sensori-moteurs, les retours d'informations de bas niveau sont calculables, donc contrôlables par logiciel, donc adaptables par l'utilisateur.

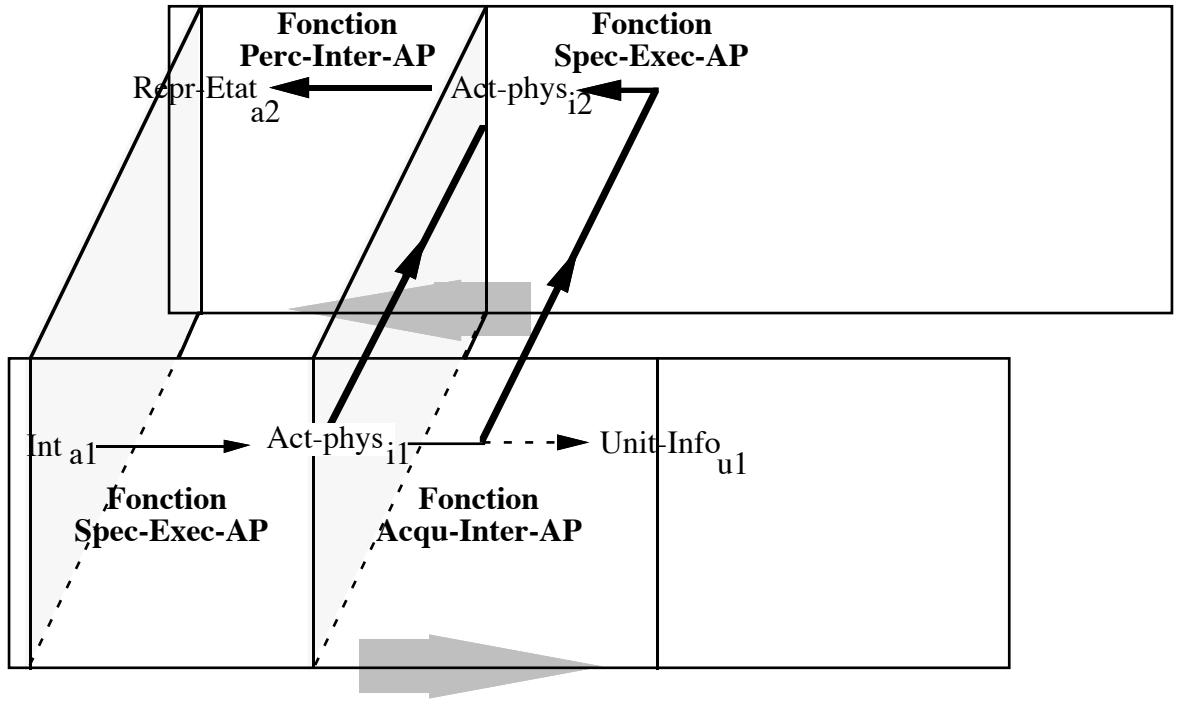


Figure 3.17 : Retours d'information de bas niveau.

5.2.3. Retour d'information de niveau intermédiaire

---

15 Les terminaux alphanumériques comme les VT100, disposaient d'un interrupteur qui permettait de supprimer le retour sonore des touches du clavier. Le niveau sonore mécanique ne pouvait cependant être masqué.

Un retour d'information de niveau intermédiaire implique l'interprétation des actions physiques en unités informationnelles et le transfert direct d'unités informationnelles entre Acqu-Inter-AP et sa fonction symétrique Spec-Exec-AP. Ce niveau de retour, qui court-circuite le niveau fonctionnel, permet d'informer l'utilisateur que ses actions physiques ont été interprétées. Dans MATIS, l'affichage de la dernière phrase reconnue par Sphinx en est un exemple. La figure 3.18 montre le cheminement des retours d'information de niveau intermédiaire.

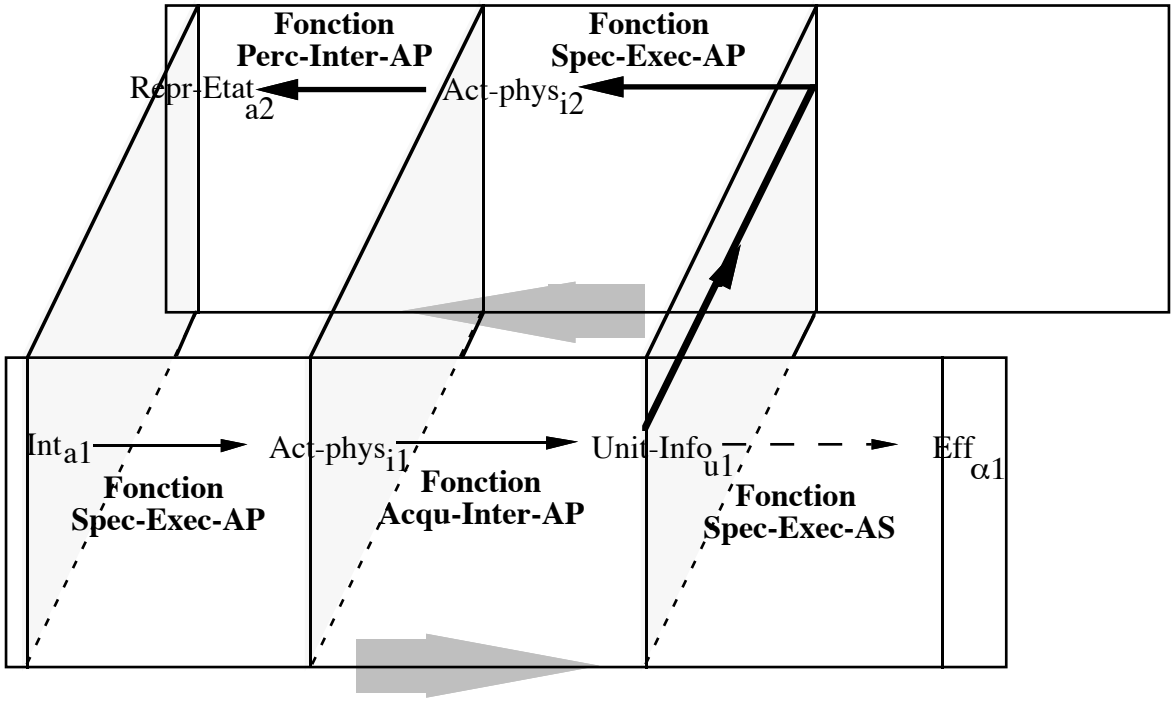


Figure 3.18 : Retour d'information de niveau intermédiaire.

5.2.4. Retour d'information de haut niveau

Un retour d'information de haut niveau consiste à traduire l'effet qui résulte de l'exécution d'une action système. Il implique donc la connaissance de la sémantique des actions utilisateur et peut nécessiter des informations issues du noyau fonctionnel. Par exemple, lors du déplacement d'une icône dans le bureau électronique de NeXT, chaque icône traversée réagit. Déplacer une icône de fichier sur l'icône d'un CD provoque l'affichage d'une étincelle dans l'image du CD montrant que le fichier peut être recopié. De même glisser une icône de fichier sur l'icône de la racine de son répertoire représentée par une maison provoque l'ouverture de la porte et l'éclairage de la maison traduisant là aussi que la commande de recopie est valide.

D'autres actions utilisateur comme celles des commandes passives ne font généralement pas intervenir le noyau fonctionnel. Une commande est passive (ou tâche syntaxique [Coutaz 90]) lorsqu'elle ne modifie pas l'état du noyau fonctionnel. Les défilements d'informations dans une fenêtre ou les tâches de réorganisation de l'écran sont de tels exemples.

La figure 3.19 montre le cheminement des retours de haut niveau .

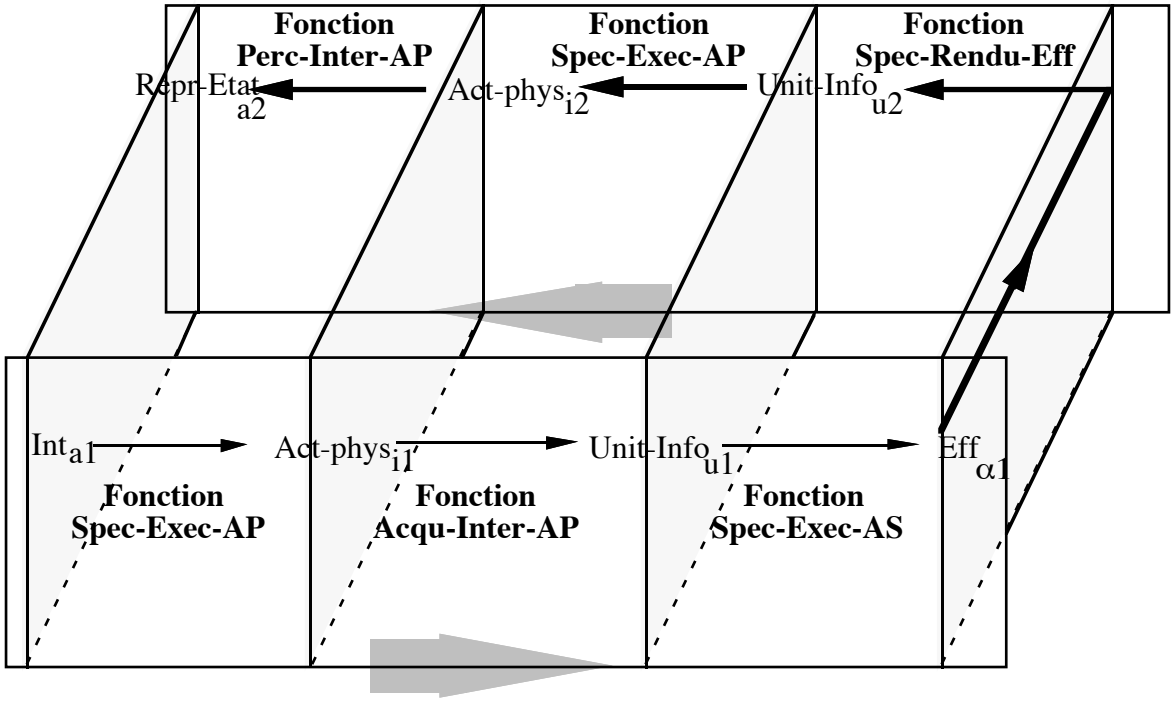


Figure 3.19 : Retour d'information de haut niveau.

5.2.3. Conclusion

Nous venons de décrire les quatre niveaux possibles de retours d'informations dans le modèle Pipe-Lines. Remarquons que la plupart des commandes de manipulation des concepts du domaine font intervenir tous les niveaux de retour. Ces niveaux renforcent la cohésion des deux plans en explicitant les passerelles possibles.

Quel que soit le type de retour d'information, il peut être ponctuel ou prolongé selon la classification de Sellen [Sellen 92]. Pour la réalisation d'un retour d'information prolongé, un flot d'information continu et circulaire est instauré tandis que d'autres traitements d'abstraction ou de concrétisation continuent ; ceci implique des traitements en parallèle. Le parallélisme fait l'objet du paragraphe suivant.

5.3. Parallélisme des traitements

Les étapes du modèle Pipe-Lines permettent d'identifier différents niveaux de parallélisme. Dans l'analyse qui suit, nous n'aborderons pas les activités parallèles au sein du "processeur humain" qui, selon Rasmussen et bien d'autres, se situent essentiellement au niveau expert (ou skill-level). Pour ce qui est du système, nous faisons une distinction entre le parallélisme perceptible à l'interface et les traitements parallèles internes au système. Le premier nous permet de faire le pont entre le système et la perspective utilisateur.

5.3.1. Parallélisme perceptible à l'interface

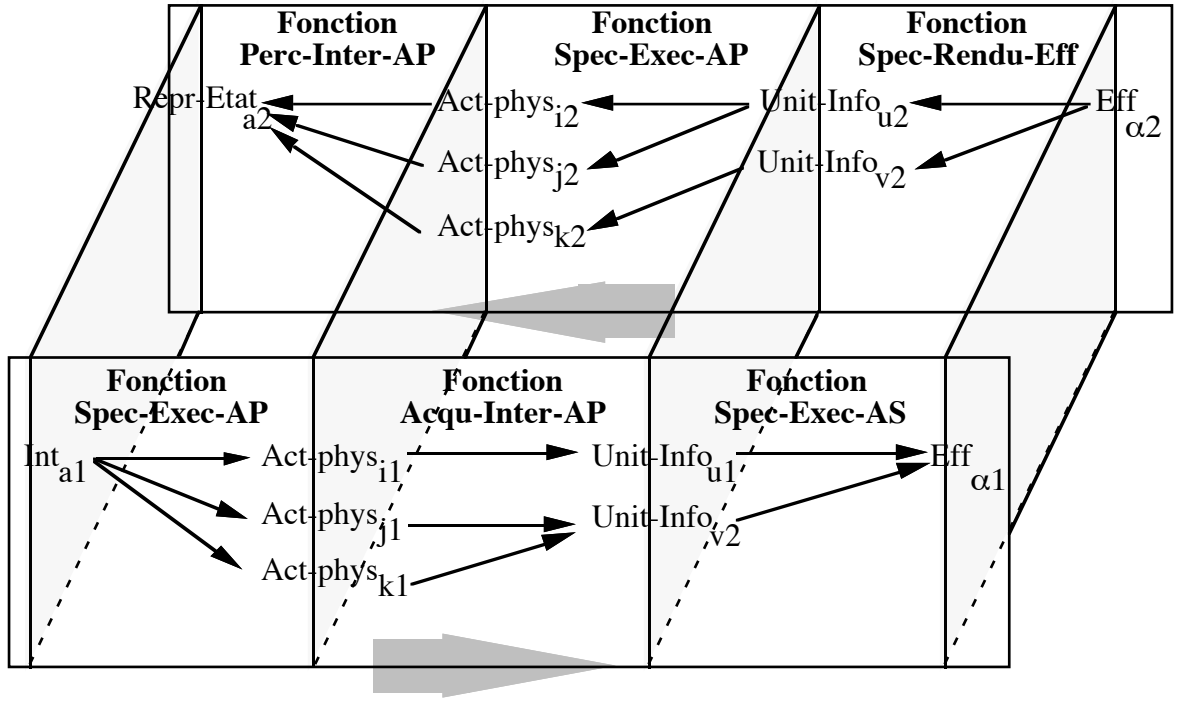
Comme l'indique MSM, le parallélisme perceptible à l'interface comprend trois niveaux de granularité : les actions physiques, les tâches élémentaires et les grappes de tâches.

- Le parallélisme au niveau physique :
  - autorise l'utilisateur à agir simultanément sur plusieurs dispositifs physiques d'entrée,
  - se traduit par la production en parallèle d'actions physiques pilotant des dispositifs physiques de sortie.

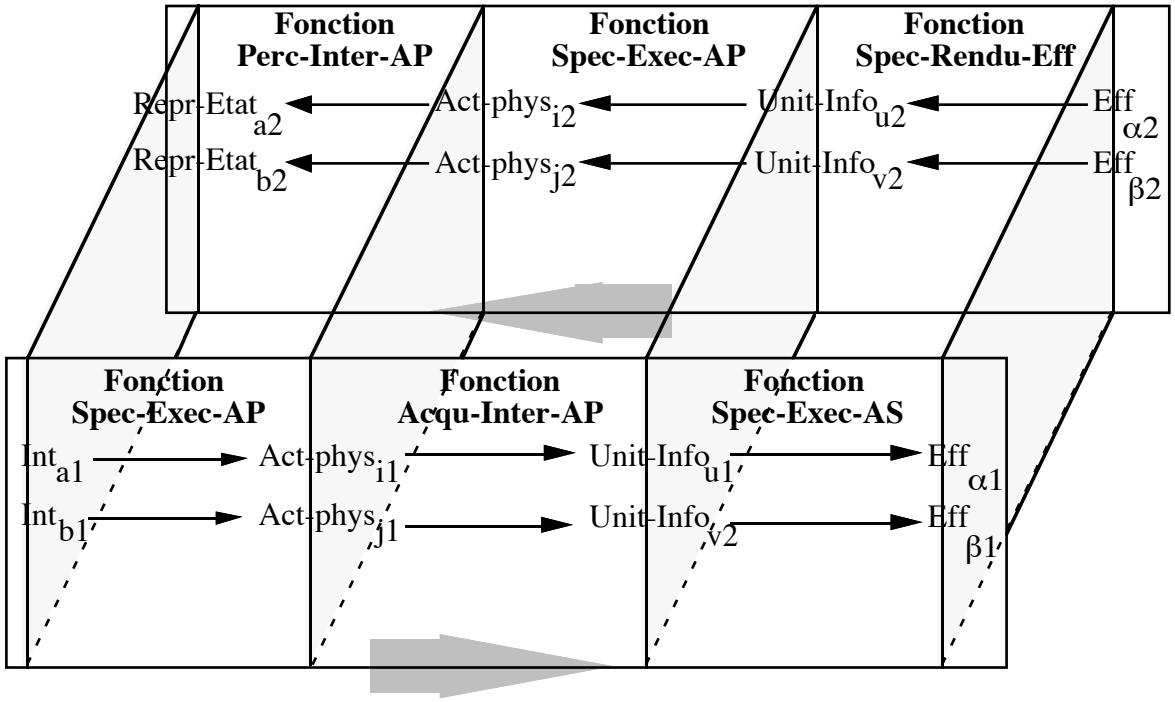
- Le parallélisme au niveau de la tâche :
  - autorise l'utilisateur à spécifier plusieurs commandes de manière concurrente,
  - se traduit en sortie par la production concurrente d'actions physiques pour rendre perceptibles des effets distincts.

- Le parallélisme au niveau des grappes de tâches permet à l'utilisateur de naviguer librement entre des tâches appartenant à des sous-espaces logiquement distincts comme l'espace des tâches de spécification de mission à un robot et celles de surveillance de robot.

La figure 3.20 traduit ces deux niveaux de parallélisme en entrée comme en sortie dans l'espace Pipe-Lines.



(a) Parallélisme au niveau des actions physiques.



(b) Parallélisme au niveau des tâches ou grappe de tâches.

Figure 3.20 : Parallélisme perceptible à l'interface.

5.3.2. Parallélisme interne au système

Nous identifions les possibilités de parallélisme interne sans évoquer les implications logicielles de réalisation. En particulier, nous ne ferons pas de distinction entre parallélisme vrai assuré par plusieurs processus s'exécutant en parallèle ou parallélisme simulé par entrelacement des traitements.

Le parallélisme des actions physiques effectuées ou perçues par l'utilisateur implique au moins les traitements parallèles des fonctions d'acquisition et de restitution sous peine de perte d'informations. De même, la possibilité de retours d'information prolongés souligne la présence de parallélisme entre les fonctions du premier et du deuxième plan.

Le modèle Pipe-Lines permet de distinguer trois types de parallélisme interne : aux niveaux d'une fonction, d'un plan et de deux plans symétriques. A cela s'ajoutent les possibilités de combinaisons entre ces catégories de base.

• Au niveau d'une fonction : le parallélisme implique alors le traitement simultané de plusieurs données par une même fonction dans le but d'en déterminer des résultats. Cette possibilité est illustrée à la figure 3.20(a) : la fonction Acqu-Inter-AP traite en parallèle des actions physiques de l'utilisateur pour en déduire deux unités informationnelles. Dans MATIS, l'utilisateur a la possibilité de parler tout en manipulant la souris : ces actions sont traitées en parallèle pour produire deux unités informationnelles.

• Au sein d'un plan : c'est le cas lorsque deux fonctions du même plan effectuent leurs traitements en parallèle. Dans MATIS, la fonction Acqu-Inter-AP peut s'exécuter en parallèle avec la fonction Spec-Exec-AS : la capture et la reconnaissance d'une phrase dictée (Acqu-Inter-AP) peuvent s'effectuer tandis qu'une action système est en cours d'exécution (Spec-Exec-AS), comme la construction d'une requête SQL pour la base de données.

• Dans les deux plans : on observe alors le travail en parallèle de deux fonctions n'appartenant pas au même plan. Le cas du retour d'information prolongé en est un exemple (figures 3.18 et 3.19).

5.4. Fusion et fission

MSM introduit les phénomènes de fusion et de fission pouvant survenir en interprétation comme en restitution. La fonction d'interprétation de MSM se traduit, dans le modèle Pipe-Lines, par les fonctions du plan des entrées, celle de restitution par le plan des sorties. Le modèle Pipe-Lines permet en plus de préciser les types d'informations impliquées par ces opérations : actions physiques, unités informationnelles ou effets (Figure 3.20)

Pour le plan des entrées, nous proposons trois niveaux de fusion et de fission [Gourdol 92] :

• La fusion sémantique correspond à cimenter plusieurs effets pour en obtenir un nouveau. Par exemple, dans VoicePaint un éditeur de dessin [Gourdol 91], deux commandes (dessiner une droite et changer l'épaisseur) sont combinées en une seule pour obtenir une droite à plusieurs épaisseurs. Chaque commande correspond à une unité informationnelle, et leur fusion aboutit à une nouvelle unité.
Au contraire, la fission consiste à éclater un effet en plusieurs effets.

• La fusion syntaxique revient à combiner plusieurs unités informationnelles pour en déduire un effet donné. La fission syntaxique se définit par l'obtention de plusieurs effets à partir d'une unité informationnelle donnée. Au vu des informations concernées, la fonction Spec-Exec-AS gère la fusion et la fission syntaxiques. Par exemple dans NoteBook, la combinaison des informations pour obtenir la commande complète "insérer une note" avec lieu d'insertion correspond à un cas de fusion syntaxique.

• La fusion lexicale combine des actions physiques pour obtenir une unité informationnelle[16]. La fission lexicale consiste à définir deux unités informationnelles à partir d'une action utilisateur. Fusion et fission lexicales sont à la charge de la fonction Acqu-Inter-AP.

16 Nous verrons au chapitre 7 sur la mise en pratique des phénomènes de fusion et de fission que les unités informationnelles du niveau lexical sont des représentations de bas niveau d'abstraction. Dans le cas précis de la fusion/fission lexicale, nous utilions le terme "unité informationnelle" de manière abusive pour plaquer aux concepts du modèle pipe-line. Il s'agit en fait d'une représentation intermédiaire entre l'unité informationnelle telle que nous l'avons définie au paragraphe 2.1 et l'action physique. Cette représentation revient aux étapes acquisition et exécution des actions physiques des fonctions Acqu-Inter-AP et Spec-Exec-AP respectivement.

Ces trois niveaux de fusion et de fission sont généralisables au plan des sorties. A titre illustratif, la fission lexicale en restitution consiste à présenter une même unité informationnelle avec différentes actions. Par exemple, un message d'alarme peut faire usage à la fois d'une sonnerie d'avertissement et d'un affichage dans une boîte de dialogue modale. Cette opération de fission lexicale est assurée par la fonction Spec-Exec-AP du plan des sorties. L'incrustation, la superposition sont des exemples de fusion lexicale qui restitue plusieurs unités informationnelles avec une même action physique. La fission syntaxique se définit par l'obtention de plusieurs unités informationnelles à partir d'un seul effet et la fusion en regroupant plusieurs effets en une seule unité informationnelle de sortie (fonction Spec-Rendu-Eff).

6. DISCUSSION ET CONCLUSION

Le modèle Pipe-Lines explicite les traitements effectués à la fois par l'utilisateur et par le système. Il permet donc de raisonner sur un système selon les deux points de vue : sa réalisation et son utilisation. Il reprend les concepts de MSM pour les lier dans un canevas rigoureux de fonctions et de données. Il exacerbe trois niveaux d'abstraction : les actions physiques, les unités informationnelles et les effets. Entre chacun de ces niveaux, il peut exister un quantum de niveaux. Parmi les trois niveaux d'abstraction, nous montrons l'existence de parallélisme, de fusion et de fission. Pipe-Lines clarifie aussi la notion de contexte en vue d'un mécanisme d'interprétation et de génération contextuelles.

Du point de vue réalisation système, le modèle Pipe-Lines établit le lien entre la description d'un système dans MSM et sa conception logicielle en un ensemble traitements. Nous concrétiserons ces liens au chapitre VI en localisant les fonctions dans notre modèle d'architecture logicielle.

Du point de vue de l'utilisation, Pipe-Lines explicite la liaison entre les traitements effectués par l'utilisateur et ceux du système. Il permet de raisonner sur les possibilités offertes par le système à son utilisateur. Ceci fait l'objet du chapitre suivant où nous décrivons une taxonomie des systèmes reposant sur les choix et possibilités offerts à l'utilisateur.

Chapitre IV

Usage, Option, Multiplicité des langages d'interaction et des dispositifs physiques : U.O.M, une méthode de classification des systèmes interactifs

"Interfaces emerge when the system is used."
- Kari Kuutti -

1. INTRODUCTION

Le modèle Pipe-Lines définit un canevas intégrateur des activités de l'utilisateur et du système. Dans cet espace, les concepts de dispositif physique et de langage d'interaction servent de points de contact entre les acteurs communicants. Les dispositifs constituent la passerelle physique et les langages forment la passerelle logique complémentaire. Nous proposons maintenant UOM, une méthode de classification des systèmes interactifs qui s'appuie sur ces deux éléments de contact entre l'utilisateur et le système.

Langage d'interaction et dispositif physique sont les deux fondements de la méthode UOM (Usage, Option, Multiplicité).

UOM se décompose en trois volets selon que l'on considère l'Usage, le caractère Optionnel ou la Multiplicité des langages d'interaction et des dispositifs physiques d'un système interactif. L'étude de la multiplicité des langages et des dispositifs conduit à une classification intemporelle, statique. L'analyse des options et de l'usage offre une perspective dynamique instantanée. L'étude des options met l'accent sur les possibilités de choix de langage et de dispositif à un instant donné ; l'usage caractérise les combinaisons possibles de ces choix.

Les paragraphes qui suivent décrivent les trois composantes de UOM. Le premier a trait à la classification des systèmes interactifs selon le critère de la multiplicité. Il conduit à une taxonomie que nous appelons M^2LD (pour disponibilité Mono- ou Multiple des Langages et Dispositifs). Le second présente O^2LD, notre technique de classification selon les options (caractère Optionnel ou Obligatoire des Langages et des Dispositifs). Enfin ULD (Usage des Langages et des Dispositifs) fait l'objet du paragraphe 5.

Ces trois volets de la méthode UOM sont motivés par une même question : Quels sont les atouts des systèmes à caractéristiques multiples? Nous montrons en conclusion comment des propriétés, telles l'équivalence et la redondance, apportent des éléments de réponse à cette question. Le paragraphe qui suit est une revue de cinq systèmes interactifs servant d'exemples illustrateurs à notre méthode de classification.

2. EXEMPLES ILLUSTRATIFS DE SYSTÈMES À CARACTÉRISTIQUES MULTIPLES

Ce paragraphe présente une description générale de cinq systèmes repris comme exemples au cours de ce chapitre.

2.1. Munix : une interface multimodale de dialogue homme-machine pour Unix (Multimodal Unix)

La richesse d'expression d'un langage comme le shell d'Unix a sa contrepartie en terme de coût d'apprentissage. L'interface multimodale proposée dans Munix tente de masquer cette complexité en intégrant au shell la manipulation directe et le langage naturel oral [Lefebvre 92, Poirier 93].

En entrée, Munix autorise le langage des commandes du shell, la manipulation directe et le langage naturel. En sortie, le langage d'interaction repose sur la métaphore du bureau où icônes représentent des fichiers et fenêtres des dossiers. Les dispositifs physiques d'entrée sont le clavier, la souris et le microphone tandis que l'écran est le seul dispositif de sortie.

Le langage de commande est saisi au clavier. La manipulation directe implique l'usage de la souris. Le langage naturel est oral mais les exemplaires d'objets, qui sont propres à chaque utilisateur, doivent être désignés avec la souris ou nommés par saisie au clavier. La figure 4.1 présente de manière synthétique les choix offerts par Munix en terme de langage et de dispositif.

Pour illustrer les capacités de Munix, prenons le cas d'un utilisateur désireux d'obtenir, page par page, la liste des fichiers du dossier */etc*. Les choix suivants lui sont offerts :

- saisir au clavier la commande shell : *ls /etc | more* ,
- sélectionner deux fois (double-click) avec la souris l'icône correspondant au dossier */etc*,
- énoncer oralement la phrase : *Peux-tu me montrer page par page?* accompagnée de la sélection, au moyen de la souris, de l'icône correspondant au dossier */etc*,
- énoncer oralement la phrase : *Peux-tu me montrer page par page?* accompagnée de la saisie au clavier du texte */etc*.

	Langages d'interaction	Dispositifs physiques
Interface en entrée	Langage de commande	
	Langage naturel combiné avec la manipulation directe ou la saisie au clavier	
	Manipulation directe	
Interface en sortie	Présentation graphique selon la métaphore du bureau	

Figure 4.1 : Les interfaces d'entrée et de sortie de Munix.

2.2. ICPplan : un éditeur de plans architecturaux

ICPplan offre un éventail d'outils élémentaires de réalisation de plans architecturaux. Il permet de créer des objets (par exemple une pièce, une chaise, une table, un escalier), de les assembler, de les modifier, etc. L'intérêt central d'ICPplan tient à son statut de plate-forme pour l'étude de la combinaison de modes de communication et pour son intégration d'un modèle dynamique du dialogue [Bourguet 92]. (Ce modèle a été évoqué au chapitre précédent à propos de la notion de contexte.)

La figure 4.2 présente les choix offerts :

- en entrée, un langage de commande articulé via le microphone ou la souris, la manipulation directe par souris, le langage naturel oral ou saisi au clavier.

- en sortie, une représentation graphique des concepts du domaine, ou des messages en langage naturel via le haut parleur ou l'écran.

Le langage de commande se traduit par la sélection d'éléments de menus ou de palettes d'outils. Pour ce langage, l'expression orale est restreinte à un sous-ensemble des possibilités menus et palettes. Le langage naturel peut se pratiquer à l'oral comme à l'écrit. Ainsi, l'écriture peut remplacer la parole lorsque les conditions d'usage ne sont pas propices (par exemple, un

niveau sonore ambiant trop élevé). En sortie, le système sélectionne le langage naturel oral lorsque l'utilisateur s'est exprimé oralement. Les informations énoncées sont cependant toujours accompagnées d'une restitution textuelle à l'écran.

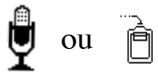
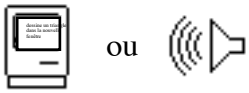
	Langages d'interaction	Dispositifs physiques
Interface en entrée	Langage de commandes	
	Langage naturel	
	Manipulation directe	
Interface en sortie	Présentation graphique des objets	
	Langage naturel	

Figure 4.2 : Les interfaces d'entrée et de sortie de l'éditeur ICPplan.

A titre d'exemple, nous présentons les possibilités offertes pour une tâche de rotation d'une entité de type escalier. Cet exemplaire est représenté à l'écran sous forme d'un objet graphique analogique (au sens de Bernsen). L'utilisateur peut :

- sélectionner et déplacer l'objet avec la souris,
- dire *pivote l'escalier*,
- dire *pivote cet objet* accompagné d'une désignation-souris de l'escalier,
- avec la souris, sélectionner l'escalier puis l'option *Pivoter* du menu *Manip*.

2.3. CUBRICON : une plate-forme de développement de systèmes intelligents

CUBRICON (The CUBRC Intelligent CONversationalist), composante du projet "Intelligent Multi-Media Interfaces" (IMMI), est une plate-forme de développement de systèmes à caractéristiques multiples [Neal 88]. Le domaine d'application de ces systèmes couvre le

contrôle aérien et la planification de missions militaires. On trouvera dans [Neal 91] une description détaillée de la conception logicielle de la plate-forme.

Pour l'interface d'entrée, l'objectif de CUBRICON est l'étude d'un mécanisme robuste de la compréhension du langage naturel complété de gestes déictiques. Ce dispositif d'inférence est capable de détecter les incohérences (lorsque, par exemple, l'utilisateur évoque oralement la notion de route mais désigne un missile) ou d'identifier l'objet pointé lorsqu'il y a ambiguïté (en raison par exemple de la superposition de représentations graphiques).

En sortie, CUBRICON aborde le problème du choix dynamique du langage d'interaction, du dispositif et de leurs combinaisons, afin de restituer au mieux les informations pertinentes. Par exemple, CUBRICON peut produire la désignation d'un objet graphique (par clignotement de l'objet) complétée d'une phrase en langage naturel qui lui fait référence. La figure 4.3 montre les différents cas de figure.

	Langages d'interaction	Dispositifs physiques
Interface en entrée	Langage naturel	 ou
	Manipulation directe Geste de désignation	
Interface en sortie	Présentation graphique des objets sous différentes formes	
	Langage naturel	 ou 

Figure 4.3 : Les interfaces d'entrée et de sortie des systèmes que l'on peut développer avec CUBRICON.

Des exemples de dialogue de contrôle aérien tactique sont présentés dans [Neal 91]. Le système est doté de deux écrans (l'un couleur, le second monochrome) :

- L'utilisateur énonce la phrase : *Display the Fulga Gap region*
- CUBRICON effectue les actions suivantes :
  - annonce du message vocal : *Look at the color graphics screen. The Fulga Gap region is being presented,*
  - affichage de la carte de la région sur l'écran couleur,

- annonce du message vocal : *The corresponding table is being presented on the monochrome screen,*
- affichage d'une table des attributs des entités de la carte sur l'écran monochrome.

• L'utilisateur énonce ensuite la phrase :*Where is the Dresden airbase?*

• CUBRICON produit le message vocal :*The Dresden airbase is located here <point>* accompagné en parallèle du clignotement et de la sélection de l'icône de la base correspondante.

2.4. MMI2 : un système expert de conception de réseaux informatiques

MMI2 est un système expert de conception de réseaux informatiques développé dans le cadre du projet européen ESPRIT MMI2 (Multi-Modal Interface for Man Machine Interaction) [Ben Amara 91, Kuijpers 92]. En entrée, MMI2 offre plusieurs langages naturels oraux et écrits (Français, Anglais et Espagnol), un langage de commande écrit, un langage gestuel et la manipulation directe des objets graphiques. En sortie, il affiche sur l'écran des phrases en langage naturel et utilise différentes modalités représentationnelles graphiques (au sens de Bernsen) : schémas de réseau, histogrammes, tableaux, etc. Nous résumons ces possibilités dans la figure 4.4.

Ce projet traite principalement deux problèmes : a) le traitement de la langue naturelle, et notamment la résolution des anaphores et des ellipses, et b) l'intégration des différents types d'informations en entrée par le biais d'un système de représentation commun appelé CMR (Common Meaning Representation). Nous reviendrons sur cette technique de représentation au chapitre VII.

Comme exemple, l'utilisateur vient de créer un réseau par manipulation directe et souhaite obtenir des informations sur les ordinateurs du réseau :

• L'utilisateur saisit la phrase suivante au moyen du clavier : *What is the cost of the Sparcstations?*

• MMI2 présente alors un camembert de prix et affiche le texte *See pie chart* en-dessous de la question saisie par l'utilisateur.

• L'utilisateur sélectionne une icône de machine du réseau avec la souris, puis saisit la phrase : *What is the cost of this machine?*

• MMI2 produit alors un histogramme montrant le prix des composants de la machine désignée et affiche le texte *See table* en-dessous de la question posée.

	Langages d'interaction	Dispositifs physiques
Interface en entrée	Langage de commandes	
	Langages naturels	
	Manipulation directe	
	Langage gestuel	
Interface en sortie	Présentation graphique des objets sous différentes formes	
	Langages naturels	

Figure 4.4 : Les interfaces d'entrée et de sortie de MMI2.

2.5. TAPAGE : un éditeur de tableaux

TAPAGE est un éditeur de TAbleaux par la PArole et le GEste [Faure 93]. Avec un ordinateur à stylo, l'utilisateur dessine un tableau à main levée. Ce tableau peut ensuite être mis en forme grâce au module "d'idéalisation des tableaux". Le stylo peut aussi servir à désigner une ligne du tableau en vue d'en modifier la taille ou la position. Une palette verticale de boutons de commandes désignables par stylo est affichée à gauche de l'écran. Parallèlement à la manipulation du stylo, l'utilisateur peut dicter une commande. Il s'agit ici d'un langage de commande oral du type *mets çà là*. Les données du tableau sont saisies de deux manières : au moyen du clavier virtuel dont les touches sont activables avec le stylo ou au moyen du stylo seul en activant le système de reconnaissance de l'écriture cursive.

Nous résumons les options de TAPAGE dans la figure 4.5.

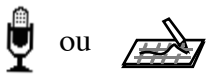
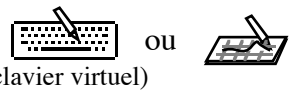
	Langages d'interaction	Dispositifs physiques
Interface en entrée	Langage de commandes	
	Langage naturel (données dans le tableau)	 (clavier virtuel)
	Manipulation directe	
Interface en sortie	Présentation graphique selon la métaphore du monde réel	

Figure 4.5 : Les interfaces de entrée et de sortie de l'éditeur de tableaux TAPAGE.

Nous choisissons un scénario d'utilisation de TAPAGE [Faure 93] qui aboutit à la réalisation d'un tableau contenant une donnée dans une de ses cases :

- Dessiner le tableau à main levée en utilisant le stylo,
- Idéaliser le tableau en énonçant la commande : *Remets en forme*,
- Sélectionner une case du tableau avec le stylo,
- Sélectionner, dans la palette d'outils, le clavier virtuel avec le stylo,
- Entrer la donnée par sélection des lettres du clavier virtuel à l'aide du stylo.

Ayant introduit nos cinq systèmes exemples, nous sommes en mesure de présenter les trois volets de notre méthode : M^{2}LD, O^{2}LD, ULD.

3. M^{2}LD : UNE TAXONOMIE SELON LA MULTIPLICITÉ DES LANGAGES ET DES DISPOSITIFS

Dans le modèle Pipe-Lines, les dispositifs et les langages forment les passerelles physiques et logiques entre l'utilisateur et le système :

- du point de vue de l'utilisateur, les langages et les dispositifs interviennent dans les fonctions Spec-Exec-AP (des intentions aux actions) et Perc-Inter-AP (des actions système aux représentations mentales) ;

• du point de vue du système, les langages et les dispositifs prennent part aux fonctions Acqu-Inter-AP (des actions utilisateur aux unités informationnelles) et Spec-Exec-AP (des unités informationnelles aux actions physiques).

Pour un système donné, M2LD considère le nombre de dispositifs et de langages distincts constitutifs des passerelles logiques et physiques. Soient :

- *de*, le nombre de dispositifs physiques d'entrée distincts du système,
- *ds*, le nombre de dispositifs de sortie,
- *le*, le nombre de langages d'interaction d'entrée autorisés par le système,
- *ls*, le nombre de langages d'interaction de sortie.

On dira que le système est :

- monodispositif en entrée (ou en sortie) si *de* =1 (ou si *ds* = 1),
- multidispositif en entrée (ou en sortie) si *de* > 1 (ou si *ds* > 1),
- monolangage en entrée (ou en sortie) si *le* = 1 (ou si *ls* = 1),
- multilangage en entrée (ou en sortie) si *le* > 1 (ou si *ls* > 1).

La figure 4.6 illustre les différentes possibilités.

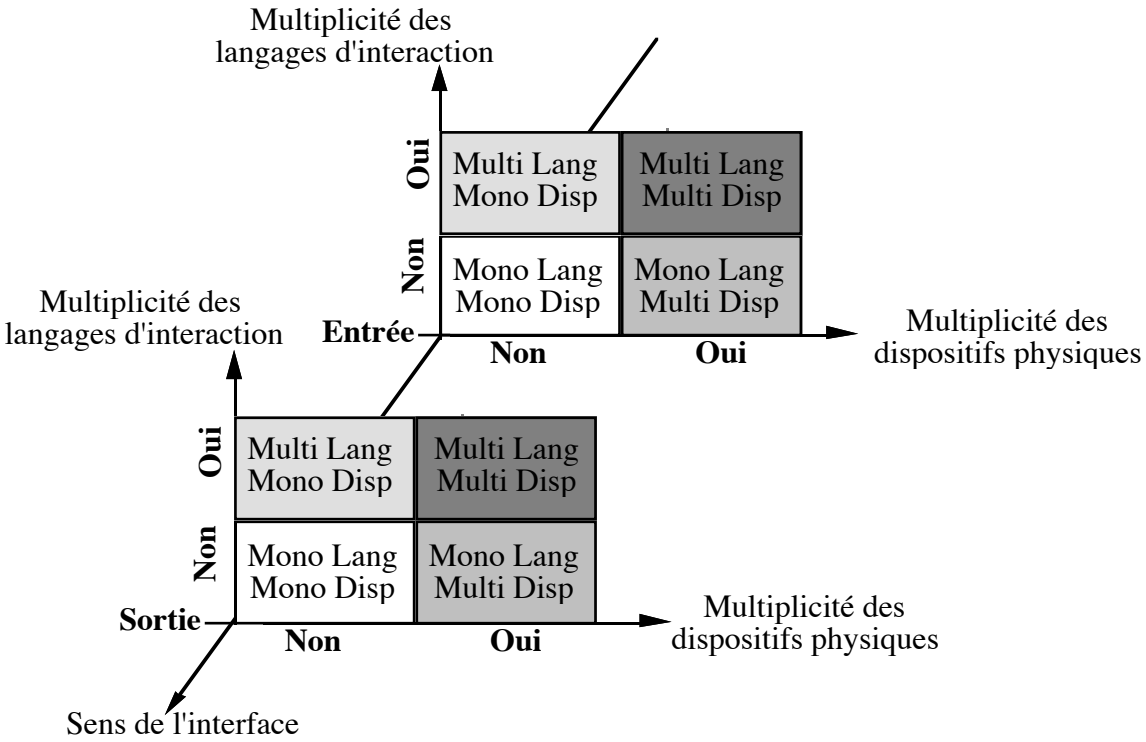


Figure 4.6 : M2LD, une classification des systèmes selon la multiplicité des langages et des dispositifs. La valeur "non" sur un axe signifie que le système est “mono” pour cette dimension. La valeur “oui” correspond au cas “multi”.

Pour une direction donnée des échanges (entrée ou sortie), un système à caractéristiques multiples peut appartenir à l'une des classes suivantes :

- "Multi Lang Mono Disp",
- "Mono Lang Multi Disp",
- "Multi Lang Multi Disp".

A titre d'exemple, MATIS est un système "Multi Lang Multi Disp" pour son interface d'entrée. En sortie, il est "Multi Lang Mono Disp".

M^2LD fournit une classification qui ne tient pas compte des variations temporelles de l'interaction. Selon le contexte, un système qualifié de "Multi Lang Multi Disp", peut temporairement se situer à l'autre extrême "Mono Lang Mono Disp". Si M^2LD convient à la pose d'une étiquette générale, l'analyse de l'utilisabilité d'un système nécessite parfois une classification plus fine. O^2LD en est une.

4. O^2LD : UNE TAXONOMIE SELON LES CHOIX TEMPORELS

O^2LD (pour caractère Obligatoire ou Optionnel des Langages et Dispositifs) organise la classification des systèmes interactifs selon deux axes : le choix, à un instant donné, du langage d'interaction et le choix, à ce même instant, du dispositif physique. De ces axes, étudiés au paragraphe suivant, nous déduisons des classes de système que nous présentons au paragraphe 4.2. En 4.3, nous analysons leurs relations et notamment leurs liens avec M^2LD.

4.1. Les axes de O^2LD

Les axes, choix du dispositif et choix du langage, concernent à la fois l'utilisateur et le système :

- A un instant donné et pour une intention formée, l'utilisateur a-t-il le choix du langage d'interaction et/ou du dispositif physique d'entrée?

- A un instant donné et pour une unité informationnelle à restituer, le système peut-il choisir le langage d'interaction et/ou le dispositif physique de sortie?

Soient :

- de_t, le nombre de dispositifs physiques d'entrée distincts dont l'utilisateur a la choix à l'instant t : $\forall t\ de_t \leq de$,

- ds_t, le nombre de dispositifs physiques de sortie distincts dont le système a le choix à l'instant t : $\forall t\ ds_t \leq ds$,

- le_t, le nombre de langages d'interaction d'entrée dont l'utilisateur a la choix à l'instant t : $\forall t\ le_t \leq le$,

- ls_t, le nombre de langages d'interaction de sortie dont le système a le choix à l'instant t : $\forall t\ ls_t \leq ls$.

A l'instant t, le système impose une situation d'obligation ou offre un ensemble d'options. On assiste à un choix :

- obligatoire de dispositif d'entrée (ou de sortie) si $de_t = 1$ (ou si $ds_t = 1$),

- obligatoire de langage d'entrée (ou de sortie) si $le_t = 1$ (ou si $ls_t = 1$),

- optionnel de dispositifs d'entrée (ou de sortie) si $de_t > 1$ (ou si $ds_t > 1$),

- optionnel de langages d'entrée (ou de sortie) si $le_t > 1$ (ou si $ls_t > 1$).

La figure 4.7 illustre les cas possibles de choix pour une orientation donnée des échanges (entrée ou sortie). La figure 4.8 explicite les seize configurations possibles. La classification notée (a) illustre la question des choix offerts à l'utilisateur pour une intention donnée (interface en entrée). La classification notée (b) correspond aux choix du système pour définir la présentation d'une unité informationnelle (interface en sortie). La combinaison des deux classifications implique la définition de seize classes de systèmes présentées en (c). Ce faisant, nous mettons en évidence les dissymétries possibles des espaces de choix offerts à l'utilisateur en entrée de celui que le système est capable de gérer en sortie.

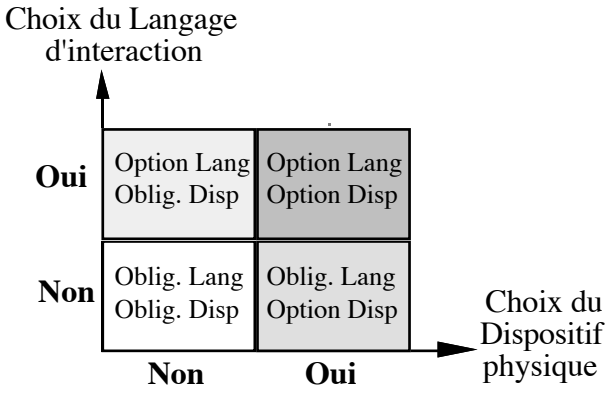


Figure 4.7 : O2LD, l'espace des possibilités de choix à un instant t en matière de langage d'interaction et de dispositif physique. Les valeurs "oui" et "non" indiquent l'existence ou l'absence de choix, c'est-à-dire le caractère optionnel ou obligatoire du choix à l'instant t.

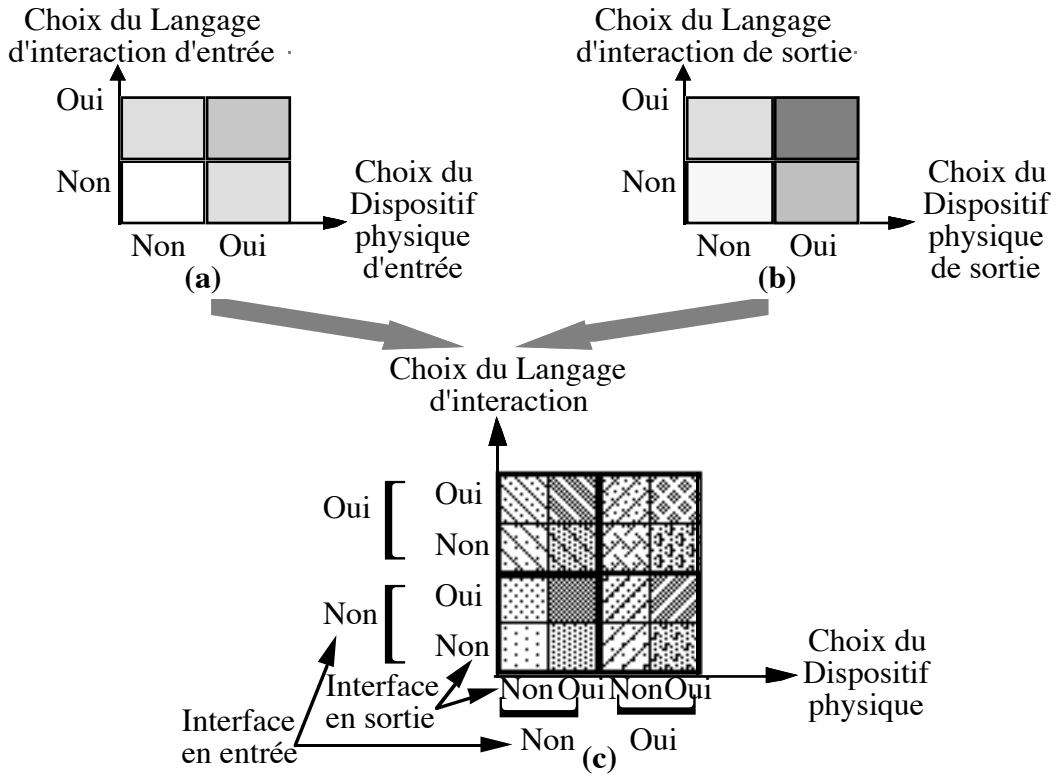


Figure 4.8 : O2LD et ses 16 classes de système selon les espaces des choix offerts en entrée et en sortie à un instant t en matière de langage d'interaction et de dispositif physique.

Au paragraphe suivant, nous approfondissons l'analyse des quatre grands couples de possibilité de la figure 4.7.

4.2. Les classes de O^{2}L D

La figure 4.9 traduit les quatre combinaisons de choix sous forme de relations entre langages et dispositifs. Les conventions sont les suivantes : la présence d'un seul carré ou d'un seul cercle traduit le caractère obligatoire du langage d'interaction ou du dispositif. Plusieurs carrés ou cercles expriment la possibilité de choix. Notons qu'un cercle peut en encapsuler d'autres pour exprimer l'association immuable de plusieurs dispositifs physiques ou virtuels.

Par exemple dans Munix, le langage naturel oral est obligatoirement combiné à l'écrit pour la saisie des noms d'objet (tels les noms de fichiers que le système de reconnaissance de la parole, sans capacité d'apprentissage, ne peut comprendre). Ainsi, dans certains contextes, le langage naturel est nécessairement associé au couple "microphone, clavier". De même dans TAPAGE, la saisie d'une donnée peut se faire en utilisant le stylo conjugué au clavier virtuel simulé sur écran. Ce faisant, la figure 4.9 explicite pour chacune des classes, le caractère monovalué ou multivalué de la relation entre un langage d'interaction et un dispositif.

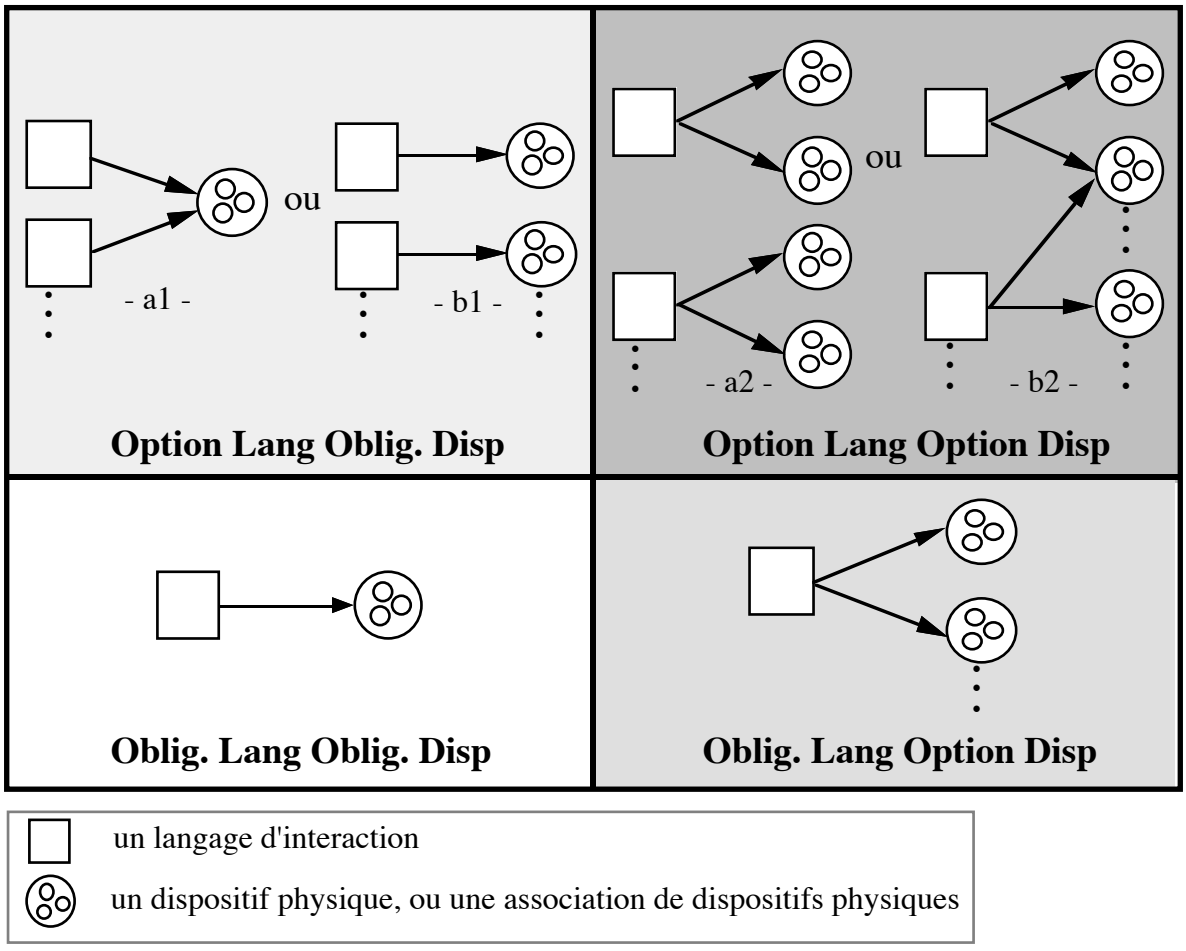


Figure 4.9 : Les quatre grandes classes de systèmes dans O^{2}LD et les relations entre langage et dispositif à un instant donné.

4.2.1. Classe "Oblig. Lang Oblig. Disp"

Les systèmes qui, à l'instant t, relèvent en entrée de la classe "**Oblig. Lang Oblig. Disp**" contraignent leurs utilisateurs à l'usage d'un seul langage mais aussi d'un seul dispositif d'entrée. En sortie, de tels systèmes n'ont, en cet instant, aucune capacité de choix du langage ni du dispositif de restitution.

Dans le système MATIS par exemple, si l'intention de l'utilisateur est de fermer une fenêtre de résultats, il ne lui est offert aucun choix : il devra sélectionner l'icône de fermeture de l'en-tête de la fenêtre. Cette procédure correspond au langage qui régit la manipulation directe et le dispositif physique est la souris.

De même pour l'interface en sortie de MATIS, si l'unité informationnelle correspond aux résultats d'une requête sur les vols entre deux villes, la seule présentation possible est une table affichée dans une nouvelle fenêtre. L'organisation de cette table ainsi que la forme des informations affichées sont définies par un langage d'interaction de sortie unique. De plus, quelle que soit l'unité informationnelle à présenter, le seul dispositif physique de sortie du système MATIS est l'écran graphique.

4.2.2. Classe "Oblig. Lang Option Disp"

Un système qui a un instant donné répond à la définition de la classe "**Oblig. Lang Option Disp**" offre un seul langage d'interaction mais y associe plusieurs dispositifs physiques. C'est le cas de systèmes qui n'autoriseraient en un instant donné que le langage naturel mais qui permettraient en entrée de saisir les phrases au clavier ou de les énoncer oralement. En sortie, ils afficheraient un texte à l'écran ou émettraient une phrase sur le haut-parleur.

4.2.3. Classe "Option Lang Oblig. Disp"

De façon inverse, un système de la classe "**Option Lang Oblig. Disp**" offre plusieurs langages d'interaction à un instant donné, mais le choix du langage fixe le dispositif. Cette définition n'implique cependant pas que le dispositif physique soit unique. Il peut être différent selon le langage d'interaction.

MMI2 peut relever de cette catégorie en sortie. Considérons par exemple la présentation d'une unité informationnelle véhiculant une liste de prix. MMI2 a le choix entre trois langages : le langage naturel écrit, une table ou un histogramme. Quel que soit le langage choisi, le

dispositif de restitution est nécessairement l'écran graphique. MMI2 illustre le cas -a1- de la figure 4.9.

CUBRICON doit choisir en sortie entre le langage naturel oral et une forme graphique de présentation. Là aussi, après avoir sélectionné le langage, aucun choix ne subsiste quant au dispositif : le haut-parleur pour le langage naturel et l'écran pour les restitutions graphiques. Contrairement à MMI2, les dispositifs physiques sont ici différents d'un langage à l'autre. CUBRICON correspond au cas -b1- de la figure 4.9.

4.2.4. Classe "Option Lang Option Disp"

Un système qui relève de la classe "**Option Lang Option Disp**" présente à la fois les caractéristiques des catégories "Oblig. Lang Option Disp" et "Option Lang Oblig. Disp".

C'est le cas de MATIS qui autorise la spécification d'une requête en plusieurs langages d'interaction distincts et notamment le langage naturel. Le langage naturel une fois fixé, l'utilisateur a encore le choix du dispositif physique : énoncer un message au moyen du microphone ou saisir le texte au clavier. La même unité informationnelle en sera déduite quel que soit le dispositif d'entrée utilisé. On se reportera à la figure 3.11 du chapitre III pour une description plus complète des possibilités de choix dans MATIS.

On relève un comportement similaire dans Munix, ICPplan, CUBRICON et MMI2 qui offrent tous le langage naturel écrit et oral. CUBRICON dispose de plusieurs langages de présentation graphiques mais, en certains points de l'interaction, le choix du dispositif subsiste entre l'écran monochrome et l'écran couleur. Notons chez ICPplan une illustration simple du modèle de l'utilisateur qui permet au système de calquer le choix du dispositif de sortie sur celui de l'utilisateur.

4.3. Relations entre classes

Après avoir présenté les propriétés de chaque classe de O^2LD au regard des relations qu'elles entretiennent entre les langages et les dispositifs, il convient de préciser les liens entre O^2LD et M^2LD sa parente (paragraphe 4.3.1). Il est également utile d'analyser les relations entre les classes même de O^2LD (paragraphe 4.3.2).

4.3.1. Les relations entre les classes de M²LD et de O²LD

M²LD permet une caractérisation statique d'un système en termes de choix de langage et de dispositif tandis qu'O²LD en permet une caractérisation instantanée. La figure 4.10 montre les relations d'inclusion entre les classes de ces deux systèmes de classification. Par exemple, un système de la classe "Mono Lang Multi Disp" est, à tout instant, soit de type "Oblig Lang Oblig Disp" soit de type "Oblig Lang Option Disp".

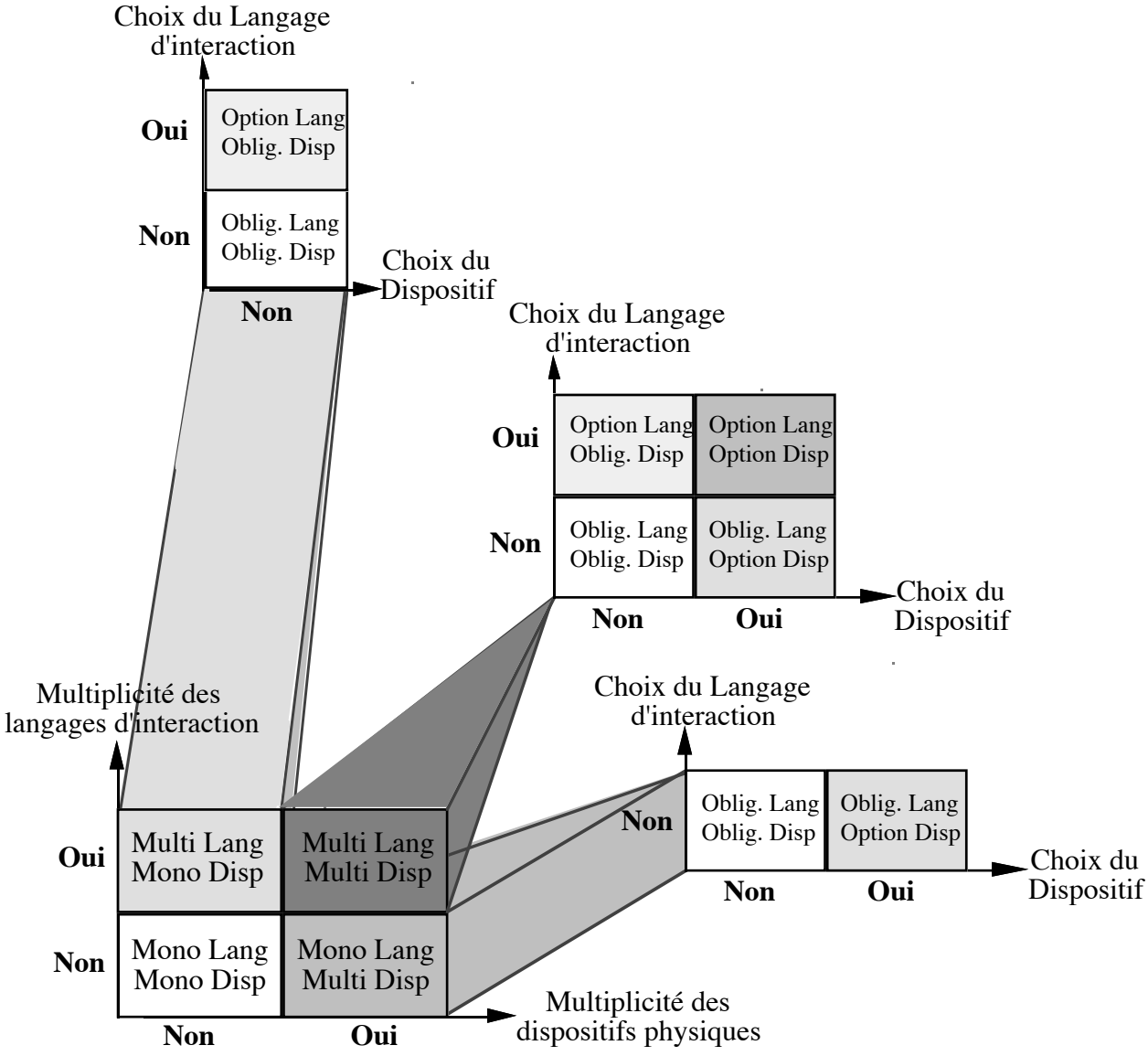


Figure 4.10 : Relations entre les classes de M²LD et de O²LD.

4.3.2. Relations entre les classes de O^2LD

La figure 4.11 montre les relations d'inclusion entre les classes de O^2LD. Considérons par exemple un système qui a l'instant t relève de la classe "Option Lang Option Disp". En entrée, un tel système peut aussi être utilisé en cet instant comme un système "Oblig. Lang. Oblig. Disp". Symétriquement, en sortie, il comporte tous les traitements nécessaires pour appartenir à la classe "Oblig. Lang. Oblig. Disp".

De manière générale,

- en sortie, un système S de classe o^2ld_S est plus **puissant à l'instant t** qu'un système S' de classe o^2ld_S' si $o^2ld_S \supset o^2ld_S'$.

- en entrée, un système S de classe o^2ld_S **peut s'utiliser à l'instant t comme** un système S' de classe o^2ld_S' si $o^2ld_S \supset o^2ld_S'$.

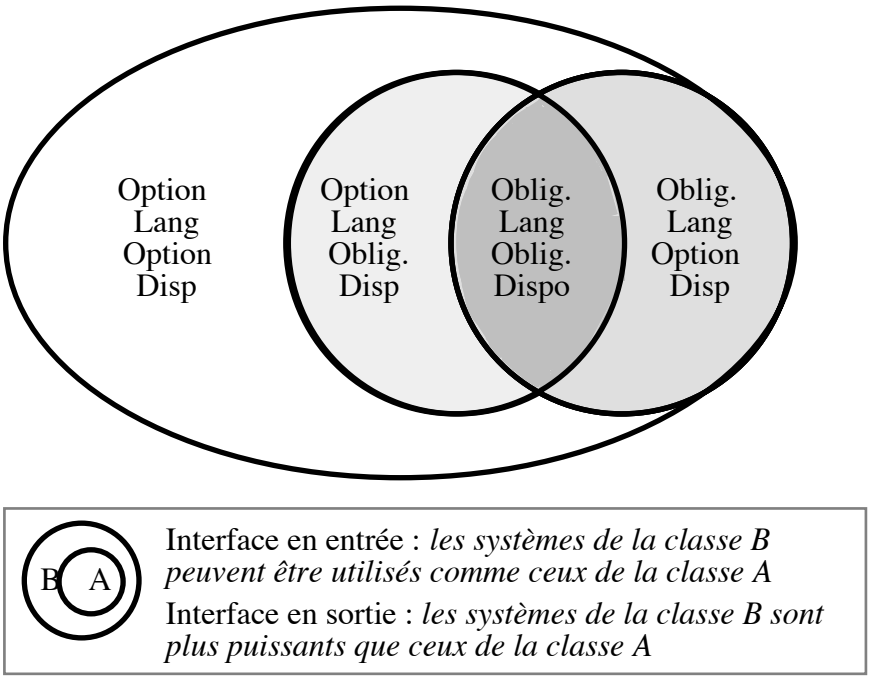


Figure 4.11 : Relations d'inclusion entre les classes de systèmes de la taxonomie O^2LD.

O^2LD tel que nous l'avons défini, fournit une classification instantanée des systèmes en terme de choix de langages et de dispositifs. Dans ce qui suit, nous étudions comment à partir d'O^2LD il est possible d'aboutir à une classification atemporelle.

4.4. Classifications empirique et analytique de système avec O2LD

Deux approches de classement sont envisageables. La première, empirique, consiste à déterminer des points d'observation pertinents et à examiner le système en ces points. La seconde, analytique, se fonde sur un modèle de l'utilisation du système et permet de raisonner sur l'interface de façon globale. Ces deux méthodes sont successivement examinées en prenant O2LD comme point de départ.

4.4.1. Méthode empirique

Comme le montre la figure 4.12, l'espace de classification discret de O2LD est étendu en espace continu. Nous relevons les quatre points remarquables qui dénotent de manière biunivoque les classes de O2LD. A chaque instant t_i d'observation, un poids est attaché au point remarquable qui correspond à la classe du système en cet instant. La position C du système dans cet espace est la position moyenne obtenue selon la formule :

$$\mathbf{C} = \frac{1}{\Sigma_t} \times \sum_i p_i \times t_i \qquad \Sigma_t = \sum_i t_i$$

Cette méthode que nous avons appliquée pour le classement d'un système dans l'espace IHMM'91 est présentée au chapitre II. Elle est complètement décrite dans [Nigay 93b].

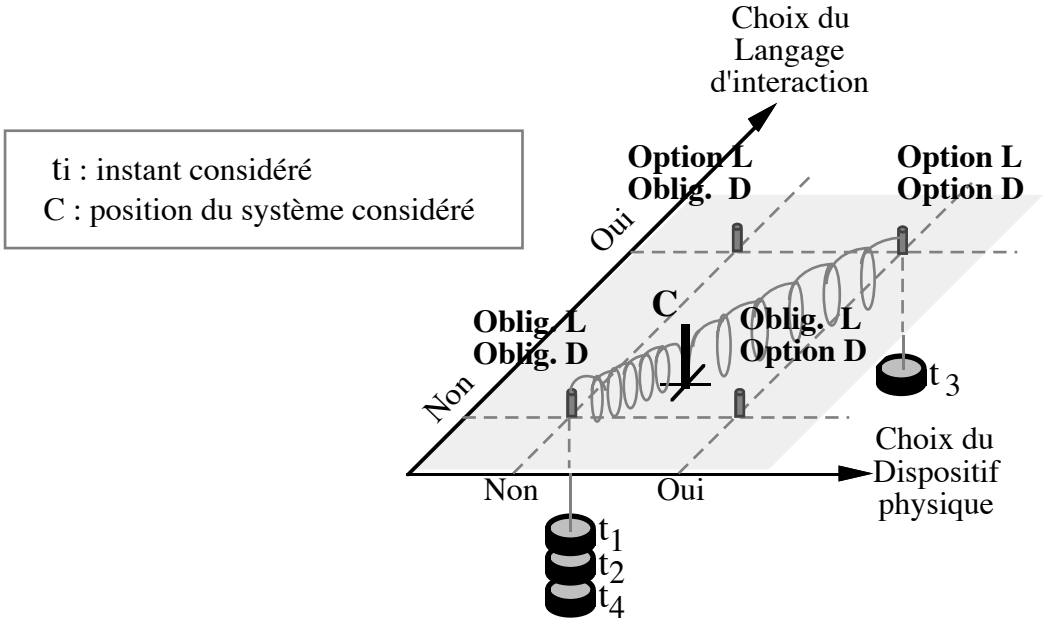


Figure 4.12 : Classification empirique d'un système dans un espace O2LD continu .

L'illustration de la figure 4.12 correspond à l'observation d'un système en quatre instants distincts t_1, t_2, t_3 et t_4. Aux instants t_1, t_2 et t_4, le système appartient à la classe

"Oblig. Lang Oblig. Disp" tandis qu'en t_3, il est positionné au point "Option Lang Option Disp". Le point moyen C se situe au voisinage de la classe "Oblig. Lang Oblig. Disp. D".

Les méthodes de classification empiriques ont leurs limites. Dans notre cas, la position d'un système dépend des instants considérés. Par conséquent, il s'agit d'observer un système en des instants significatifs et donc de déterminer les tâches pertinentes. Pour MATIS, un scénario type serait une tâche de planification d'un voyage entre plusieurs villes avec des contraintes d'horaire, de repas etc. Un exemple de scénario est présenté en Annexe A et décrit dans [Nigay 93d].

Les scénarios pertinents une fois fixés, les résultats dépendent aussi des sujets observés. Dans MATIS, pour un même scénario, il est probable que deux utilisateurs adopteraient des procédures distinctes. Il conviendrait alors de conduire l'expérimentation sur une population représentative de sujets pour minimiser la dépendance du classement du système aux caractéristiques des utilisateurs[17].

4.4.2. Méthode analytique

Nous illustrons l'utilisation analytique de O^{2}LD au moyen de systèmes dont la position dans l'espace diffère significativement : MATIS, CUBRICON et ICPplan.

4.4.2.1. O^{2}LD analytique et MATIS

Nous commençons par étudier l'interface d'entrée de MATIS. Pour spécifier une requête, l'utilisateur a le choix entre trois langages d'interaction et trois dispositifs physiques. Pour spécifier une commande ou manipuler des objets graphiques, il n'y a au contraire qu'une possibilité : un langage de manipulation directe et la souris pour dispositif. La commande d'envoi d'une requête, qui peut s'exprimer en langage naturel oral, écrit, ou par manipulation directe, fait toutefois exception. Ces constatations nous conduisent à placer MATIS, quant à son interface d'entrée, à équidistance des quatre points remarquables de l'espace. Nous obtenons le point noté "MATIS e" dans la figure 4.13.

En sortie, MATIS offre deux langages d'interaction : le langage naturel écrit et la présentation graphique des résultats sous forme de table. Si les langages de sortie sont multiples, MATIS n'effectue pas véritablement de choix : le langage de sortie dépend

17 Tous les problèmes liés à la prise en compte des caractéristiques des utilisateurs participant à une expérimentation relève de la psychologie expérimentale, un vaste domaine de recherche.

directement du langage d'entrée. Le dispositif physique de sortie, l'écran, est unique. La position de MATIS en sortie, notée “MATIS s”, est par conséquent très proche du point "Oblig. L Oblig. D".

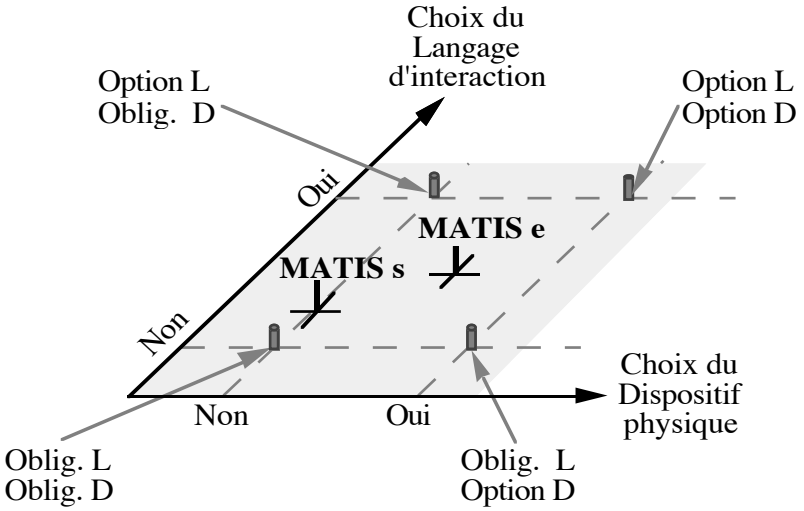


Figure 4.13 : Positions de MATIS (interface en entrée et en sortie) dans l'espace O^2LD.

4.4.2.2. O^2LD analytique et CUBRICON

L'une des motivations du projet CUBRICON est l'étude de la combinaison du langage naturel au geste de désignation. Dans cette association, le langage naturel domine et la manipulation directe lui sert de complément. En cela, le langage d'interaction est obligatoire. L’utilisateur a cependant le choix des dispositifs physiques d'entrée en écrivant la phrase au clavier ou en l'énonçant via le microphone. Au vu de ces caractéristiques, l’interface d’entrée de CUBRICON, notée “CUBRICON e” dans la figure 4.14, est proche du point "Oblig. L Option. D".

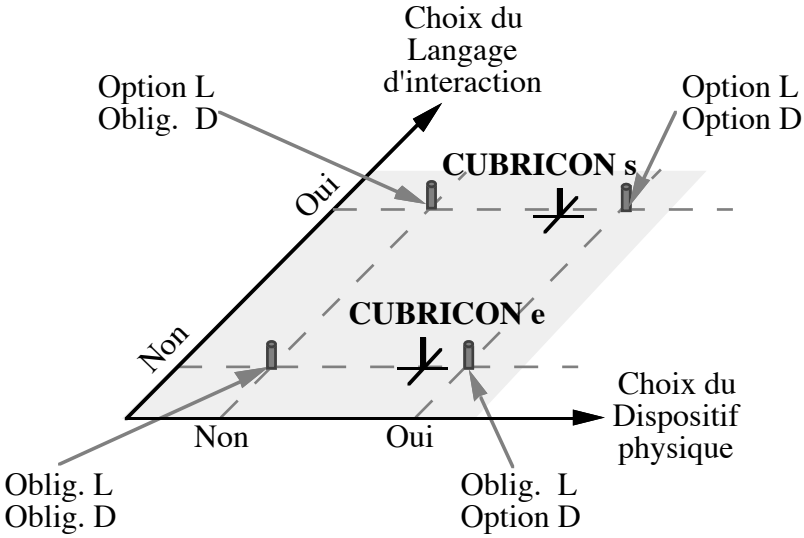


Figure 4.14 : Positions de CUBRICON (interface en entrée et en sortie) dans l'espace O^2LD.

L'interface de sortie, notée "CUBRICON s" est au contraire proche du point "Option L Option D". En effet, le système effectue le choix du langage d'interaction en fonction de l'unité informationnelle à présenter. Ce choix établit par exemple la forme de la présentation graphique (tableaux, camembert etc.). CUBRICON choisit de plus le dispositif physique de sortie à piloter : c'est le cas de l'affichage avec le choix de l'écran (monochrome ou couleur) et pour un message en langage naturel, une restitution orale ou affichée.

4.4.2.3. O2LD analytique et ICPplan

ICPplan offre le choix du langage pour toutes les commandes de dessin. En cela, il est situé sur l'axe vertical au point le plus haut (Option L). Pour les options possibles des dispositifs physiques, ICPplan est comparable à MATIS. En considérant l'interface en entrée, nous le situons au point "ICPplan e" de la figure 4.15.

Pour l'interface en sortie, ICPplan propose deux langages d'interaction. Pour l'un d'eux, le langage naturel, le système effectue le choix du dispositif physique : comme pour CUBRICON, le message est soit restitué oralement, soit affiché à l'écran sous forme d'un texte. Le critère de sélection est le dispositif que l'utilisateur vient de choisir en entrée. Par exemple, si l'utilisateur vient de parler, le message en sortie est oral. (Il est à noter qu'ICPplan affiche toujours le texte du message en parallèle de la production orale.) La position d'ICPplan pour son interface de sortie est notée "ICPplan s" dans la figure 4.15.

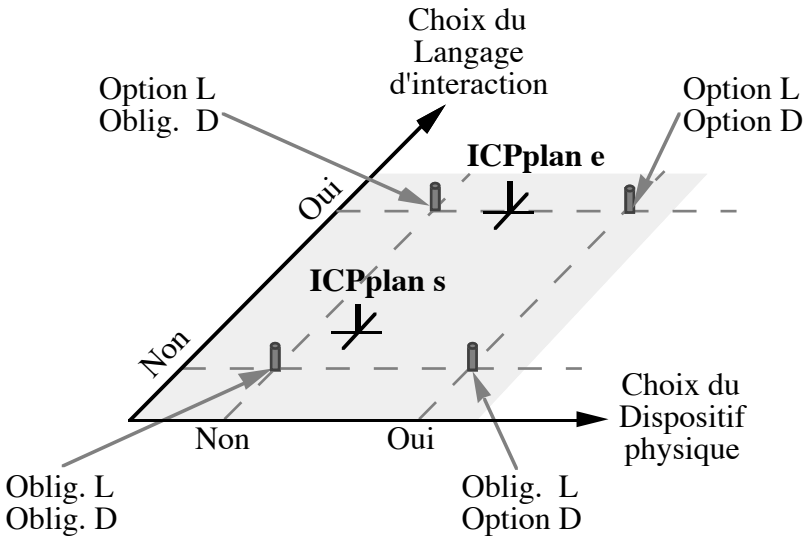


Figure 4.15 : positions d'ICPplan (Interface en entrée et en sortie) dans l'espace O2LD.

4.5. O2LD : en résumé

O2LD fournit, dans ses fondements, quatre classes de système qui se différencient par leurs capacités, à un instant donné, à effectuer ou à offrir des choix en matière de langages et de dispositifs. Ces choix varient en fonction des intentions communicationnelles de l'utilisateur mais aussi du système. De ce modèle instantané, il est possible de dériver une classification absolue fondée sur l'observation empirique ou sur l'analyse.

L'approche empirique pose le double problème de la définition des instants d'observation et du choix de sujets représentatifs. La méthode analytique, prédictive mais au risque d'être réductrice, consiste en une réflexion générale sur les choix offerts. Nous retrouvons ces deux approches duales en ergonomie pour l'évaluation des interfaces [Senach 90].

5. ULD : UNE TAXONOMIE SELON LES USAGES DES LANGAGES ET DES DISPOSITIFS

ULD (Usage des Langages et des Dispositifs) ouvre une perspective de classification orthogonale à celle d'O2LD. Alors qu'O2LD centre l'analyse sur les capacités de choix en matière de langage et de dispositif, ULD oriente l'examen sur les combinaisons temporelles de ces choix. Nous présentons ci-dessous les dimensions de l'espace ULD que nous illustrons ensuite avec nos systèmes exemples.

5.1. Les dimensions de l'espace ULD

Comme le montre la figure 4.16, les dimensions d'ULD traduisent l'utilisation (simultanée ou séquentielle) de plusieurs langages d'interaction, l'utilisation combinée ou indépendante des langages, et la mise à contribution exclusive, concurrente, alternée ou synergique, des dispositifs physiques. En reprenant la terminologie de IHMM'91 présentée au chapitre II, nous définissons quatre types d'usage remarquables de langage et/ou de dispositifs : exclusif, concurrent, alterné, synergique.

- exclusif signifie "absence de combinaison et usage séquentiel",
- concurrent entraîne "absence de combinaison mais usage simultané",
- alterné implique "combinaison mais usage séquentiel",
- synergique recouvre "combinaison et usage simultané".

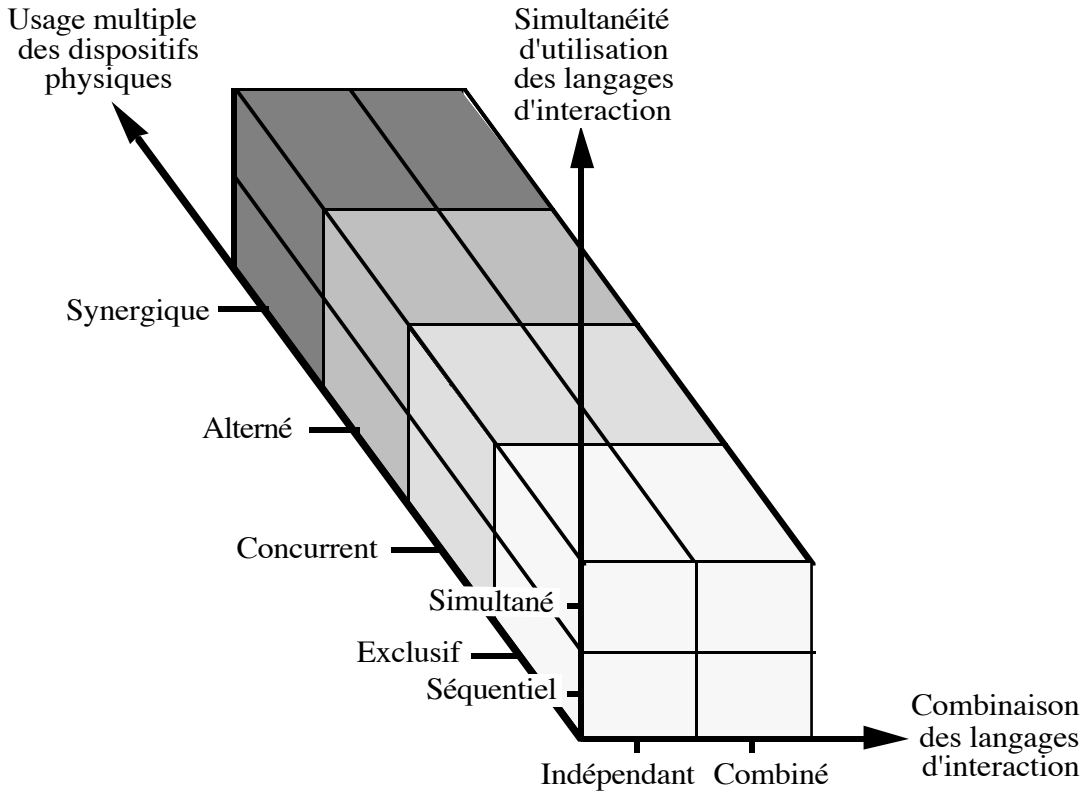


Figure 4.16 : ULD, une classification des systèmes selon les usages des langages et des dispositifs.

ULD a le mérite de mettre en relief l'indépendance de l'usage des langages de celui des dispositifs physiques. En particulier, un système "Multi Lang Multi Disp" dans la classification M^2LD, peut relever de la classe "synergique" pour l'usage des langages d'interaction mais de la classe "exclusif" pour celui des dispositifs physiques. Un tel système autoriserait par exemple la saisie au clavier d'une phrase dont la forme reposerait sur plusieurs langages. Inversement, un système qui appartient à la classe "exclusif" pour l'usage des langages d'interaction peut être synergique du point de vue de l'usage des dispositifs. C'est le cas de Munix qui offre le langage naturel oral complété à l'écrit par la saisie au clavier des noms d'objets. La figure 4.17 montre les relations possibles entre les classes M^2LD et celles d'ULD.

La nature des usages offerts trouve son reflet dans les fonctions du modèle Pipe-Lines. Au paragraphe 5.3 du chapitre III, nous avions évoqué les notions de parallélisme, de fusion et de fission. La figure 4.18 montre les liens entre ces phénomènes internes et les usages observables des langages et des dispositifs. On constate, par exemple, que l'usage synergique des dispositifs de sortie implique une fission des unités informationelles et le parallélisme des actions de pilotage des dispositifs de restitution (figure 4.18 -a). Inversement, l'usage synergique des dispositifs d'entrée entraîne le parallélisme et la fusion des actions physiques (figure 4.18 -b).

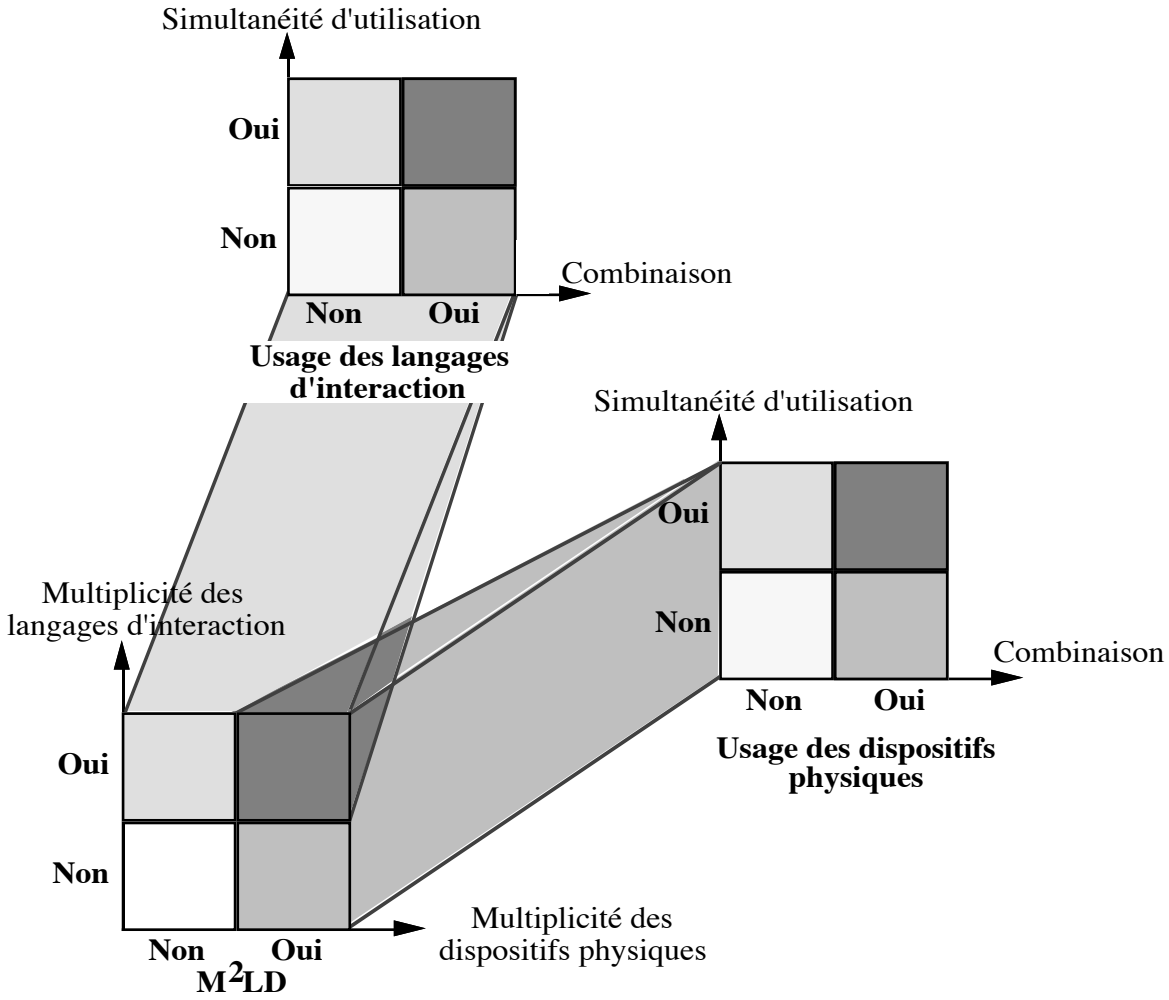
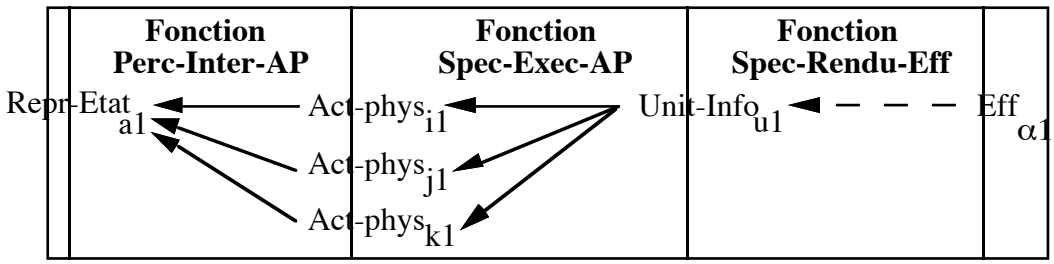
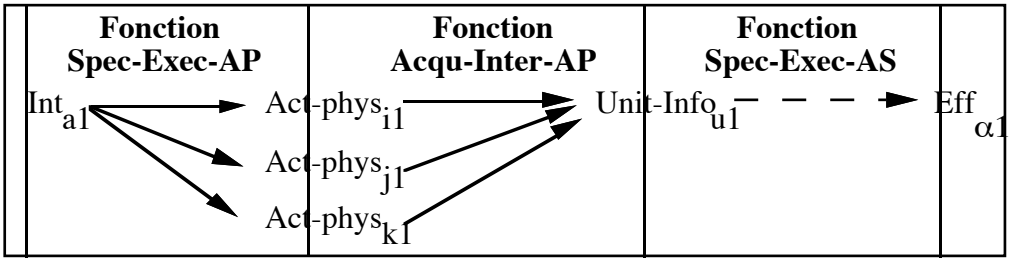


Figure 4.17 : Relations entre les classes M^2LD et ULD.



-a- Usage synergique des dispositifs de sortie



-b- Usage synergique des dispositifs d'entrée

Figure 4.18 : Usage synergique des dispositifs physiques.

Nous illustrons maintenant les classes ULD au moyen de nos exemples types.

5.2. Les systèmes exemples dans ULD

Nous organisons l'illustration d'ULD en considérant les usages en entrée puis en sortie des langages et dispositifs de nos systèmes exemples. Nous soulignerons la distinction qu'il convient de relever entre l'usage des langages et l'usage des dispositifs.

5.2.1. Interface en entrée

En entrée, MATIS offre l'usage synergique des langages et des dispositifs. L'énoncé oral *Flights from Pittsburgh to this city* accompagné de la sélection d'une ville au moyen de la souris, est un usage synergique de deux langages (naturel et manipulation directe) et de deux dispositifs (microphone et souris). Il peut aussi y avoir usage exclusif de langage mais synergie de dispositifs. Par exemple, l'utilisateur articule une phrase en langage naturel, *Flights from Pittsburgh*, tout en saisissant au clavier un texte en langage naturel, *serving a meal and arriving in the afternoon* .

De la même manière que MATIS, Munix offre l'usage synergique des langages et des dispositifs : par exemple l'utilisateur peut dire oralement la phrase *Peux-tu me montrer page par page?* accompagnée de la sélection de l'icône du dossier à visualiser. Munix permet aussi un usage exclusif de langage complété d'un usage synergique de dispositifs lorsqu'une commande orale nécessite la saisie au clavier de noms d'objet.

TAPAGE est un système synergique pour l'usage des langages et des dispositifs dans le cas où l'utilisateur articule la phrase "*Mets ça là*" tout en sélectionnant avec le crayon l'objet à déplacer et le nouvel emplacement.

MMI2 offre un usage alterné des langages et des dispositifs : par exemple l'utilisateur sélectionne l'icône d'un ordinateur sur le réseau puis saisit la phrase en langage naturel : *What is the cost of this machine?* Ici, deux langages et deux dispositifs sont utilisés de façon alternée.

5.2.2. Interface en sortie

ICPplan est un exemple de système synergique pour l'usage des dispositifs de sortie mais est exclusif ou concurrent dans son usage des langages. En effet, tout message oral en langage naturel est accompagné de sa représentation textuelle. Il y a alors usage combiné de

l'écran et du haut-parleur pour le seul langage naturel. Le langage graphique qui sert à la restitution des concepts du domaine, est utilisé de manière concurrente au langage naturel qui permet l'expression des messages.

CUBRICON présente les caractéristiques inverses. Il est synergique dans l'usage des langages de sortie sans l'être pour les dispositifs physiques. Par exemple, il peut afficher un tableau et l'accompagner d'une légende en langage naturel comme *Le tableau des prix est affiché ci-dessus*. Deux langages de sortie sont utilisés en synergie pour un dispositif physique unique : l'écran. On relève des exemples similaires dans MMI2.

5.3. Conclusions

ULD s'appuie sur les dimensions de l'espace IHMM'91 mais en précise la portée. Dans IHMM'91, l'applicabilité des concepts d'exclusion, de synergie, de concurrence, d'alternance, était sous-spécifiée voire ambigüe. Avec ULD, nous montrons comment ces notions caractérisent de manière orthogonale l'usage possible, à un instant donné, des langages et des dispositifs.

6. LA SOUPLESSE "LANGAGE ET DISPOSITIF" DANS LES SYSTÈMES À CARACTÉRISTIQUES MULTIPLES

L'utilisabilité, bien que sous-estimée par les praticiens du génie logiciel, est un facteur pertinent de la Qualité [McCall 77]. Dans [Abowd 92], nous structurons l'utilisabilité en termes de robustesse et de souplesse et nous mesurons la souplesse d'un système par sa capacité à permettre des choix : choix offerts à l'utilisateur ou pratiqués dynamiquement par le système en fonction du contexte. Dans notre étude préliminaire, nous explicitons ces choix en termes de préemption, de représentation multiple d'un même concept, d'égale opportunité, d'adaptativité, de multimodalité, etc. et nous observons que ces critères concernent plusieurs niveaux d'abstraction depuis les dispositifs physiques jusqu'au noyau fonctionnel. La limite de ce travail est l'absence d'éléments de mesure de ces critères.

La méthode UOM et ses trois volets, M^2LD, O^2LD et ULD, offrent une métrique d'évaluation de la souplesse d'un système en termes de choix de langages et de dispositifs. Sans vouloir développer trop avant les liens avec notre travail antérieur sur l'utilisabilité et ses critères, nos notions de dispositif, de langage et d'unité informationnelle dénotent trois niveaux d'abstraction distincts dont la nature était sous-spécifiée dans [Abowd 92]. Un bon nombre de critères telles la représentation multiple, la multimodalité, l'égale opportunité, etc. peuvent se projeter en termes de langages et de dispositifs. UOM permet aussi d'unifier certains de ces

critères sous forme de nouvelles relations qualifiantes entre langages et entre dispositifs : l'équivalence, l'assignation, la redondance et la complémentarité.

L'équivalence et l'assignation ont trait à l'existence de choix. La redondance et la complémentarité se rapportent à la combinaison de choix. Dans ce qui suit, on appelera expression ou phrase, une suite d'actions (utilisateur ou système) qui fait sens pour un langage (d'entrée ou de sortie).

6.1. Equivalence de langages et équivalence de dispositifs

L'équivalence est une relation qui se définit entre plusieurs langages ou entre plusieurs dispositifs. Pour chacun de ces cas, il convient de distinguer l'équivalence totale de l'équivalence partielle.

L'équivalence entre plusieurs langages d'un système est totale lorsque toutes les unités informationnelles de ce système peuvent s'exprimer dans chacun des langages. Elle est partielle lorsqu'un sous-ensemble seulement des unités informationnelles sont exprimables dans chacun des langages. Dans MATIS, il y a équivalence totale entre le langage naturel et le langage de manipulation directe des requêtes. L'équivalence entre langages implique, dans la taxonomie M^2LD, que le système soit de type "Multi langage" et, dans l'espace O^2LD, qu'il soit de type "Langage Optionnel".

Plusieurs dispositifs sont totalement équivalents pour un langage donné, si toutes les expressions permises par ce langage sont spécifiables avec chacun d'entre eux. L'équivalence est partielle si l'équivalence s'applique à un sous-ensemble des expressions du langage. Dans MATIS et bien d'autres, le clavier et le microphone sont totalement équivalents pour le langage naturel. L'équivalence entre dispositifs entraîne, dans la taxonomie M^2LD, que le système soit de type "Multi Dispositif" et "Dispositif Optionnel" dans l'espace O^2LD.

6.2. Assignation

L'assignation est une relation qui se définit entre un langage et une classe d'unités informationnelles d'un système. Elle peut aussi exister entre un dispositif et un langage.

Il y a assignation entre une classe d'unités informationelles et un langage lorsque cette classe ne peut s'exprimer que dans ce langage. Par exemple, dans MATIS, les réponses aux requêtes ne sont exprimables qu'au moyen de tableaux. Un système "Mono Langage" est

nécessairement “assignant” pour toutes ses unités informationnelles. Dans O^2LD, l’assignation implique “Langage Obligatoire”.

On parle d’assignation entre un langage et un dispositif lorsque les expressions de ce langage ne peuvent être spécifiées qu’au moyen de ce dispositif. Dans MATIS, le langage des tableaux implique le choix de l’écran. Un système “Dispositif Obligatoire” à un instant donné, satisfait cette propriété à cet instant. Un système “Mono Dispositif” la vérifie à tout instant.

6.3. Redondance

La redondance est une relation qui se définit entre deux langages ou entre deux dispositifs. Elle repose sur l’équivalence.

Il y a redondance totale/partielle entre deux langages d’un système si ces langages sont totalement/partiellement équivalents, s’ils peuvent être utilisés de manière concurrente et si tout couple d’expressions équivalentes correspond à un exemplaire unique d’unité informationnelle. Dans MATIS, il y a redondance totale entre le langage naturel et le langage de manipulation directe : l’énoncé *flights to Boston* et la saisie concurrente et équivalente de *Boston* dans le champ de destination d’un formulaire donne naissance à un seul exemplaire de requête contenant Boston. Nous revenons sur cet exemple de redondance au chapitre VII.

Il y a redondance totale/partielle entre deux dispositifs pour un langage s’ils sont totalement/partiellement équivalents, s’ils peuvent être utilisés de manière concurrente et si tout couple d’actions équivalentes correspond à un exemplaire d’expression. Un tel fonctionnement s’applique par exemple en vision stéréoscopique. Un autre exemple possible est le montage “redondant” d’un microphone et d’une caméra qui observe le mouvement des lèvres pour augmenter la robustesse d’un système de reconnaissance de la parole.

6.4. Complémentarité

La complémentarité est une relation qui se définit entre deux langages ou entre deux dispositifs.

La complémentarité entre deux langages d’un système est totale (ou partielle) lorsque toutes les (ou un sous-ensemble des) unités informationnelles de ce système utilisent, pour leur construction ou leur restitution, des expressions non équivalentes dans ces langages. Typiquement, les expressions coréférencielles entre deux langages sont complémentaires. Dans MATIS, le langage naturel et la manipulation directe sont partiellement complémentaires :

articuler la phrase *flights from this city* ou *flights from* accompagné de la sélection de la ville est un exemple de complémentarité entre expressions de langages distincts. Les deux expressions sont nécessaires à la constitution de l'unité informationnelle "lieu de départ". Notons que la complémentarité autorise un usage synergique ou alterné de langages : il y a toujours combinaison mais les expressions peuvent être produites en parallèle ou de manière séquentielle.

La complémentarité entre deux dispositifs pour un langage donné est totale (ou partielle) lorsque toutes les (ou un sous-ensemble des) expressions de ce langage s'expriment au moyen d'actions non équivalentes avec ces dispositifs. Munix offre un exemple de complémentarité partielle entre le clavier et le microphone mais aussi entre le clavier et la souris pour le cas du langage naturel : les commandes en langage naturel qui font intervenir des noms d'objet imposent un usage complémentaire du microphone et du clavier (ou de la souris) pour la désignation des objets. (On observe ici l'équivalence du clavier et de la souris pour la tâche de nommage des objets. Cette équivalence est partielle : elle ne vaut pas pour toutes les tâches du système.)

6.5. Discussion

En résumé, l'équivalence et l'assignation caractérisent la portée des choix en matière de langage et de dispositif. La première joue en faveur de la souplesse, la seconde augmente le caractère préemptif du système. La redondance et l'équivalence qualifient les combinaisons des choix. La première intervient, selon l'intention communicationnelle, comme un facteur de souplesse ou de robustesse. La seconde, qui impose un usage complémentaire de langages et/ou de dispositifs, peut se voir comme un facteur restrictif. Sous cet angle, le paradigme "mets ça là" souvent cité comme modèle d'interaction "naturelle", apparaît d'une utilisabilité douteuse : il requiert la complémentarité de deux langages mais aussi l'assignation de deux dispositifs (typiquement, le microphone pour l'énoncé oral et la souris pour la désignation).

Notre analyse jusqu'ici centrée sur les choix et contraintes permet de mesurer un aspect de la souplesse d'un système dès sa conception. Une évaluation selon les mêmes critères peut être pratiquée de manière expérimentale avec tout ou partie du prototype en centrant l'analyse sur le comportement de l'utilisateur. Il est alors utile d'identifier l'usage que les sujets observés font effectivement des langages et des dispositifs. En particulier, il est intéressant d'expliquer, voire corriger, l'usage restrictif que font certains sujets des possibilités du système. Par exemple, des observations de type Magicien d'Oz, ont montré que les sujets, en phase exploratoire, utilisent volontiers l'équivalence et la redondance. Mais après avoir cerné les capacités du système, les solutions procéduralisées laissent une large place à l'assignation

[Mignot 93]. Dans ces phases d'évaluation, il est intéressant d'expliquer la raison de ces choix puis de revenir sur la conception, s'il n'est pas déjà trop tard!

7. CONCLUSION

Dans cette première partie du mémoire, nous avons présenté un état de l'art sur la communication multimodale et multimédia. Face à la complexité et à la confusion de la terminologie dans ce domaine, nous avons choisi deux notions simples, le langage d'interaction et le dispositif physique, comme fil directeur de nos espaces de raisonnement.

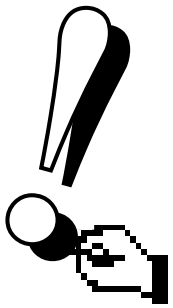
Le premier de nos espaces, le modèle Pipe-Lines, structure en étapes les échanges entre les entités communicantes que sont l'utilisateur et le système. On y voit le rôle des langages et des dispositifs, la place du contexte. Le modèle Pipe-Lines sert de fondement à la méthode UOM, dont les volets, usage, optionnalité et multiplicité, permettent de caractériser les systèmes interactifs en terme de choix et de composition de choix, de langages et de dispositifs. On obtient ainsi des espaces de classification statique ou instantanée, à la fois simples et riches, que nous avons illustrés par les systèmes exemples connus de notre communauté.

Avec la perspective "usage", nous avons aussi clarifié la terminologie imprécise d'IHMM'91 en l'appliquant aux combinaisons remarquables de langages et de dispositifs. Certaines d'entre elles se sont vues prolongées par une définition précise de critères qualifiant la souplesse d'un système : l'équivalence, l'assignation, la redondance et la complémentarité.

Langages et dispositifs constituent les points de rencontre de l'utilisateur et du système. Fondées sur ces concepts, UOM et ses dimensions classifiantes définissent des axes pour la conception d'interface mais aussi des critères de mesure de la souplesse de ces interfaces. Il convient maintenant d'apprécier ces concepts, axes et propriétés comme des éléments directeurs en conception logicielle d'interfaces. Cette étude fait l'objet de la seconde partie de ce mémoire.

PARTIE 2

ESPACE SOLUTION



"La science classique dissolvait la complexité apparente des phénomènes pour révéler la simplicité cachée des Lois immuables de la Nature. Aujourd'hui, la complexité commence à apparaître non pas comme l'ennemi à éliminer, mais comme le défi à relever."

- Edgar Morin -

Chapitre V

Interface Homme-Machine :

Cycle de développement et outils

"*People use computers to accomplish tasks. Consequently, understanding human capabilities and tasks is as important to the design of effective computer systems as understanding computer technologies.*"
- Bonnie E. John -

1. INTRODUCTION

Dans la première partie de ce mémoire, l'exploration de l'espace problème nous a permis de cerner les propriétés et les besoins des systèmes à caractéristiques multiples. Nous abordons maintenant le problème de la réalisation de tels systèmes en accord avec leurs propriétés. Ce chapitre étudie le support pratique de l'activité de réalisation par une revue des outils logiciels d'aide au développement. L'objectif est de situer, dans cet ensemble, nos travaux sur les architectures logicielles. Dès lors nous n'avons pas l'ambition de présenter un état de l'art exhaustif sachant aussi que de nombreuses études taxonomiques recouvrent déjà cet objectif. Il est intéressant néanmoins de rappeler quelques points de vue complémentaires.

Certains auteurs, distinguent les outils par la nature des concepts manipulés, d'autres par les classes de langage de spécification, d'autres encore par les niveaux de services offerts. Certains ne sont concernés que par les outils de spécification des interfaces, d'autres par les services et techniques de codage :

- V. Normand organise sa taxonomie en fonction du niveau des concepts manipulés par l'outil [Normand 92, chap. III] : concepts de gestion des ressources de l'interaction, techniques d'interaction, concepts de gestion du dialogue, services d'interfaces, concepts du domaine et concepts architecturaux. Nous retrouverons ces niveaux dans notre étude sur les modèles d'architecture.

- J. Nanard propose une classification centrée sur les langages de spécification [Nanard 90] et distingue les langages déclaratifs à événements, les diagrammes de transition, les réseaux de Petri [Palanque 92], etc. B. Myers [Myers 89] augmente ce point de vue en élargissant l'analyse aux techniques de spécification qui viennent s'ajouter au formalisme. Ainsi, il distingue les langages textuels des formalismes gestuels par démonstration. Dans ce dernier cas, la spécification se fait par manipulation directe [Cardelli 88].

- J. Coutaz met l'accent sur les niveaux des services offerts pour le codage des interfaces [Coutaz 90, Partie 3] : les boîtes à outils, les squelettes d'application et les environnements de développement.

Nous adopterons une approche complémentaire qui montre le point d'ancrage des outils dans les processus de développement utilisés en génie logiciel.

Les travaux taxonomiques cités ont en commun l'étude d'outils pour le développement d'interfaces purement graphiques. A l'heure actuelle, très peu d'outils abordent la réalisation de systèmes à caractéristiques multiples. Quand ils existent, l'approche la plus souvent adoptée est l'extension des méthodes et outils destinés aux interfaces usuelles. Le projet ARCHIE illustre cette technique [Smart 93].

ARCHIE permet d'étendre une interface graphique par le rajout de dispositifs physiques sans modifier le noyau fonctionnel. Les actions utilisateur sur les nouveaux dispositifs sont captées par le noyau ARCHIE, traduites en terme d'actions souris ou clavier, puis transmises au logiciel graphique. La solution ARCHIE, qui permet d'étendre sans modifier l'existant, suppose qu'il y ait équivalence totale, au sens où nous l'avons défini au chapitre précédent, entre les nouveaux dispositifs et le clavier ou la souris, et ceci pour l'ensemble des langages du système.

L'approche par extension a le mérite de réutiliser le savoir-faire mais il convient de ne pas réduire les propriétés des systèmes à caractéristiques multiples à celles des interfaces graphiques. Si cette approche se justifie en tant que technique de génie logiciel, elle nous paraît contestable en matière de conception ergonomique et d'évaluation. Du point de vue ergonomique, on sait qu'une interface à caractéristiques multiples n'est pas la somme de ses caractéristiques prises individuellement. En particulier, nous voyons émerger les problèmes de choix, et de combinaison de choix, en matière de langages et de dispositifs.

Ce chapitre est organisé ainsi : après une définition des différentes étapes du processus de développement d'une interface (section 2), nous analysons quelques outils représentatifs de chaque étape (section 3). Parmi ces outils, nous accorderons une place de choix au modèle d'architecture, objet central de notre recherche (section 4). Les classes d'outils présentées dans ce chapitre sont actuellement et principalement destinées à des interfaces graphiques. Dans le cas contraire, nous soulignerons les services spécifiques aux systèmes à caractéristiques multiples.

2. CYCLE DE DÉVELOPPEMENT : LES ÉTAPES

Le génie logiciel est concerné par la production et la maintenance de logiciels sous le contrôle d'un ensemble de contraintes. C'est dans les années 70 que l'on prit conscience que le processus de développement lui-même était mal défini et qu'il nécessitait un cadre structurel. C'est ainsi que Royce proposa le modèle en cascade [Royce 70] qui fut ensuite affiné sous diverses formes avec notamment le modèle en V [McDermid 84] puis, récemment, le modèle en spirale [Boehm 88].

Le modèle en spirale est en fait un métamodèle qu'il convient d'instancier en un processus de développement adapté au cas précis d'un logiciel donné. Son avantage tient à sa généricité mais aussi à l'introduction explicite d'activités de contrôle telles que l'analyse des coûts et des risques, la validation et la vérification. Le modèle en V, bien que simpliste, a l'avantage de spécifier les étapes clés de tout processus de développement ainsi que les documents qui s'y rattachent. Pour cette raison, nous l'utilisons comme support pédagogique pour la présentation des outils d'interfaces.

Comme le montre la figure 5.1, le modèle en V distingue l'analyse des besoins, la conception, le codage et les tests. Ces activités se décomposent à leur tour en étapes sur lesquelles nous viendrons greffer les outils d'interfaces.

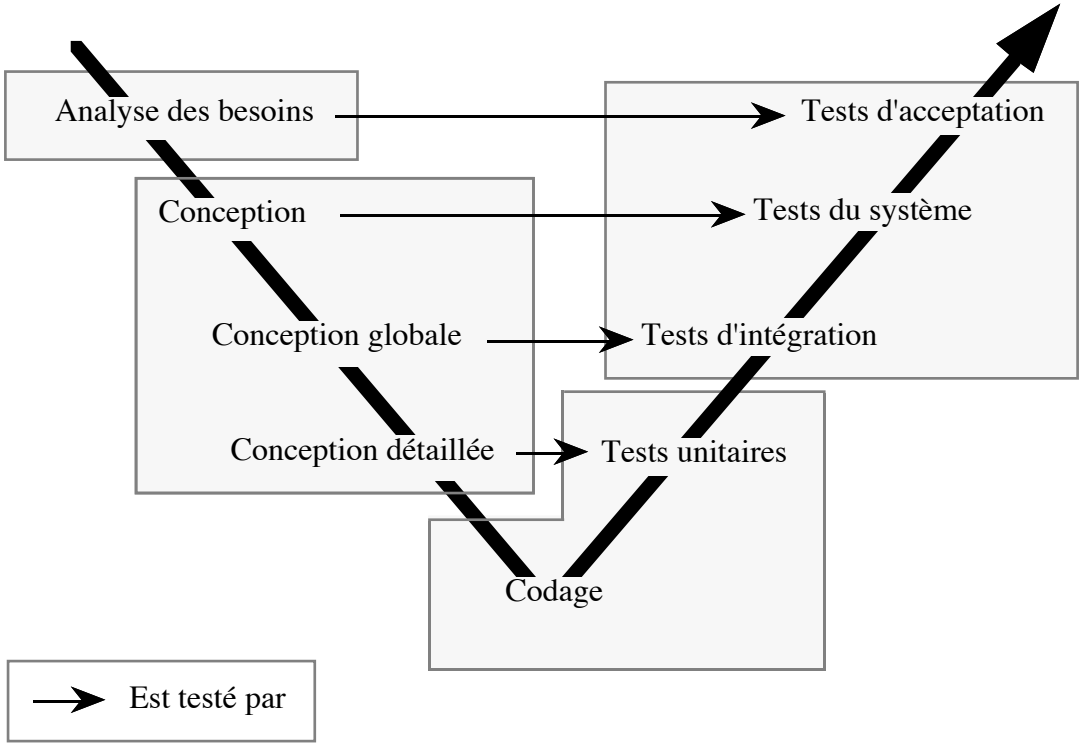


Figure 5.1 : Les étapes de conception et développement d'un système.

2.1. Analyse des besoins

L'analyse des besoins a pour objet la définition des services requis du système et des contraintes de développement. Cette activité, qui se fait en accord avec le client, donne lieu à un cahier des charges, document à valeur contractuelle entre le maître d'œuvre et le client. La structure du cahier des charges suit généralement une norme comme la norme AFNOR qui regroupe à la fois la définition et la spécification des besoins. La définition des besoins est

exprimée de manière informelle en langue naturelle ; pour la spécification, on utilise des schémas ou des diagrammes formels comme le propose SADT [Jaulent 89].

Dans l'activité d'analyse des besoins, il convient de bien distinguer les besoins fonctionnels des contraintes :

- les besoins fonctionnels décrivent les services attendus du système, les tâches qu'il doit permettre d'effectuer. Idéalement, l'identification des besoins passe par une analyse de tâche rigoureuse. Une activité en soi!

- les besoins non fonctionnels définissent les contraintes matérielles ou logicielles et les qualités requises du système. C'est ici que l'utilisabilité du système, évoquée au chapitre précédent, doit être cernée avec précision et consignée dans le plan qualité.

2.2. Conception

L'activité de conception consiste à définir une solution matérielle et logicielle qui répond à l'analyse des besoins. Elle comprend deux volets essentiels : la conception proprement dite, et les conceptions globale et détaillée. Pour un système interactif, la première étape inclut une étude ergonomique, tandis que les conceptions globale et détaillée sont résolument orientées vers la réalisation. La première requiert une compétence en psychologie et en ergonomie, les secondes un savoir-faire en informatique. Pour lever toute ambiguité sur le terme "conception", nous parlerons désormais de conception ergonomique et de conception logicielle.

La conception ergonomique du système se traduit, entre autres, par un document de spécifications externes. Ce dernier décrit le système tel qu'il sera perçu par l'utilisateur. Comme pour le cahier des charges, la description est généralement informelle. En particulier, la description de l'interface homme-machine est le plus souvent exprimée en langue naturelle et complétée de schémas d'écran.

La conception logicielle globale constitue l'étape préliminaire orientée vers la mise en œuvre logicielle. Elle produit un dossier d'architecture générale. Dans le cas d'un logiciel interactif, l'architecture générale devra s'appuyer sur le principe de séparation modulaire entre le noyau fonctionnel qui réalise les services du domaine, et l'interface homme-machine (IHM) qui gère les échanges avec l'utilisateur [Coutaz 90]. Ce principe est illustré à la figure 1.1 du chapitre I. Les modèles conceptuels d'architecture tels Seeheim, puis Arch [UIMS 92] et notre contribution, PAC-Amodeus [Nigay 91], aident à l'application de ce principe de base.

La conception logicielle détaillée décrit les choix algorithmiques, fixe la signature des procédures et des fonctions, définit les structures de données les plus pertinentes et spécifie les protocoles de communication. Elle peut être guidée par le principe de réutilisation logicielle ou encore se pratiquer par spécification formelle. L'avantage de la spécification formelle est la génération automatique de code conforme aux spécifications.

2.3. Implémentation et tests

L'implémentation ou codage traduit les spécifications issues de l'étape de conception logicielle en un code exécutable. Le codage constitue la dernière étape de la pente descendante du modèle en V.

Sur la pente ascendante, à chacune des étapes exposées ci-dessus, correspond un ensemble de tests qui permet de vérifier et/ou de valider[18] les étapes par rapport au code produit. Les tests unitaires permettent de vérifier que les composants modulaires du système répondent chacun à leurs spécifications. Les tests d'intégration servent à vérifier que les modules réalisés indépendamment interagissent correctement. Les tests du système permettent de vérifier que les éléments de la solution exprimés dans le dossier de spécifications externes sont présents. Les tests d'acceptation servent à vérifier que les besoins exprimés dans le cahier des charges du logiciel sont couverts. C'est le dernier contrôle avant la livraison du produit.

Bien qu'ils soient effectués en séquence une fois le codage réalisé, les tests doivent être définis en même temps que leur correspondant. Parfois, certaines vérifications ou validations sont effectuées lors des phases de la pente descendante, avant même le codage. Par exemple, une maquette logicielle permet de tester le bien fondé de certains choix ergonomiques, un story-board est utile à la vérification de l'enchaînement des tâches, etc.

---

18 La validation se différencie ainsi de la vérification [Boehm 81] :

- La ***validation*** s'interroge sur l'adéquation du logiciel produit : "are we building the right product?". L'adéquation n'est pas une caractéristique mesurable mais relève d'un jugement subjectif. Le génie logiciel rejoint ici les constatations de l'ergonomie sur la pertinence des données qualitatives.

- La ***vérification*** concerne la conformité du produit avec une description de référence : "are we building the product right?". La vérification n'a de sens que s'il existe un document de référence, par exemple, un dossier de spécifications détaillées.

3. OUTILS ET CYCLE DE DÉVELOPPEMENT

Notre présentation des outils procède selon les étapes du cycle en V.

3.1. Analyse des besoins et générateurs “descendants”

Les générateurs “descendants” tels UIDE [Foley 88], SIROCO [Normand 92] et ADEPT [Johnson 93], sont des outils de spécification de haut niveau d’abstraction. Ils produisent automatiquement le code exécutable de l’IHM à partir d’une description formelle des concepts et des tâches du domaine : ils procèdent de manière descendante depuis les concepts jusqu’aux objets de présentation. Comme le montre la figure 5.2, ces générateurs masquent les étapes de conception ergonomiques et logicielles. Ce faisant, ils augmentent les chances de conformité du code exécutable à la spécification des besoins et sont très efficaces pour la production de prototypes.

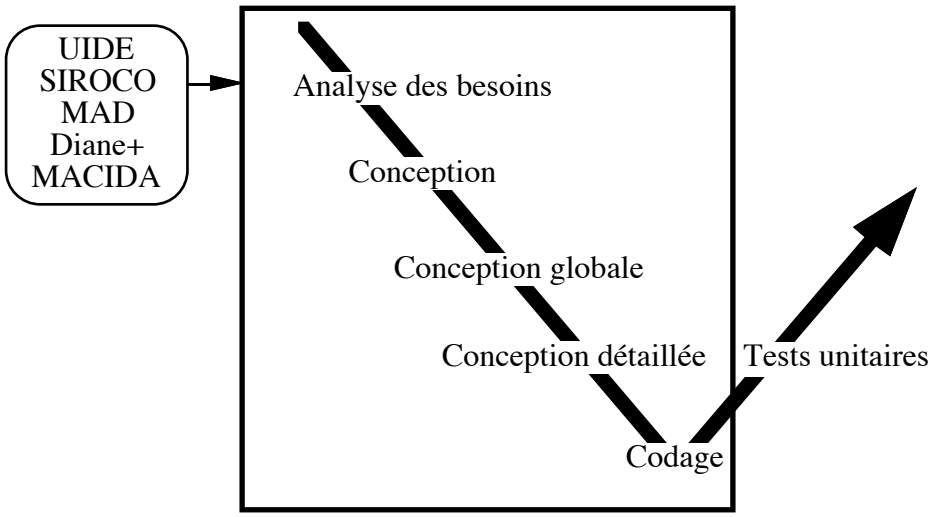


Figure 5.2 : UIDE,SIROCO, ADEPT, MAD, Diane+, MACIDA.

UIDE propose un langage de spécification des informations relatives au noyau fonctionnel sans considération pour leurs présentations. Le langage est donc fonctionnel. Par exemple, la définition d'une fonction du noyau fonctionnel se décrit par :

- ses conditions d'activation et de terminaison,
- ses paramètres décorés d’attributs tels que “optionnel”, “par défaut”,
- sa fonction inverse,
- les fonctions non disponibles au même instant.

A partir d'une telle description, UIDE prend en charge la génération du code de l'interface. Par exemple, une fonction peut être présentée comme l'option d'un menu dont la sélection entraîne l'ouverture d'un formulaire de saisie des paramètres. Ce principe de correspondance entre type de concept et type d'objet d'interaction est appliqué par la plupart de cette catégorie d'outils tels Mickey [Olsen 89], MacIda UIMS [Petoud 89], ITS [Wiecha 89]. Dans ITS toutefois, la correspondance n'est pas prédéfinie mais programmable au moyen de règles.

Dans SIROCO, la spécification conceptuelle d'un système comprend deux volets : le premier consiste à décrire le fonctionnement du système en termes d'objets conceptuels et de fonctions. Le second définit l'utilisation du système sous forme d'espaces de travail et de perspectives. Un espace de travail regroupe des tâches et des concepts logiquement connectés. Une perspective sur un concept définit les constituants pertinents du concept qu'il convient de présenter à l'utilisateur.

Sur le plan du génie logiciel, SIROCO va plus loin que l'environnement UIDE. Le code produit par SIROCO respecte une organisation rationnelle fondée sur l'approche à objets. Contrairement à UIDE, SIROCO obéit à un modèle d'architecture explicite. L'une des retombées d'une organisation à la fois systématique et orientée objet est de permettre l'amélioration ou la personnalisation "manuelles" de l'interface par retouche du code produit.

Sur le plan ergonomique, SIROCO n'offre pas les mêmes garanties de conformité à l'analyse de tâche que le système ADEPT [Johnson 93]. ADEPT est un environnement de développement qui s'appuie sur la méthode d'analyse de tâche TKS [Johnson 91]. A la différence de SIROCO qui exige du concepteur de transcrire dans son formalisme de spécification, l'analyse, éventuellement informelle, des tâches, ADEPT force d'emblée une spécification formelle de l'espace des tâches. Ce faisant, il garantit la conformité du code à l'analyse des besoins fonctionnels. MAD [Scapin 90] et Diane$^+$ [Tarby 93] visent des objectifs similaires.

En résumé, les outils de spécification de tâches augmentés de générateurs de code couvrent toutes les étapes de la branche descendante du cycle en V. En raison de la génération automatique, les tests de nature informatique sont superflus. Mais cette vision idéale du tout automatique fondé sur l'analyse de tâche convient à des domaines d'application pour lesquels les classes d'objets de présentation prêtes à l'emploi conviennent. Par souci de généralité, il est important que le code généré soit adaptable. Mais un code modifié manuellement pose le problème de modifications ultérieures faites par l'outil automatiquement.

3.2. Conception ergonomique et outils de spécifications externes

Les spécifications externes qui résultent de la conception ergonomique sont le plus souvent exprimées en langue naturelle. Il existe cependant des formalismes tels CLG [Moran 81], TAG [Payne 86], UAN [Hartson 92] qui répondent à ce besoin de formalisation. La figure 5.3 montre leur point d'intervention et leur portée dans le processus de développement.

Les uns, comme TAG et ses dérivés ETAG [Tauber 90], visent l'évaluation prédictive de l'utilisabilité des interfaces. L'expérience montre que les détections d'anomalies concernent, pour l'essentiel, des défauts de bas niveau d'abstraction comme l'absence de congruence dans les noms de commandes. Pour d'autres, tel UAN, la fonction première est d'éviter les ambiguïtés d'une spécification en langue naturelle. Une description formelle précise de l'interface est toujours précieuse. Elle est indispensable lorsque les phases suivantes, orientées logiciel, ne sont pas prises en charge par la même équipe.

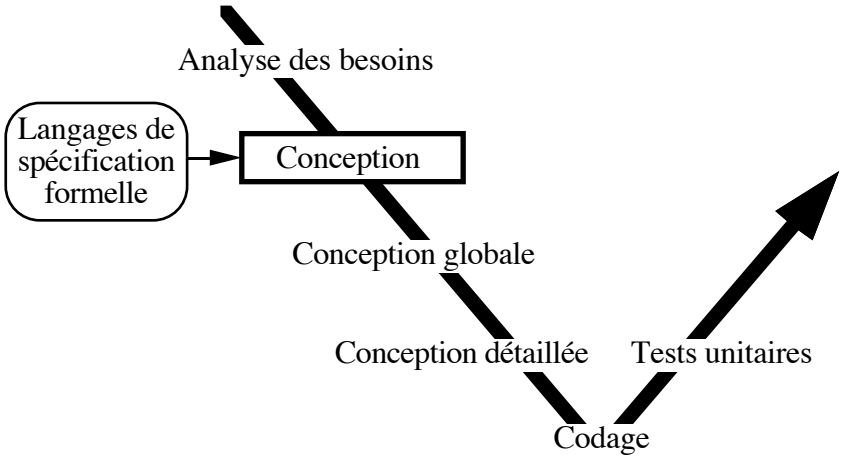


Figure 5.3 : Les langages de spécification formelle.

Les langages permettant la spécification du comportement perceptible d'un système sont consacrés aux interfaces graphiques. Nous avons, pour notre part, étendu le langage UAN pour l'appliquer aux interfaces à caractéristiques multiples [Coutaz et al. 93d]. On trouvera en annexe B, une partie des spécifications de l'interface de MATIS exprimée en UAN étendu. (Une version complète est disponible dans [Coutaz et al. 93e].)

Pour introduire le langage UAN et son extension, nous décrivons la tâche de spécification d'une requête MATIS dans le cas d'une expression orale comportant un déictique, par exemple *flights from this city*.. La coréfence est résolue par une désignation d'information au moyen de la souris. La première colonne décrit les actions physiques de l'utilisateur : par exemple ~[objet] désigne un déplacement de la souris sur l'objet. [objet]v correspond à la

pression du bouton de la souris et au contraire le symbole ∧ représente le relâchement du bouton de la souris. En particulier UAN offre la possibilité de décrire des actions physiques effectuées en parallèle, nécessaire pour décrire l'usage synergique des dispositifs physiques. Pour cela l'opérateur noté || dénote le parallélisme. (sentence, fps, 👄) représente une phrase articulée, "fps" désignant le contenu de la phrase (requête complète "*full*", incomplète"*partial*", ou envoi d'une requête "*show*").

Task: SpecRSpeech&Mouse (R, fps)	**is atomic**	
User Action	Interface Feedback	Interface State
SelAppli = MatisAppli ∧ SelR = R: case SpeechMode = Push&Hold ~[x,y in OMEarW]∨ ProduceDeicticSentence (sentence, fps, 👄) ∧ SelectValues	 OMEarW! *(Listening)* NLFeedback&State(sentence, 👄)	
SpeechMode = Continuous ProduceDeicticSentence (sentence, fps, 👄) ‖ SelectValues	 ($t > t_{silence}$) NLFeedback&State(sentence, 👄)	
SpeechMode = PushToStart ~[x,y in OMEarW]∨∧ ProduceDeicticSentence (sentence, fps, 👄) ‖ SelectValues ~[x,y in OMEarW]∨∧	 NLFeedback&State(sentence, 👄)	
endcase		

La première colonne traduit les trois manières de spécifier une requête en correspondance avec les trois modes de fonctionnement du système de reconnaissance de la parole Sphinx. Par exemple le dernier cas de la première colonne correspond au mode "PushToStart" de Sphinx, où l'utilisateur sélectionne l'icône Sphinx pour indiquer le début et la fin d'une phrase articulée. Entre les deux sélections, l'utilisateur spécifie d'autres valeurs qui seront combinées avec les informations de la phrase articulée. La sélection de l'icône Sphinx s'exprime par [OMEarW]∨∧. La deuxième colonne traduit les retours d'information associés

aux actions physiques. Ainsi la sélection de l'icône Sphinx représentant un microphone dans le mode "Push&Hold" provoque l'affichage d'une icône contenant une oreille et le texte "*listening*". Cette nouvelle icône signifie que le système de reconnaissance est en écoute. Ceci se traduit dans la deuxième colonne par OMEarW! *(Listening)*. La troisième colonne traduit l'état du système suite à une action physique de l'utilisateur. Par exemple la variable notée "SelAppli" qui sert de condition à la première colonne est une variable traduisant l'état du système.

Le langage UAN original présente quelques faiblesses sémantiques compensées par des apports intéressants :

- Comme le montre l'exemple ci-dessus, UAN permet l'expression de relations temporelles entre les tâches : parallélisme, entrelacement, attente avec délai de garde. Ces opérateurs, indispensables à la description des activités multiples de l'utilisateur, augmentent les expressions ET/OU des arbres de tâches usuels.

- La colonne état du système est un indicateur utile pour le concepteur logiciel : les variables utilisées pour exprimer un changement d'état interne sont des concepts pertinents de l'interaction qui devront, à l'exécution, être modélisés et maintenus dans le contexte.

- Le troisième avantage de ce langage tient à sa dimension formelle. A travers l'exercice de MATIS, nous avons indentifié plusieurs propriétés qu'un outil pourrait vérifier de manière automatique. En voici deux exemples :

Si la précondition de l'exécution d'une tâche n'est pas spécifiable en termes d'attributs perceptibles, alors l'utilisateur n'a pas les moyens de déterminer que cette tâche est exécutable. La propriété d'"*observabilité*" [Abowd et al. 92] n'est pas atteinte.

Si, à une action utilisateur (colonne de gauche d'une spécification UAN), ne correspond aucun retour d'information (colonne centrale d'une spécification UAN), le principe de "retour immédiat" est transgressé.

Bien que réalisée a posteriori, la spécification UAN de MATIS nous a permis d'identifier plusieurs erreurs de conception. En particulier, notre présentation du concept de dictionnaire actif transgresse le principe d'observabilité. A un instant donné, MATIS peut être actif sans que son dictionnaire le soit et cet état n'est pas décelable par l'utilisateur. Il peut donc arriver que l'utilisateur énonce une requête à destination de MATIS alors que le système de

reconnaissance utilise à cet instant-là d'un autre dictionnaire. La phrase ne sera pas comprise et l'utilisateur aura quelques difficultés à déterminer la raison.

En résumé, les formalismes de spécifications externes interviennent dans la phase de conception ergonomique. Ils permettent, de produire une spécification non ambigüe et précise du comportement perceptible du système. Ils peuvent en outre être prolongés d'outils de vérification automatique de propriétés ergonomiques. A l'inverse des générateurs d'interface ascendants présentés ci-dessous, ces outils ne couvrent que l'étape de conception ergonomique.

3.3. Conceptions et générateurs "ascendants"

Les générateurs d'interfaces "ascendants" produisent automatiquement le code d'une interface à partir d'une spécification externe de l'interface. La figure 5.4 montre leur point d'ancrage et leur portée. Alors que les générateurs descendants partent d'une description abstraite des concepts, les générateurs ascendants reposent sur une description concrète des objets de présentation. Les premiers couvrent toute la branche descendante du cycle en V, les seconds viennent en prolongement de l'analyse des besoins mais vont plus loin que les outils de spécifications externes présentés au paragraphe précédent.

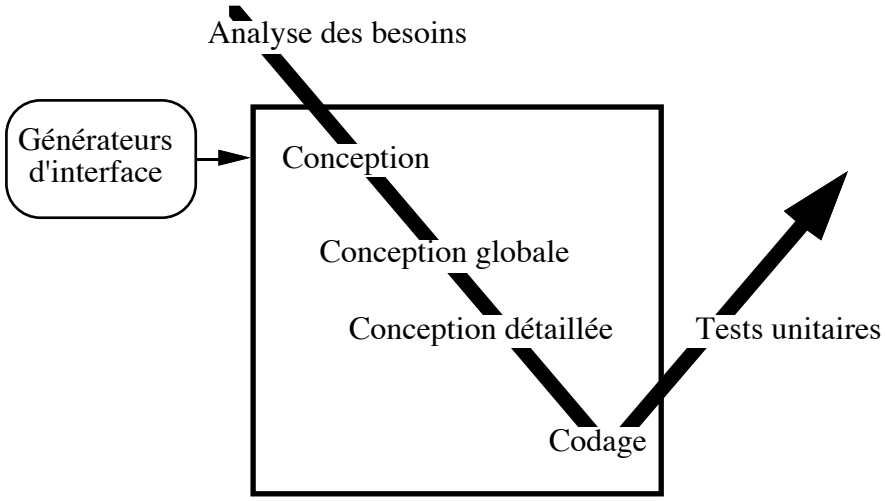


Figure 5.4 : Les générateurs d'interfaces.

Les générateurs d'interface ascendants sont, en général, fondés sur un modèle d'architecture qui définit les points d'ancrage du code généré entre le noyau fonctionnel et les objets de présentation d'une boîte à outils. Pour certains générateurs les protocoles d'accès au noyau fonctionnel et aux boîtes à outils sont clairement définis. C'est le cas de SERPENT qui satisfait au modèle d'architecture ARCH.

Dans SERPENT, les échanges entre le noyau fonctionnel et l'interface sont spécifiés dans le langage SLANG. SLANG permet, entre autres, de décrire à un haut niveau d'abstraction les données partagées entre le noyau fonctionnel et l'interface. Par exemple, la définition de l'interface entre le noyau fonctionnel et l'IHM d'un calendrier serait la suivante [Bass 90] :

```
Calendar: shared data
            Appts: record
                     year : integer;
                     month : integer;
                     day : integer;
                     ....
                     end record
            ...
end shared data;
```

La variable "Appts" ou rendez-vous désigne :

- une donnée que l'IHM fournit au noyau fonctionnel comme un nouveau rendez-vous fixé dans l'agenda,
- mais aussi une donnée que le noyau fonctionnel retourne à l'IHM suite à une requête de l'utilisateur comme "quels sont les rendez-vous de mardi ?".

SERPENT est l'un des rares générateurs ascendants a gérer de manière formelle l'interface du code généré avec le noyau fonctionnel. Mais tous définissent un protocole d'accès aux objets de présentation des boîtes à outils via un langage de spécification d'interface. Certains, comme SERPENT, se préoccupent de la portabilité du code généré vis-à-vis des boîtes à outils, d'autres comme Interface Builder, font l'hypothèse de l'existence immuable d'un environnement extensible orienté objets.

Concernant le langage de spécification d'interface, on distingue les générateurs qui offrent un langage de spécification textuelle tel SERPENT [Bass 88] et son langage SLANG, des générateurs, comme Interface Builder [Webster 89], dont le langage est gestuel par manipulation directe [Myers 88].

Pour illustrer les différences de spécification d'un composant graphique, nous prenons l'exemple de la création d'un bouton d'ouverture d'une fenêtre :

• En utilisant un générateur ascendant non interactif, la création d'un bouton se fait par déclaration selon les conventions du formalisme de spécification. Par exemple, dans SERPENT la création d'un bouton s'écrit en langage déclaratif SLANG :

```
OBJECTS:
        confirm_button : command_widget
        ATTRIBUTES:
        …
        label_text: "OPEN";  # texte affiché à l'intérieur du bouton
```

• Avec un générateur ascendant interactif, la création se fait par sélection du bouton dans une palette d'objets graphiques prédéfinis, puis placement par manipulation de l'exemplaire de bouton à l'emplacement désiré. Il est ensuite possible d'en modifier les attributs. Par exemple, dans Interface Builder, plusieurs sons peuvent être associés au bouton par manipulation directe : sélection de l'icône de son et glissement de l'icône sur le bouton.

Alors que dans SERPENT, la dynamique du dialogue est spécifiée avec SLANG, elle peut être effectuée par Manipulation directe dans Interface Builder. Ainsi définir que la sélection du bouton provoque l'ouverture d'une fenêtre se fait par l'enfoncement de la touche <Ctrl> accompagné du déplacement de la souris allant du bouton vers la barre de titre de la fenêtre. Un retour d'information immédiat constitué d'un trait épais entre le bouton et la fenêtre explicite le lien dynamique créé.

Les générateurs ascendants interactifs semblent très séduisants car ils réduisent la phase d'apprentissage. Cependant il est souvent nécessaire de revenir au niveau de la boîte à outils pour des besoins spécifiques de l'interface. De plus, il est rare que la spécification entière puisse se faire par manipulation directe. Généralement la partie statique de l'interface graphique est spécifiée par manipulation directe comme les fenêtres, leurs tailles, leurs positions etc. Au contraire la partie dynamique de l'interface, comme les liens d'ouverture entre deux composants graphiques, nécessite souvent de programmer. C'est le cas du générateur Egéria [Egeria 90] et de bien d'autres. Interface Builder, qui permet une spécification interactive des liens dynamiques est une exception.

En résumé, les générateurs ascendants couvrent les étapes de conception et d'implémentation des IHM. En ce sens, ils constituent, comme les générateurs descendants, de bons outils de prototypage. Du prototype au système livré, les implémenteurs, soucieux de réutiliser le produit de leurs efforts, ne sont pas toujours prêts à faire la différence. Aussi d'outils de prototypage, les générateurs ascendants d'interface visent le statut d'outils de

développement. De nouvelles difficultés surgissent alors : la portabilité vis-à-vis des boîtes à outils sans nuire à l'efficacité, la richesse de l'expression du contrôle du dialogue, la formalisation des échanges avec le noyau fonctionnel, la conformité à l'analyse de tâche, la capacité, comme pour les outils de spécifications externes, d'auto-évaluation par rapport à des critères ergonomiques. A notre connaissance, aucun générateur ascendant ne satisfait actuellement toutes ces contraintes. Et si l'on ajoute la création d'interfaces à caractéristiques multiples, aucun générateur, même à titre de maquettage, ne remplit la condition. Sur ce terrain, les modèles d'architecture ont leurs enseignements à fournir.

3.4. Conception logicielle et modèles d'architecture

Un modèle d'architecture logicielle définit un guide pour l'organisation du logiciel. Comme le montre la figure 5.5., il intervient dans la phase de conception logicielle globale.

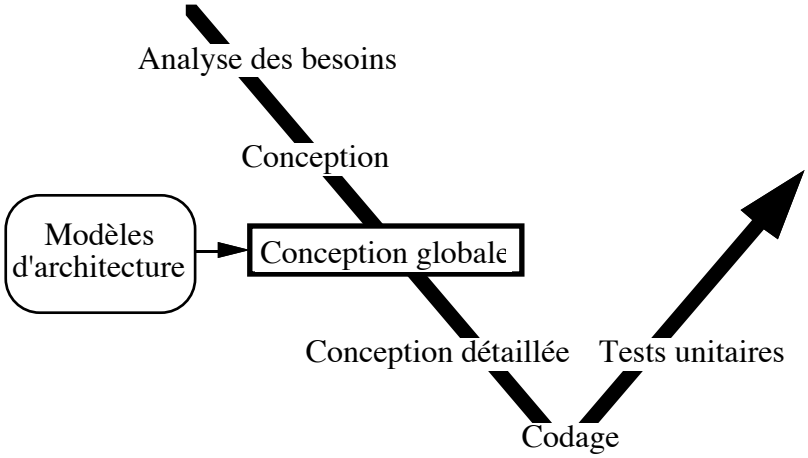


Figure 5.5 : Les modèles d'architecture.

Objets de nos travaux, les modèles d'architecture sont étudiés plus en détail au paragraphe 4.

3.5. Codage et outils de développement

Comme le montre la figure 5.6, les outils de développement sont destinés à la phase de codage uniquement. Nous en distinguons deux classes : les boîtes à outils et les squelettes d'application.

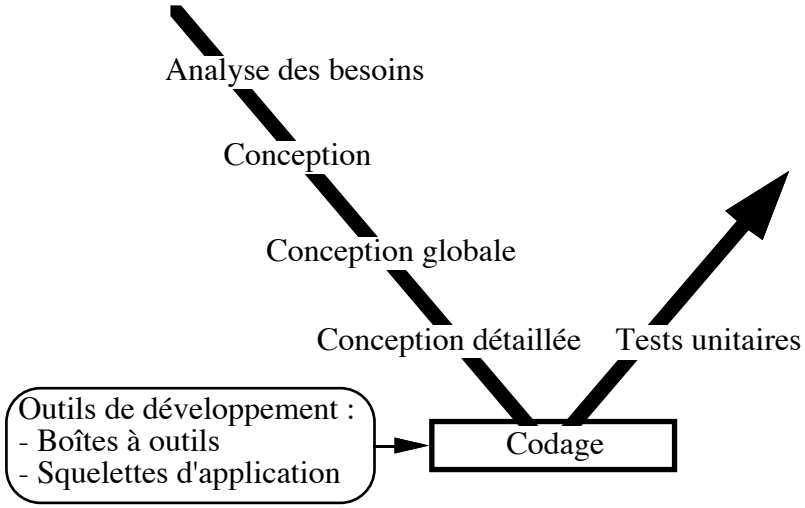


Figure 5.6 : Les outils de développement.

3.5.1. Les boîtes à outils

Une boîte à outils est une bibliothèque de composants logiciels accessibles au programme client via des appels procéduraux. Ces composants prennent généralement la forme d'objets graphiques organisés en hiérarchies de niveaux d'abstraction extensibles [OSF/MOTIF 90, Apple 92]. Jusqu'ici, leur conception est restée centrée sur l'aide à la programmation d'interfaces de type manipulation directe, complétée accessoirement de services élémentaires pour l'acquisition et la restitution de sons (voir par exemple, l'Application Kit de NeXT [NeXTstep 91]).

Depuis peu, un nouvel outillage est apparu pour la synchronisation d'informations multimédia et pour l'intégration de dispositifs d'entrée sortie innovants. L'environnement Macintosh offre un tel exemple d'avancée technique : NTK, la boîte à outils du Newton, inclut les services nécessaires à la gestion de l'encre. Le Quadra AV comprend un système de reconnaissance de la parole et, depuis quelques années, l'extension QuickTime du Système 7 permet la synchronisation de données audio et vidéo ou dépendantes de l'application cliente. Ces services restent néanmoins de bas niveau.

On trouvera dans Artkit présentée dans [Henry 90] un effort de conceptualisation au niveau du geste. Dans Artkit, le geste est géré en deux étapes. La première, la segmentation, correspond à une analyse lexicale des événements de bas niveau. L'analyse est effectuée par un agent dérivé de l'agent chargé des déplacements de la souris ("*inking-drag agent*"). L'étape suivante, la reconnaissance du geste, est dépendante de la base de gestes répertoriés pour le

système. Elle est effectuée par un objet de reconnaissance qui sert d'intermédiaire entre l'agent segmentation et l'objet graphique concerné par le geste ("*focus*").

De manière générale, le niveau d'abstraction offert par une boîte à outils conditionne l'effort de développement d'une interface utilisateur et fixe les styles d'interface et d'interaction. Plus il est élevé, plus le temps d'implémentation est réduit et plus l'interface produite est "normalisée". Il n'en reste pas moins vrai que l'apprentissage de la programmation pour une boîte à outils donnée reste élevé. Les squelettes d'application visent à réduire ce temps de formation.

MOTIF de l'Open Software Foundation [OSF/MOTIF 90], ou Toolbox Macintosh [Apple 92] sont des exemples de boîtes à outils. L'apprentissage de telles boîtes à outils est d'environ trois mois.

3.5.2. Les squelettes d'application

Au contraire des boîtes à outils, les squelettes d'application réalisent les fonctions générales d'une interface sous forme d'un logiciel réutilisable et extensible. L'utilisation d'un squelette consiste à greffer des composants spécifiques au système à implémenter. En général, un squelette est construit en corrélation avec une boîte à outils. MacApp en est un exemple.

Exploitant l'environnement à objets, MacApp [Schmucker 86] offre un ensemble d'objets prédéfinis surchargeables. En particulier, la classe d'objet Tapplication fournit la boucle d'acquisition et de traitement des événements utilisateur constituant ainsi le programme principal de l'interface.

Le moteur de fusion que nous décrivons au chapitre VII est aussi un squelette. Mais ce composant intervient dans la réalisation des systèmes à caractéristiques multiples. Il a pour fonction la gestion des actions utilisateur produites ou non en parallèle pour des langages d'interaction et des dispositifs d'entrée éventuellement distincts.

En résumé, les outils de codage constituent le logiciel de base des IHM. De fait, ils servent de plates-formes d'accueil aux outils évolués comme les générateurs d'interface. Par enrichissements successifs, on aboutit à un empilement de machines abstraites dédiées à l'IHM. Pour que l'édifice forme un tout à la fois robuste, souple, maintenable et efficace, il convient qu'il réponde à des principes d'architecture.

4. MODÈLES D'ARCHITECTURE

4.1. Définition et justification

Un modèle d'architecture définit les éléments directeurs d'aide à l'organisation modulaire du logiciel de l'interface d'un système informatique. L'expérience montre que dans le domaine des systèmes interactifs, cette modularité est non seulement difficile à établir mais incontournable. La difficulté a de multiples causes : une IHM se trouve à la jonction d'un noyau fonctionnel et d'un utilisateur ; une IHM est mise au point par itération.

Prise entre deux mondes, l'IHM est pilotée par des événements conceptuellement asynchrones et de nature différente. En IHM, la gestion de l'asynchronisme a souvent été détournée au profit de la séquentialité. La séquentialité est une propriété séduisante pour l'informaticien mais elle correspond mal au comportement opportuniste de l'utilisateur et notamment aux dialogues à plusieurs fils d'activité. A chaque fil d'activité mentale de l'utilisateur doit correspondre un mini-dialogue dans l'IHM. En l'absence d'un modèle d'architecture adapté, la description de ces mini-dialogues est une tâche difficile, longue, voire impossible.

Dans l'état actuel des connaissances, la mise au point itérative des interfaces est la seule stratégie possible et effective. Le succès de cette approche repose sur la possibilité de modifier une IHM sans mettre en cause la fiabilité ni atteindre des coûts prohibitifs de mise à jour. Une organisation explicite du logiciel de l'IHM est un facteur de réduction de la complexité des modifications.

Bien que les modèles d'architecture diffèrent, ils répondent tous au même principe de base : celui de la séparation fonctionnelle entre le noyau fonctionnel et l'IHM. Le premier modélise les concepts du domaine tandis que le second les présente à l'utilisateur. Outre cette séparation, le modèle d'architecture détermine la stratégie de répartition des services ou traitements de l'interface. Enfin, il décrit avec précision les relations entre les constituants logiciels.

Un modèle d'architecture :

- préconise la séparation entre les services du noyau fonctionnel et ceux de l'interface,
- définit une stratégie de répartition des services de l'interface qui se traduit par un ensemble de constituants logiciels,
- définit le protocole d'échange entre les constituants logiciels qu'il a permis d'identifier.

Ayant défini le rôle, nous allons identifier les propriétés qu'un modèle d'architecture doit satisfaire pour remplir sa mission.

4.2. Propriétés requises

Nous proposons quatre propriétés que tout modèle d'architecture devrait satisfaire : modifiabilité, réduction de la complexité, décomposition du travail, intégration des outils de mise en œuvre.

La modifiabilité du code va de pair avec la conception itérative des interfaces. Cette propriété est difficile à vérifier. Plusieurs critères sont mis en jeu et notamment :

- La lisibilité du code qui peut se mesurer par des normes de programmation et de présentation, le choix des identificateurs, et le pourcentage de commentaires ;
- L'organisation du code qui dépend directement d'un modèle d'architecture.

Règle 1

Un modèle d'architecture logicielle implique une organisation spécifique du code de l'interface utilisateur. Cette organisation doit faciliter les modifications dues, pour l'essentiel, à la mise au point itérative des interfaces.

Le développement d'une interface utilisateur est une tâche complexe et coûteuse en temps de travail. Des études ont montré que cinquante à quatre-vingts pour-cent du code d'un système peuvent être dédiés à son interface utilisateur. L'application d'un modèle d'architecture semble donc incontournable afin d'en organiser la structure. Une organisation explicite du code permet de réduire la complexité de réalisation et offre un cadre de référence pour la décomposition du travail.

Règle 2

Un modèle d'architecture logicielle doit réduire la complexité de réalisation d'une interface utilisateur en proposant des éléments de structuration.

Règle 3

Un modèle d'architecture logicielle doit favoriser la décomposition du travail.

Ayant souligné qu'un modèle d'architecture est utile à la phase de conception globale (paragraphe 3.4.), il est nécessaire que celui-ci prenne en compte les résultats des phases précédentes et en particulier de l'analyse des besoins. Lors de la phase d'analyse des besoins, les contraintes logicielles et matérielles sont fixées ; par conséquent leur prise en compte doit pouvoir s'effectuer au sein du modèle d'architecture.

Règle 4

Un modèle d'architecture doit tenir compte de l'existence des outils logiciels de développement d'interfaces utilisateur et doit englober la plate-forme physique d'accueil. En d'autres termes, un modèle d'architecture doit prendre en compte les contraintes logicielles et matérielles stipulées dans le cahier des charges.

4.3. Classes de modèles

Nous trouvons de nombreuses propositions de classification des modèles d'architecture. Nous en avons retenu trois. La première s'appuie sur les niveaux de raffinement et le degré de parallélisme [Coutaz&Nigay 91]. La seconde organise l'espace des modèles en fonction du caractère homogène ou hybride des composants [Carlsen 91]. La dernière considère l'application du modèle par le concepteur d'interface [Cockton 91].

4.3.1. Niveaux d'affinement, séquentialité et parallélisme

Dans [Coutaz&Nigay 91], nous prenons comme point de départ, la séparation entre le noyau fonctionnel et l'interface que nous affinons progressivement :

- Le noyau fonctionnel modélise le domaine à l'aide d'objets sémantiques liés par des relations.

- L'interface ou présentation est un ensemble organisé d'objets interactifs qui définissent la partie perceptible du système.

La conception itérative et sa compagne immédiate, la réutilisation logicielle, suggèrent que le noyau fonctionnel n'ait aucune connaissance des objets de présentation et réciproquement. Ici, l'absence de connaissance ne signifie pas absence de communication mais implique des références par indirection et/ou des changements de formalisme.

La double nécessité d'assurer à la fois l'indépendance et les échanges d'information entre le noyau fonctionnel et l'interface décrivant la présentation, se traduit par la présence d'un troisième composant : le Contrôleur de Dialogue. Cette remarque nous conduit au modèle Seeheim qui a su clairement identifier le principe de séparation entre un noyau fonctionnel qui implémente les concepts du domaine, et une interface (IHM) chargée de présenter ces concepts. A son tour, cette IHM se voit structurée en trois composants : l'interface avec le noyau fonctionnel, le contrôleur de dialogue et les techniques de présentation [Pfaff 85].

Bien que le modèle de Seeheim serve de fondement à de nombreux outils, on a assisté à des interprétations quelques peu erronées. Certains ont vu dans ce modèle une analogie entre IHM et langage. Selon la théorie, un langage est défini par une sémantique, une syntaxe et un lexique. En plaquant cette décomposition sur l'organisation du modèle de base, nous observons que le noyau fonctionnel correspond aux actions sémantiques, que le Contrôleur de Dialogue remplit les fonctions d'un analyseur syntaxique et que le Composant de Présentation effectue l'analyse lexicale. Ce faisant, le fonctionnement du système obtenu procède séquentiellement de l'analyse lexicale à l'analyse sémantique pour les entrées et vice versa pour exprimer un changement d'état interne.

Or dans le domaine de l'interaction homme-machine la séquentialité peut être gênante : la forme d'une expression d'entrée peut évoluer en parallèle avec la production d'une expression de sortie ; ou encore plusieurs expressions d'entrée, éventuellement indépendantes, peuvent être spécifiées en parallèle, ou de manière alternée :

- En particulier, l'entrelacement des expressions d'entrée et de sortie qui se manifeste déjà dans les interfaces à manipulation directe, exige de la part du système, une réaction immédiate et informative.

- La spécification parallèle ou entrelacée de plusieurs expressions d'entrée est le fait des interfaces à fils d'activité multiples ("multithread dialogues"). Un fil d'activité modélise une branche du plan de résolution de l'utilisateur. Ce dernier peut ainsi passer d'une branche à l'autre de manière opportuniste.

Ces observations ont servi de moteur à la conception des modèles multi-agent qui, comme PAC [Coutaz 87] ou MVC [Krasner 88], structurent un système interactif en un ensemble d'agents spécialisés réactifs. Ces modèles se caractérisent par une organisation fortement modulaire, des traitements exécutés en parallèle et une communication par événements. A la gestion centralisée de l'état de l'interaction de Seeheim se substitue une totale répartition des charges. Ce parallélisme et cette répartition sont les conditions nécessaires à la prise en compte des méthodes de résolution de l'utilisateur. La répartition présente en sus deux

retombées intéressantes : une grande modularité et l'ouverture vers les traitements physiquement distribués. PAC est à l'origine de notre modèle et sera détaillé au chapitre suivant.

4.3.2. Modèles homogènes et hétérogènes

Carlsen identifie trois classes de modèles selon la structure du logiciel préconisée [Carlsen 91] :

• La structure en composants hétérogènes implique une organisation en un ensemble de composants qui ne remplissent pas les mêmes services. Typiquement, le modèle Seeheim en est un exemple.

• La structure en espace d'objets hétérogènes organise le logiciel en une collection d'agents qui ne remplissent pas les mêmes services mais qui sont structurés de la même manière. Un agent est alors perçu comme une "micro interface". Nous verrons que le modèle PAC-Amodeus que nous proposons vérifie ce type de structure.

• La structure en espace d'objets homogènes organise le logiciel en une collection d'agents de même caractéristiques et de même rôle fonctionnel. Le modèle PAC en est un exemple.

4.3.3. Modèles et application du phénomène d'abstraction

Gilbert Cockton propose une autre classification en considérant l'application du modèle par le concepteur [Cockton 91]. Il distingue ainsi trois modèles :

• Abstractions architecturales du haut vers le bas. Un modèle qui préconise un mécanisme d'abstraction du haut vers le bas définit un ensemble de composants de haut niveau d'abstraction. La tâche du concepteur consiste à répartir les services parmi les composants fixés. Seeheim et Arch induisent ce type d'application.

• Abstractions architecturales à partir du milieu. Un modèle de cette classe propose un ensemble de composants de bas niveau et une stratégie pour combiner ces composants afin d'obtenir des composants de plus haut niveau d'abstraction. Le modèle des dispositifs d'entrée de Mackinlay (voir chapitre II) et celui des interacteurs de Faconti&Paterno [Faconti 93] impliquent ce type de raisonnement.

• Abstractions architecturales du bas vers le haut. Un modèle de cette classe s'apparente à un modèle de la classe précédente sans toutefois proposer de stratégie de combinaison. Il propose un ensemble de composants de bas niveau.

D'autres approches existent comme celle consistant à distinguer les modèles selon la communication préconisée entre les composants logiciels ou selon la topologie des composants (hiérarchie, réseau, pipe-line etc.) [Cockton 91].

Au chapitre suivant, nous situons le modèle d'architecture que nous proposons et le caractérisons selon les critères de classification exposés ci-dessus.

5. CONCLUSION

Au terme de cette analyse sur les outils existants, nous constatons l'absence, ou presque, de techniques d'aide au développement d'interfaces à caractéristiques multiples. Nous estimons que la conception d'outils élaborés comme les générateurs d'interface requiert, au préalable, la définition d'une ossature conceptuelle réutilisable. Sans modèle d'architecture conceptuel, nous prenons le risque de développer des outils de portée limitée.

Au vu de ces constatations, nous visons un triple objectif :

• l'élaboration d'un modèle d'architecture conceptuel qui réponde aux propriétés énoncées plus haut comme la modifiabilité et l'intégration d'outils existants,

• la définition d'une stratégie d'identification des composants logiciels répondant à la mise en œuvre d'un système donné,

• l'identification de composants génériques réunis en un squelette réutilisable.

Cette seconde partie du mémoire présente nos travaux selon les étapes de notre objectif. Au chapitre VI, nous décrivons le modèle d'architecture conceptuel, PAC-Amodeus, que le concepteur logiciel trouvera utile comme cadre de raisonnement. Pour l'aider à instancier le modèle au cas particulier du système à implémenter, nous proposons ensuite une stratégie d'identification des composants. Nous la formulons sous forme de règles heuristiques consignées dans un système expert : PAC-Expert. Le modèle contient de manière implicite des composants réutilisables, et notamment un moteur de fusion, que nous présentons au chapitre

VII et que nous validons par la mise en œuvre de deux systèmes à caractéristiques multiples : MATIS et NoteBook. Ces architectures de ces deux systèmes font l'objet du chapitre VIII.

Chapitre VI

PAC-Amodeus : notre modèle d'architecture

"Envoie toutes sortes de messages
Aux inconnus et lucioles de passage.
Prends le parti du risque et de l'erreur
Le silence est toujours complice ou trompeur."
- Yves Simon -

1. INTRODUCTION

La première partie de ce mémoire, "espace de conception", nous a permis d'explorer les propriétés des systèmes à caractéristiques multiples via le point de jonction entre l'utilisateur et le système : les langages d'interaction et les dispositifs. Il convient maintenant de traiter le problème de leur conception logicielle et d'identifier des solutions qui soient en accord avec les propriétés et les relations caractérisantes décrites dans la première partie du mémoire.

La conception logicielle, nous l'avons vu, doit reposer si possible, sur le cadre structurant d'un modèle d'architecture. Dans ce chapitre, nous présentons PAC-Amodeus, notre propre contribution au domaine des architectures logicielles. PAC-Amodeus résulte de la combinaison de deux modèles conceptuels d'architecture: PAC [Coutaz 88, Coutaz 90] et ARCH [UIMS 92]. L'un et l'autre sont présentés aux paragraphes 2 et 3 respectivement. PAC-Amodeus et ses liens avec l'espace Pipe-Lines font l'objet d'une description détaillée aux paragraphe 4 et 5.

La généralité de PAC-Amodeus a ses contreparties : l'imprécision et l'ambiguïté. Cette liberté peut être exploitée fructueusement par le concepteur qui maîtrise le modèle mais elle est source de difficultés pour le néophyte. On trouvera au paragraphe 6 des règles heuristiques de mise en application du modèle. Ces règles nous ont conduits à développer un système expert, PAC-Expert, qui inclut une méthode de conception d'architecture logicielle répondant au modèle PAC-Amodeus. PAC-Expert est présenté au paragraphe 7. Nous concluons en 8 par un bilan comparatif de PAC-Amodeus avec d'autres modèles multi-agent.

2. LE MODÈLE MULTI-AGENT PAC

PAC, point de départ de notre étude, est un modèle multi-agent [Coutaz 90]. Il a comme principes directeurs le concept d'agent à facettes et une décomposition récursive. Nous reprenons chacun de ces aspects dans les paragraphes qui suivent. Nous concluons cette présentation par une analyse critique de PAC qui nous conduit à proposer un formalisme comme véhicule des règles conceptuelles du modèle.

2.1. Le concept d'agent et architecture multi-agent

Un agent est un système de traitement de l'information : il se distingue par un jeu d'opérations, des mécanismes d'entrée/sortie et une capacité à représenter un état que l'on appelle vecteur d'état. Un agent est un acteur communicant : il assure un rôle (c'est un acteur) et

se manifeste par un comportement observable via l'acquisition et la production d'informations (il communique). L'acquisition et la production d'information sont des actions dont la réalisation se traduit par des événements. Les informations proviennent ou sont destinées à d'autres agents : l'agent vit en communauté. Il conduit ses activités en parallèle avec celles de ses confrères. Il faut l'opposer à milieu d'activités séquentielles et à entité passive et isolée.

En Intelligence Artificielle, on convient de distinguer les agents réactifs des agents intelligents [Demazeau 92]. Les premiers, du type stimuli-réponse, ont un comportement câblé. Les seconds, dotés de mécanisme de planification, poursuivent des buts de manière adaptative. En Interface Homme-Machine, on parle implicitement d'agents réactifs. Par exemple, dans le modèle des dispositifs d'entrée de Mackinlay et de ses coauteurs (voir chapitre II), un dispositif est un agent réactif. Les interacteurs décrits dans [Faconti 93], ou les agents de Abowd [Abowd 92], sont également des agents réactifs. Dans PAC, la distinction n'est pas significative bien que toutes nos applications de PAC aient conduit à l'implémentation d'agents réactifs.

Un système multi-agent est une société d'agents dont la nature et l'organisation définissent ce qu'on appelle l'architecture du système. Cette architecture répond, le plus souvent, à un modèle. Par exemple, en Intelligence Artificielle, on a largemement adopté le modèle du tableau noir [Reddy 74] . PAC est un modèle d'architecture applicable au cas des systèmes interactifs.

2.2. Agent PAC : un agent à facettes

Un agent PAC, constituant d'un système interactif, est un système interactif. D'après le modèle de Seeheim, un système interactif peut s'analyser selon trois points de vue : le noyau fonctionnel, le contrôleur de dialogue, et la présentation. Un agent PAC adopte ces trois perspectives complémentaires : la Présentation, l'Abstraction et le Contrôle.

La Présentation définit le comportement perceptible de l'agent pour un agent humain. Elle concerne à la fois les entrées et les sorties c'est-à-dire les modalités d'action accessibles à l'utilisateur et la restitution perceptible. Si l'on reprend la terminologie du chapitre III, la Présentation : (1) interprète les événements résultant des actions physiques que l'utilisateur applique sur l'agent via des dispositifs d'entrée et (2) engendre des événements qui traduisent des actions physiques sur les dispositifs de sortie.

L'Abstraction, avec ses fonctions et ses attributs internes, définit la compétence de l'agent indépendamment des considérations de présentation. Elle constitue, au sens de Seeheim, le noyau fonctionnel de l'agent.

Le Contrôle a un double rôle : (1) il sert de pont entre les facettes Présentation et Abstraction de l'agent, (2) il gère des relations avec d'autres agents PAC. Tout échange d'informations entre l'Abstraction et la Présentation s'effectue via le Contrôle. C'est aussi par leurs parties Contrôle que deux agents PAC communiquent.

Le rôle d'intermédiaire dans la communication attribue au Contrôle les fonctions d'arbitrage et de traduction :

- L'arbitrage recouvre la synchronisation et la coordination. Ces services sont nécessaires entre agents mais aussi entre l'Abstraction et la Présentation d'un agent. En effet, ces dernières peuvent, dans certaines circonstances, être le lieu d'activités asynchrones qui requièrent alors une coordination.

- La traduction concerne la transformation de formalismes entre l'Abstraction et la Présentation qui, en raison de rôles distincts, utilisent des modèles de représentation différents.

A titre d'illustration, nous reprenons l'exemple du thermomètre introduit au chapitre III (Figure 3.7) pour représenter la notion de température dans un système de contrôle industriel. Dans le cadre de ce chapitre, le thermomètre devient l'agent PAC de la figure 6.1. La Présentation de cet agent sait dessiner une forme de thermomètre et reçoit les événements qui traduisent les actions de l'utilisateur ; l'Abstraction, avec son vecteur d'état "température initiale, température finale, et variation de température", constitue un modèle abstrait de l'état du thermomètre. Le Contrôle effectue le pont entre les deux facettes en gérant la correspondance des phénomènes abstraits et concrets.

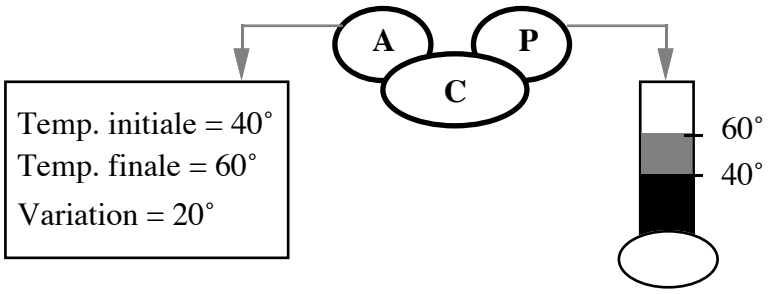


Figure 6.1 : Un exemple d'agent PAC.

Par exemple, l'action utilisateur qui consiste à déplacer la barre de mercure pour spécifier la température souhaitée se manifeste à la Présentation par un événement de type "localisation (x, y)". La Présentation en déduit une nouvelle hauteur de mercure qu'elle transmet

au Contrôleur selon un protocole convenu. Par exemple, la Présentation pourrait transmettre une valeur liée au formalisme graphique, et notamment une longueur de segment de droite. Cette longueur n'est pas nécessairement la longueur effective de la droite qui représente le niveau de mercure sur l'écran, mais une valeur correspondant à une taille de thermomètre normalisée : cette valeur serait la même quelle que soit la taille de l'exemplaire de thermomètre effectivement utilisé, en supposant par exemple que l'image du thermomètre soit proportionnelle à la taille de la fenêtre support.

Le Contrôleur de l'agent thermomètre traduit la longueur du segment de droite en fonction du protocole convenu avec l'Abstraction. Ici, on échangera une valeur de température normalisée (éventuellement indépendante du fait qu'il s'agit d'une représentation en Celsius ou en Farenheit). L'Abstraction en déduit une température finale, calcule la variation et transmet le ou les résultats pertinents au Contrôleur (par exemple la variation et la température finale). Ce dernier exprime les résultats dans le formalisme attendu de la Présentation qui peut alors effectuer une mise à jour du rendu perceptible de l'agent : un mercure en grisé pour dénoter la variation et la valeur en Celsius de la température souhaitée.

On peut imaginer que le flux d'échanges ci-dessus soit déclenché dès la réception, par la Présentation, d'un événement "mouse-down" et qu'il soit entretenu jusqu'au relâchement du bouton de la souris. Une fois le bouton relâché, l'agent thermomètre et l'utilisateur mettent fin à la transaction mais l'agent thermomètre se doit de signaler à d'autres agents du système le souhait final de l'utilisateur. Une autre possibilité est que le flux d'échanges ci-dessus soit déclenché seulement lorsque l'utilisateur relâche le bouton de la souris. Entre les actions enfoncement et relâchement, la Présentation prend à sa charge un retour d'information de nature purement lexicale. Par exemple, la hauteur du mercure suit le mouvement de la souris mais sans impression de la valeur correspondante de la température. Dans le permier cas, le retour d'information pendant la transaction fait intervenir les compétences sémantiques de l'agent. Dans le second cas, il ne s'agit que de feedback lexical.

De manière générale, un événement utilisateur peut avoir un effet sémantique ou non. Un événement sans effet sémantique est traité en local par la Présentation qui produit un retour d'information perceptible. Les événements à effet de bord sémantique sont non seulement traités localement par la Présentation pour les retours d'informations lexicaux mais, en plus, sont transmis à l'Abstraction par l'intermédiaire du Contrôle pour complément de traitements. L'enrichissement sémantique du retour d'information peut se poursuivre lorsque l'agent fait appel à d'autres agents de la hiérarchie y compris le noyau fonctionnel.

La description que nous venons de donner d'un agent PAC est applicable à tous les niveaux d'affinement ou d'encapsulation d'un système : PAC est un modèle récursif.

2.3. PAC : un modèle récursif

Comme le montre la figure 6.2, PAC structure récursivement un système interactif sous forme d'une hiérarchie d'agents. La hiérarchie traduit des relations entre agents et reflète un continuum de niveaux d'abstraction depuis le noyau fonctionnel jusqu'aux éléments fins de l'interaction, c'est-à-dire les actions physiques (au sens du chapitre III).

De manière générale, les agents PAC d'un système sont les entités qui réalisent les fonctions d'abstraction et de concrétisation du modèle MSM, ou encore dans le modèle Pipe-Lines, les fonctions Acqu-Inter-AP et Spec-Exec-AS du plan des entrées et, dans le plan des sortie, les fonctions Spec-Rendu-Eff et Spec-Exec-AP. Alors que dans MSM et Pipe-Lines, nous insistions sur les transformations successives entre les représentations conceptuelles et les phénomènes concrets, PAC montre comment des agents peuvent se répartir les tâches de concrétisation et d'abstraction et comment ils coopèrent à leur réalisation.

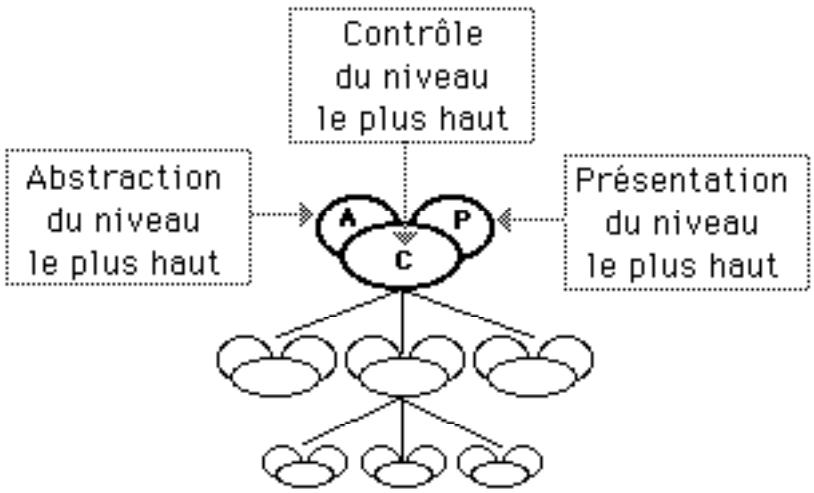


Figure 6.2 : Une architecture PAC.
Les traits pleins figurent des relations entre agents de niveaux d'abstraction distincts.

Notons que les liens entre les agents ne sont pas nécessairement des canaux de communication. Ils peuvent traduire aussi des relations de composition c'est-à-dire des opérations de conception logicielle que sont l'encapsulation et l'affinement. Pour illustrer cette double interprétation du modèle PAC, analysons la racine de la hiérarchie. L'"abstraction du niveau le plus haut" correspond au noyau fonctionnel du modèle de Seeheim. Le "contrôle du niveau le plus haut" peut s'interpréter de deux façons distinctes : (a) comme le contrôleur de dialogue de Seeheim ou (b) comme un agent simple dont les unités de traitement sont des unités informationnelles (au sens du chapitre III), c'est-à-dire un système de représentation des

concepts qui fait sens pour le noyau fonctionnel. Dans le premier cas, le sous-arbre d'agents représente un affinement du contrôleur. La liaison entre l'agent du niveau le plus haut et le sous-arbre est une relation de composition (encapsulation/affinement). Dans le second cas, le sous-arbre désigne les agents dont l'agent racine a la charge. La liaison entre l'agent du niveau le plus haut et le sous-arbre est une relation d'échange d'information.

De même, la "présentation du niveau le plus haut" peut se voir comme le composant Présentation du module de Seeheim. Cette interprétation, semblable au cas (a) de la composante contrôle, correspond à l'encapsulation de la présentation du système. Dans ce cas, la présentation du niveau le plus haut se définit comme l'ensemble des présentations des agents du sous-arbre. Dans l'interprétation (b), elle est la facette de concrétisation de la racine considérée comme agent simple.

Ces deux formes d'interprétation, encapsulation/affinement et communication, sont applicables ensemble à tout agent de la hiérarchie. Dans la pratique, la vision encapsulation/affinement définit le niveau de détail auquel le concepteur logiciel s'intéresse. Par exemple, Seeheim est une architecture PAC super encapsulante. Un concepteur logiciel peut s'en contenter pour des cas de figure simples où il n'existe pas de dialogue à plusieurs fils et pour lesquels les objets de présentation d'une boîte à outils conviennent à la restitution des concepts du noyau fonctionnel. La vision communication met en évidence les liens de dépendance entre les futurs composants logiciels.

Nous présentons maintenant une brève revue des avantages et des limitations du modèle PAC.

2.3. Analyse critique

PAC fournit un cadre de construction systématique applicable à tous les niveaux d'abstraction d'un système interactif. Cette approche récursive permet de concevoir une architecture progressivement de façon ascendante ou descendante. Il permet de traduire à la fois l'encapsulation/l'affinement et la dépendance fonctionnelle, deux activités de conception logicielle intimement reliées. Notons que nous retrouvons cette double propriété dans le modèle des dispositifs d'entrée de Mackinlay avec les opérateurs de fusion et de connexion.

PAC permet de distinguer les services abstraits des techniques d'interaction en introduisant un intermédiaire explicite : le Contrôle. Cette propriété d'indépendance présente plusieurs avantages :

• la satisfaction du critère qualité de réutilisabilité des constituants de l'interface et ceci de manière systématique à tous les niveaux d'abstraction.

• la modification de l'interface à moindre coût. Notre expérience avec PAC (Mithra [Chabert 89a], Expert Data-Base [Chabert 89b], Compo [Nigay 90b]) montre qu'il est possible de mettre au point une interface de façon itérative et de la faire évoluer facilement.

PAC encourage la répartition des traitements sémantiques et syntaxiques. Cette propriété présente plusieurs retombées intéressantes :

• Les constituants de l'interface communiquent au niveau d'abstraction voulu. En particulier, le noyau fonctionnel a la possibilité de s'exprimer dans son formalisme, à son niveau d'abstraction, ce qui augmente son indépendance vis-à-vis des constituants perceptibles de l'interface.

• Une part de la connaissance du noyau fonctionnel peut être déportée dans l'interface. Cette forme de délégation combine performance et qualité sémantique des retours d'information sans remettre en cause la distinction entre le noyau fonctionnel et l'interface. Dans l'exemple du thermomètre de la figure 6.2, le retour d'information aurait pu faire appel aux compétences du noyau fonctionnel pour décider du bien fondé de l'augmentation de température. La vérification de cette contrainte, pour des raisons de performance, aurait pu être déportée dans la partie Abstraction de l'agent thermomètre. La décision de cette migration peut être dynamique (le système en prend la décision, suite à une analyse de charge) ou bien statique décidée a priori par le concepteur logiciel. Nous reviendrons sur la délégation sémantique à propos de PAC-Amodeus.

PAC, comme tous les modèles multi-agent, permet de modéliser le parallélisme et l'entrelacement que nous avions relevés dans MSM (voir Chapitre II). PAC et ses niveaux expriment de manière naturelle la granularité de ces phénomènes : grappes de tâches, tâches élémentaires, dispositifs physiques auxquels correspondent dans PAC, des hiérarchies d'agents, des agents feuilles, voire des agents capables de gérer plusieurs dispositifs physiques d'entrée ou de sortie simultanément.

Les modèles multi-agent comme PAC sont de bons candidats pour l'implémentation des applications réparties dont les collecticiels et les "espaces médiatiques"[19] sont des exemples

[19] En anglais "mediaspace".

novateurs (mais exigents). On relèvera notamment le modèle ALV (Abstraction-Link-View), calqué sur PAC, qui sert de cadre à la mise en œuvre du système RENDEZVOUS [Hill 92].

PAC, MVC et bien d'autres [Faconti 93, Duce 90] se présentent comme des variations d'un thème commun. Tous mettent en évidence la répartition des charges et la distinction entre comportement interne et comportement perceptible, mais aucun, jusqu'ici, ne s'est imposé comme solution véritable. Cela tient essentiellement à leur manque de précision et à la présence d'ambiguïtés qui laissent entier le problème de l'identification des agents.

L'imprécision du modèle PAC nous a conduit à proposer un formalisme qui véhicule les règles de conception du modèle.

2.4. Formalisation du modèle PAC

L'une des motivations qui nous conduit à formaliser le modèle PAC sous forme d'un langage de spécification est la possibilité de vérifier les contraintes et propriétés que le modèle implique et donc d'atteindre les critères de qualité offerts par le modèle. La réutilisation des agents PAC est un exemple de critère de qualité. Notre motivation rejoint en fait les objectifs de toute technique de spécification : celle de vérifier un ensemble de contraintes.

L'approche adoptée est la définition d'un sur-langage qui s'appuie sur un langage de description de systèmes réactifs ou sur une algèbre de processus. Ce sur-langage est indépendant du langage de base choisi, celui-ci étant un paramètre du sur-langage. La formalisation est constituée des définitions syntaxiques et sémantiques du sur-langage que l'on trouvera complètement décrites dans [Nigay 90].

2.4.1 Les hypothèses sur le langage de base

Dans la suite de l'exposé du langage, nous utilisons les notations suivantes :
Soit **B** le nom du langage de base.
Soit **Proc** l'ensemble des processus de base du langage B.
Soit **p** un processus du langage de base : p est un élément de Proc.
Soit **Ev(p)** l'ensemble des événements du processus p.

Le langage de base B doit fournir au moins un opérateur de mise en parallèle, un opérateur d'abstraction et un opérateur de composition ou somme.

Opérateur de mise en parallèle :

Si p et p' sont deux éléments de Proc, alors **p || p'** est la mise en parallèle de p et p'. Ces deux éléments peuvent communiquer de manière à ce qu'ils puissent s'influencer. Cette communication s'effectue par des événements en commun, l'un recevant un événement émis par l'autre.

L'ensemble des événements de p || p' est noté : Ev(p || p'). Il est défini par :

Ev(p || p') = Ev(p) U Ev(p').

Opérateur d'abstraction ou masquage d'événements internes à un processus :

Si p est un élément de Proc et evts un sous-ensemble de Ev(p), alors **p / evts** exprime une abstraction par rapport aux événements de evts, en dissimulant les événements evts.

L'ensemble des événements de p / evts est noté : Ev(p / evts). Il est défini par :

Ev(p / evts) = Ev(p) / evts.

Opérateur somme :

Si p et p' sont deux éléments de Proc, alors **p + p'** est la somme indéterministe de p et de p'.

L'ensemble des événements de p + p' est noté : Ev(p + p'). Il est défini par :

Ev(p + p') = Ev(p) U Ev(p').

Un langage de base possible est CSP (Communicating Sequential Processes [Hoare 85]). CSP permet de décrire un système réactif en termes de processus communicants et offre les opérateurs de base requis.

2.4.2. La syntaxe du sur-langage

La syntaxe du sur-langage comprend les définitions de l'élément de base et des éléments construits. L'ensemble de base est l'ensemble des processus noté **Proc**. Nous illustrons l'utilisation de cette syntaxe à la description d'une architecture très simple.

L'élément de base correspond à **un agent PAC**. Un agent PAC comportant trois parties (Abstraction, Contrôle et Présentation), l'élément de base est constitué de trois processus s'exécutant en parallèle. Perçu de l'extérieur, l'agent est néanmoins qu'un seul processus auquel on a masqué les événements internes. Ces événements internes sont ceux de la communication entre les processus qui composent l'agent.

Elément de base : un agent PAC

Syntaxe concrète :

pac ::= "(" a "|| " c "||" p ")" / ens_evc

a ::= Proc "+" a | Proc

c ::= Proc

p ::= Proc "+" p | Proc

Syntaxe abstraite :

Un élément de base est de la forme :

(a || b || c) / evt

où a, b et c sont des éléments de Proc et evt un sous-ensemble de Ev(a) U Ev(b) U Ev(c).

Nompac : noms globaux d'élément de base

Un élément construit correspond à **une architecture PAC**. Nous introduisons un seul opérateur de construction : l'opérateur binaire **fils_de**.

Elément construit : une hiérarchie d'agents

a) Syntaxe concrète :

pac ::= pac "fils_de" pac | Nompac

b) Syntaxe abstraite :

pac ::= pac fils_de pac | Nompac

En instanciant Nompac par sa définition, on obtient :

(a1 || c1 || p1) / evt1 fils_de (a2 || c2 || p2) / evt2 =

((a2 || (c2 || (a1 || c1 || p1) / evt1) || p2) / evt2) / Ev(c1) ∩ Ev(c2)

a) Syntaxe concrète de : (a || c || p) / evt :

pac ::= "(" a "||" c "||" p ")" / ens_evt

a ::= Proc "+" a | Proc

c ::= Proc | "(" Proc || ens_de_fils ")"

ens_de_fils ::= pac | pac "||" ens_de_fils

p ::= Proc "+" p | Proc

b) Syntaxe abstraite :

Les éléments sont de la forme :

(a || c || p) / evt ,

et c de la forme :

Proc || Liste_de_pac

A titre d'illustration, nous décrivons la hiérarchie de la figure 6.3

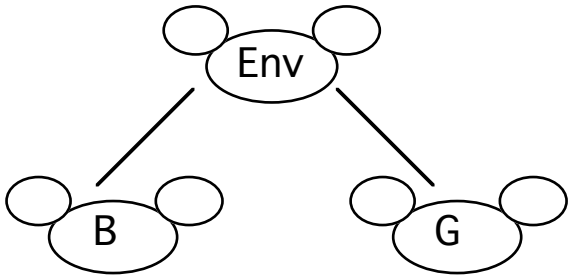


Figure 6.3 : Un exemple d'architecture PAC.

Le terme correspondant à cette architecture est :
(ENV.A || (ENV.C || (B.A || B.C || B.P) || (G.A || G.C || G.P)) || ENV.P)

Nous définissons un formalisme graphique équivalent à la forme textuelle exposé ci-dessus dans [Nigay 90]. Par rapport à l'expression textuelle, le formalisme graphique offre l'avantage d'être plus facile à utiliser et, sous certaines conditions, d'être plus lisible.

2.4.3. La sémantique statique du sur-langage

Comme pour la syntaxe, la sémantique statique du sur-langage doit être définie pour l'élément de base et les éléments construits.

Pour les éléments de base, deux contraintes véhiculées par PAC mais non décrites par la syntaxe, doivent être exprimées ici :

1) Soit (a || b || c) / evts un élément de base. Aucune communication directe entre le processus a et le processus c n'est autorisée.

2) De l'extérieur, l'élément de base est perçu comme un seul processus auquel on a dissimulé les événements internes ; les événements internes servent à la communication entre les processus qui constituent l'élément.

Soit Evts-internes, l'ensemble des événements internes de l'élément. Il est défini par :

Evts-internes (a || b || c) = (Ev(a) ∩ Ev(b)) ∪ (Ev(c) ∩ Ev(b))
= (Ev(a) ∪ Ev(c)) ∩ Ev(b)

Soit PACb le prédicat suivant :

PACb(P) : P est une expression décrivant un élément de base "bien formé" c.-à-d. respectant les deux contraintes 1) et 2) ci-dessus.

$$\frac{Ev(a) \cap Ev(c) = \emptyset,\ evts = Evts\text{-}internes(a \parallel b \parallel c)}{PACb\ ((a \parallel b \parallel c) / evts)}$$

De la même manière, pour les éléments construits nous décrivons dans [Nigay 90], la contrainte suivante :

Si un élément PAC *pac1* est fils d'un élément PAC *pac2* alors ils ne communiquent que par leur partie Contrôle ; et *pac1* ne peut communiquer avec les éventuels autres fils de *pac2*.

2.4.4. Discussion sur la formalisation de PAC

La formalisation de PAC présente plusieurs retombées intéressantes : la garantie d'une conception conforme aux propriétés du modèle, la vérification de la validité de la description, la possibilité de décrire une interface homme-machine à différents niveaux d'abstraction et l'expression facilitée.

2.4.4.1. Adéquation au modèle PAC

Fournir une formalisation de PAC par un langage de description permet de s'assurer qu'une conception exprimée dans ce langage respecte le modèle. En l'occurrence, les propriétés suivantes peuvent être garanties :

- distinction entre services abstraits et techniques d'interaction,
- répartition des traitements sémantiques et syntaxiques,
- réutilisation possible d'agents généraux.

2.4.4.2. Vérification

Le langage permet de vérifier qu'à tout événement émis correspond un traitement déclenché par sa réception. En particulier, il devient possible de vérifier que tout événement en provenance de l'utilisateur est pris en compte et que, par conséquent, le comportement de l'interface est entièrement défini vis-à-vis de l'utilisateur. Cette remarque s'applique également aux événements en provenance du noyau fonctionnel.

2.4.4.3. Description à différents niveaux d'abstraction

Le langage autorise différents niveaux de description. En effet, il est possible de spécifier une architecture PAC et ne pas s'attacher au détail des traitements assurés par chaque processus. Le concepteur peut aussi atteindre un niveau de description plus fin et décrire chacun

des processus constituants d'un agent PAC. A l'extrême, si le concepteur décrit tous les processus intervenant dans le système, la génération automatique des logiciels devient possible.

Associés à la notion de niveaux d'abstraction sont les méthodes de conception et d'encapsulation. Dans notre sur-langage, l'opérateur Fils_de répond à l'effet de ces techniques. Fils_de permet à chaque étape d'abstraire les événements internes. A la dernière étape, c'est-à-dire au niveau d'abstraction le plus élevé, la spécification de l'interface est constituée d'un seul élément composé qui communique vers l'extérieur par sa partie Présentation uniquement. Ainsi l'on obtient au plus haut niveau un seul agent dont les événements extérieurs sont ceux avec l'utilisateur.

2.4.4.4. Facilité d'expression

La spécification avec notre langage évite au concepteur d'expliciter la mise en œuvre du parallélisme et la communication entre processus. De plus, à l'image de toute IHM, le langage est dirigé par les événements. En conséquence, il est bien adapté à l'expression des réponses aux questions que le concepteur est amené à se poser (typiquement : "quel comportement adopter lorsque l'utilisateur effectue telle action?").

2.5. Conclusions et constats

Nous avons présenté les éléments directeurs du modèle PAC en mettant en évidence la répartition des charges et le mécanisme d'abstraction sous-jacent au modèle. La hiérarchie d'agents PAC traduit les niveaux d'abstraction identifiés lors de la conception d'un système. Le manque de précision du modèle nous a conduit à définir une formalisation en proposant un langage. La sémantique statique de celui-ci permet de traduire les règles conceptuelles véhiculées par le modèle. Nous utiliserons ce formalisme pour la spécification de notre modèle PAC-Amodeus.

Notre expérience avec l'application du modèle PAC nous a amenés à considérer l'agent racine non pas comme une encapsulation du système mais comme un agent particulier de la hiérarchie. Nous convenons de l'appeler "super contrôleur". Dans ces conditions, l'Abstraction du super contrôleur est bien, comme le modèle l'indique, le noyau fonctionnel. De plus, pour tous les systèmes que nous avons réalisés en appliquant PAC :

- le super contrôleur assure le changement de formalisme entre le noyau fonctionnel et l'interface utilisateur. Il maintient la correspondance entre les concepts du noyau fonctionnel et les agents PAC qui les rendent perceptibles à l'utilisateur. C'est le seul agent à connaître et à manipuler des concepts du noyau fonctionnel. En cela,

nous pouvons considérer que le super contrôleur est un agent frontière jouant le rôle d'adaptateur avec le noyau fonctionnel ;

• seul le super contrôleur communique avec les autres agents PAC directement sans respecter le chemin induit par la hiérarchie. En effet, le Super Contrôleur, qui maintient la correspondance entre les concepts du domaine et les agents PAC de restitution, connaît tous les agents PAC correspondant à un concept du domaine. Il peut donc communiquer directement avec eux ;

• le super contrôleur a généralement une Présentation vide.

L'identification de ce rôle particulier nous a conduits à ne pas considérer le super contrôleur comme un agent PAC. Cette exception nous a amenés à percevoir le modèle PAC d'un autre point de vue et à restreindre son utilité à la description d'un seul des constituants de l'interface : le contrôleur de dialogue. Pour définir les autres composants nous avons considéré le modèle ARCH présenté ci-après.

3. LE MODÈLE ARCH

Comme le montre la figure 6.4, le modèle ARCH [UIMS 92] s'appuie sur les composants conceptuels du modèle Seeheim [Pfaff 85]. On y retrouve les notions de noyau fonctionnel, de contrôleur de dialogue et de présentation.

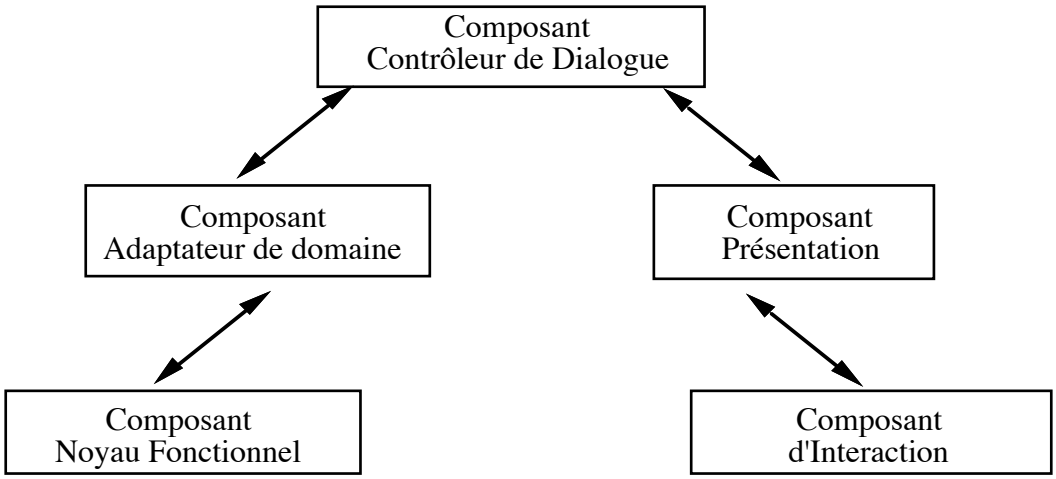


Figure 6.4 : Les composants du modèle ARCH.

Les pieds de l'arche constituent les composants imposés par la réalité : le noyau fonctionnel réalise les concepts du domaine ; le composant d'interaction, en contact direct avec

l'utilisateur, est mis en œuvre au moyen des objets d'interaction d'une boîte à outils. En clé de voûte, le contrôleur de dialogue gère l'enchaînement des tâches ainsi que les liens entre les objets regroupés dans les deux composants voisins : l 'adaptateur de domaine et la présentation.

L'adaptateur de domaine sert, pour l'essentiel, à ajuster les différences de modélisation des objets conceptuels entre le noyau fonctionnel et le contrôleur de dialogue. Le composant de présentation joue un rôle similaire : il permet au contrôleur de dialogue de s'affranchir du fonctionnement de la boîte à outils du niveau interaction. La présentation peut se voir comme une boîte à outils virtuelle, telle XVT [Valdès 89] qui implémente des objets de présentation concrétisés en fin de compte par les objets d'interaction d'une boîte à outil réelle.

La figure 6.4 doit se voir comme une représentation canonique. En réalité, l'importance relative des cinq composants d'ARCH varie en fonction du cas à traiter. Cette migration des fonctions entre les composants s'exprime métaphoriquement au moyen du métamodèle *slinky*, du nom de ce jouet qui, une fois mis en mouvement, voit sa masse se déplacer dynamiquement. Le choix de la répartition des fonctions entre les composants relève du savoir-faire en ingénierie du logiciel. L'équilibre doit être le résultat d'un dosage analytique des facteurs qualité comme l'efficacité et la portabilité.

Nous appellerons *instance d'ARCH*, un exemplaire de modèle ARCH produit en appliquant slinky à un système interactif particulier. L'IHM d'un système regroupe tous les composants d'une instance d'ARCH à l'exception du noyau fonctionnel.

Les composants du modèle ARCH sont décrits plus en détail lors de la présentation de notre modèle PAC-Amodeus au paragraphe suivant.

4. LES COMPOSANTS DU MODÈLE PAC-AMODEUS

PAC-Amodeus, qui se veut à la fois pratique et conceptuel, préconise la décomposition fonctionnelle d'ARCH et intègre les capacités d'affinement et de prise en compte du parallélisme de PAC. La figure 6.5 en présente les composants [Nigay 93b]. Dans ses grandes lignes, PAC-Amodeus reprend les composants du modèle ARCH dont il affine le contrôleur de dialogue, composant principal, en termes d'agents PAC.

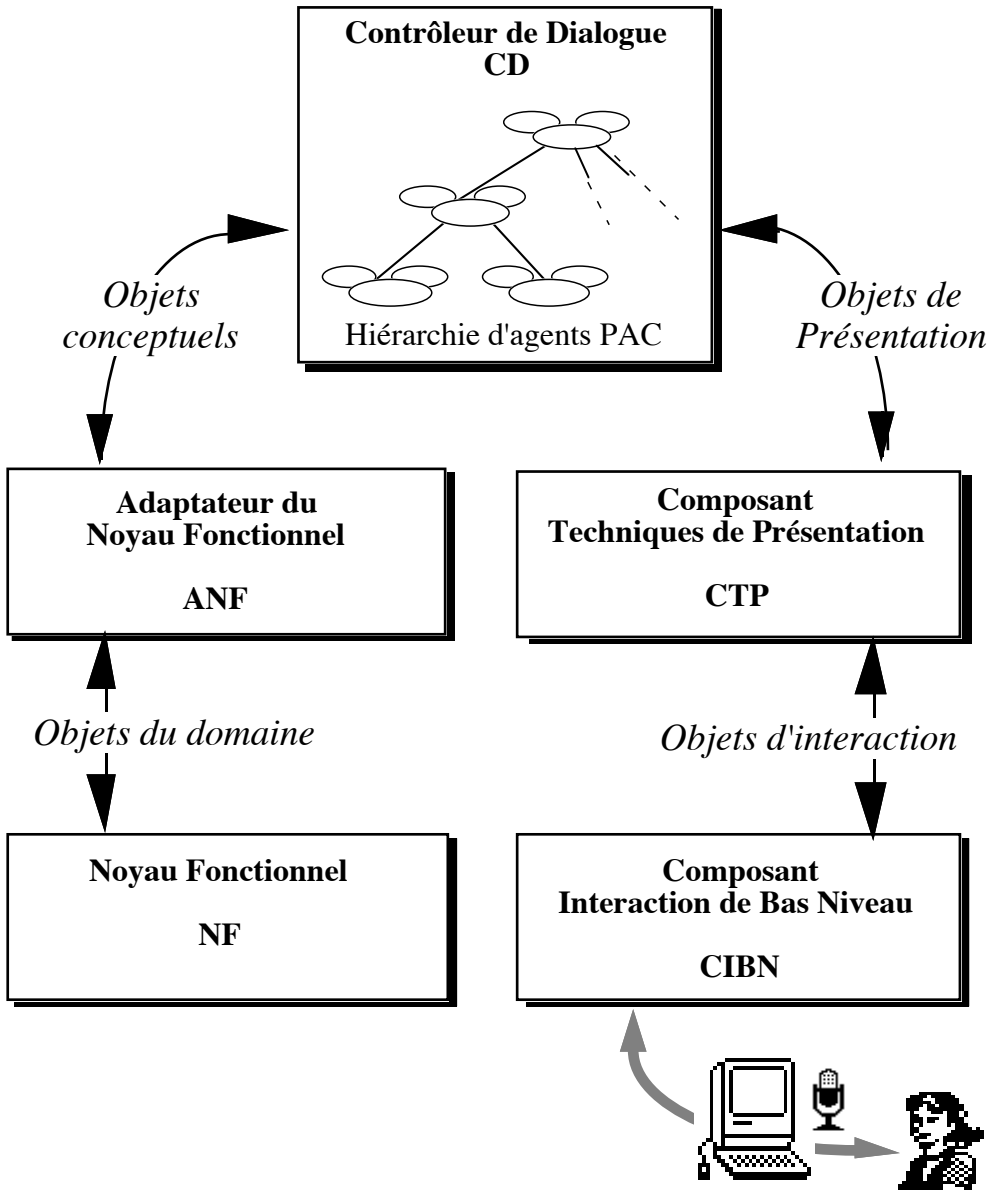


Figure 6.5 : Les composants du modèle PAC-Amodeus.

La définition de notre modèle d'architecture obéit à une double motivation, l'une liée à l'ergonomie, l'autre au génie logiciel.Il s'agit :

1. d'identifier les composants logiciels du système en sorte que soient intégrés des principes issus de l'ergonomie cognitive. En particulier, il convient de respecter le comportement opportuniste de l'utilisateur. Ceci signifie qu'il faut rendre possible la modélisation des dialogues à plusieurs fils. Nous avons vu avec PAC comment une architecture multi-agent pouvait répondre à cette demande ;

2. de satisfaire les critères qualité des logiciels énoncés par McCall [McCall 77]. La conception itérative des IHM, la seule approche qui soit réaliste actuellement, requiert la modifiabilité du logiciel. La décomposition modulaire des architectures multi-agent contribue à la modifiabilité ; et les agents à facettes comme ceux de PAC et de MVC [Goldberg 84], augmentent encore les capacités d'adaptation du logiciel. Si l'on considère les coûts de développement des IHM, la réutilisabilité et la portabilité sont deux autres facteurs essentiels. ARCH, avec ses niveaux Présentation et Interaction, fournit des éléments de réponse réalistes.

Nous allons montrer comment les cinq composants de PAC-Amodeus répondent à ces objectifs : le noyau fonctionnel, l'adaptateur de noyau fonctionnel, le contrôleur de dialogue, le composant des techniques de présentation et le composant d'interaction de bas niveau.

4.1. Le Noyau fonctionnel et l'utilisateur

Le Noyau Fonctionnel (NF) et l'utilisateur sont les deux extrêmes du modèle. A gauche de la figure 6.5, le NF réalise les concepts du domaine en ignorant la façon dont ces concepts sont rendus perceptibles à l'utilisateur. Une base de données comme celle de MATIS, le savoir-faire d'un robot mobile de surveillance sont des exemples de noyaux fonctionnels. Les autres composants du système, l'Adaptateur du Noyau Fonctionnel, le Contrôleur de Dialogue et les Composants de Présentation et d'Interaction, constituent l'IHM du système.

L'utilisateur et le Noyau Fonctionnel produisent et consomment des informations par l'intermédiaire de l'IHM. Ce fonctionnement symétrique laisse libre le choix du contrôle de l'interaction en fonction du cas à traiter. Avec l'introduction du parallélisme, les notions de contrôle interne, externe, et mixte [Tanner 83] sont maintenant surannées.

Le Noyau Fonctionnel échange des concepts du domaine ou *objet du domaine* avec son voisin immédiat : l'Adaptateur du Noyau Fonctionnel.

4.2. L'Adaptateur du Noyau Fonctionnel

L'Adaptateur du Noyau Fonctionnel (ANF) est une interface conçue pour absorber les changements entre ses voisins directs. Comme toute frontière, il implémente un protocole. Un protocole comprend généralement trois dimensions : stratégies temporelles, nature des informations échangées, et mécanisme de liaison. On trouvera dans [Coutaz 91], une analyse de

ces trois axes. Les stratégies temporelles instaurent par exemple une communication synchrone ou asynchrone :

• Communication synchrone : le Noyau Fonctionnel et l'IHM sont mutuellement asservis. Ce type de communication permet au mieux des fils de dialogue entrelacés.

• Communication asynchrone : le Noyau Fonctionnel et l'IHM sont des partenaires égaux dans la conduite de l'interaction. Ce type de communication permet plusieurs fils de dialogue simultanés. C'est le cas du système MATIS. La base de données est un processus distant lié par une connexion serveur avec l'interface. L'utilisateur peut continuer à travailler tandis que la base de données est en cours de recherche d'informations suite à la soumission d'une requête.

L'ANF est le lieu approprié pour la mise en œuvre des *améliorations sémantiques* et de la *délégation sémantique* :

• Améliorations sémantiques : en raison d'une erreur d'analyse ou par contraintes logicielles et matérielles, un concept peut être modélisé dans le noyau fonctionnel d'une manière inadaptée aux besoins de l'utilisateur. Par exemple, ce concept est découpé en de multiples structures disjointes alors qu'il constitue une unité cognitive. Une amélioration (ou une réparation sémantique) peut être pratiquée dans l'ANF en regroupant plusieurs entités du Noyau Fonctionnel au sein d'un objet conceptuel ou inversement, par découpage d'entités en plusieurs objets conceptuels. Par exemple dans NoteBook, une note est un fichier texte au niveau du Noyau Fonctionnel dont le rôle est de gérer un ensemble de fichiers. Comme le montre la figure 6.6, l'ANF traduit ce fichier texte ou objet du domaine en plusieurs objets conceptuels utiles à l'interface. La notion de zone que nous avons introduite pour l'interface d'un robot mobile de surveillance est un autre exemple d'amélioration sémantique [Chabert 89a]. On en trouvera la description dans [Bass &Coutaz 90].

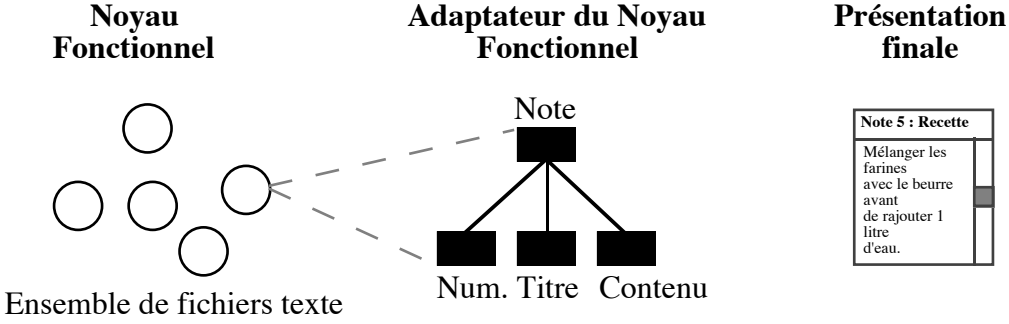


Figure 6.6 : Améliorations sémantiques dans NoteBook.

Le groupement et le découpage impliquent pour l'ANF des opérations de fusion et de fission au sens de MSM. Ici la fusion et la fission se pratiquent à un haut niveau d'abstraction.

• La délégation sémantique relève de l'ingénierie logicielle. Elle consiste à déporter dans l'IHM, et notamment dans l'ANF, les compétences du Noyau Fonctionnel en vue d'améliorer les performances. La qualité sémantique des retours d'information exige de fréquents allers et retours entre l'IHM et le noyau fonctionnel. Lorsque la chaîne de transformations et de transferts entre l'utilisateur et le Noyau Fonctionnel ne présente pas les performances requises (c.-à-d. n'est pas en accord avec les temps de réponse attendus de l'utilisateur), la décharge dans l'interface (par exemple, sous formes de caches) doit être pratiquée. L'ANF constitue l'un de ces lieux de délégation lorsque le noyau fonctionnel est exécuté à distance. Mais, nous l'avons vu au pragraphe 4.2, la délégation peut se pratiquer dans tous les composants de l'interface.

Dans MATIS, l'ANF sert d'interface entre l'IHM et la base de données (langage S.Q.L) constituant le Noyau Fonctionnel. L'ANF rend ainsi l'interface de MATIS complètement indépendante de la base de données. La modification de celle-ci impliquerait la modification de l'ANF uniquement.

L'ANF échange des *objets conceptuels* avec son voisin de droite immédiat : le Contrôleur de Dialogue.

4.3. Le Contrôleur de dialogue

En clé de voûte de l'architecture, le contrôleur de dialogue (CD) assure, comme le contrôleur de ARCH, plusieurs fonctions :

• l'enchaînement des tâches,

• la transformation de formalisme (les représentations internes utilisées dans l'IHM ne sont pas nécessairement en accord avec celles du Noyau Fonctionnel ou de l'ANF, son représentant),

• la mise en correspondance entre le monde des concepts et celui de l'interface (par exemple, il faut être capable de modéliser le fait que le réel T du Noyau Fonctionnel

est rendu perceptible à l'utilisateur au moyen de la jauge J), ou mise en correspondance entre objets conceptuels et objets de présentation,

• la construction d'une interaction coopérante comme la gestion des erreurs de communication.

Contrairement au contrôleur des modèles ARCH ou Seeheim, le Contrôleur de Dialogue (CD) de PAC-Amodeus n'est pas un monolithe obscur mais une collection organisée d'agents PAC formant un pont entre l'ANF et le Composant Techniques de Présentation. Nous illustrons l'intérêt de cette solution en traitant successivement les fonctions du CD, sa décomposition à résolution multiple, sa formalisation possible.

4.3.1. Les fonctions du Contrôleur de Dialogue

Nous reprenons le rôle premier du CD : l'enchaînement des tâches. Une tâche est un objectif que l'utilisateur se fixe accompagné de la procédure qui permet de l'atteindre. Une procédure est une encapsulation de sous-tâches. Du point de vue système, cette définition se traduit par l'état du système que l'utilisateur souhaite observer associé à une procédure. Ici, la procédure est une encapsulation de sous-tâches dont les tâches élémentaires sont des commandes du système.

Dans leur modélisation des tâches et sous-tâches, les concepteurs ergonomes ont pu identifier la nécessité de permettre l'entrelacement, voire le parallélisme entre tâches. Du point de vue conception logicielle, l'occurence de ces phénomènes, se traduit, nous l'avons vu, par des agents à raison d'une grappe par fil de dialogue. Au-delà d'un sous-dialogue, une grappe traduit également des niveaux d'abstraction. Chaque niveau correspond à des traitements dans les processus d'interprétation et de restitution de l'espace MSM.

La deuxième fonction du CD est d'assurer la transformation de formalisme. Par nature, l'ANF et le Composant Techniques de Présentation utilisent des formalismes distincts, le premier dirigé par des considérations opératoires, le second par les techniques multimédia des boîtes à outils. Afin que ces formalismes se rejoignent, les informations en transit dans le CD doivent être transformées. On parle d'opérations d'abstraction (dans le sens Utilisateur - Noyau Fonctionnel) et de concrétisation (dans le sens Noyau Fonctionnel - Utilisateur).

Enfin le CD doit maintenir la relation entre les objets conceptuels et les objets de présentation. Les changements d'état dans l'ANF doivent être reflétés dans la présentation perceptible par l'utilisateur (et vice versa). Aussi des liens doivent-ils être maintenus entre les objets conceptuels et les objets de présentation. Un objet conceptuel peut avoir plusieurs

représentations. Dans ces conditions, la cohérence doit être assurée entre les divers objets de présentation reliés au même objet conceptuel. Nous verrons au paragraphe 6 comment cette cohérence peut être assurée.

L'expérience montre que les trois grandes classes de fonction d'un CD doivent être distribuées dans des entités spécialisées, hautement réactives : des agents. Le paragraphe qui suit nous en précise la nature.

4.3.2. Le Contrôleur de Dialogue à résolution multiple

Au premier niveau de résolution, le CD peut être perçu comme le monolithe d'ARCH. Par raffinements successifs, le concepteur logiciel est capable d'identifier les niveaux d'agents nécessaires aux opérations d'abstraction/concrétisation. Orthogonalement à cet axe vertical du processus de raffinement et de composition, nous utilisons les trois facettes horizontales d'un agent PAC introduites au paragraphe 2. Ainsi, nous caractérisons un agent selon deux axes :

- son niveau d'intervention dans les transformations successives de l'information,

- ses trois facettes complémentaires fortement couplées issues du modèle PAC : Présentation, Abstraction, Contrôle.

Comme nous l'avons expliqué au paragraphe 2, la facette abstraction d'un agent modélise sa compétence opératoire tout comme le Noyau Fonctionnel représente le fonctionnement conceptuel du système. Comme pour l'ANF, cette facette peut être exploitée pour y pratiquer des délégation et amélioration sémantiques. Comme le montre la figure 6.7, elle est généralement reliée à des objets conceptuels de l'ANF.

La facette présentation d'un agent définit son rendu vis-à-vis de l'utilisateur. Comme le montre la figure 6.7, elle est reliée à un (ou plusieurs) objets de présentation et peut, si besoin est, maintenir des informations d'état en vue de contrôler son comportement perçu par l'utilisateur. La définition de la présentation est indépendante du langage d'interaction et du dispositif physique utilisés pour la rendre perceptible à l'utilisateur. De façon symétrique pour les informations reçues de l'utilisateur, la connaissance du langage et du dispositif d'entrée n'est pas pertinente pour l'agent, sauf si celle-ci est un concept du Noyau Fonctionnel.

La facette contrôle est un transformateur de formalisme entre les deux facettes horizontales qu'il relie et intervient dans la chaîne verticale des processus d'abstraction et de concrétisation.

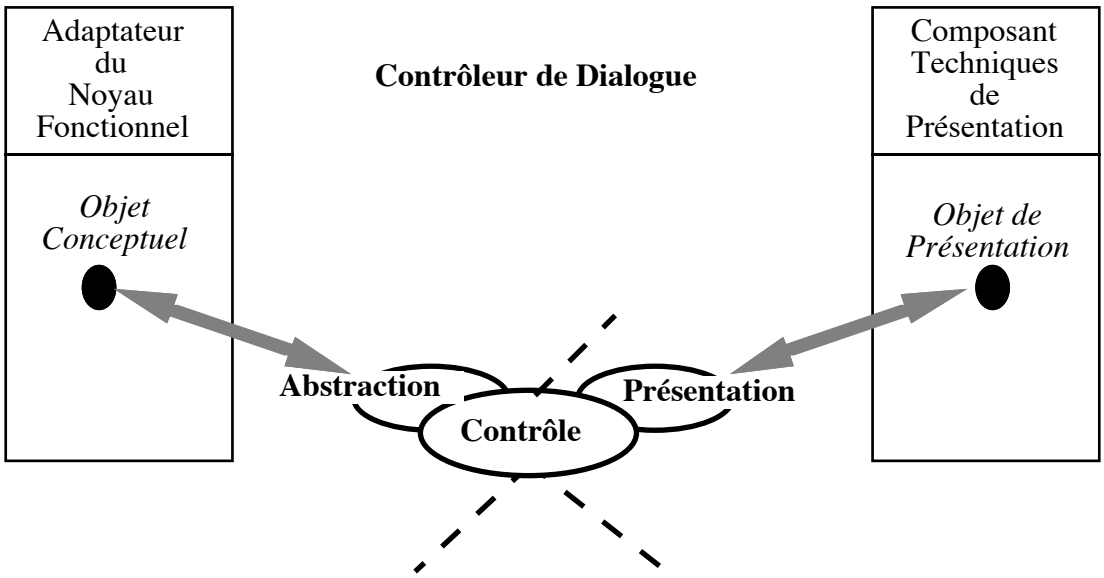


Figure 6.7 : Un agent PAC du Contrôleur de Dialogue.

Les lignes en pointillé représentent les liens avec d'autres agents dans l'axe vertical du processus d'abstraction/concrétisation. Les flèches montrent les liens avec les composants logiciels.

L'illustration de la figure 6.7 est une vue idéalisée de la réalité. Elle définit le canevas de conception logicielle. L'expérience montre que certains agents ne présentent pas nécessairement toutes les propriétés recommandées par le modèle. Aussi avons-nous abouti expérimentalement à la classification suivante.

Liaison avec des objets conceptuels

Un agent peut ou non être relié à un objet conceptuel :

- relié à un objet conceptuel signifie que l'agent participe à la chaîne verticale de transformations ;

- non relié indique qu'il ne dépend pas du noyau fonctionnel mais qu'il participe à l'accomplissement de la tâche par amélioration ou réparation sémantique (c'est un "agent syntaxique").

Présence des facettes

Un agent a toujours une facette contrôle (il doit communiquer avec ses pairs) mais ses facettes Abstraction et Présentation peuvent être absentes.

- La facette Abstraction peut être absente : c'est le cas lorsque l'agent est en correspondance directe avec un objet conceptuel. Afin d'éviter des duplications

d'information, la compétence abstraite de l'agent est reléguée dans l'objet conceptuel et l'abstraction se limite à l'identification de l'objet conceptuel.

• La facette Présentation est absente lorsque l'agent est utilisé comme unité de calcul ou de contrôle. De tels agents ne sont pas des feuilles de la hiérarchie et doivent avoir des agents fils.

Considérant ses fonctions et son indépendance voulue vis-à-vis des langages d'interaction et des dispositifs physiques, le contrôleur de dialogue a la charge des fonctions Spec-Exec-AS et Spec-Rendu-Eff de l'espace Pipe-Lines. Par conséquent, par la facette Présentation de chaque agent, le CD reçoit et émet des unités informationnelles qui sont équivalentes à des Objets de Présentation.

4.3.3. Formalisation du Contrôleur de Dialogue

Composant principal du modèle, le Contrôleur de Dialogue peut être spécifié formellement en utilisant le langage présenté au paragraphe 2.4. L'opérateur Fils-de, qui permet de construire la hiérarchie d'agents, abstrait à chaque niveau les événements internes à un agent PAC. Reprenant la définition introduite au paragraphe 2.4 :

$$(a \parallel b \parallel c) \,/\, \text{Evts-internes}\ \text{un élément de base avec}$$
$$\text{Evts-internes}\ (a \parallel b \parallel c) = (\text{Ev}(a) \cup \text{Ev}(c)) \cap \text{Ev}(b)$$

Dans PAC-Amodeus, les événements externes à un agent sont :

• ceux reçus de l'utilisateur ou émis vers l'utilisateur via sa facette Présentation : Evts-U,

• ceux reçus du Noyau Fonctionnel ou émis vers le Noyau Fonctionnel via sa facette Abstraction : Evts-NF,

• ceux reçus des agents ou émis vers des agents via sa facette Contrôle : Evts-Agent.

Ainsi :

$$\text{Ev}(a \parallel b \parallel c) = \text{Evts-internes} \cup \text{Evts-U} \cup \text{Evts-NF} \cup \text{Evts-Agent}$$

A la dernière étape, au niveau d'abstraction le plus élevé, la spécification du Contrôleur de Dialogue est constituée d'un seul agent qui communique par sa Présentation pour les

événements en provenance ou à destination de l'utilisateur et par son Abstraction pour les événements en provenance ou à destination du Noyau Fonctionnel.

Le contrôleur de dialogue échange des *objets de présentation* avec son voisin de droite immédiat : le Composant Techniques de Présentation.

4.4. Le Composant Techniques de Présentation

Le Composant Techniques de Présentation définit le comportement perceptible conceptuel du système en termes d'objets conceptuels de présentation. Un objet conceptuel de présentation est un interacteur abstrait, idéalisé, qui traduit un besoin communicationnel. Le choix, l'alerte, le patron[20], la mesure, la description, la liste, etc. sont des exemples de besoins communicationnels. Ces objets conceptuels de présentation peuvent donner lieu à différentes incarnations selon leurs attributs et la disponibilité des objets d'interaction du composant Techniques d'Interaction. Par exemple, le choix devient un menu, l'alerte est traduite en une boîte de dialogue "modale", le patron vient en correspondance avec un formulaire, la mesure est concrétisée par une jauge, la description est véhiculée par un message en langage naturel et la liste devient un tableau.

Dans ces conditions, le composant Techniques de Présentation est une couche logicielle qui assure la portabilité du contrôleur de dialogue sur différentes plates-formes d'accueil (tels Motif, MS Windows) et qui rend le contrôleur de dialogue indépendant des langages et dispositifs du système. Il peut aussi se voir comme l'extension de boîtes à outils si celles-ci ne comportent pas les objets d'interaction idoines. Par exemple, Motif qui, dans notre architecture, relève du composant Techniques d'Interaction, ne dispose pas de widget équivalent aux "cones trees" de Robertson et Mackinlay [Mackinlay 90]. Dans notre architecture, ce widget serait réalisé comme extension de Motif et serait installé dans le composant Techniques de Présentation. De même, les machines graphiques à images abstraites relevant de la technique des boîtes [Nanard 84, Coutaz 85, Quint 87] relèvent, par leur fonction d'abstraction, de ce composant.]

En conséquence, et de manière idéale, le composant Techniques de Présentation comprend deux niveaux d'abstraction : la couche extension des techniques d'interaction et la couche conceptuelle qui assure la portabilité. En pratique, si la portabilité n'est pas requise, la couche conceptuelle peut être éliminée. (On voit ici, un exemple d'application du métamodèle "slinky" par réduction de l'importance relative des techniques de présentation.)

---

[20] En anglais : "template".

Quelle que soit son organisation pratique, le CTP définit la correspondance entre les objets de présentation conceptuels et les objets d'interaction du composant Techniques d'interaction :

• Dans le sens Contrôleur de Dialogue vers l'utilisateur, le CTP traduit les objets de présentation en termes d'objets d'interaction. C'est là que le système effectue le choix, au sens de ULD, du langage d'interaction. Le ou les objets d'interaction sont ensuite concrétisés par le Composant d'Interaction de Bas Niveau qui inclut le pilotage des dispositifs physiques.

• Dans le sens inverse (de l'utilisateur vers le Contrôleur de Dialogue), le CTP abstrait les objets d'interactions reçus en termes d'objet de présentation qui sont indépendants des langages et dispositifs utilisés en entrée.

Pour illustrer la correspondance entre les composants Techniques de Présentation et Techniques d'Interaction, considérons un message de confirmation que le contrôleur de dialogue aurait décidé d'émettre à destination de l'utilisateur. Le message de confirmation est un objet de présentation qui peut être concrétisé par une boîte de dialogue graphique ou bien par un message en langage naturel vocal. La boîte de dialogue est un objet d'interaction composé d'objets d'interaction élémentaires (bouton, chaîne de caractères, etc.). Le message vocal est également un objet d'interaction réalisé par le service de synthèse du composant d'interaction de bas niveau.

Le CTP réalise donc en partie les fonctions Acqu-Inter-AP et Spec-Exec-AP de l'espace Pipe-Lines. L'autre partie revient au Composant d'Interaction de Bas Niveau.

4.5. Le Composant Interaction de Bas Niveau

Le Composant d'Interaction de Bas Niveau désigne la plate-forme d'accueil à la fois logicielle et matérielle. Il comprend le système de fenêtrage tel X window et ses boîtes à outils. Si l'on considère l'interaction vocale, il inclut un moteur de reconnaissance de la parole et de manière générale tout système de reconnaissance : reconnaissance de geste à partir de la souris, du gant numérique, du stylo, ou de caméras ; reconnaissance de visage, localisation du regard, détection de la présence de l'utilisateur devant le terminal au moyen de caméras [DeMarconnay 93], système d'analyse de l'écriture.

Le regroupement des systèmes de reconnaissance en un composant unique est une modélisation conceptuelle qui traduit un niveau d'abstraction commun. En pratique, ce niveau

est géré par un ensemble de processus qui représentent les capacités computationnelles des canaux digitaux de MSM. Dans un modèle à agents systématique, ces processus sont les agents du système du niveau d'abstraction le plus bas, voire une encapsulation d'agents. Ils ont comme fonction celle de serveur de dispositifs physiques. Ils peuvent être centralisés sur un même processeur ou bien répartis sur une architecture matérielle multiprocesseur fortement couplée ou distribuée.

Dans l'espace Pipe-Lines, le composant Techniques d'Interaction réalise les fonctions d'acquisition et de restitution de bas niveau. Le paragraphe qui suit récapitule le rôle des composants de PAC-Amodeus par rapport aux fonctions de la perspective Pipe-Lines.

5. PAC-AMODEUS ET L'ESPACE PIPE-LINES

Notre mise en relation de PAC-Amodeus avec l'espace Pipe-Lines va se faire selon deux perspectives : la localisation des fonctions de Pipe-Lines dans une architecture PAC-Amodeus et l'impact des langages d'interaction et des dispositifs physiques dans PAC-Amodeus.

5.1. PAC-Amodeus et les fonctions de l'espace Pipe-Lines

L'espace Pipe-Lines, présenté au chapitre III, explicite les traitements effectués par l'utilisateur et le système en définissant trois couples de fonctions. Nous avons montré au chapitre IV que la caractérisation des fonctions de Pipe-Lines permet de décrire différentes classes de systèmes. En localisant ces fonctions au sein du modèle PAC-Amodeus, nous démontrons que Pipe-Lines est aussi un espace de conception logicielle qui préconise un découpage fonctionnel.

La figure 6.8 indique la localisation des fonctions de l'espace Pipe-Lines dans le modèle d'architecture PAC-Amodeus. La position d'une fonction définit son lieu d'exécution. Nous déterminons ainsi les composants logiciels de PAC-Amodeus qui interviennent ou peuvent intervenir dans l'exécution des fonctions. Nous notons que les deux étapes successives des fonctions Acqu-Inter-AP et de sa réciproque, Spec-Exec-AP, sont exécutées dans des composants distincts : le Composant Techniques de Présentation et le Composant Interaction de Bas niveau. En se rapportant à l'espace de classification ULD, le premier choisit le langage d'interaction : en entrée, pour effectuer l'analyse des informations en provenance de l'utilisateur, en sortie, pour définir l'expression de sortie. Le Composant Interaction de Bas Niveau, qui englobe les plates-formes d'accueil matérielles et logicielles, a la charge de l'acquisition et de la restitution en pilotant les dispositifs physiques. Il choisit les dispositifs physiques.

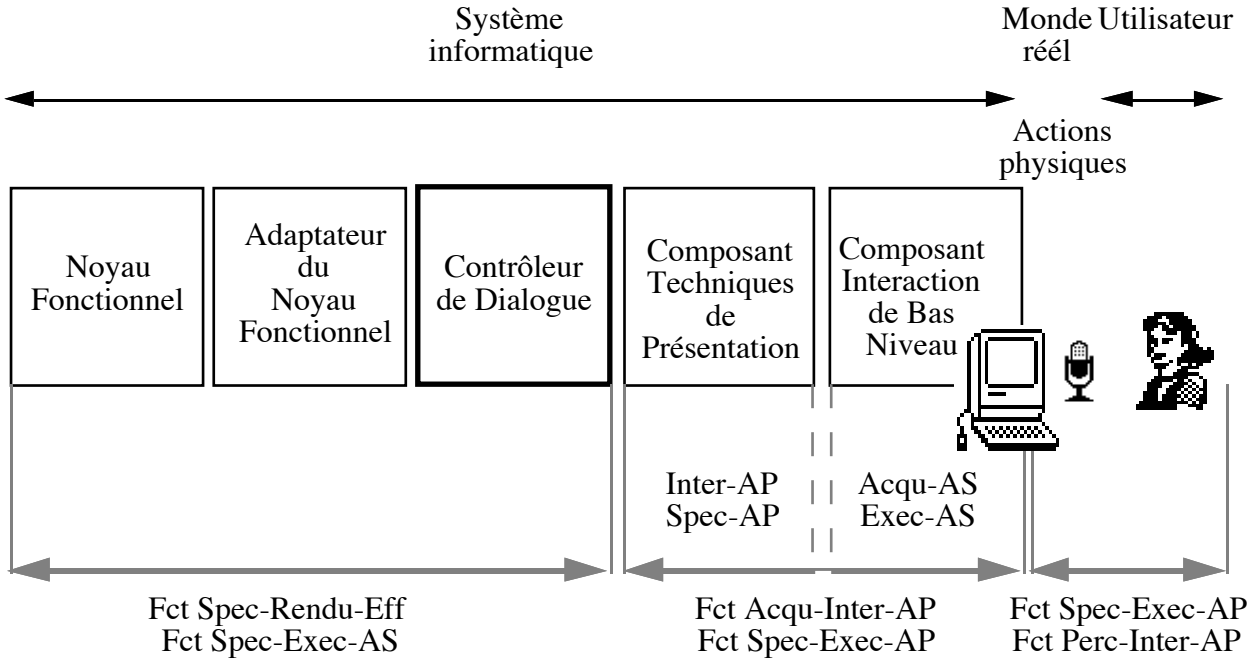


Figure 6.8 : Les fonctions de l'espace Pipe-Lines et leurs relations avec les composants de PAC-Amodeus.

5.2. PAC-Amodeus, langages d'interaction et dispositifs physiques

Dans la présentation du modèle PAC-Amodeus, nous avons souligné l'indépendance des trois composants Noyau Fonctionnel, Adaptateur du Noyau Fonctionnel et le Contrôleur de Dialogue vis-à-vis des langages d'interaction et des dispositifs physiques. En effet, les agents PAC du Contrôleur de Dialogue communiquent par leur facette Présentation avec le Composant Techniques de Présentation en échangeant des objets de présentation. Ceux-ci sont indépendants des langages et des dispositifs.

Cette indépendance est également mise en évidence par les relations de localisation des fonctions de l'espace Pipe-Lines dans les composants de PAC-Amodeus (voir paragraphe précédent). En effet, la résultante de la fonction Acqu-Inter-AP du premier plan est une unité informationnelle qui définit le plus haut niveau d'abstraction des informations en provenance de l'utilisateur. Une unité informationnelle équivalente à un objet de présentation est donc indépendante des langages et dispositifs. De même, en sens inverse, l'unité informationnelle est rendue perceptible par la fonction Spec-Exec-AP effectuée par les composants Technique de Présentation et d'Interaction de Bas Niveau qui, respectivement, choisissent le langage d'interaction de sortie de l'expression et le dispositif physique support.

Cette indépendance permet d'ajouter ou d'éliminer des langages et dispositifs en ne modifiant que les deux Composants Technique de Présentation et d'Interaction de Bas Niveau. De plus le Composant Technique de Présentation est dédié au langage d'interaction tandis que le Composant d'Interaction de Bas Niveau est dépendant des dispositifs d'entrée/sortie. Par exemple dans le système MATIS, la suppression du langage pseudo-naturel écrit (saisi au clavier) ne provoquerait pas de modification dans le Composant Technique de Présentation. De même, l'ajout d'un langage de commande oral impliquerait des modifications dans le Composant Technique de Présentation et non pas dans le Composant d'Interaction de Bas Niveau, à condition que le vocabulaire soit le même que celui du langage pseudo-naturel actuel.

Pour illustrer l'indépendance du Contrôleur de Dialogue par rapport aux langages et dispositifs, nous proposons un exemple issu de MATIS. Nous considérons que l'utilisateur a articulé la phrase "*Flights from Pittsburgh to*" tout en sélectionnant la ville de Boston dans une fenêtre résultat. Les phases de traitement correspondant sont illustrées dans la figure 6.9.

La phrase dictée est acquise et représentée par le Composant Interaction de Bas Niveau. La phrase est ensuite analysée par le Composant Techniques de Présentation. Ainsi l'objet de présentation reçu par le Contrôleur de Dialogue est l'information [From Pittsburgh To Nil] qui est indépendante du langage et du dispositif d'entrée utilisés.

En parallèle, la sélection est captée et représentée en termes de positions du curseur par le Composant Interaction de Bas Niveau. L'objet d'interaction est ensuite abstrait par le Composant Techniques de Présentation qui produit l'objet de présentation contenant l'information [From Boston To Boston]. En effet à ce niveau d'abstraction il n'est pas possible de déduire si la sélection correspond à la ville de départ ou d'arrivée.

Nous observons que les deux objets de présentation ne sont pas reçus par le même agent PAC : l'objet [From Boston To Boston] est reçu par l'agent qui gère la fenêtre de résultat. L'objet [From Pittsburgh To Nil] est émis vers l'agent "Langage Naturel" qui gère le retour d'information des phrases en langage pseudo-naturel. Ce retour d'information est particulièrement utile en phase de développement. La suppression de ce retour d'information impliquerait la réception de l'objet [From Boston To Boston] par l'agent racine de la hiérarchie.

PAC-Amodeus définit des niveaux d'abstraction, indique le principe du support des dialogues à plusieurs fils, montre les points d'ancrage avec les boîtes à outils et le noyau fonctionnel. Mais, parce qu'il est conceptuel, PAC-Amodeus ne fournit aucune indication sur la nature et le rôle des agents du Contrôleur de Dialogue qu'il convient de définir pour chaque cas de système. Cette imprécision est une source de difficultés que nous souhaitons combler. On

trouvera au paragraphe suivant un ensemble de règles heuristiques d'aide au processus d'identification des agents.

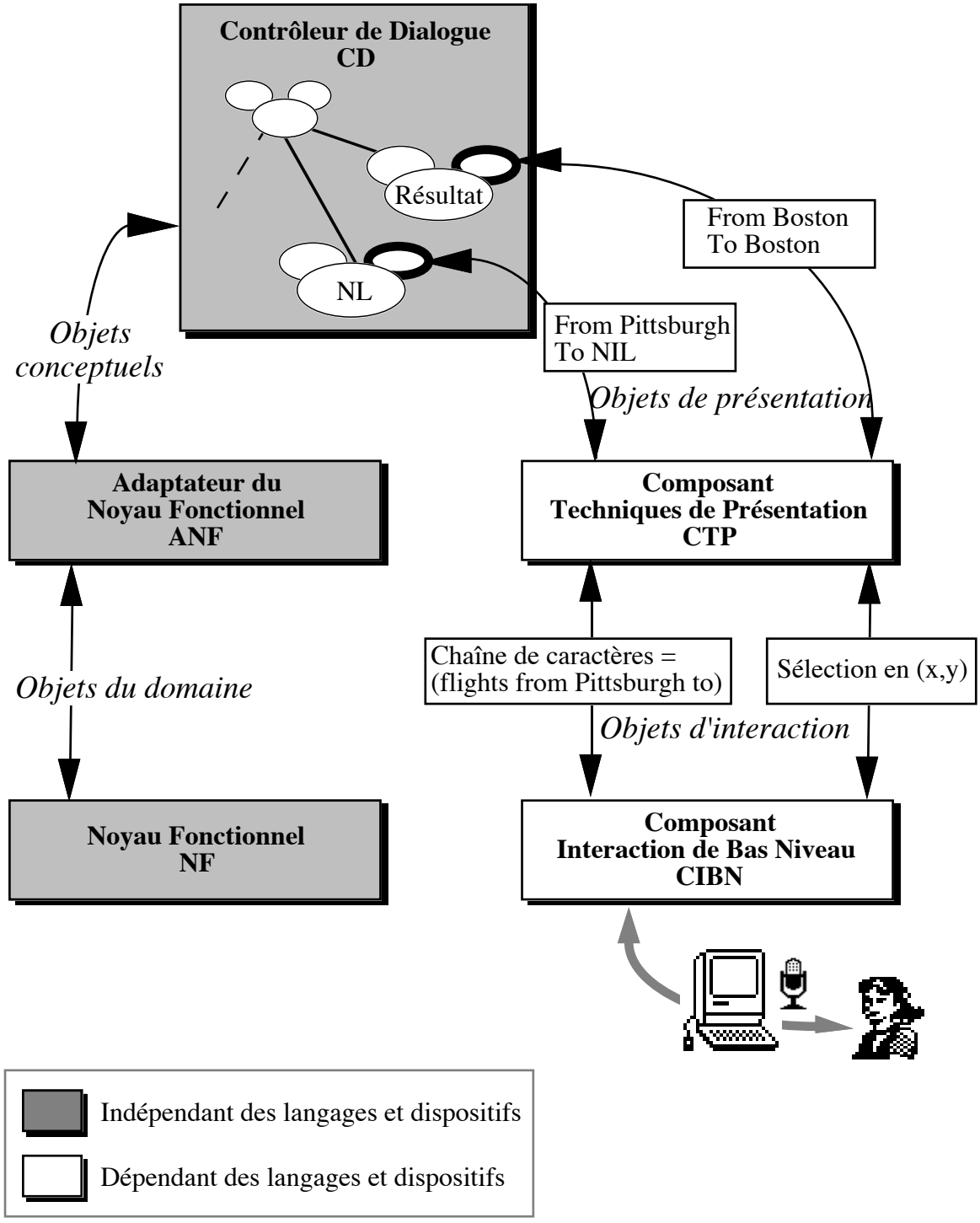


Figure 6.9 : Indépendance des composants par rapport aux langages et dispositifs.

6. RÈGLES HEURISTIQUES DE MISE EN ŒUVRE

L'approche adoptée est purement heuristique. Nos règles de structuration sont le résultat d'expériences avec PAC et PAC-Amodeus appliqués à la conception logicielle de systèmes de plusieurs dizaines, voire plusieurs centaines, de milliers d'instructions : Mithra [Chabert 89a], Expert Data-Base [Chabert 89b], Solo [Martin 89], Compo [Nigay 90b], Palas-X [Nigay 92b], les scénarios des projets AMODEUS1 et AMODEUS2 [Coutaz et al. 91b], NoteBook et MATIS.

L'application des règles exige la connaissance des spécifications externes du système. Dans cette description, la fenêtre, lieu d'interaction privilégiée avec l'utilisateur, sert d'indicateur de choix. Le jeu de règles est organisé en six catégories selon qu'elles se fondent sur l'existence de fenêtres, sur le contenu des fenêtres ou sur leurs enchaînements. Deux classes de règles ont respectivement trait aux services généraux et au mécanisme de référence entre agents. La dernière catégorie, qui regroupe les règles de réduction, procède à l'élimination d'agents superflus.

6.1. Règles et existence de fenêtre

6.1.1. Agent "espace de travail"

Il convient de distinguer les *fenêtres de commodité* qui servent de support aux boîtes de dialogue ou aux formulaires, des fenêtres qui concrétisent des *espaces de travail* comme les fenêtres d'édition de document.

Règle 1 :
» Une fenêtre qui sert de support à un espace de travail est modélisée par un agent.

La Présentation d'un agent "espace de travail" gère la fenêtre en tant que telle au moyen des services du système de fenêtrage : déplacement, changement de taille, mise sous forme d'icônes, etc. Si cet agent est une feuille de hiérarchie PAC :

- sa Présentation a la charge de restituer les concepts contenus dans l'espace de travail,
- son Abstraction maintient une représentation abstraite des concepts restitués, par exemple les liens logiques entre ces concepts.

6.1.2. Agent "vue multiple"

Pour des raisons de confort, et selon les besoins de la tâche en cours, certains concepts du noyau fonctionnel nécessitent d'être présentés plusieurs fois, quelquefois simultanément, et sous des formes éventuellement distinctes. On parle alors de *vues multiples d'un même concept*. Par exemple, dans NoteBook le titre de la note figure à la fois dans la zone d'édition de la note et dans la liste des titres du carnet (Figure 6.10). Dans l'exemple du système de contrôle de processus du chapitre III (figure 3.7), le concept de température est restitué sous deux formes : un thermomètre et une valeur numérique. Dans le premier exemple, il s'agit de redondance ; dans le second cas, il peut y avoir redondance ou complémentarité (voir chapitre IV).

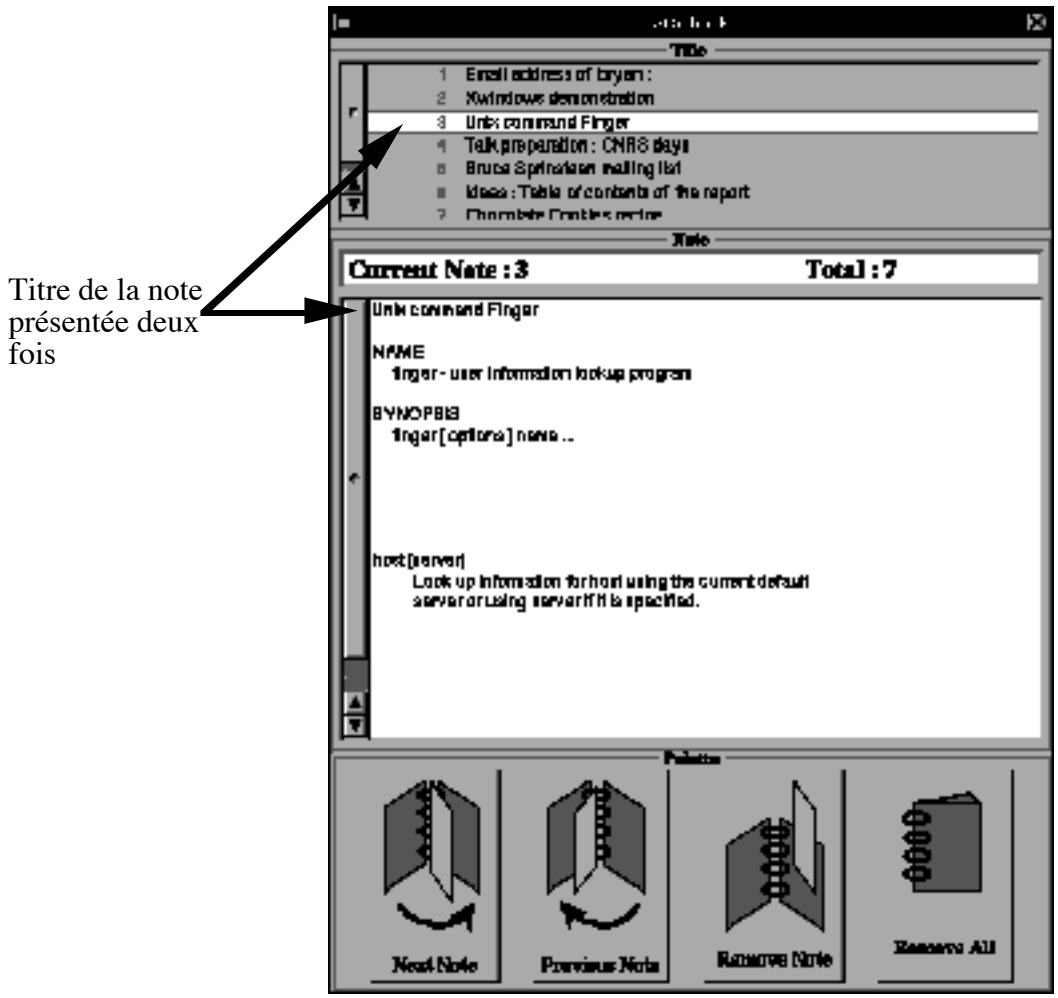


Figure 6.10 : NoteBook, vue multiple du titre pour chaque note du carnet.

Qu'il y ait redondance et/ou complémentarité, la cohérence de la présentation entre les vues doit être gérée : si l'utilisateur ou le noyau fonctionnel agit sur l'une des vues du concept, la modification doit être répercutée sur les autres. Si la répercussion ne doit pas avoir lieu, il faut aussi être capable de le décider. Comme le montre la figure 6.11 :

- Les vues sont assurées par les Présentations d'agents distincts.

- Un agent père "Vue multiple" assure la cohérence en répercutant ou non les modifications.

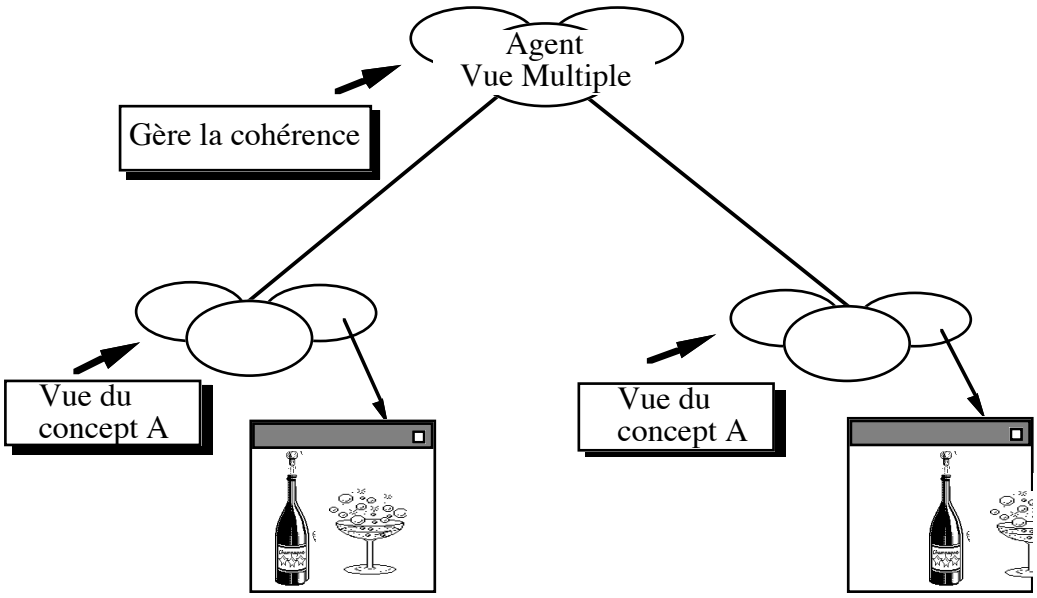


Figure 6.11 : Vue multiple d'un même concept. Un agent Père gère la cohérence.

Règle 2 :
» Les vues multiples d'un même concept sont gérées par un agent "vue multiple".

6.2. Règles et contenu de fenêtre

6.2.1. Agents "palette" et "barre de menus"

Il est fréquent que les classes de concepts du noyau fonctionnel soient regroupées dans une palette. C'est le cas de MacDraw avec sa liste des figures géométriques cercle, rectangle, droite, etc. à partir desquelles il est possible de créer des exemplaires. C'est le cas aussi d'HyperCard avec ses menus détachables. Il est fréquent que ces palettes aient à mémoriser un "mode", par exemple le mode "cercle" dans MacDraw et que ce mode nécessite d'être répercuté. Dans ce cas, et pour augmenter la réutilisabilité de ce type de comportement, nous recommandons de modéliser une palette de concepts par un agent.

Règle 3 :
» Une palette de classes de concepts est modélisée par un agent.

L'Abstraction d'un agent palette contient la liste des classes de concept et maintient l'identité de la classe sélectionnée (le mode).

La Présentation d'un agent palette est réalisée, selon le cas, dans les termes du composant Techniques de Présentation ou Techniques d'Interaction. Par exemple, si l'on dispose de Motif, la Présentation de la palette fait référence à un widget de classe "rowcolumn" dont les conditions "callback" sont reliées à des procédures de la facette Présentation. Le widget s'occupe des retours d'information lexicaux (par exemple la mise à jour des éléments en reverse vidéo de la palette). Les retours "sémantiquement plus riches" sont activés via les procédures callback. La facette Présentation peut effectuer des compléments de feedback puis transmettre le résultat de la sélection au Contrôleur de l'agent. Si la Présentation n'a pas lieu de compléter le "feedback" câblé du widget, les procédures callback peuvent alors être directement attachées au Contrôle de l'agent.

Le Contrôle de l'agent palette traduit l'action de sélection en une information sémantiquement plus riche (opération d'abstraction). Dans le cas de MacDraw, il s'agirait de traduire la sélection du ième élément du "rowcolumn" en la spécification du mode de dessin. Ensuite, il avertit son environnement (en général, l'agent père. Voir règle 9) de la sélection d'une nouvelle classe de concept.

Un raisonnement similaire peut être appliqué au cas de la barre de menus qui gère les menus jaillissants dans sa partie Présentation. Toutefois, un sous-menu détachable, qui prend le statut de palette de concepts, devient, lorsqu'il est détaché, un exemplaire d'agent dont le père est identique à celui de la barre de menus.

Règle 4 :
» Une barre de menus est modélisée par un agent.

6.2.2. Agent "zone d'édition"

Si une fenêtre contient une zone d'édition, zone où l'utilisateur peut créer et modifier des concepts, celle-ci doit être modélisée par un agent. (S'il s'agit d'une simple surface d'affichage, elle est alors assurée par l'agent "espace de travail" englobant.)

Règle 5 :
» Une zone d'édition est modélisée par un agent.

L'agent "zone d'édition" gère les manipulations possibles sur les concepts qu'il présente ainsi que les éventuelles contraintes applicables à ces concepts. Sa compétence consiste à gérer son contenu en tant qu'ensemble de concepts. A leur tour, les concepts sont modélisés par des agents distincts fils de l'agent "espace de travail" sauf si ceux-ci sont atomiques (se référer à la règle suivante).

Les contraintes entre les concepts fils peuvent être de nature géométrique et propres à l'agent de restitution. Il s'agit alors de contraintes lexicales. D'autres peuvent être de nature sémantique et se traduire ensuite en termes de contraintes lexicales. Si la gestion des contraintes sémantiques est à la charge de l'agent, on assiste alors à une délégation sémantique.

6.2.3. Agent et concept complexe

Nous dirons qu'un concept est complexe s'il remplit l'une au moins des conditions suivantes : il est composé ou bien il est atomique mais sa présentation est composée.

Un concept est composé lorsqu'il est constitué de sous-concepts et que cette structuration doit être reflétée à l'interface dans une même fenêtre. En général une hiérarchie d'agents isomorphique à la structure du concept composé convient à la restitution du concept composé.

Une Présentation composée résulte de la fusion de plusieurs objets de présentation du niveau Technique de Présentation (ou du niveau Techniques d'Interaction) mais cette fusion ne peut être modélisée comme un objet de présentation (ou d'interaction). Dans ces conditions, l'agrégation doit être réalisée par la facette Présentation d'un agent.

Par exemple, les concepts de mur et de route dans le système Mithra se perçoivent comme l'assemblage d'un segment de droite, d'un bouton de sélection et d'un menu déroulant attaché au bouton. Dans l'environnement Motif, un widget occupe une surface rectangulaire (tout au moins la version utilisée lors de l'implémentation du système Mithra). Alors un segment de droite diagonal qui serait modélisé comme un widget se verrait alloué une surface d'écran qui ne correspondrait pas au seul segment. Dans ces conditions, nous sommes contraints de faire appel aux primitives graphiques du serveur X (en particulier XDrawLine) pour dessiner le segment de droite, la boîte à outils Motif pour créer un bouton de sélection au centre du segment et un menu déroulant qui jaillit lorsque le bouton est sélectionné. Cet assemblage et ce comportement ne pouvant être définis comme une nouvelle classe de widget, se voit encapsulés comme la Présentation d'un agent mur (ou de route).

Règle 6 :
» Un concept complexe d'une zone d'édition est modélisé par un agent.

6.3. Règles et liens entre fenêtres

Les liens entre fenêtres de type "espace de travail" sont de trois sortes : les liens d'ouverture et le lien syntaxique.

6.3.1. Lien d'ouverture entre espaces de même type

Il peut arriver que depuis un espace de travail qui contient un concept de classe donnée, on ouvre un nouvel espace sur un autre exemplaire de cette classe. Par exemple, plusieurs zones d'édition peuvent être ouvertes sur des documents distincts, chaque zone disposant d'un bouton d'appel à l'opérateur "ouvrir". Dans ce cas, il convient de relier les agents d'édition sous le contrôle d'un père commun qui peut, par exemple, maintenir des variables d'état communes aux zones. La vue multiple d'un même concept est un cas particulier de cette règle puisque, au lieu de l'ouverture de plusieurs exemplaires de la même classe, il s'agit de l'ouverture multiple du même exemplaire de concept.

Règle 7 :
» Si, depuis une fenêtre "espace de travail" contenant un concept de classe donnée, il est possible d'ouvrir un espace de travail sur un autre exemplaire de la même classe, ces deux espaces sont modélisés comme des agents fils d'un agent père commun.

6.3.2. Lien d'ouverture structurel

Les objets d'interaction des boîtes à outils incitent les concepteurs à restituer les concepts composés hiérarchiques sous forme de fenêtres que l'utilisateur ouvre en cascade. Le Finder du Macintosh offre un tel exemple : le contenu d'un répertoire est dévoilé par l'ouverture d'une nouvelle fenêtre et ainsi de suite jusqu'à la rencontre des fichiers feuilles. Chaque fenêtre, au nom de la règle 5, est un agent d'édition. L'ouverture en cascade des fenêtres se traduit par la création d'agents d'édition liés par une relation père-fils.

Règle 8 :
» Si, depuis une fenêtre d'édition qui restitue, de manière synthétique, un ou plusieurs concepts composés, il est possible d'ouvrir une nouvelle fenêtre représentant plus en détail le contenu d'un de ces concepts composés, l'agent qui modélise la nouvelle fenêtre est fils de l'agent source.

6.3.3. Liens syntaxiques et agent “ciment syntaxique”

Il est fréquent que la spécification d'une commande se traduise par un ensemble d'actions utilisateur réparties sur des agents distincts. On parle alors de syntaxe distribuée.

L'exemple le plus courant de ce phénomène est la création d'exemplaires d'objet dans le domaine de l'édition graphique comme MacDraw (Macintosh) et Idraw (Xwindows). Les classes d'objets (ou les concepts du domaine) sont regroupées dans une palette tandis que les exemplaires sont manipulés dans une fenêtre de travail. Typiquement, la création d'un exemplaire s'effectue en sélectionnant dans la palette la classe d'objet désirée puis en actionnant la souris dans la zone de travail pour spécifier les attributs de l'objet (localisation, taille, etc.). Dans cet exemple, l'action sur la palette détermine le nom de la commande (Créer-objet-de-classe-X) tandis que les actions dans la fenêtre de travail spécifient les paramètres de la commande. Ces actions réparties doivent être cimentées pour construire une action plus abstraite : une commande. Comment résoudre une telle analyse syntaxique distribuée?

Un agent PAC, père des agents qui interviennent dans la définition d'une commande, assure l'analyse syntaxique des actions sur ses agents fils. Ces derniers, après analyse lexicale des actions de l'utilisateur, transmettent des unités d'entrée à l'agent père qui effectue l'analyse syntaxique.

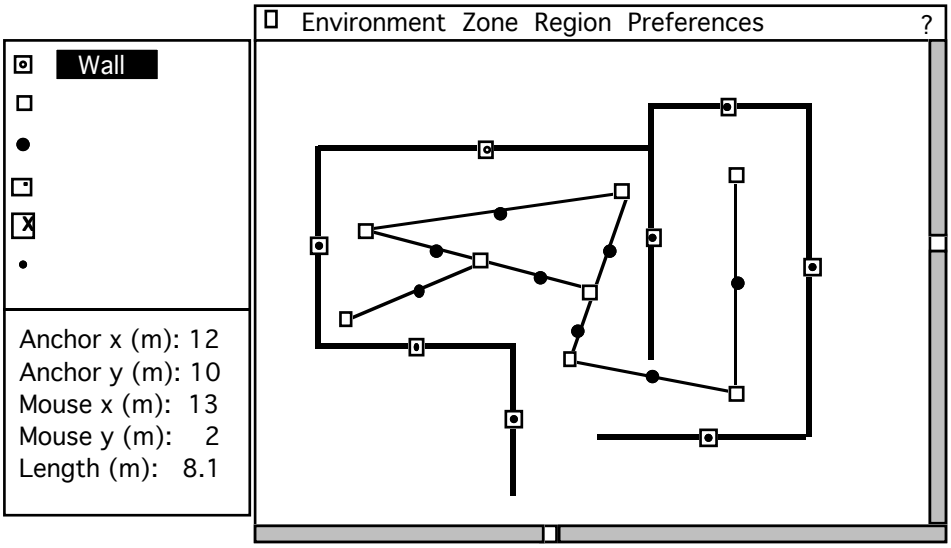


Figure 6.12 : Dans Mithra, la création d’un mur implique des actions distribuées sur la palette des concepts et la zone d’édition qui représente l’environnement d’évolution du robot.

L'application de cette technique se retrouve dans le système Mithra. Comme le montre la figure 6.12, une palette regroupe les notions de mur, de route, etc. et l'environnement d'évolution du robot est présenté dans une fenêtre d'édition. Pour créer un exemplaire de concept dans l'environnement, l'utilisateur sélectionne la classe du concept dans la palette puis positionne l'exemplaire dans l'environnement en désignant sa localisation dans la fenêtre d'édition.

La palette est gérée par un agent PAC *Palette* et la zone d'édition est réalisée par un agent PAC *Edition*. La première action de l'utilisateur est constatée par l'agent *Palette* qui effectue quelques traitements locaux (et notamment fournit un retour d'information visuel), puis avertit son père de la sélection d'une classe de concept ; la seconde action de l'utilisateur est détectée par l'agent *Edition* qui procède en local puis transmet à son père l'identification d'une localisation. L'agent père, *Superviseur d'interface*, qui assure l'analyse syntaxique, décide du sens à donner à la réception conjuguée des informations "sélection-classe-X", "localisation".

Règle 9 :
» Si la spécification d'une commande implique des actions distribuées sur plusieurs agents, ceux-ci doivent être placés sous le contrôle d'un agent qui cimente les actions réparties en une commande.

6.4. Règles et services généraux

Les messages d'erreur peuvent être dynamiques ou statiques. Le texte d'un message dynamique est construit à la volée alors qu'un message statique fournit toujours la même information quel que soit le contexte de l'interaction. Entre ces deux extrêmes, les messages paramétrés comprennent des éléments fixés une fois pour toute et des trous remplis à la volée avec les valeurs des paramètres. Nous nous intéressons ici aux messages d'erreurs paramétrés.

Nous proposons que les messages d'erreur calculés par le noyau fonctionnel soient restitués par un agent PAC spécialisé. Cet agent est à placer comme fils de la racine de la hiérarchie. En effet, le noyau fonctionnel, qui a la compétence pour détecter les erreurs sémantiques, se doit de transmettre les causes d'erreur au monde extérieur. Par souci d'efficacité, il est raisonnable de placer l'agent *Erreur* à proximité de la greffe avec le noyau fonctionnel.

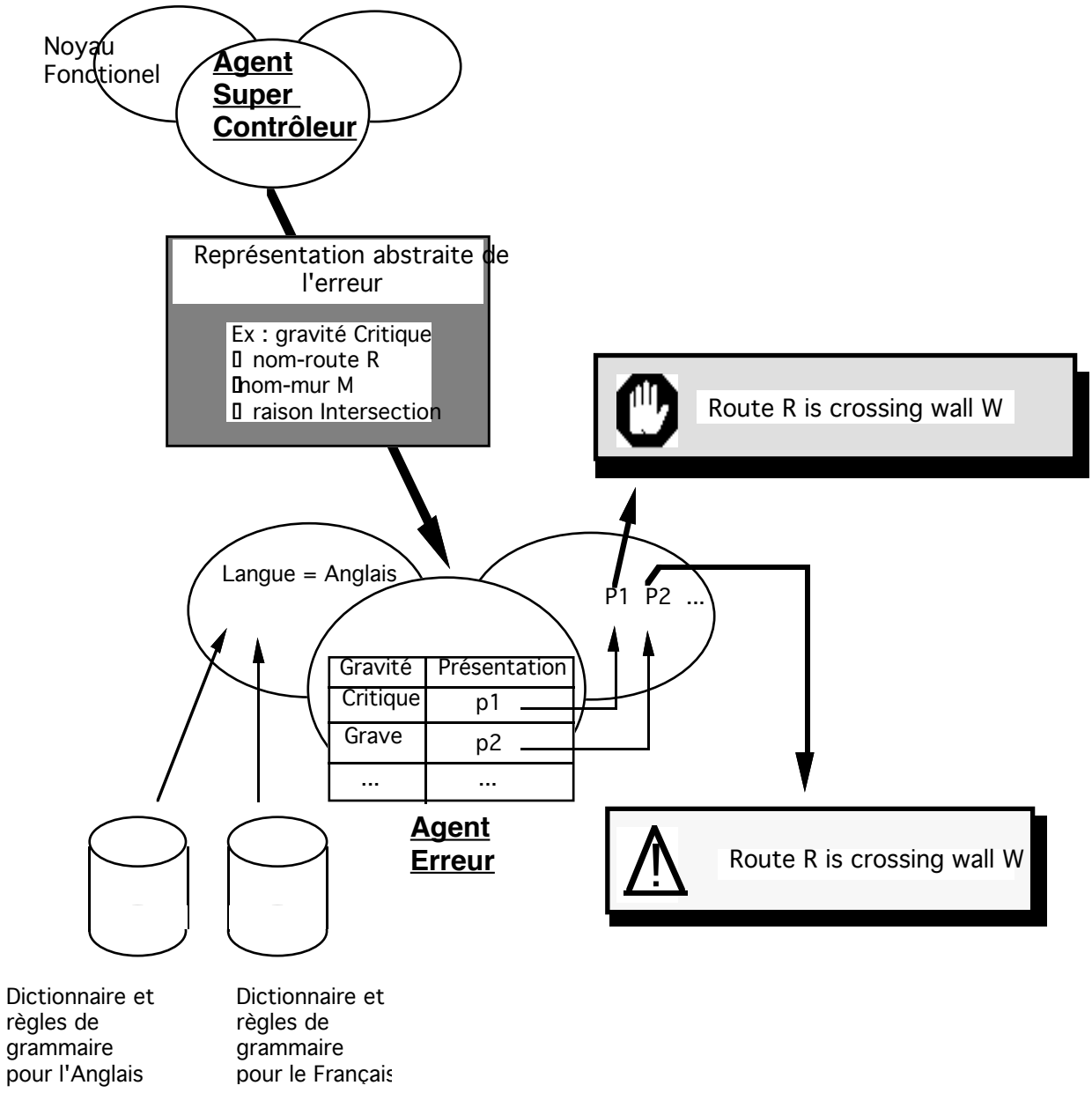


Figure 6.13 : Un agent PAC dédié à la gestion des erreurs dans Mithra.

L'erreur remise par le noyau fonctionnel est, au nom du principe de séparation entre le noyau fonctionnel et l'interface, constituée d'éléments abstraits indépendants des agents PAC de restitution. Elle contient par exemple, un code de gravité, un code d'identification et l'identité des concepts du domaine impliqués dans l'erreur. L'agent *Erreur* se charge de la restitution concrète du message abstrait. Sa Présentation contient les différentes formes de présentation à l'écran des messages d'erreur. Sa partie Contrôle assure la correspondance entre l'erreur et sa représentation. Sa partie Abstraction détermine la phrase à afficher à partir du message abstrait et en fonction du contexte de l'interaction.

La figure 6.13 illustre l'application de notre règle au cas du système Mithra. L'agent *SuperContrôleur* reçoit du noyau fonctionnel un message abstrait qu'il transmet à l'agent *Erreur*. La partie Abstraction de cet agent élabore une phrase en langue naturelle à partir de la représentation abstraite, de dictionnaires et de règles de construction. Dans l'exemple, la langue utilisée est l'anglais. La partie Contrôle de *Erreur*, maintient une correspondance entre la gravité de la situation et la forme de la présentation. Par exemple, au cours de l'édition de l'environnement, l'intersection de route et de mur est une anomalie qu'il faut signaler avec le symbole "danger". En spécification de mission, cette intersection devient inacceptable et le symbole "arrêt" s'impose!

Notons que cette solution diffère de l'approche présentée en 5.2.1 qui délègue dans l'IHM le contrôle des intersections de murs et de routes. De manière générale, les deux approches peuvent cohabiter : certains cas d'erreurs sémantiques peuvent être détectés dans l'interface, d'autres sont réservés au noyau fonctionnel. A chacun des cas, sa solution en terme d'agent. Le choix de la solution résulte, par application du métamodèle Slinky, du bon dosage de la répartition des traitements entre les composants d'un système.

Nous pouvons maintenant énoncer la règle 10 :

Règle 10 :
»□Un agent PAC doit être dédié à la gestion des erreurs sémantiques.

6.5. Règle sur les références entre agents

Comme nous l'avons expliqué au paragraphe 2, les mécanismes d'indirection assurés par le Contrôle de chaque agent PAC augmente la réutilisabilité des parties Abstraction et Présentation d'un agent. Nous nous intéressons maintenant à la réutilisabilité de l'agent lui-même.

Les règles, telles les règles de l'agent syntaxique et de l'agent vue multiple, qui introduisent des agents contrôleurs intermédiaires, contribuent à augmenter la réutilisabilité des agents feuilles. Ces feuilles ne connaissent pas l'existence des agents frères mais s'adressent à un père commun qui régit leur comportement de frère. Elles peuvent donc être réutilisées dans un autre logiciel avec d'autres frères et sous le contrôle d'un père aux règles différentes. Un agent qui effectuerait une référence explicite à un agent frère ne pourrait exister sans ce frère et ne saurait être réutilisable. De façon à rendre cette réutilisation possible, il convient que les agents se désignent par référence symbolique.

Une référence symbolique permet d'exprimer la coopération entre des entités communicantes sans que ces entités se nomment de manière explicite. La correspondance entre le symbole de désignation et l'entité effective est résolue dynamiquement à l'exécution. L'exemple de la figure 6.14, tiré du système NoteBook, illustre ce point. (On pourra se reporter à la figure 6.10 pour une présentation de l'interface perceptible de NoteBook.)

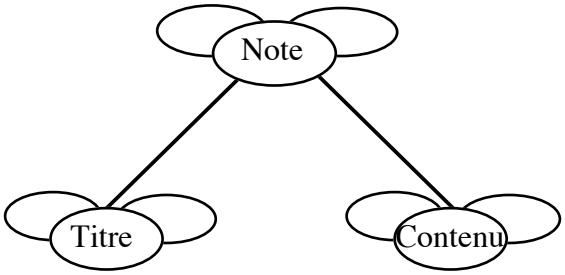


Figure 6.14 : NoteBook, relation entre deux agents via leur père commun.

Les agents *Titre* et *Contenu* communiquent par l'intermédiaire de l'agent *Note* en utilisant la référence symbolique PERE. Par exemple la première ligne du contenu d'une note définit le titre de la note ; sa modification est reçue par l'agent *Contenu* qui la répercute à l'agent *Titre* pour sa mise à jour. Celle-ci se fait par l'intermédiaire de l'agent Note : l'agent *Contenu* appelle la fonction [PERE Modifier prem-ligne = <texte>]. Les agents *Titre* et *Contenu* sont ainsi réutilisables sous condition, naturellement, d'avoir un père. Dans une architecture PAC, cette condition est véhiculée de manière automatique.

Nous pouvons maintenant énoncer la règle 11 :

Règle 11
» Utiliser des références symboliques pour la coopération entre agents afin d'assurer leur réutilisation.

6.6. Règles de réduction

L'application systématique des règles énoncées jusqu'ici peuvent conduire à une prolifération d'agents. Ce phénomène est en accord avec le principe de la modularité mais peut, selon les plates-formes de mise en œuvre, conduire à un excès de communication par message. Nous énonçons ci-dessous deux règles visant à éliminer des agents.

Par exemple, un agent et son fils unique peuvent être combinés en sorte d'éliminer une communication par messages. Les fonctions des deux agents peuvent néanmoins faire l'objet de

deux modules distincts. Cette règle ne doit être appliquée que si le père ne sera pas amené dans une version future du logiciel, à contrôler une seconde grappe d'agents.

Règle 12 :
»❑Un agent PAC et un agent fils unique peuvent être regroupés en un seul agent.

Selon les boîtes à outils dont on dispose pour la mise en œuvre, il se pourrait que les fonctions d'un agent sont couvertes par un objet de présentation ou d'interaction. Dans ce cas, il convient de lui changer de statut : il devient un composant de la Présentation de l'agent père. Les barres de menu tombent souvent sous l'effet de cette règle.

Règle 13 :
»❑Un agent dont le rôle peut être encapsulé par un objet de présentation ou d'interaction peut être éliminé et apparaître comme un composant de la Présentation de son agent père.

6.7. Synthèse

En résumé, les règles recommandent au concepteur d'architecture de s'interroger sur les points suivants :

1) Quels sont les lieux privilégiés d'interaction avec l'utilisateur : fenêtres "espace de travail", fenêtres "d'édition", palettes, barres de menus?

2) Quelle est la nature des concepts présentés : composés ou atomiques? S'il sont atomiques sont-ils realisables directement avec la boîte à outils sous-jacente?

3) Quels sont les liens entre les fenêtres : ouverture en cascade, ouverture sur une même classe, actions utilisateur réparties, vues multiples?

4) Peut-on éliminer des agents en fonction des outils de mise en œuvre?

Toutes ces règles répondent aux motivations suivantes : la réutilisabilité, la modifiabilité, le parallélisme. Si ces arguments relèvent du génie logiciel, ils trouvent leur prolongement dans la qualité ergonomique des interfaces : conception itérative des interfaces, support pour les dialogues à plusieurs fils, actions distribuées, cohérence des vues multiples.

Dans tous les cas, nos règles répondent à des principes généraux. Elle conduisent donc à concevoir des solutions générales avec l'introduction d'agents généraux pour l'élaboration du

Contrôleur de Dialogue. En conséquence, nous pouvons envisager un outil informatique qui fournirait de façon automatique l'architecture en agents PAC du Contrôleur de Dialogue. Cet outil, qui allégerait considérablement l'effort du concepteur, fait l'objet du paragraphe suivant.

7. PAC-EXPERT : UN GÉNÉRATEUR D'ARCHITECTURE LOGICIELLE

PAC-Expert est un système expert développé en CLIPS (version 4.2) [Giarratano 88] sur Macintosh : il inclut une méthode de conception d'architecture logicielle répondant au modèle PAC-Amodeus. A partir de la description du comportement externe d'un système, PAC-Expert produit l'architecture PAC-Amodeus correspondante. La représentation graphique de l'architecture constitue la spécification interne globale du système qui doit, selon les préceptes du génie logiciel (chapitre V) précéder la conception interne détaillée. La figure 6.15 localise PAC-Expert dans le cycle de développement en V décrit au chapitre 5. On trouvera dans le rapport LGI-IMAG [Nigay 92c] et le document Amodeus [Nigay 92d] une description détaillée de PAC-Expert.

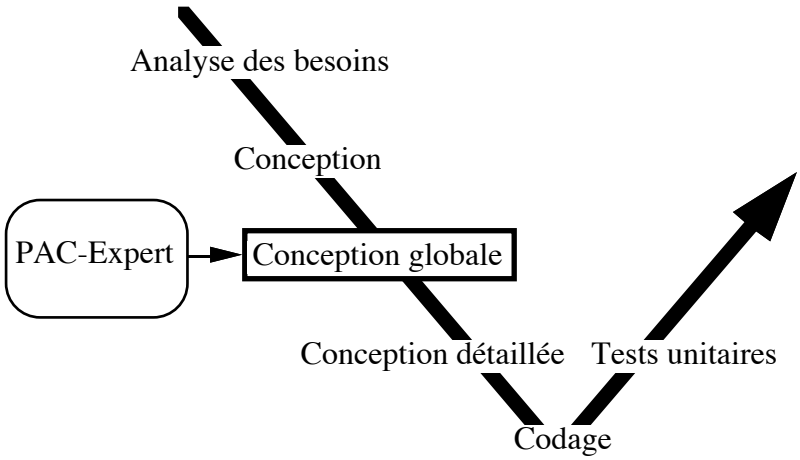


Figure 6.15 : PAC-Expert et le cycle de développement d'un système.

Comme le montre la figure 6.16, PAC-Expert est organisé en cinq phases successives de traitement :

1- La phase d'acquisition des données a lieu sous forme d'une suite de questions posées à l'utilisateur en langue naturelle. La structuration du questionnaire est guidée par les règles de génération des agents présentés précèdemment. Lors de cette phase, les spécifications externes du système sont entièrement acquises.

2- La phase de génération des agents et de leurs relations s'appuie sur les règles heuristiques qui prennent comme description l'état initial, la spécification externe du système acquise dans la phase 1.

3- La phase de réduction de la hiérarchie consiste à regrouper des agents et à supprimer des niveaux dans la hiérarchie produite à la phase 2.

4- La phase d'affichage des résultats consiste à afficher la hiérarchie d'agents sous forme graphique[21]. La figure 6.17 offre un exemple de copie d'écran.

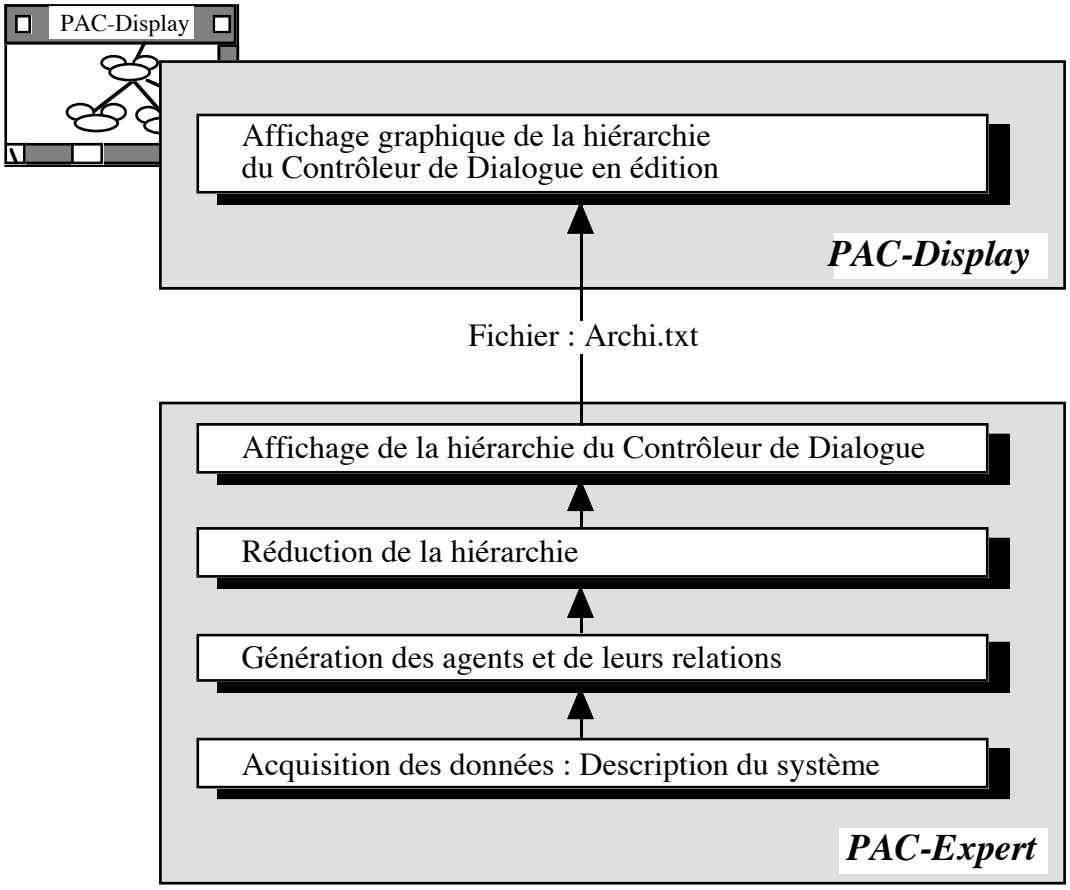


Figure 6.16 : Les phases de traitement de PAC-Expert et PAC-Display.

[21] Nous devons ce service graphique, PAC-Display, à Arno Gourdol. PAC-Display permet de visualiser mais aussi d'éditer la hiérarchie produite par PAC-Expert.

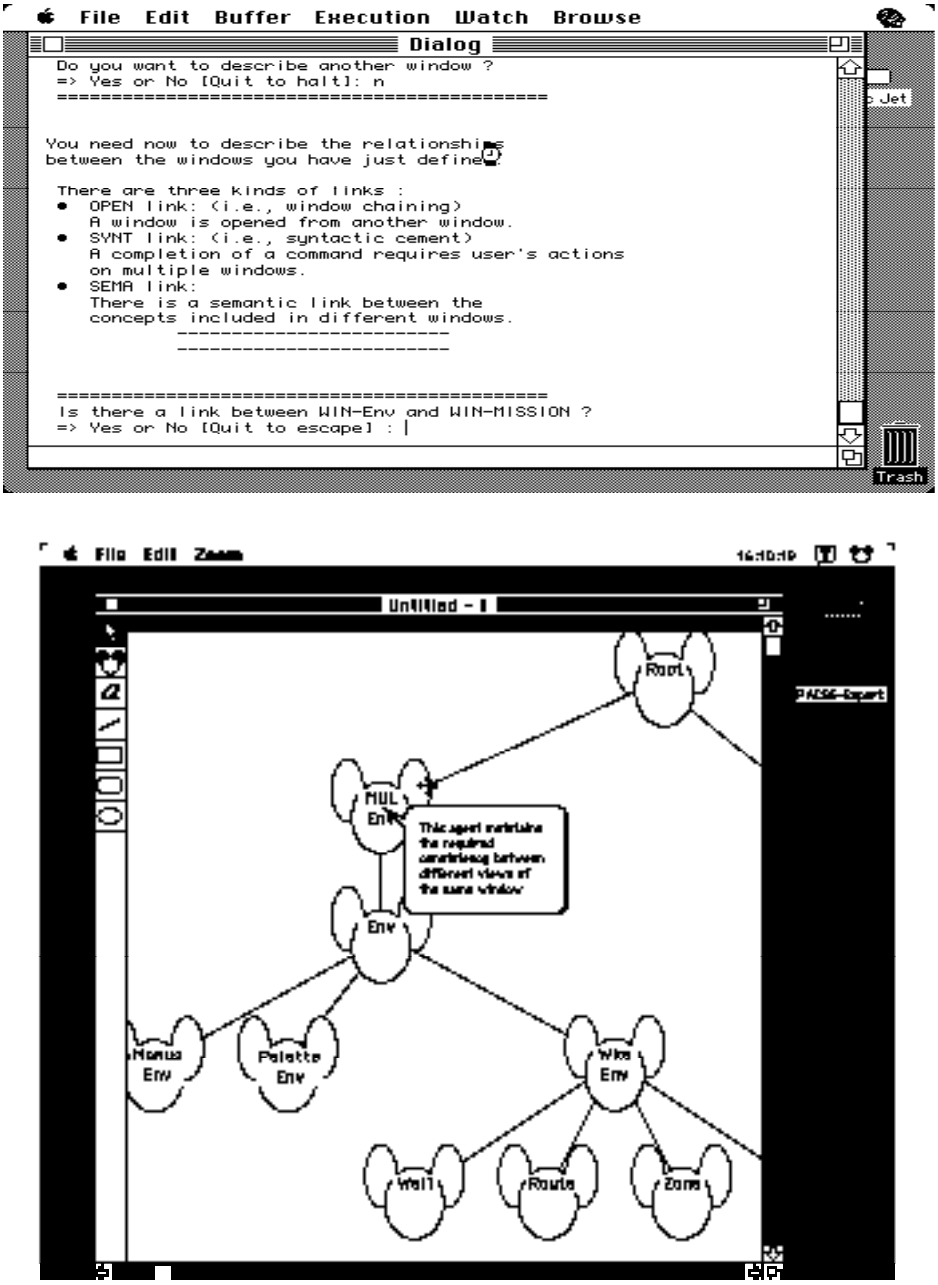


Figure 6.17 : Copies d'écran de PAC-Expert. Chaque agent PAC peut être renommé et désigné : une bulle apparaît expliquant le rôle de l'agent.

8. BILAN COMPARATIF

PAC-Amodeus, fondé sur ARCH et PAC, est un modèle d'architecture hybride. Ce faisant, il recouvre à la fois les aspects pratiques et conceptuels de l'activité de spécification d'architecture logicielle. Le canevas ARCH fournit les éléments structurants pratiques qui permettent de localiser les outils réutilisables et de pratiquer les compromis de type génie logiciel (portabilité, réutilisabilté) à la manière "slinky". PAC apporte la dimension conceptuelle avec

ses agents à facettes : capacité de concrétisation et d'abstraction, parallélisme, encapsulation et affinement. A notre connaissance, aucun modèle n'offre à la fois ces deux aspects complémentaires. Les modèles les plus innovants, qui relèvent de l'approche multi-agent, sont proches des principes de PAC : MVC [Golberg 84], ALV [Hill 92], GIO [Faconti 92]. Le modèle conceptuel de Whizz, motivé par la mise en œuvre d'interfaces animées, occupe une place particulière [Chatty 92].

Dans MVC, ALV et GIO, les agents sont comme dans PAC structurés en facettes. Le "Model", qui définit la compétence de l'agent MVC correspond à la composante Abstraction de PAC et d'ALV. La "View" recouvre la fonction de restitution de l'agent et le "Controller", sa fonction d'acquisition des actions utilisateur. Le couple "View, Controller" est équivalent à la facette Présentation de PAC et à la facette "View" d'ALV.

GIO est inspiré de la pratique graphique. Comme le montre la figure 6.18, un agent GIO est une unité computationnelle qui définit un niveau d'abstraction. La facette "Measure" acquiert des informations de l'agent GIO plus bas dans la hiérarchie des niveaux. La facette Abstraction délivre une information enrichie à un agent plus haut dans la hiérarchie. Le même processus est appliqué entre les facettes "Collection" et "Restitution". L'agent GIO se veut réutilisable. En conséquence, les contraintes propres à un assemblage particulier d'agents sont exprimées via des agents externes au modèle appelés "Control". On retrouve ici le concept d'agent PAC sans facettes A et P.

Alors que PAC et ALV explicitent les liens entre les perspectives fonctionnelles d'un agent via une facette spécialisée (le "Contrôle" de PAC et le "Link" d'ALV), GIO et MVC laissent ce problème en suspens. De fait, il est fréquent de voir des amendements pratiques de MVC en M^2VC dans lequel un "Model" sert d'intermédiaire explicite entre les trois autres facettes.

Le modèle conceptuel de Whizz s'appuie sur une métaphore musicale : il met en scène des danceurs, des instruments, des notes, des rythmes et des tempos. Les danceurs sont des agents qui produisent des effects perceptibles. Ils changent d'état en fonction des notes (ou informations) qu'ils entendent. Par exemple, une position, une couleur. Les instruments jouent des notes en fonction de leur état et de leur compétence. Ils jouent des notes sous le contrôle d'un tempo qui construit une suite d'instants d'intervalles réguliers. Le rythme détermine, dans cette suite, les dates pour lesquelles l'instrument doit jouer une note. Prenons un exemple.

Soit la représentation du concept de vitesse par un compteur à aiguille : la valeur entière qui modélise la vitesse (une valeur active) est reliée à un instrument lui-même connecté à un danceur. La valeur active produit des notes entières qui sont transformées en positions par

l'instrument. Ces positions sont des notes que le danceur écoute et traduit en représentation graphique. En résumé, le modèle conceptuel de Whizz consiste à peupler l'univers d'agents de rôle déterminé. Certains ont des rôles de présentation, d'autres de traducteurs, d'autres de producteurs d'informations conceptuelles. Le fonctionnement de l'ensemble est régi par le temps que modélisent les tempos et les rythmes. Remarquons que la composition de la valeur active de l'exemple précédent avec son instrument et son danceur définit les trois facettes d'un agent PAC. Mais le modèle de Whizz va plus loin que PAC et les autres avec une représentation explicite d'agents gestionnaires du temps.

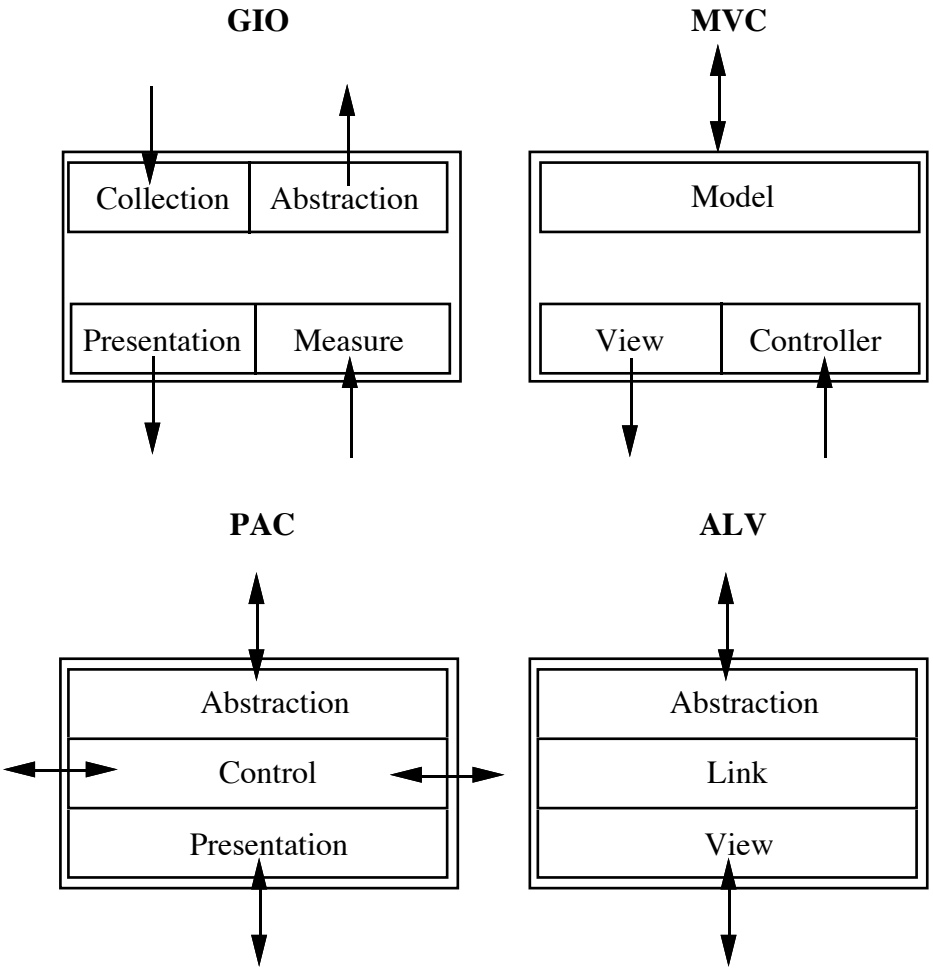


Figure 6.18 : Les facettes d'agents dans les modèles GIO, MVC, PAC, ALV.

Tous les modèles multi-agents que nous venons de référencer, répondent à des préoccupations communes : modularité, distinction explicite entre présentation et abstraction, encapsulation, parallélisme. PAC, ALV, MVC et le modèle de Whizz ont vu des mises en œuvres pratiques accompagnées, s'il le fallait, de mode d'emploi : les règles heuristiques de PAC, la boîte à outils Whizz, et pour MVC, une plate-forme d'accueil Smalltalk qui explicite la correspondance entre facette et classe d'objet. GIO se distingue du lot par un objectif plus

théorique : par une formalisation équivalente Lotos d'une composition d'agents GIO, il sera possible de prouver des propriétés sur le comportement de cette société. A ce titre, ces travaux rejoignent ceux de Harrison et Duke sur la "théorie des interacteurs" [Duke 93].

Après avoir présenté notre modèle de référence, nous montrons maintenant où le parallélisme des traitements et la fusion des informations en entrée, deux caractéristiques essentielles des systèmes à caractéristiques multiples, interviennent dans une telle architecture. VoicePaint, NoteBook et MATIS servent d'exemples pour étayer cette discussion.

Chapitre VII

PAC-Amodeus : intégration des langages et des dispositifs

"*Beaucoup de choses sur la terre à entendre et à voir,*
choses vivantes parmi nous !"
- Saint-John Perse -

1. INTRODUCTION

Dans le chapitre III sur l'espace Pipe-Lines, nous avons évoqué les phénomènes de parallélisme, de fusion et de fission. Au chapitre suivant, nous avons explicité avec la méthode UOM, les liens entre ces phénomènes, qui sont internes au système, et les usages observables des langages et des dispositifs. Par exemple, l'usage synergique de langages ou de dispositifs implique, pour le système, parallélisme, fusion et/ou fission.

Avec ce chapitre, nous voulons montrer comment le parallélisme et les opérations de fusion/fission s'inscrivent dans une architecture logicielle et comment PAC-Amodeus en permet et facilite la prise en compte. Une des retombées de cette analyse est une démonstration de l'apport et de l'adéquation de PAC-Amodeus à la conception logicielle des systèmes à caractéristiques multiples. Bien que nos résultats semblent généralisables aux interfaces de sortie, le parallélisme, la fusion et la fission ont été étudiés via PAC-Amodeus pour des interfaces d'entrée uniquement.

Nous abordons le parallélisme des traitements au paragraphe 2. Nous étudions ensuite la fusion des informations en entrée aux paragraphes 3 et 4 : nous identifions trois catégories de fusion puis nous proposons les stratégies, les critères et les représentations de données sous-jacentes au processus de fusion. Nous étayons la discussion par des systèmes existants. Après avoir posé les dimensions susceptibles d'orienter le processus de fusion, le paragraphe 5 présente notre moteur de fusion. Celui-ci est générique et constitue une plate-forme réutilisable pour la réalisation de systèmes qui nécessitent la fusion des informations en entrée. NoteBook et MATIS réalisé selon PAC-Amodeus serviront d'exemple pour étayer cette discussion.

2. PARALLÉLISME DANS PAC-AMODEUS

Dans PAC-Amodeus, des traitements parallèles sont possibles dans chacun des composants à tous les niveaux d'abstraction. Nous passons en revue chacune des possibilités.

2.1. Parallélisme dans le Composant Interaction de Bas Niveau

Dans le Composant Interaction de Bas Niveau (CIBN), les actions physiques utilisateur sont captées en parallèle par des processus d'acquisition[22]. Ces processus modélisent les

[22] En réalité, un dispositif d'entrée/sortie est géré par un pilote dédié du système d'exploitation. Sur une station monoprocesseur, ces pilotes sont activés en séquence suivant le mécanisme de traitement des interruptions. Il se

actions utilisateur sous forme d'événement dont le type caractérise à la fois la classe du dispositif utilisé et l'action appliquée au dispositif. Un événement porte une estampille, notée Ti, qui indique la date d'occurrence de l'action. La figure 7.1 illustre le parallélisme au niveau de l'acquisition des actions utilisateur dans le cas où chaque dispositif est géré par un processus dédié. MATIS offre un exemple concret : un processus NeXTStep produit des événements parole tandis qu'un autre processus produit les événements clavier et souris. MATIS dispose donc, au sens MSM, de deux canaux digitaux d'entrée.

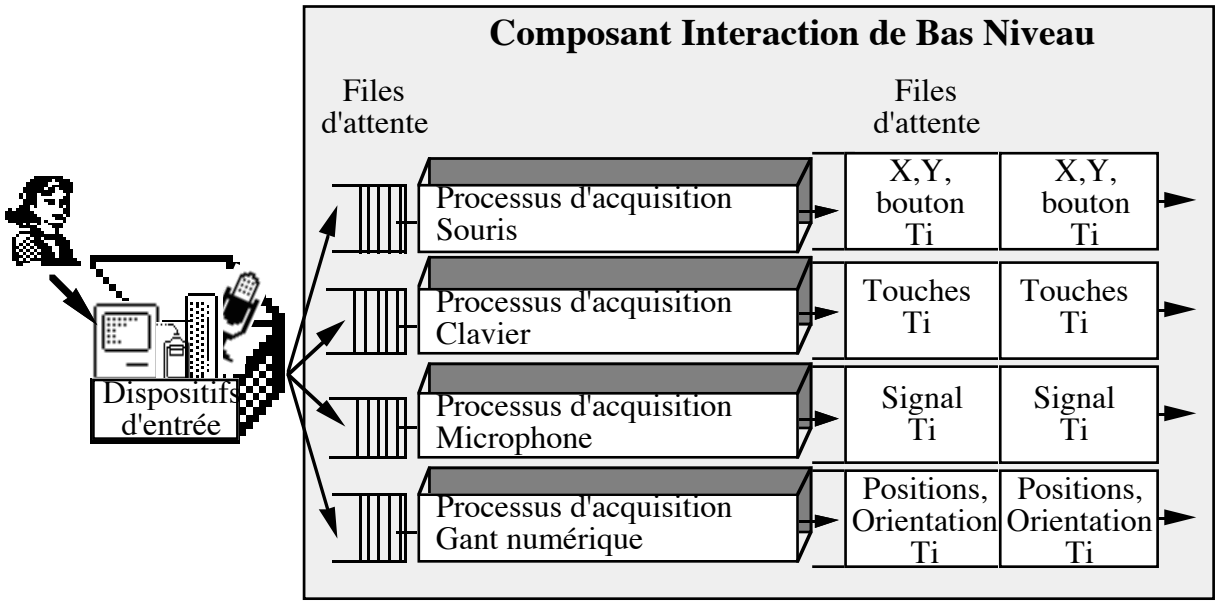


Figure 7.1 : CIBN, parallélisme des processus d'acquisition des actions physiques utilisateur.

Les événements produits par les processus d'acquisition sont ensuite abstraits par les processus d'abstraction pour former des objets d'interaction. Ces objets, on le rappelle, constituent la monnaie d'échange entre le CIBN et le Composant Techniques de Présentation. Les traitements d'abstraction, qui sont assurés par des processus dédiés, ont lieu en parallèle. La figure 7.2 en montre le principe. Considérons quelques exemples courants.

Le traitement d'un événement qui résulte de l'articulation d'une phrase, produit une suite de mots représentée sous forme d'une chaîne de caractères. Parallèlement, les événements issus

---

trouve que les plates-formes d'accueil peuvent placer plusieurs pilotes sous le contrôle d'un seul processus. Par exemple le serveur X, qui gère les "ressources physiques de l'interaction" [Coutaz 90], est un processus Unix lié entre autre aux pilotes du clavier et de la souris. Considérant ce niveau d'abstraction comme acquis sur toute plate-forme de développement, nous pouvons affirmer que les actions utilisateur sont captées en parallèle même s'il peut s'agir de pseudo-parallélisme.

de la saisie d'un mot au clavier sont abstraits pour former un mot sous forme d'une chaîne de caractères. De même, les événements issus du mouvement d'un gant numérique correspondent à la matrice des positions des doigts et de l'orientation. Ces événements sont abstraits en termes de poses définies au préalable. Nous dirons que les poses sont des exemples d'objet d'interaction.

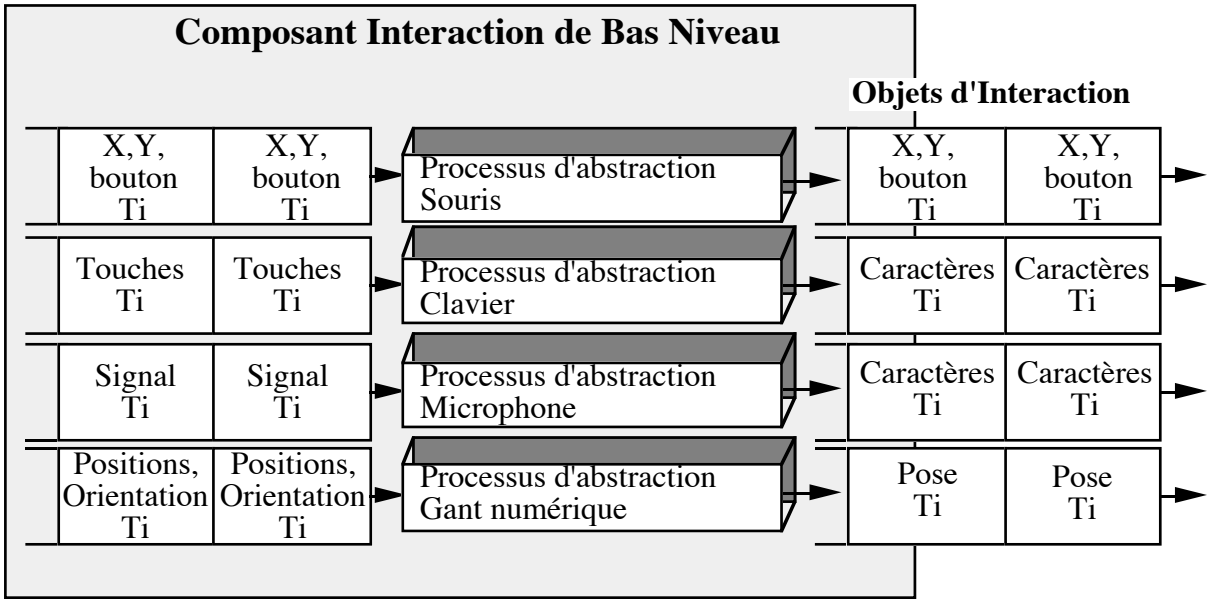


Figure 7.2 : CIBN, parallélisme des processus d'abstraction des actions physiques déjà acquises.

Les processus d'acquisition et d'abstraction du Composant d'Interaction de Bas Niveau sont indépendants du domaine d'application. Il faut entendre par "indépendance vis-à-vis du domaine", la réutilisabilté des traitements dont ces processus ont la charge. De manière générale, les traitements sont organisés autour d'un moteur paramétré par les données caractérisantes du domaine. Dans le cas de Sphinx, ces données sont déclarées dans un dictionnaire. Lorsque MATIS est actif, Sphinx doit disposer du dictionnaire de MATIS. De même, pour un gant numérique [Bordegoni 93], le développeur doit définir les poses spécifiques au système considéré. Définir le dictionnaire de mots reconnus par Sphinx ou décrire les poses pour le système VPL sont deux activités de même nature.

2.2. Parallélisme dans le Composant Techniques de Présentation

Le niveau Composant Techniques de Présentation (CTP) qui reçoit les objets d'interaction issus du CIBN, est le siège d'activités d'abstraction à raison d'une activité par langage d'interaction. La résultante de ces activités est une unité informationnelle ou objet de Présentation.

Comme exemples d'activités d'abstraction, un clic souris en provenance du CIBN est transformé dans le CTP en une sélection d'objet ; une chaîne de caractères issue du système de reconnaissance de la parole du CIBN est traduite en un arbre syntaxique. Par exemple, dans MATIS, le CTP est le lieu de l'analyse syntaxique des chaînes de caractères ou suites de mots que le reconnaisseur de parole du niveau CIBN a su identifier.

2.3. Parallélisme dans le Contrôleur de Dialogue

Le Contrôleur de Dialogue (CD) qui est organisé en agents est conceptuellement le siège d'activités parallèles. Chaque agent traite le contenu des objets de présentation que le CTP lui transmet. Le CTP distribue les objets de présentation aux parties Présentation des agents en fonction de l'intérêt que ces agents ont exprimé pour les classes possibles d'objet.

2.4. En résumé

En synthèse, PAC-Amodeus autorise la présence d'activités parallèles à tous les niveaux d'abstraction de son arche. Nous avons montré comment d'un composant de l'arche à l'autre les informations du plus bas niveau d'abstraction, c.-à-d. les événements, sont progressivement enrichis en informations de haut niveau d'abstraction, les unités informationnelles. Ce processus d'abstraction correspond à la fonction d'interprétation de Pipe-Lines.

Dans le sens inverse, on assiste à un processus de concrétisation (que MSM appelle fonction de restitution). Les activités parallèles se répartissent de manière similaire au processus d'abstraction dans tous les composants de l'arche. Dans le CD, deux agents peuvent produire en parallèle des objets de présentation. Il y a alors parallélisme dans la restitution des concepts que ces agents représentent. Le CTP peut aussi déterminer en parallèle les formes de l'expression de sortie de plusieurs objets de présentation ou spécifier simultanément plusieurs formes pour un même objet de présentation : la résultante est alors plusieurs objets d'interaction que le CIBN peut rendre perceptibles à l'utilisateur en pilotant simultanément plusieurs dispositifs physiques de sortie. Ainsi, de façon symétrique au processus d'abstraction, le parallélisme est présent dans le processus de concrétisation.

Le parallélisme des traitements est un concept introduit originellement par MSM et que nous avons repris à notre compte dans l'espace Pipe-Lines. Par sa localisation et ses effets dans PAC-Amodeus, nous en décrivons la réalisation logicielle. Nous concluons par l'établissement des liens entre le parallélisme des traitements dans les composants de PAC-Amodeus et la classification ULD :

• Des activités parallèles dans le CIBN permettent le parallélisme des actions physiques.

• Des activités parallèles dans le PTC permettent le parallélisme au niveau des langages d'interaction.

• Des activités parallèles dans le CD permettent le parallélisme au niveau des tâches.

En cela nous traduisons en termes de réalisation logicielle les liens déjà esquissés entre le parallélisme des traitements au sein de l'espace de conception Pipe-Lines et la classification ULD qui caractérise le système selon le point de vue de l'utilisateur. Nous schématisons cette relation triangulaire par la figure 7.3. La flèche en trait épais (3) correspond aux liens que nous venons de présenter.

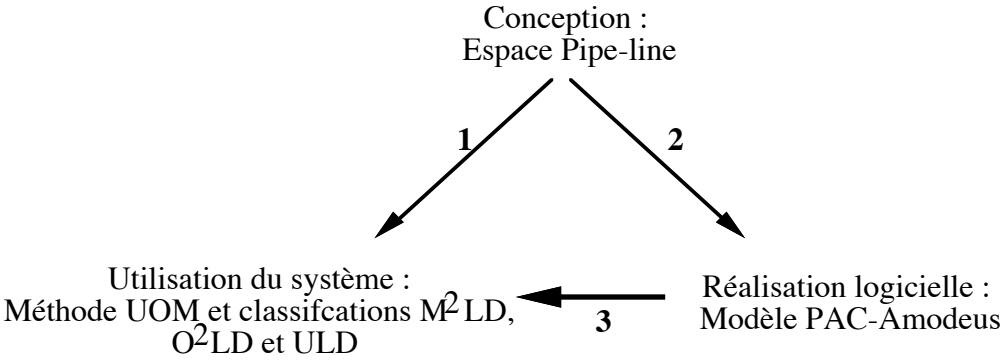


Figure 7.3 : Le triangle composé de l'espace Pipe-Lines, la méthode UOM et le modèle PAC-Amodeus.

3. FUSION DES INFORMATIONS DANS PAC-AMODEUS : TROIS CATÉGORIES

Le système transforme les objets d'interaction en objets de présentation, puis en objets conceptuels, enfin en objets du domaine et vice versa. Lors de chaque phase de transformation, des objets peuvent être combinés ; cette fusion se fait donc à différents niveaux. Au chapitre III, nous avons identifié trois catégories de fusion au sein de l'espace Pipe-Lines : lexicale, syntaxique et sémantique. La figure 7.4 les localise dans PAC-Amodeus.

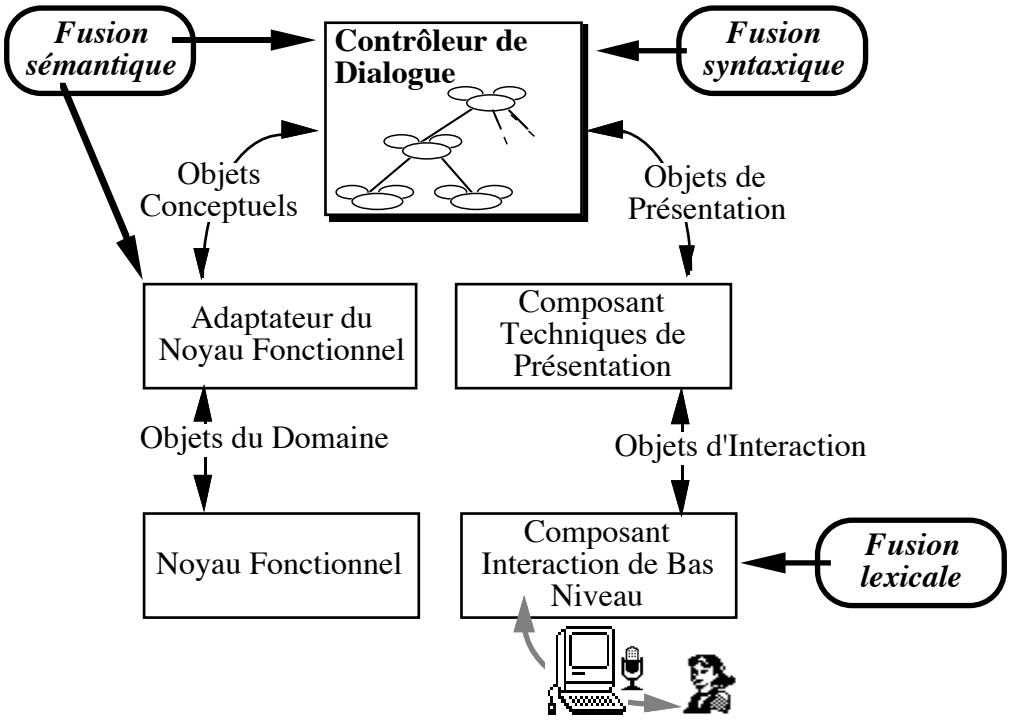


Figure 7.4 : Les fusions lexicale, syntaxique et sémantique dans PAC-Amodeus.

La *fusion lexicale*, réalisée dans le Composant Interaction de Bas Niveau, traduit une synchronisation entre des actions physiques. Par exemple, dans le Macintosh, l'enfoncement conjoint de la touche "shift" et d'un clic souris sont fusionnés par le système en un seul événement. Les informations acquises et fusionnées correspondent aux lexèmes du langage utilisé. Cette fusion est qualifiée de bas niveau puisque chaque information considérée isolément ne constitue pas un lexème.

L'opérateur de fusion de la théorie des dispositifs d'entrée de Mackinlay [Mackinlay 90] (Chapitre II) exprime de manière formelle la fusion lexicale lorsqu'il s'agit de relier des dispositifs physiques. Nous rappelons que dans cette théorie, un dispositif d'entrée est modélisé par un sextuplet qui contient, entre autre, le domaine des valeurs d'entrée, le domaine des valeurs de sortie et une fonction de résolution qui explicite la correspondance entre les deux domaines. Par application de l'opérateur de fusion, le domaine d'entrée du dispositif composé est le produit cartésien des domaines d'entrée des dispositifs reliés par cet opérateur. Deux ascenseurs, l'un vertical, l'autre horizontal, reliés par l'opérateur de fusion simulent les fonctions d'une souris. Le "<Shift> Clic" cité en exemple ci-dessus fournit aussi une illustration du phénomène de fusion lexicale.

Pour un système donné, les fusions lexicales sont en nombre fini et peuvent, par exemple, être décrites par une grammaire. Le langage UAN [Hartson 92], qui permet l'expression d'actions parallèles, convient à ce type de spécification. Ainsi, de même que l'on définit le dictionnaire des mots autorisés d'un langage naturel, de même la liste des fusions lexicales permises peut être formalisée. Certaines boîtes à outils intègrent ces services sous forme de modules prédéfinis [Olsen 87] .

La *fusion syntaxique* revient au Contrôleur de Dialogue. Elle consiste à combiner des unités informationnelles pour obtenir la description complète d'une commande comme "insérer une note" du système NoteBook. Une commande est le terme usuel qui désigne une procédure atomique que le système fournit à l'utilisateur pour accomplir une tâche élémentaire.

La *fusion sémantique* est effectuée dans le Contrôleur de Dialogue ou l'Adaptateur du Noyau Fonctionnel. Elle combine des commandes pour aboutir à une nouvelle fonction. Par exemple, dans VoicePaint un éditeur de dessin [Gourdol 91], deux commandes (dessiner une droite et changer l'épaisseur) sont combinées en une seule pour obtenir une droite à plusieurs épaisseurs.

Nous énonçons les liens entre la classification ULD et les trois catégories de fusion situées dans les composants de PAC-Amodeus :

- L'usage combiné des dispositifs physiques d'entrée (une dimension d'ULD) peut impliquer l'exécution de fusion lexicale dans le CIBN ou de fusions syntaxique ou sémantique dans le CD ou l'ANF.

- L'usage combiné des langages d'interaction par l'utilisateur (une dimension d'ULD) peut impliquer l'exécution de fusion syntaxique dans le CD ou de fusion sémantique dans le CD ou l'ANF.

- Un usage non combiné de dispositifs ou de langages peut cependant entraîner l'exécution de fusion sémantique.

Nous venons d'expliciter les lieux de réalisation des trois niveaux de fusion dans PAC-Amodeus et d'énoncé les liens avec les caractéristiques d'utilisation du système. Nous allons maintenant décrire la réalisation logicielle du processus de fusion des informations en nous consacrant tout particulièrement à la fusion syntaxique.

4. LES PROCESSUS DE FUSION : GRILLE D'ANALYSE

Notre analyse sur la fusion des informations en entrée s'appuie sur les dimensions que nous avons identifiées lors des journées de travail de l'atelier "interfaces multimodales" d'IHM'92 [IHM 92]. Cette liste de dimensions n'est pas exhaustive. Toutefois, elle nous a permis de clarifier le processus de fusion et de mettre en évidence les points communs, les divergences et les aspects complémentaires des solutions existantes. Définissant l'orientation du processus de fusion, ces dimensions sont ici appliquées et étendues au contexte de nos travaux établi par l'espace Pipe-Lines et la méthode UOM. Les voici : existence d'un langage dominant, stratégie d'intégration, critères d'intégration, stratégie d'exploration et technique de représentation. Nous développons chacun de ces points puis en offrons une illustration avec ICPplan avant de présenter une description complète de notre approche.

4.1. Existence d'un langage d'interaction d'entrée dominant

Un *langage d'interaction d'entrée est dominant* lorsque les expressions déclenchantes de l'interaction doivent être exprimées dans ce langage. On appelle expression déclenchante une expression bien formée du langage dont l'interprétation conduit à une unité informationnelle qui marque le début d'une nouvelle transaction. On constate l'assignation du langage aux unités informationnelles marqueurs de début de transaction. En général, le langage dominant est assigné à la dénotation des noms de commandes, les paramètres pouvant être spécifiés dans un autre langage.

Par exemple, dans le système 3DDE (3D Drawing Editor) [Bordegoni 93], un éditeur d'objets en trois dimensions, le langage gestuel du gant numérique est assigné à la désignation des commandes telle la commande de coloriage tandis que les paramètres de la commande, comme la couleur du pinceau, sont spécifiés dans un autre langage au moyen des boutons d'une space-ball. Il est clair que l'existence d'un langage dominant réduit les choix offerts à l'utilisateur (classification O^2LD) par assignation du langage à des classes prédéfinies d'unités informationnelles.

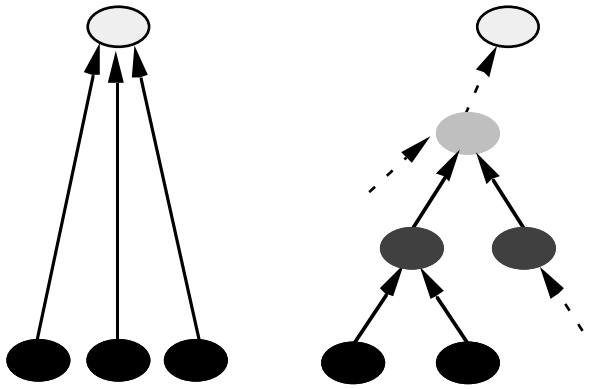
Il est intéressant d'observer que même si le système, tel MATIS, n'impose pas de langage dominant, l'utilisateur a tendance à se fixer un langage dominant et cette fixation varie d'un sujet à l'autre. Ce résultat expérimental, certes préliminaire, nous vient d'observations faites par Magicien d'Oz [Mignot 93]. Dans cette expérimentation, les sujets devaient meubler un appartement représenté sous forme d'un plan d'architecte. Pour cela, ils disposaient de deux langages d'interaction, le langage naturel oral et la manipulation directe par souris. Les sujets, en phase d'exploration utilisent l'un ou l'autre des langages mais très vite opérationalisent leurs tâches dans un langage donné.

Du point de vue mise en œuvre, la notion de langage dominant doit pouvoir être utilement exploitée comme pivot d'intégration. En général, un pivot spécialisé accélère les calculs. On pourrait ainsi espérer améliorer les performances du processus de fusion. MMI2 (voir chapitre IV paragraphe 2.4) utilise une structure pivot intitulée CMR (Common Meaning Representation). Une structure CMR sert de véhicule du sens des expressions d'entrée, quel que soit le langage utilisé. Nous constatons que la structure d'une CMR obéit aux besoins de l'analyse du langage naturel. Elle contient par exemple la valeur illocutoire (question, affirmation, etc.) de la phrase en langage naturel.

4.2. Stratégie d'intégration

La stratégie d'intégration précise la politique du mécanisme de fusion : celle-ci peut-être *précoce* ou *différée*. Si la politique adoptée est précoce, le mécanisme de fusion ne se met jamais en attente d'informations, et par là risque d'effectuer des fusions incorrectes. Inversement, une politique différée induit des mises en attente du mécanisme dans le but d'obtenir des informations complémentaires avant d'effectuer une fusion. Une stratégie précoce offre les moyens de produire les retours d'information réactifs de la manipulation directe qui ne relèvent pas de la sémantique du domaine. A l'inverse de la stratégie retardée, une stratégie précoce peut être amenée à défaire des fusions.

Orthogonalement à la politique qui est précoce ou différée, la stratégie adoptée peut conduire à une intégration *progressive* procédant par étapes, ou bien unique. La figure 7.5 illustre les deux formes d'intégration. L'intégration unique fait intervenir deux niveaux : l'avant et l'après fusion. Inversement, l'intégration progressive fait intervenir plusieurs niveaux.



Intégration unique Intégration progressive

Figure 7.5 : Stratégie d'intégration, intégration unique ou progressive.

4.3. Critères d'intégration

Un critère d'intégration définit une condition de déclenchement de fusion. Nous avons identifié les critères suivants :

• La *proximité temporelle* sert à mettre en correspondance des informations issues de langages et dispositifs différents mais produits en des instants très proches. La définition d'une fenêtre temporelle permet de détecter des proximités temporelles. La fenêtre temporelle est aussi utilisée pour délimiter l'attente d'informations complémentaires dans le cas de stratégie différée.

• La *complémentarité logique ou structurelle* des informations permet dans certains cas de les fusionner même si la condition de proximité temporelle n'est pas respectée.

• La *complétude* d'une structure d'intégration peut constituer une condition de passage entre niveaux d'abstraction. Cette complétude peut servir également à limiter l'attente d'informations pour la stratégie différée.

• Les informations contenues dans le contexte interviennent dans la résolution des coréférences, anaphores et ellipses. Les informations contenues par le contexte sont présentées au paragraphe 4 du chapitre III.

• L'assignation de langages à des classes d'unités informationnelles (voir paragraphe 6.2 du chapitre IV), évite des tentatives de fusion d'unités informationnelles issues de langages ne pouvant être utilisés pour l'unité informationnelle considérée.

4.4. Stratégie d'exploration

Le processus de fusion élabore un espace de possibilités de relations entre les informations à traiter. Cet espace est parcouru soit *en largeur d'abord soit* en *profondeur d'abord*:

• En largeur d'abord, il s'agit d'identifier les relations possibles entre les informations à un niveau donné et de choisir la plus prometteuse avant de passer au niveau suivant (et ceci de manière répétitive).

• En profondeur d'abord, le processus de fusion exploite la première relation (souvent temporelle) et l'enrichit à travers les niveaux en fonction des relations structurelles et sémantiques ; c'est seulement au niveau sémantique que sont validées ou réfutées les interprétations. Les retours arrière dans l'espace des possibilités se font progressivement à travers les niveaux.

4.5. Technique de représentation

Nous reprenons à notre compte, la réflexion du groupe IHM'92 : "La technique de représentation précise la nature des structures de données employées comme support aux informations fusionnées. Le formalisme de représentation peut être unique (quel que soit le niveau d'abstraction) ou multiple (adapté au niveau d'abstraction). Et les structures de données peuvent être dynamiques ou statiques." [IHM 92] Dans notre terminologie, la technique de représentation détermine le format des unités informationnelles.

4.6. Illustration

Dans le rapport [IHM 92], plusieurs expériences et solutions menées par les membres de l'atelier sont présentées :

• le processus de fusion au niveau conceptuel de F. Azémard [Azémard 93],

• le processus de fusion basé sur un modèle connexionniste de J.C. Martin [Martin 92],

• et les processus de fusion des systèmes LIMSI-DRAW [Bellik 92], Modeleur Déclaratif [Vigouroux 92], ICPplan [Bourguet 92] et MATIS .

Chacun de ces exemples illustre les dimensions de la grille d'analyse élaborée lors de l'atelier.

Avant de décrire notre processus de fusion, nous présentons l'approche d'ICPplan. Nous rappelons que dans nos exercices de classification d'ICPplan selon la méthode UOM (voir chapitre IV), l'interface en entrée d'ICPplan :

• est multilangage, multidispositif (M^2LD),
• est optionnel pour le langage et parfois optionnel pour le dispositif (O^2LD),
• offre un usage synergique des dispositifs et des langages (ULD).

Nous nous intéressons maintenant au classement d'ICPplan dans la grille d'analyse du processus de fusion. La figure 7.6 en synthétise les résultats :

- Dans ICPplan, toutes les commandes de dessin s'expriment aussi bien en langage naturel qu'en langage de commande. Il y a équivalence totale entre les deux langages : ICPplan n'a pas de langage dominant.

- La stratégie de fusion opère en une seule étape à haut niveau d'abstraction et gère une seule structure d'intégration à la fois (qui correspond, on s'en doute, à la commande en cours de spécification). Ainsi deux commandes indépendantes ne peuvent être définies en parallèle ou de manière entrelacée. Le processus se met en attente d'informations pour compléter la commande en cours. Dès lors, la stratégie est différée.

- Les critères d'intégration sont la proximité temporelle et la complémentarité des informations par rapport à la commande en cours de spécification. La proximité temporelle est un critère prioritaire sur la complémentarité. Ainsi deux informations appartenant à la même fenêtre temporelle sont fusionnées même si les informations sont contradictoires : de cette fusion impossible découle un cas d'erreur géré par le dialogue.

- Le processus de fusion n'élabore qu'une seule solution de relation entre les informations. La stratégie d'exploration n'est donc pas en largeur. De plus aucun retour arrière (cas de défusion) n'est possible, le dialogue gérant les cas d'erreurs ou d'incompatibilité. Aussi la stratégie d'exploration n'est pas non plus en profondeur.

- Le formalisme sur lequel se fonde le processus de fusion est unique. Il est déduit de la description des commandes du système.

Langage dominant	*Stratégie d'intégration*	*Critères d'intégration*	*Stratégie d'exploration*	*Techniques de représentation*
Non	Différée Unique	Proximité temporelle Complémentarité structurelle	Pas en largeur Pas en profondeur	Fondée sur la tâche (Structure d'une commande)

Figure 7.6 : Caractéristiques du processus de fusion du système ICPplan.

Nous présentons maintenant notre moteur de fusion. Nous avons veillé à le rendre générique, c'est-à-dire réutilisable ou instanciable par la technique du paramétrage : les éléments qui dépendent du système font l'objet de déclarations externes au moteur. Cette approche permet d'envisager son intégration dans un squelette d'application pour systèmes à caractéristiques multiples.

5. NOTRE MOTEUR DE FUSION GÉNÉRIQUE

Comme pour l'analyse d'ICPplan, nous allons décrire notre moteur de fusion selon les dimensions de la grille de la figure 7.6. Nous présentons ensuite notre technique de représentation et le mécanisme que nous illustrons au moyen de MATIS. En conclusion, nous évoquerons des améliorations qu'il conviendrait d'apporter à la solution actuelle.

5.1. Présentation selon notre grille d'analyse

5.1.1. Langage dominant

Le processus de fusion ne présuppose pas l'existence d'un langage dominant : la technique de représentation que nous présentons plus loin en 5.1.4, est indépendante des langages d'interaction du système. Cette approche facilite la mise en œuvre de relations d'équivalence entre langages d'interaction comme c'est le cas pour MATIS entre la manipulation directe utilisable pour la spécification des requêtes et le langage naturel.

5.1.2. Stratégie d'intégration et d'exploration

La stratégie du moteur de fusion est précoce ("*eager*"). Le moteur ne se met jamais en attente d'une information complémentaire. L'avantage, nous l'avons vu, est la possibilité de produire des retours d'informations lexicaux et syntaxiques immédiats. L'inconvénient est le risque d'avoir à défaire des fusions incorrectes. Cette situation peut se présenter lorsque les actions physiques ne sont pas traitées dans l'ordre chronologique d'occurrence. Cet ordre est remis en cause par les différences de temps d'acquisition et de traitement des actions dans les composants de bas niveau de l'arche : le CIBN et le CTP.

Le moteur élabore une solution par fusions successives : sa stratégie est donc progressive et précoce. Il opère en profondeur d'abord avec retour arrière en cas de fusion incorrecte. Contrairement au processus de fusion d'ICPplan, le moteur autorise la construction de plusieurs solutions en parallèle. Si deux informations ne sont pas fusionnées, elles deviennent des candidates potentielles à la fusion avec des informations à venir. Cette approche permet de gérer le cas de l'usage concurrent (au sens de ULD) de plusieurs langages.

La figure 7.7 présente de manière synthétique la stratégie d'intégration et d'exploration de notre moteur.

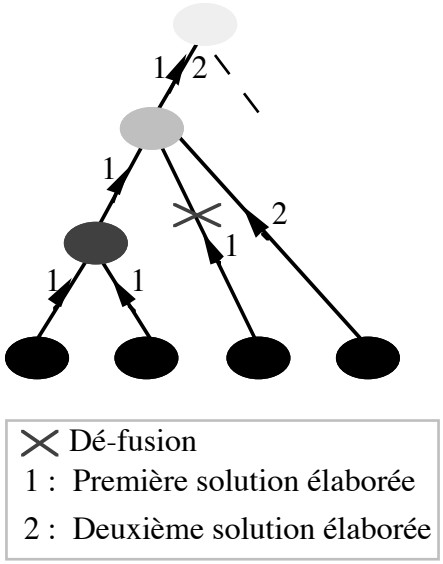


Figure 7.7 : Stratégie d'intégration précoce et progressive, opérant en profondeur.

5.1.3. Critères d'intégration : trois niveaux de fusion

La fusion est effectuée sur la base de trois critères : la structure des objets à combiner, le temps, le contexte. A chacun de ces critères, correspond un traitement spécifique de fusion que nous appelons respectivement *microfusion*, *macrofusion* et *fusion contextuelle*. En complément, il convient de considérer l'anticritère suivant : si les structures logiques des unités informationnelles sont incompatibles, leur fusion est impossible. La complémentarité structurelle est donc une condition nécessaire mais non suffisante à la fusion que le critère soit micro, macro ou contextuel. Nous présentons ces trois conditions de fusion par ordre de priorité décroissante :

- Le *critère de microfusion* s'appuie sur le temps et combine des informations, structurellement complémentaires ayant des intervalles de temps qui s'entrelacent (parallélisme ou pseudo-parallélisme) ;

- le *critère de macrofusion* s'appuie sur le temps et combine des informations structurellement complémentaires qui appartiennent à une fenêtre temporelle (épaisseur du présent ou proximité temporelle) ;

- le *critère de fusion contextuelle* utilise uniquement le contexte (au sens Pipe-Lines) et fusionne des informations structurellement complémentaires. Dans le système MATIS par exemple, la requête la plus récente en cours de construction est une information maintenue par le contexte et est utlisée par le moteur comme critère d'intégration.

Il est important de noter que les critères de fusion sont indépendants du domaine de l'application.

5.1.4. Technique de représentation : le creuset

Les fusions syntaxique et sémantique s'appuient sur une représentation uniforme : le creuset. (Les fusions lexicales, qui relèvent du Composant d'Interaction de Bas Niveau, ne sont pas considérées dans notre moteur.) Le creuset sert de représentation commune aux unités informationnelles émises par le Composant de Présentation et aux unités informationnelles qui résultent du processus de fusion. Ainsi de la fusion de deux creusets, résulte un creuset.

Comme le montre la figure 7.8, un *creuset* est un objet structuré à deux dimensions. L'un des axes définit les composants structurels des unités uniformationnelles que l'Adaptateur du Noyau Fonctionnel sait reconnaître. Par exemple, chaque constituant de la signature d'une commande occupe un emplacement sur l'axe vertical "composants structurels" du creuset.

L'axe des temps identifie l'ordre d'occurrence des composants structurels, c'est-à-dire l'ordre d'intégration dans le creuset des unités informationnelles correspondant à ces composants. Cet ordre dépend de la date de production des unités informationnelles par le Composant de Présentation. La présence de plusieurs valeurs pour un même composant peut révéler une redondance ou des incohérences. Le creuset doit être rempli à l'intérieur d'une fenêtre temporelle (appelée aussi "épaisseur du présent"), sinon il est abandonné.

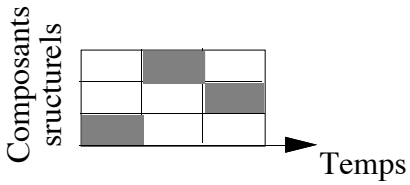


Figure 7.8 : L'objet creuset comme structure de représentation commune pour la fusion.

La stratégie que nous avons adoptée nécessite, par sa "précocité", la capacité défusionner. Le creuset, formalisme unique de représentation, facilite cette activité. La figure 7.9 illustre le principe d'une défusion dont la condition s'appuie sur les relations temporelles des composants structurels. Dans l'étape 1, deux creusets viennent d'être combinés (partie gauche de la figure) alors qu'un troisième survient (partie droite de la figure). Ce dernier contient un composant structurel plus récent que son correspondant dans le creuset résultat de la fusion. Une condition de défusion est donc satisfaite et les traitements de l'étape 2 sont

entrepris. Dans l'étape 3, on constate que le creuset dernier venu occupe la place qui lui revient dans le creuset fusionné et le creuset intégré à tort devient libre. Les creusets au sommet de la figure 7.9 sont deux candidats possibles à la prochaine fusion. Nous reviendrons sur les opérations de dé-fusion au paragraphe 5 consacré à la description détaillée du fonctionnement du moteur de fusion.

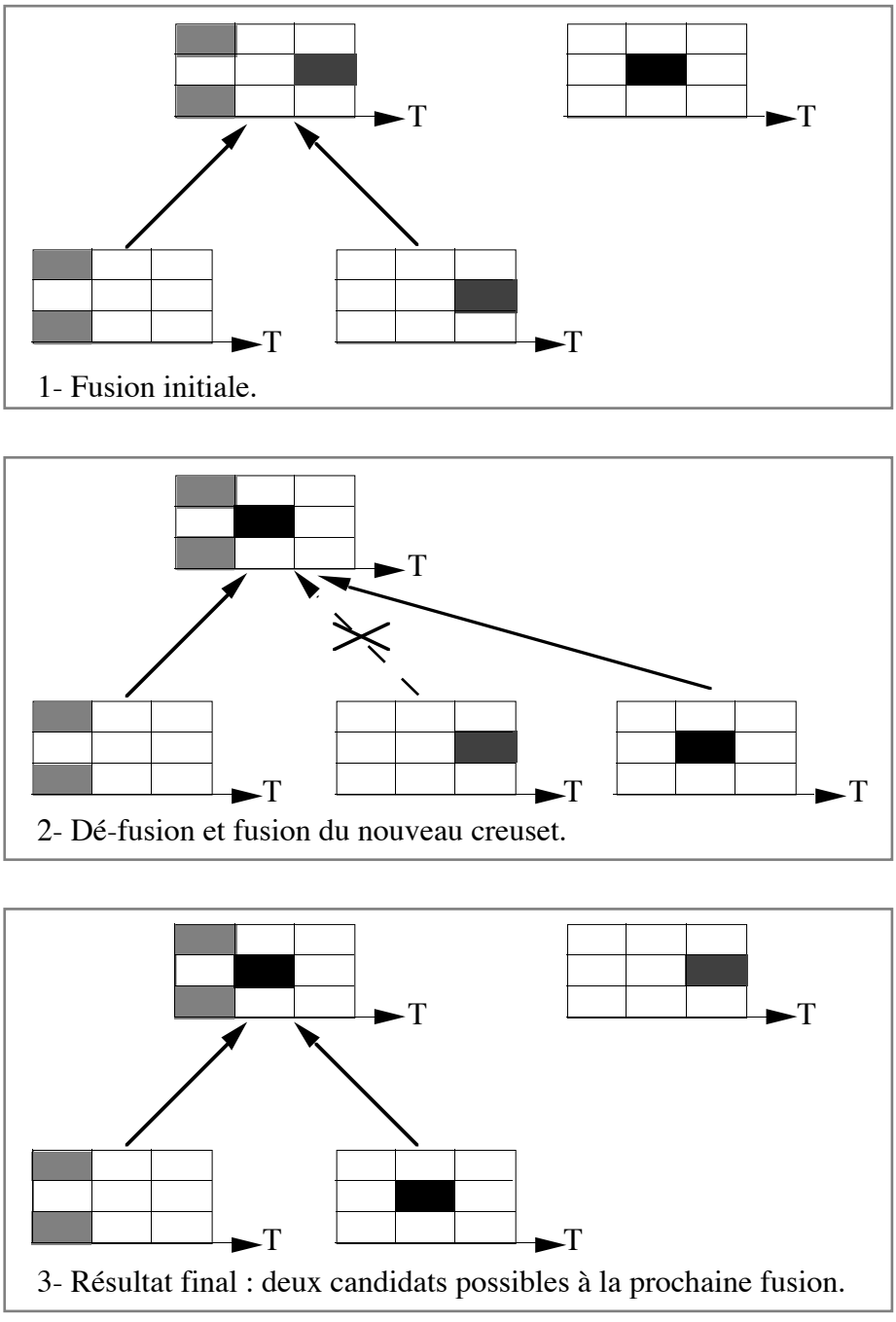


Figure 7.9 : Fusion et dé-fusion de creusets.

Le moteur de fusion n'accorde aucune sémantique à une case du creuset. Son fonctionnement vise un objectif systématique fondé sur une définition structurelle des unités

informationnelles, à savoir : combler les cases vides des creusets candidats à la fusion en s'appuyant sur les critères micro, macro, contextuels énoncés ci-dessus. Par exemple, pour MATIS, la case numérotée 1 correspond à la ville de départ. Dans une autre application, comme ICPplan, la case 1 permettrait de stocker l'identité d'un objet graphique (mur, porte par exemple). Ainsi la représentation sous forme de creuset est indépendante de la sémantique du domaine mais requiert une description préalable des unités informationnelles que manipule l'Adaptateur de Noyau Fonctionnel.

5.2. Ancrage du moteur dans PAC-Amodeus

Les fusions syntaxiques et sémantiques reviennent au Contrôleur du Dialogue (CD). Elles s'appuient sur les agents PAC qui le composent. L'organisation non séquentielle du CD autorise des fusions en parallèle. Son organisation hiérarchique permet une fusion incrémentale et contextuelle. La discussion qui suit et l'exemple de la commande "Mets ça là", présenté au paragraphe 5.4.1, montrent l'intérêt d'une hiérarchie d'agents.

La figure 7.10 désigne le point d'ancrage du moteur de fusion dans PAC-Amodeus. On constate que la partie Contrôle de chaque agent fait appel au moteur. Chaque agent a la possibilité d'ajouter des unités informationnelles issues de son état interne qu'il souhaite soumettre au moteur comme candidat à la fusion. Dans une implémentation à objet selon laquelle un agent serait réalisé par un exemplaire d'objet, le moteur serait une méthode héritée par toutes les classes d'agent. Alors, le mécanisme de fusion serait commun mais exécuté de manière répartie par une société d'agents.

Une sous-hiérarchie d'agents PAC correspond, on l'a vu, à une tâche ou à une grappe de tâches. Dans notre analyse du mécanisme de fusion, une structure hiérarchique permet d'organiser l'espace des creusets en sous-ensembles ou contextes. Ce procédé permet d'accroître l'efficacité du moteur de fusion : deux creusets qui n'appartiennent pas au même ensemble ne peuvent être fusionnés. Les creusets de deux ensembles distincts peuvent avoir ou non un nombre distinct de composants structurels.

Par leur Présentation, chaque agent feuille de la hiérarchie, reçoit du Composant Techniques de Présentation (CTP) les creusets pour lesquels ils ont exprimé leur intérêt. Chaque case d'un creuset peut être déclarée intéressante par un ou plusieurs agents. Un creuset émis par le CTP peut être complet ou incomplet. L'agent traite localement l'information reçue : même si le creuset est incomplet, il peut cependant spécifier un retour d'information qui est alors partiel.

Si le creuset reçu par un agent feuille est incomplet ou s'il est complet mais intéresse l'agent père, l'agent feuille le transmet au père. Ce dernier fait appel au moteur de fusion en lui fournissant comme paramètre le creuset nouvellement reçu. Si aucune fusion n'a été effectuée, le creuset incomplet est renvoyé à l'agent. Mais si le creuset est fusionné, le creuset résultant est envoyé à l'agent le plus bas dans la hiérarchie et père des agents responsables des cases remplies du creuset.

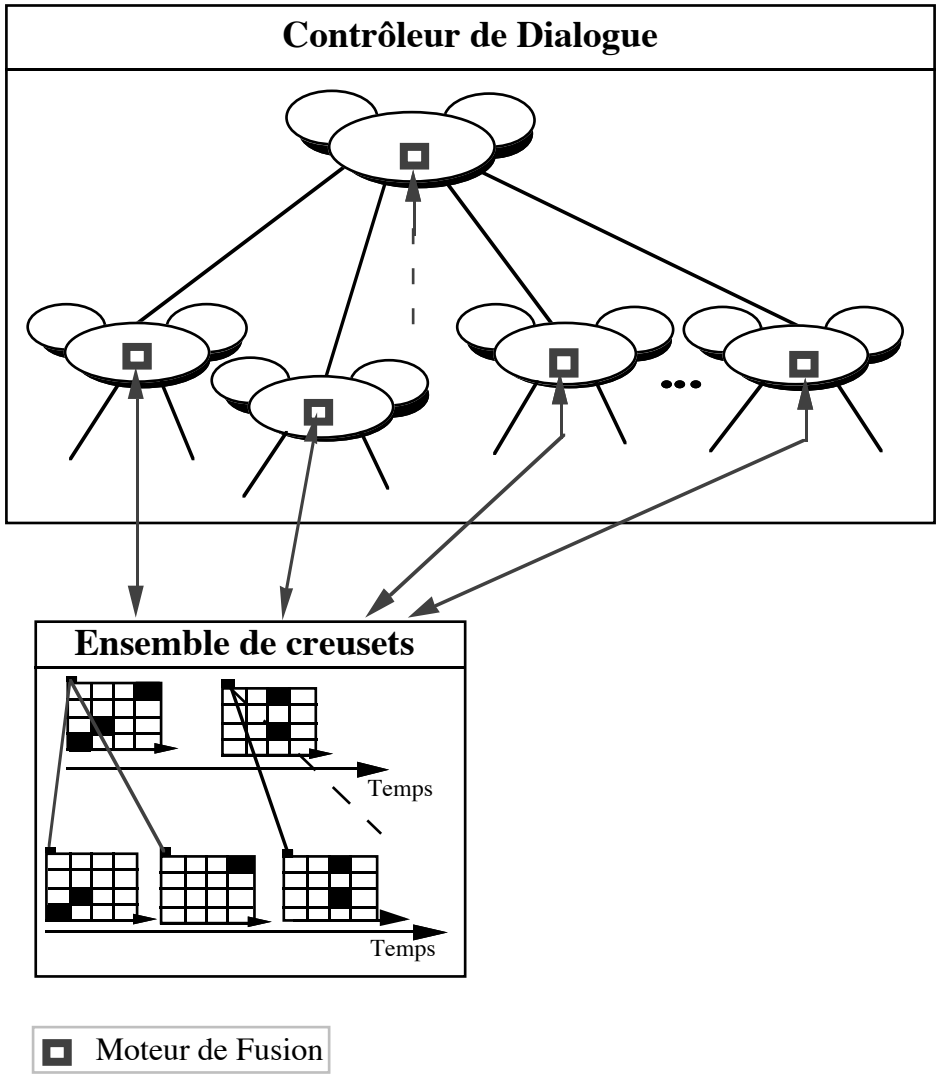


Figure 7.10 : Point d'attache du moteur de fusion au sein de PAC-Amodeus.

La détermination de l'agent père se fait par la fonction PlusBasCommunPère(creuset). Le moteur de fusion maintient associé à un creuset un agent considéré comme responsable du creuset. Ainsi dans le cas de défusion, l'agent responsable est averti et gère alors la répercussion à ses agents fils. (Par définition l'agent responsable est le père de tous les agents concernés par la défusion.) La figure 7.11 montre l'agent responsable de chaque creuset dans le cas d'une fusion en deux étapes.

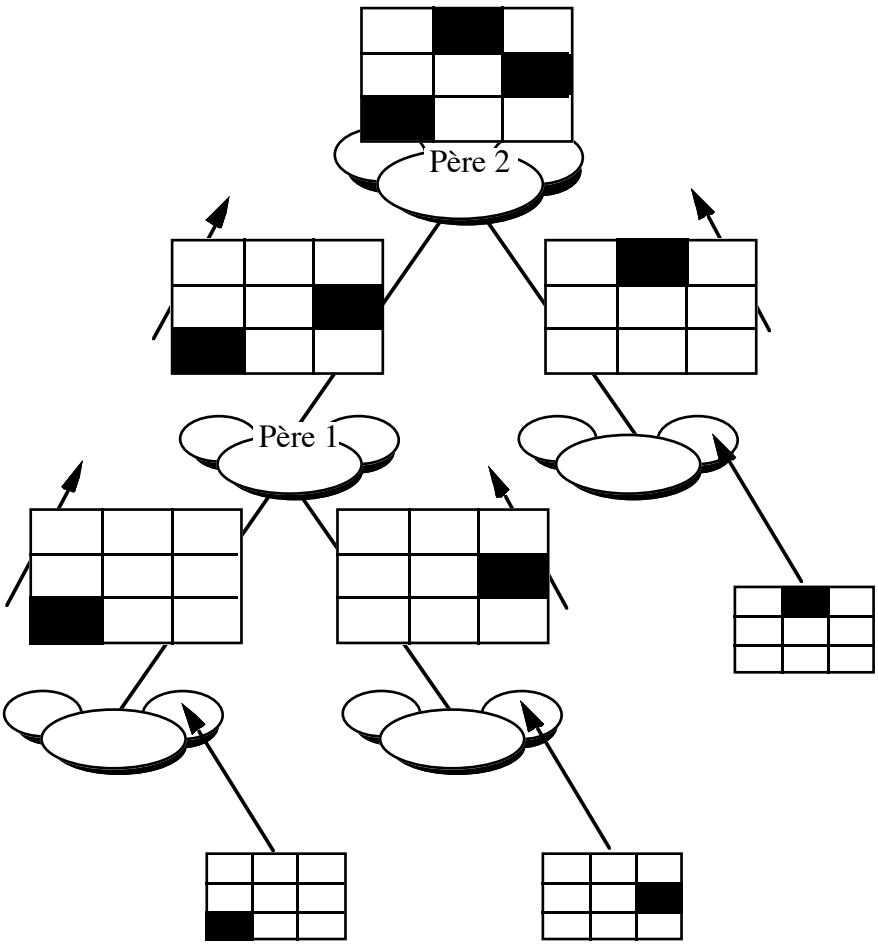


Figure 7.11 : Fusion en deux étapes au sein de la hiérarchie d'agents.

5.3. Le mécanisme de fusion et ses règles

Le mécanisme de fusion repose sur un ensemble de métriques. L'une des conséquences de la fusion est le retrait de creusets devenus inutiles. Une stratégie précoce nécessite des retours arrière. Ces points font l'objet des trois paragraphes suivants. Nous décrivons ensuite la suite de traitements entrepris à l'arrivée d'un creuset. Nous terminons cette description par un exemple illustratif.

5.3.1 Les métriques d'un creuset

Chaque case d'un creuset est estampillée par une date. Cette date est héritée de l'événement qui est à l'origine de l'unité informationnelle rangée dans cette case. Soient :

- $c_i = (p1, p2, ..., p_j, ..., p_n)$: le creuset i constitué de n composants structurels.
- $info_{ij}$: l'unité informationnelle rangée dans le composant j du creuset i.
- $Tinfo_{ij}$: la date de l'unité informationnelle rangée dans le composant j du creuset i.
- $Tmax_i$: la date de la case la plus récente de creuset c_i.
- $Tmin_i$: la date de la case la plus ancienne du creuset c_i.
- Fen_temp_i : la fenêtre temporelle du creuset c_i.
- Tempo : la temporisation du creuset c_i déclenchée à sa complétude.

Les dates, la fenêtre temporelle et l'espérance de vie d'un creuset se définissent ainsi :

Dates:

$\forall$ i,j tel que $c_i = (p1, p2 , ..., pj, ..., pn)$ alors

$Tmax_i = Max (Tinfo_{ij})$

$Tmin_i = Min (Tinfo_{ij})$

Fenêtre temporelle:

$\forall$ i,j tel que $c_i = (p1, p2 , ..., pj, ..., pn)$ alors

$Fen\_temp_i = [Tmin_i - \Delta t , Tmax_i + \Delta t]$

Espérance de vie d'un creuset une fois complet :

$\forall$ i,j tel que $c_i = (p1, p2 , ..., pj, ..., pn)$ alors

$Esp_i = T_{maxi} + tempo = Max (Tinfo_{ij}) + Tempo = Max (Tinfo_{ij}) + \Delta t$

La figure 7.12 représente graphiquement les différentes métriques d'un creuset.

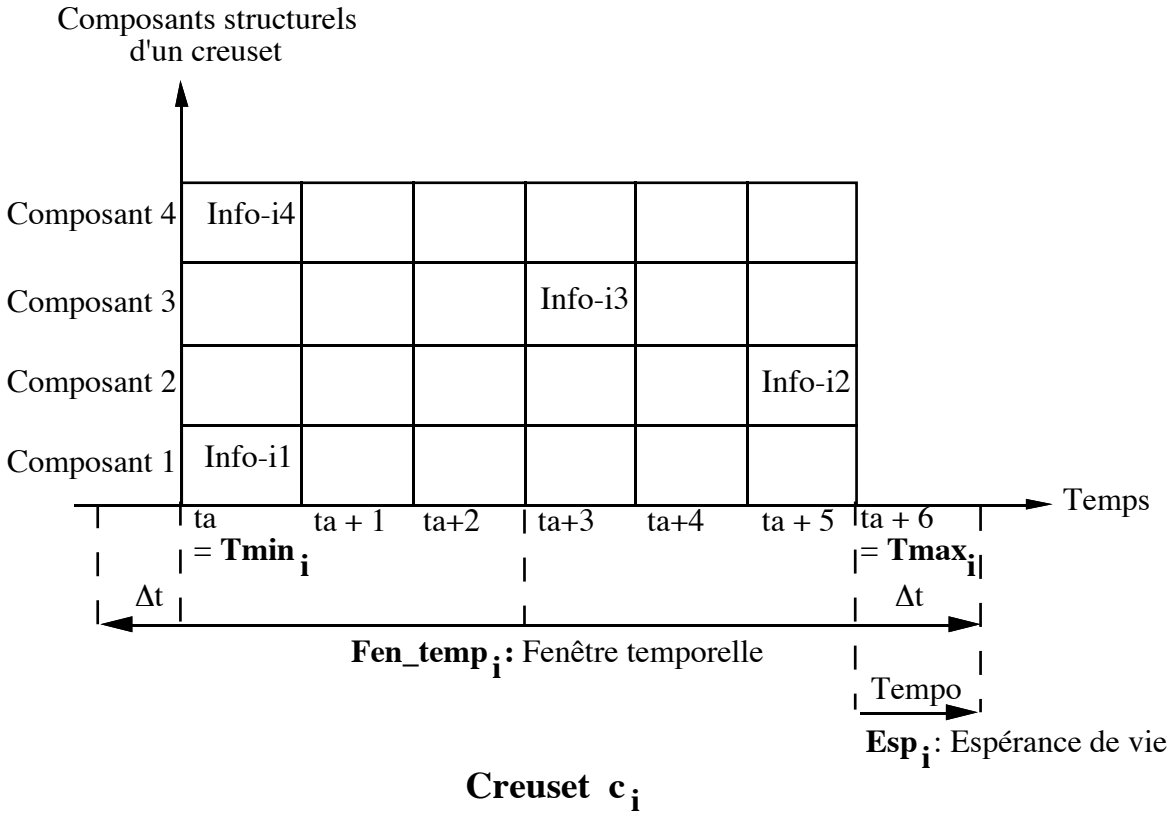


Figure 7.12 : Les métriques d'un creuset.

5.3.2 Le retrait d'un creuset de la liste des candidats à la fusion

Le retrait d'un creuset de la liste des candidats à la fusion intervient à la conjonction de deux conditions distinctes : le creuset est complet (toutes ses cases sont remplies) et le creuset a consommé son espérance de vie (l'heure actuelle est $Tmax_i+\Delta t$ où Δt est une valeur de temporisation déclenchée dès que le creuset est complet). En fin de temporisation, c'est-à-dire à l'heure $Tmax_i+\Delta t$, les fusions effectuées au sein du creuset ne peuvent plus être remises en cause.

Règle 1

Conditions de retrait d'un creuset dans la liste des candidats à la fusion

Un creuset $c_i = (p_1, p_2, ..., p_j, ..., p_n)$ est supprimé de la liste si :

• il est complet : $\forall p_j \in c_i, \exists info_{ij}$

• et sa temporisation est arrivée à échéance : l'heure actuelle = Esp_i

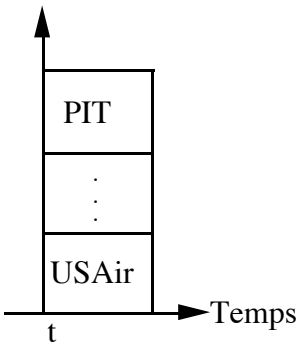
5.3.3 La défusion

Tout creuset défusionné devient un creuset susceptible d'être refusionné. Si plusieurs creusets ont été défusionnés, le moteur tente de les fusionner du plus ancien au plus récent en s'appuyabt sur $Tmin_i$. Le moteur s'arrête lorsqu'il n'a plus de creuset à fusionner. L'arrivée d'un nouveau creuset c_i peut provoquer la défusion d'un creuset $c_{i'}$. Le moteur tente alors de le fusionner avec les creusets candidats. La fusion de $c_{i'}$ peut à son tour provoquer la défusion d'un autre creuset $c_{i''}$ et ainsi de suite.

5.3.4 A l'arrivée d'un nouveau creuset

Nous rappelons que la stratégie adoptée est précoce. Si la fusion est impossible, les unités informationnelles sont traitées indépendamment et en parallèle. Plusieurs creusets sont alors candidats à la fusion.

Lorsqu'un creuset est nouvellement ajouté à la liste des candidats, la première étape du moteur consiste à le combiner en appliquant le critère de *microfusion*. La microfusion considère la date des colonnes d'un creuset. Une colonne correspond au contenu structurel d'une unité informationnelle. Par exemple dans MATIS l'énoncé de la phrase "*USAir flights from Pittsburgh*" à l'instant t, conduit à la création du creuset suivant :



Le moteur recherche si des colonnes complémentaires ont des dates qui sont très proches à Δmicrot près ($\Delta microt << \Delta t$) de la date du nouveau creuset. Dans le cas positif, les informations sont fusionnées, entraînant parfois une défusion. La microfusion garantit que des informations très proches dans le temps soient combinées. Nous schématisons la microfusion aux figures 7.13 et 7.14.

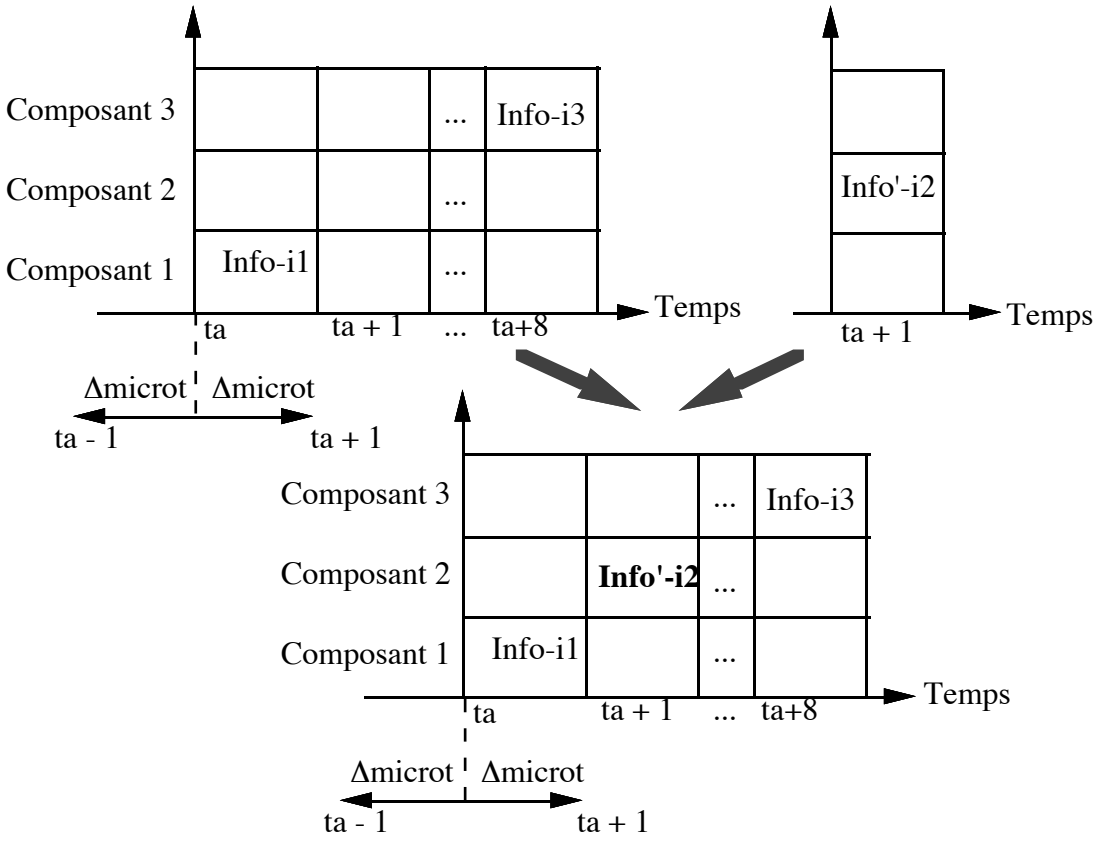


Figure 7.13 : Microfusion sans dé-fusion. (Δmicrot = 1 unité)

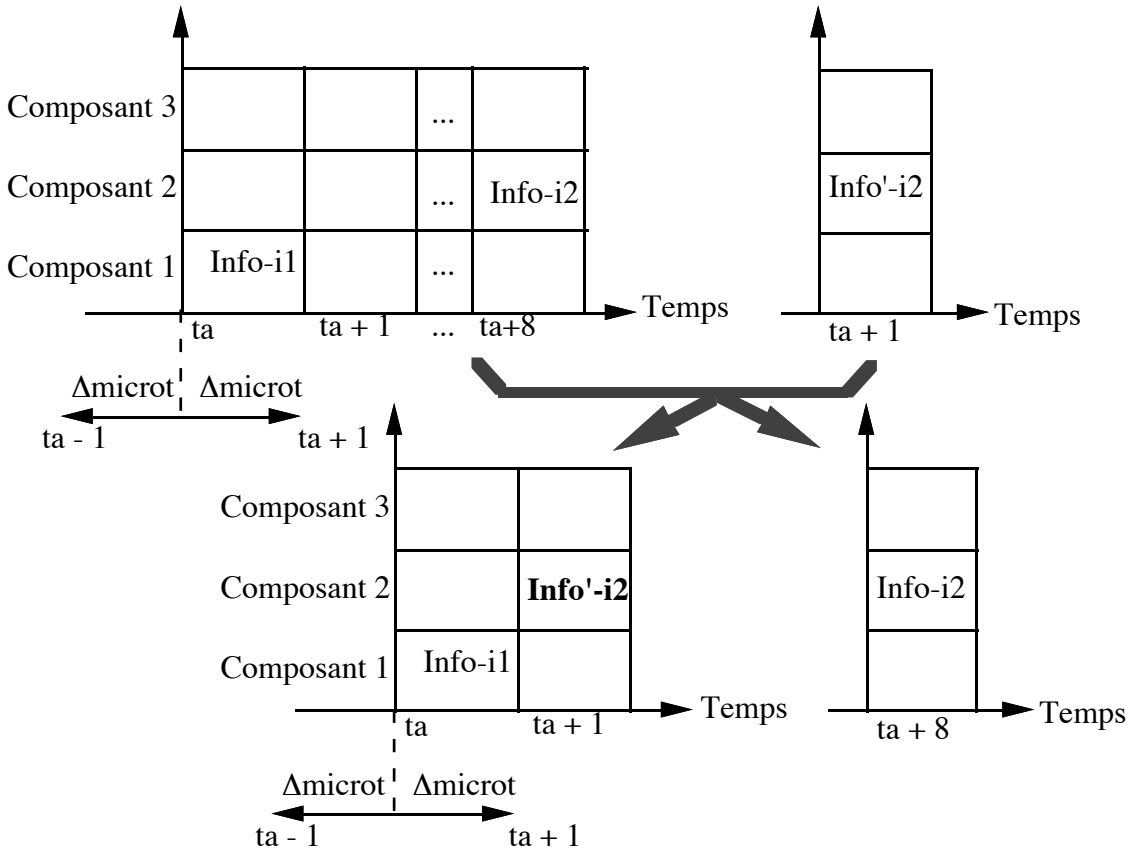


Figure 7.14 : Microfusion avec dé-fusion. (Δmicrot = 1 unité de temps)

Règle 2

La microfusion

Soient :

- col_{it} = (p1, p2, ..., pj, ..., pn) :
  une colonne à l'instant t du creuset c_i de composants structurels p1, p2, ..., pj, ..., pn.
- $col_{jt'}$ = (p'1, p'2, ..., p'j, ..., p'n)
  une colonne à l'instant t' du creuset c_j de composants structurels sont
  p'1, p'2, ..., p'j, ..., p'n.
- $i \neq j$

col_{it} et $col_{jt'}$ sont combinées :

- si elles sont complémentaires :
  complémentaire (col_{it} , $col_{jt'}$) est défini par :
  $\forall k \in [1..n] : \exists info_{ik} \wedge \neg (\exists info_{jk})$

- si leurs dates sont très proches à Δmicrot près :
  proche (col_t, $col_{t'}$) est défini par :
  $t' \in [t-\Delta microt, t+\Delta microt]$

La redondance de dispositifs et de langages (paragraphe 6.3 du chapitre IV) est analysée en pré-traitement de la microfusion. Avant de décider s'il y a lieu de pratiquer une microfusion, le moteur vérifie que les informations ne sont pas redondantes. Si c'est le cas le nouveau creuset est ignoré. Dans MATIS, l'énoncé "*Flights to Boston*" et la saisie concurrente et équivalente de "Boston" dans le champ de la requête est un cas de redondance (voir figure 7.15).

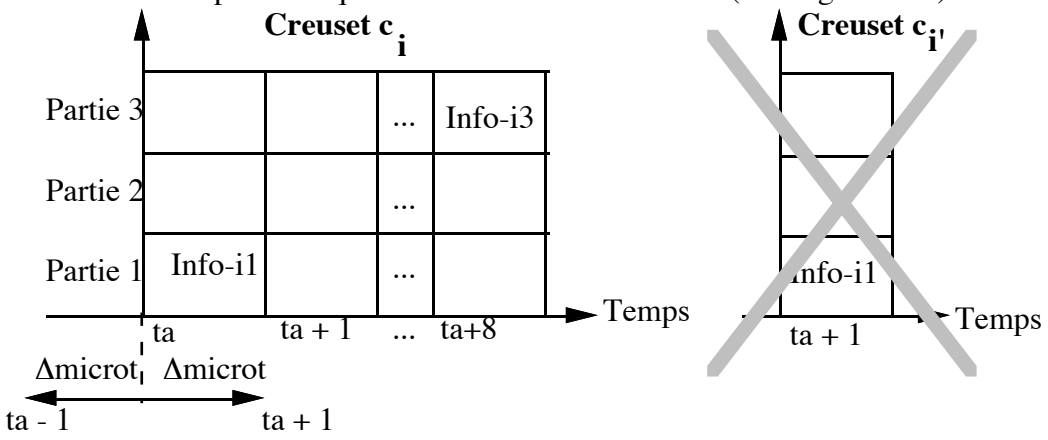


Figure 7.15 : Redondance.

Dans cet exemple, la saisie de "Boston" (Info-i1) est reçue en premier par le moteur (temps de traitement inférieur à celui de la parole) qui fusionne l'information dans un creuset c_i. La phrase "Flights *from Boston*" est classée comme redondante et le creuset $c_{i'}$ qui lui correspond est ignoré.

Règle 3

Redondance détectée avant la microfusion

Soient

• col_{it} = (p1, p2, ..., pj, ..., pn) :
une colonne à l'instant t du creuset c_i de composants structurels p1, p2, ..., pj, ..., pn.

• $col_{jt'}$ = (p'1, p'2, ..., p'j, ..., p'n)
une colonne à l'instant t' du creuset c_j de composants structurels sont
p'1, p'2, ..., p'j, .., p'n.

• $i \neq j$

col_{it} et $col_{jt'}$ sont déclarées redondantes :

• si elles contiennent les mêmes informations pour les mêmes composants structurels :
redondant (col_t, $col_{t'}$) est défini par :
$\forall k \in [1..n] : \exists info_{ik} \wedge \exists info_{jk} \wedge info_{ik} = info_{jk}$
$\wedge$
$\forall k' \in [1..n] : \neg (\exists info_{ik'}) \wedge \neg (\exists info_{jk'})$

• si leurs dates sont très proches à Δmicrot près :
proche (col_t, $col_{t'}$) est défini par :
$t' \in [t-\Delta microt, t+\Delta microt]$

En cas d'échec de la microfusion, le moteur tente d'appliquer le critère de macrofusion. Cette étape consiste à fusionner un creuset c_i dont la fenêtre temporelle englobe la date du nouveau creuset c_{nouv} à condition que les deux creusets soient complémentaires. La figure 7.16 illustre la macrofusion.

La macrofusion peut aussi entraîner la défusion. La complémentarité de deux creusets ne s'appuie pas uniquement sur le contenu des cases mais aussi sur leurs dates : si le nouveau creuset c_{nouv} est plus ancien que les cases correspondantes dans le creuset c_i alors il y a défusion du creuset c_i et fusion de creuset c_{nouv} avec le creuset c_i. Cette règle permet de respecter l'ordre de spécification des informations, ordre remis en cause par les différences de

temps d'acquisition et de traitement des composants de bas niveau (le CINB et le CTP). Nous illustrons ce cas par la figure 7.17.

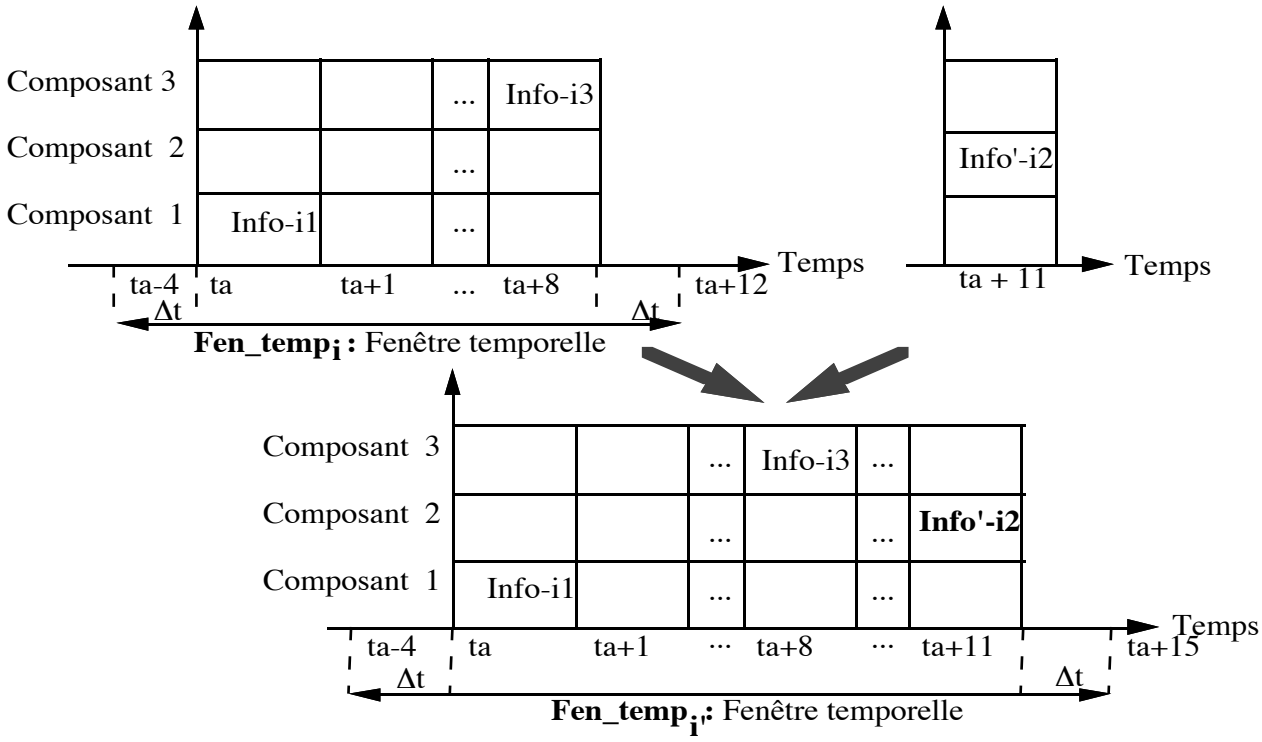


Figure 7.16 : Macrofusion sans défusion.

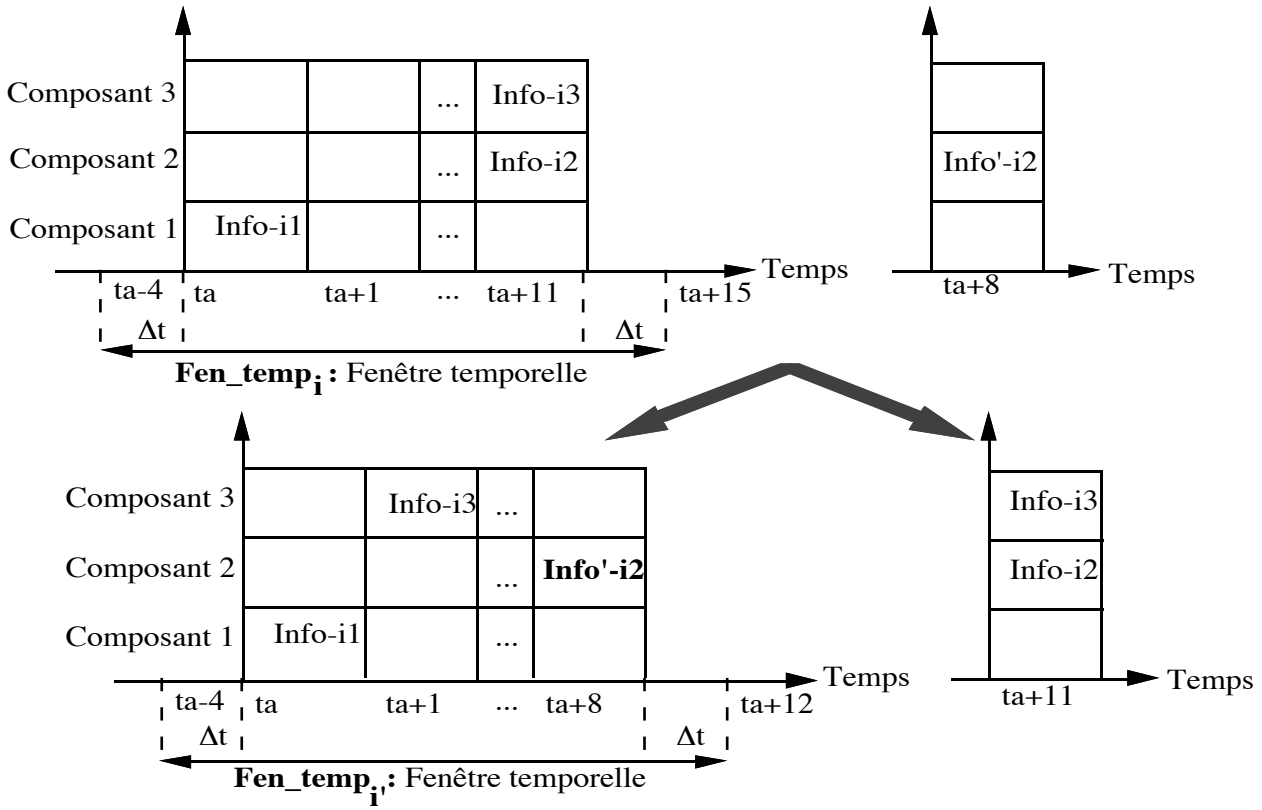


Figure 7.17 : Macrofusion avec défusion.

Règle 4

La macrofusion

Soient :

• $c_i = (p_1, p_2, ..., p_j, ..., p_n)$: le creuset i constitué de n composants structurels $p_1, p_2, ..., p_j, ..., p_n$.

• $c_{nouvt} = (p'1, p'2, ..., p'j, ..., p'n)$
le nouveau creuset arrivé à l'instant t constitué de n composants structurels $p'1, p'2, ..., p'j, ..., p'n$

• $i \neq nouvt$

c_i et c_{nouvt} sont combinés :

• s'ils sont complémentaires :
creusetComplémentaire (c_i, c_{nouvt}) est défini par :

$\forall k \in [1..n] : \exists info_{ik} \wedge \neg (\exists info_{nouvtk})$
$\wedge$
$\forall j \in [1..n] : (\exists info_{ij} \wedge \exists info_{nouvj}) \wedge (Tp'j=t < Tpj)$

• et si la date de c_{nouvt} appartient à la fenêtre temporelle de c_i :
$t \in Fen\_temp_i$

La règle suivante, qui concerne l'ordre de parcours de la liste des creusets candidats, s'applique aussi bien à la micro qu'à la macro fusion. Comme le montre la figure 7.18, le moteur procède du plus ancien au plus récent.

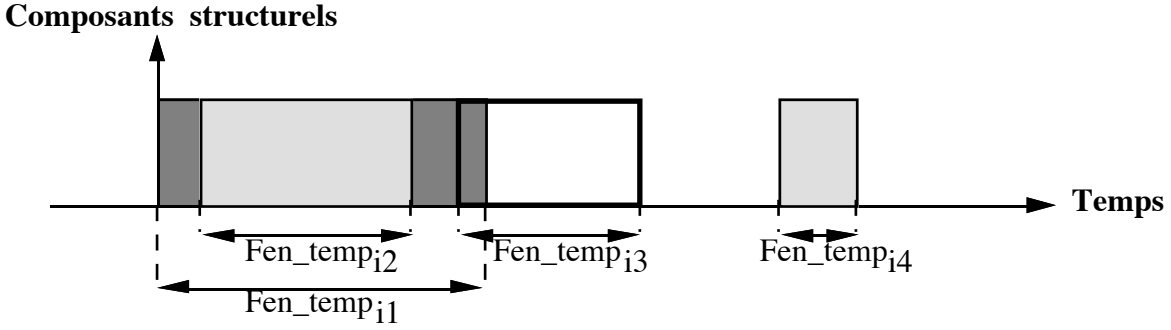


Figure 7.18 : Ordre de parcours des creusets.

Règle 5

Ordre de parcours des creusets

L'ordre de parcours des creusets candidats à la *macrofusion* et *microfusion* va du plus ancien au plus récent, en se référant à la borne inférieure de la fenêtre temporelle de chaque creuset ($Tmin_i - \Delta t$).

La dernière étape du moteur de fusion en cas d'échec des deux étapes précédentes (microfusion et macrofusion) est la fusion contextuelle. Celle-ci ne prend pas en compte le temps mais tente de fusionner le nouveau creuset avec celui qui correspond au contexte courant, ou focus courant de l'utilisateur. Il s'agit du creuset le plus récent. La fusion contextuelle autorise des pauses dans l'interaction. Par exemple l'utilisateur de MATIS peut commencer à spécifier une requête, s'arrêter de travailler avec MATIS et finir la requête après un délai non borné.

Règle 6

Fusion contextuelle

Soient :

- $c_{contexte} = (p_1, p_2, ..., p_j, ..., p_n)$: le creuset tel que
  $\forall i\ ci \neq c_{contexte} : Tmax_i < Tmax_{contexte}$
- $c_{nouvt} = (p'1, p'2, ..., p'j, ..., p'n)$
  le nouveau creuset arrivé à l'instant t constitué de n composants structurels
  $p'1, p'2, ..., p'j, ..., p'n$
- $i \neq nouvt$

$c_{contexte}$ et c_{nouvt} sont combinés :

- s'ils sont complémentaires :
  complémentaire ($c_{contexte}, c_{nouvt}$) est défini par :
  $\forall k \in [1..n] : \exists\ info_{ik}\ \Lambda\ \neg\ (\exists\ info_{contextek})$

Après ces trois étapes, si le nouveau creuset n'est pas fusionné, il devient le contexte courant.

5.3.5 Commandes au moteur de fusion

Par expérimentation, nous avons identifié trois commandes qu'un agent PAC peut envoyer au moteur de fusion pour en influencer le comportement.

La première commande permet à un agent PAC de retirer de la liste un creuset incomplet et par conséquent avant la date d'expiration du creuset (Espérance de vie). Dans MATIS, l'envoi d'une requête à la base de données provoque la suppression du creuset de la liste des candidats à la fusion.

La deuxième est la possibilité de modifier explicitement le creuset contexte courant. Il y a alors modification de l'ordre des creusets. La commande de modification est envoyée par l'agent responsable du creuset. Par exemple lorsque l'utilisateur dit "Je souhaite compléter ma commande initiale avec ...", le changement de tâche ou de contexte est spécifié par l'utilisateur et doit être répercuté au niveau du moteur de fusion. Dans MATIS, la sélection d'une ancienne fenêtre de requête incomplète provoque un changement de contexte répercuté au moteur de fusion.

La troisième commande identifiée permet de forcer la fusion de deux creusets. En cas d'incompatibilité structurelle de ces derniers, ce sont les parties du creuset le plus récent qui sont prises en compte. Cette commande est utile à chaque fois que l'utilisateur souhaite annuler des informations déjà fusionnées. Le moteur force alors la fusion bien que structurellement les creusets soient incompatibles. Cette fusion forcée consiste alors à remplacer des informations déjà fusionnées. Par exemple, l'utilisateur en train de dessiner dit "Non, ce n'est pas rouge mais vert". Dans MATIS, le cas se présente lorsque l'utilisateur modifie une requête incomplète par saisie directe dans le formulaire. Des informations déjà fusionnées doivent être remplacées par les nouvelles informations saisies.

5.4. Deux exemples

Le premier exemple illustre l'apport du support que constitue la hiérarchie d'agents tandis que le deuxième issu de MATIS montre les différents niveaux de fusion.

5.4.1. Le paradigme "Mets ça là"

L'exemple suivant montre le processus de fusion syntaxique au sein du Contrôleur de Dialogue et l'apport de la hiérarchie d'agents. L'utilisateur d'un éditeur de dessins dispose de

deux canevas. Il dit “mets ça là” et, en parallèle, désigne avec la souris un objet du premier canevas puis un point du deuxième canevas. La figure 7.19 montre l’architecture d’agents PAC intervenant dans cet exemple. Les agents *dessin1* et *dessin2* modélisent respectivement les canevas source et destination. L’agent *palette* correspond au panneau des outils graphiques associé au premier canevas. L’agent palette du deuxième canevas n’est pas représenté dans la figure. Les agents de type *éditeur* servent à cimenter des actions réparties sur plusieurs agents en une unité d’abstraction plus riche. La *racine* supervise l’ensemble et se trouve prête à communiquer avec le Noyau Fonctionnel ou son représentant, l’Adaptateur du Noyau Fonctionnel.

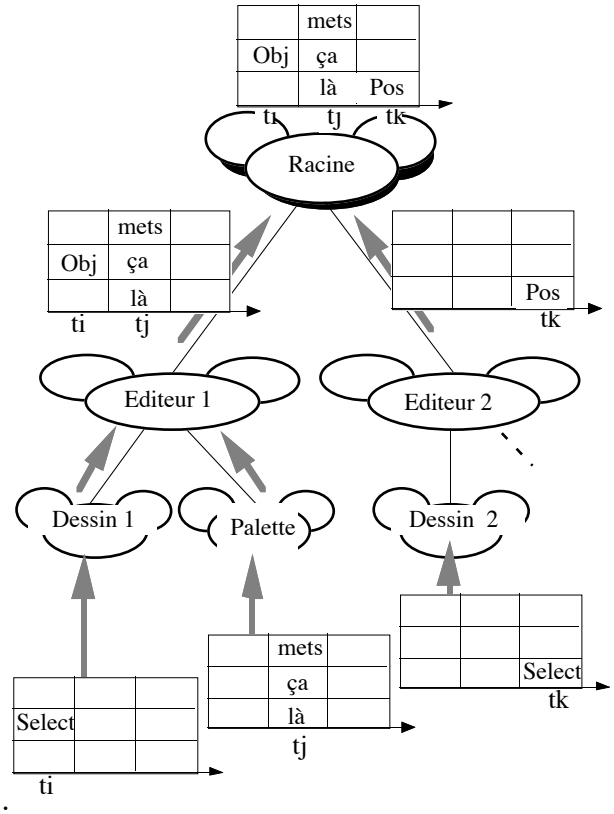


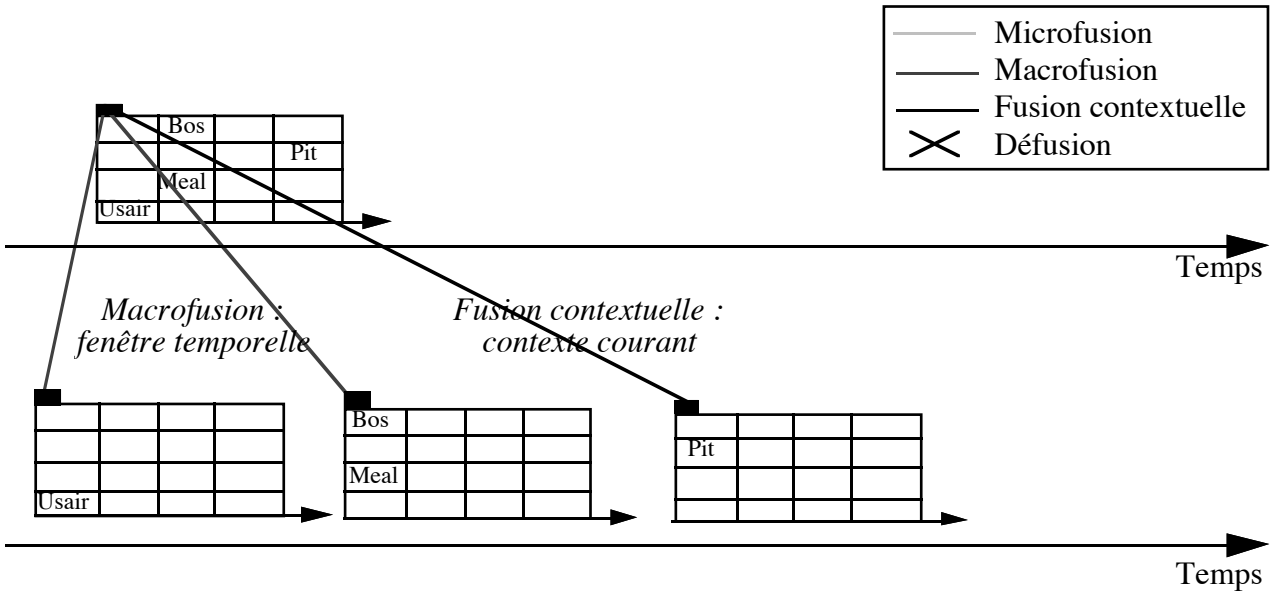
Figure 7.19 : “mets ça là”, un exemple de fusion à différents niveaux au moyen d’une hiérarchie d’agents PAC du Contrôleur de Dialogue.

Dans notre exemple, les agents de dessin et la palette reçoivent chacun un creuset estampillé en provenance du Composant Techniques de Présentation. L'agent Dessin1 transforme la sélection en l’identification de l’objet sélectionné, définit un retour d'information partiel et fait appel au moteur de fusion en lui fournissant le nouveau creuset. L'agent Palette fait de même : l'appel au moteur par l'agent Palette provoque la fusion des deux creusets. Le creuset résultant de la fusion est envoyé à l'agent Editeur 1, père des deux agents concernés, qui peut produire à son tour un retour d'information. L'agent Dessin 2 recevant un nouveau creuset transforme la sélection en l’identification d'une position, produit un retour d'information partiel et fait appel au moteur de fusion. Le moteur va alors fusionner ce nouveau creuset et fournir le

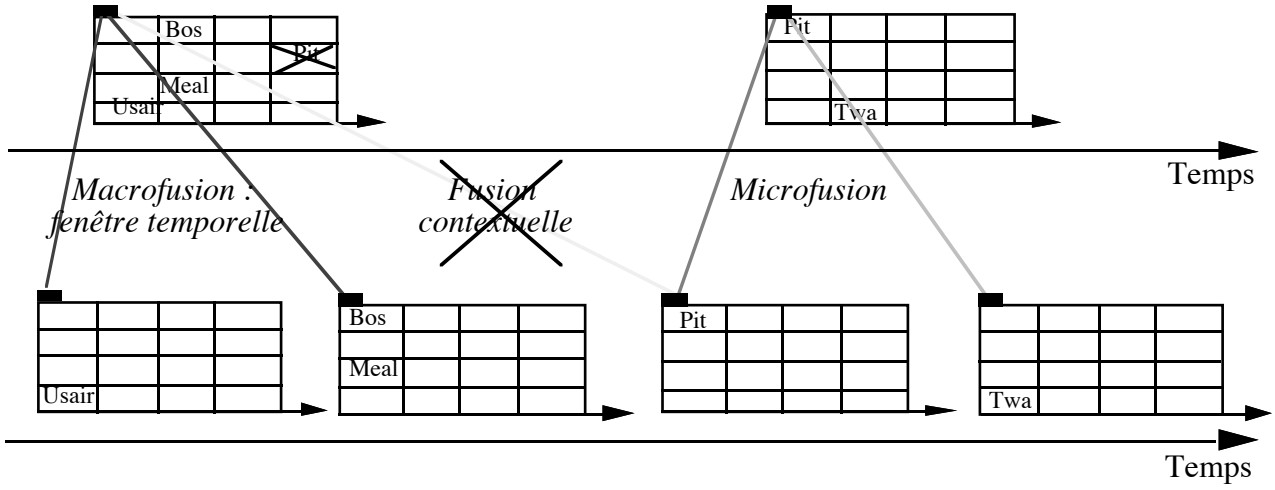
creuset résultant à l'agent Racine, qui est le père commun de tous les agents concernés par le creuset ; la commande est alors complète et peut être exécutée par l'agent Racine.

5.4.2. Un exemple d'énoncé issu de MATIS

Pour illustrer les différents types de fusion, nous avons choisi un exemple d'énoncés accepté par MATIS. La figure 7.20 présente les étapes de fusion et de défusion de cet exemple.



-1- Première étape : macrofusion puis fusion contextuelle.



-2- Deuxième étape : dé-fusion et microfusion.

Figure 7.20 : Illustration du moteur de fusion avec MATIS.

L'utilisateur sélectionne avec la souris "*USAir*" (liste dans une palette) et prononce ensuite la phrase "*Flights from Boston serving a meal*". Ces deux informations sont fusionnées

puisqu'elles appartiennent à une même fenêtre temporelle (macrofusion). L'utilisateur prononce ensuite la phrase "*TWA flights from*" tout en sélectionnant la ville "*Pittsburgh*". La sélection est traitée en première (son temps de traitement étant inférieur à celui de la parole). La sélection "*Pittsburgh*" est donc fusionnée avec la requête courante (fusion contextuelle). Lorsque l'information parole "*TWA flights from*" est disponible, il est alors nécessaire de défusionner la sélection "*Pittsburgh*", qui peut alors être fusionnée avec l'acte de parole émis en parallèle de la sélection (microfusion).

Dans notre exemple, des agents distincts reçoivent les informations à fusionner à chaque étape. Ces agents fournissent avant même que la commande soit complète des retours d'informations spécifiques au niveau d'abstraction. Par exemple, l'agent qui reçoit la sélection *Pittsburgh* produit un retour d'information immédiat traduisant la sélection en grisant la case sélectionnée. Comme le montre la figure 7.20, chaque information est rangée dans les cases structurelles idoines d'un creuset. Le creuset résultant de la macrofusion et de la fusion contextuelle de la figure 7.20 cas -1- est reçu par un agent, père des agents ayant reçu les creusets contenant l'information originelle. Lors de la deuxième étape, figure 7.20 cas -2-, la fusion contextuelle est défaite et le creuset contenant l'information *Pittsburgh* est fusionné avec le nouveau creuset.

5.5. Conclusion et extensions envisagées

Nous souhaitons souligner le fait que notre moteur de fusion est générique : il ne dépend pas d'un système particulier. Ses critères de fusion sont la proximité temporelle et la complémentarité structurelle. Les objets ou unités informationnelles qu'il manipule, notés creuset, sont génériques : un creuset correspond à un ensemble de cases à compléter par fusion. La sémantique attachée à chaque case n'est pas connue du moteur. Seule la composition structurelle doit être déclarée de manière externe au moteur. Cette technique, qui permet au moteur d'engendrer des formats de creuset à façon sans en modifier les traitements, permet d'envisager l'intégration du moteur dans un squelette d'application pour systèmes à caractéristiques multiples. Dans la figure 7.21, nous en résumons les caractéristiques.

Langage dominant	*Stratégie d'intégration*	*Critères d'intégration*	*Stratégie d'exploration*	*Techniques de représentation*
Non	Précoce Progressive	Proximité temporelle Complémentarité structurelle	En profondeur	Creuset

Figure 7.21 : Caractéristiques de notre moteur de fusion.

Nous avons organisé les traitements du moteur en explicitant trois types de fusion de priorités distinctes. La *microfusion* permet un usage parallèle et combiné des dispositifs et langages ; la *macrofusion* autorise l'usage séquentiel néanmoins proche dans le temps (fenêtre temporelle) et combiné des dispositifs et langages ; la *fusion contextuelle* permet d'ignorer la notion de temps pour prendre en compte les changements de fils de dialogue.

Nous avons montré l'apport du modèle PAC-Amodeus et notamment celui de la hiérarchie d'agents PAC du Contrôleur de Dialogue comme support à la fusion. Les services du moteur de fusion sont accessibles par la partie Contrôle de chaque agent PAC. La hiérarchie d'agents permet de structurer l'espace des creusets et par conséquent d'augmenter l'efficacité du moteur. Le moteur effectue la fusion d'unités informationnelles et les conséquences de ces fusions en rapport avec l'interface de sortie sont gérées par les agents. L'organisation hiérarchique du Contrôleur de Dialogue permet des retours d'information immédiats et partiels dans le cas de fusions successives. Nous avons illustré ce point par un exemple au paragraphe 5.4.1.

Nous envisageons plusieurs extensions possibles. Une première classe d'extensions consiste à introduire de nouveaux paramètres qui permettraient d'agir de manière générique sur le comportement du moteur.

Un premier paramètre est la largeur de la fenêtre temporelle. De la même manière que l'utilisateur du Macintosh définit la vitesse du double-clic (Très lent, Lent ou Rapide), nous proposons de définir la largeur de la fenêtre temporelle qui pourrait être calculée dynamiquement par le système en fonction du contexte.

Un deuxième paramètre envisagé permettrait de spécifier les types de fusions possibles. Par exemple, un système qui n'autorise que l'usage synergique des dispositifs ou langages ne requiert que la *microfusion*.

Outre l'ajout de paramètres, il nous semble crucial d'associer un facteur de confiance à chaque case de creuset. Cette notion a été introduite de manière ad hoc pour le système MATIS. Il faudrait le généraliser. Typiquement lorsque l'utilisateur sélectionne une ville dans une fenêtre, le Composant Techniques de Présentation crée un creuset qui contient deux fois le nom de la ville dans des cases différentes. En effet le CTP ne peut décider s'il s'agit d'une ville de départ ou d'arrivée. Les cases contenant les noms de ville ont alors un facteur de confiance très faible, qui permet au moteur de fusionner le creuset en ignorant l'une des cases.

La figure 7.22 illustre le cas où l'utilisateur désigne la ville "Pittsburgh" sans préciser s'il s'agit du départ ou de la destination : le creuset de gauche contient le nom de cette ville pour

les constituants départ et arrivée avec un facteur de confiance nul. L'énoncé oral "From Boston" est non ambigu : le creuset de gauche voit le constituant départ rempli avec la valeur Boston associé à un facteur de confiance maximum : ce constituant remplace la valeur PIT du champ départ du creuset initial.

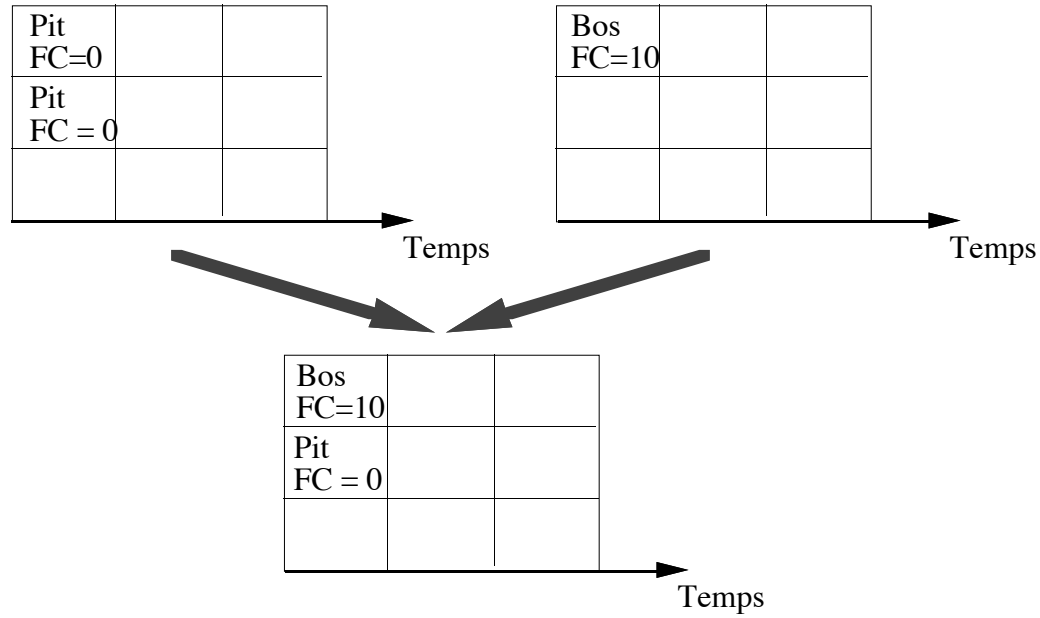
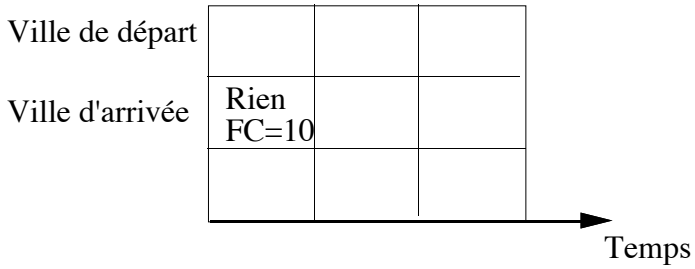
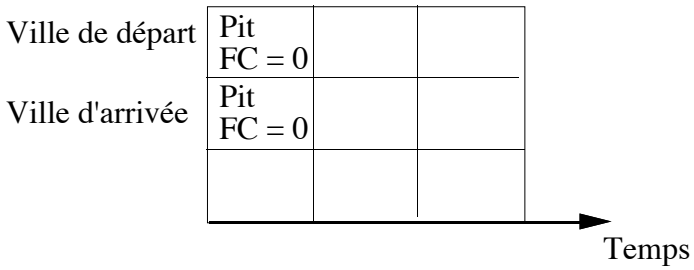


Figure 7.22 : Facteur de confiance associé à chaque case de creuset.

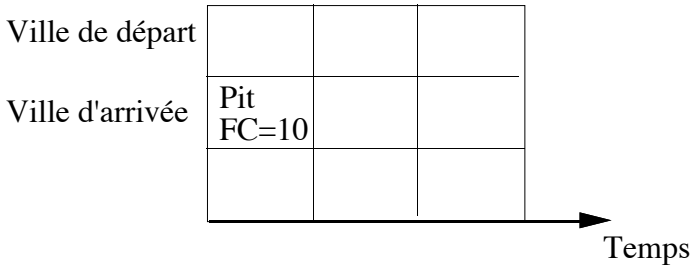
Les facteurs de confiance modélisent l'incertitude des informations dans chaque case. Ils permettent aussi de spécifier un ordre de remplissage des cases d'un creuset, très utile à la résolution des coréférences. Par exemple si l'utilisateur de MATIS articule la phrase "*Flights to this city*" puis sélectionne une ville avec la souris. Le PTC crée un premier creuset correspondant à la phrase :



Le creuset correspondant à la sélection avec la souris est le suivant :



Le moteur de fusion sera alors capable de fusionner les deux creusets en s'appuyant sur les facteurs de confiance. Un très haut facteur de confiance dans une case non remplie, fonctionne comme un aimant et va influencer le moteur à fusionner des informations dans cette case. Ainsi de ces deux creusets, le moteur déduit le creuset suivant [23]:



6. CONCLUSION

Notre contribution à la réalisation logicielle des systèmes est multiple :

• nous proposons un modèle d'architecture pour la réalisation logicielle des systèmes à caractéristiques multiples,

• nous avons intégré dans un modèle d'architecture les phénomènes de fusion et de parallélisme, deux caractéristiques essentielles des systèmes à caractéristiques multiples,

• nous proposons un mécanisme de fusion indépendant du domaine de l'application du système.

---

[23] Dans le système MATIS actuel, de cet exemple d'énoncé découle une requête contenant Pit comme ville de départ et non d'arrivée. Le moteur parcourt dans un sens arbitraire les cases du creuset et la première case est la villle de départ.

Nous avons montré que PAC-Amodeus autorise des traitements en parallèle à tout niveau d'abstraction et fournit un cadre de réalisation pour la fusion d'informations. Ce sont deux conditions nécessaires à la réalisation de systèmes à caractéristiques multiples. De plus, notre modèle offre un support intéressant aux retours d'information immédiats : un retour d'information peut être engendré par chaque agent de la hiérarchie. Ainsi, pour une commande donnée, des retours partiels et spécifiques au niveau d'abstraction peuvent être restitués avant même que le mécanisme de fusion soit achevé. L'indépendance du Contrôleur de Dialogue vis-à-vis des langages et des dispositifs limite la répercussion de l'ajout ou de la modification d'un langage d'interaction aux composants d'Interaction de Niveau Bas et Techniques de Présentation.

Grâce à sa généricité et en tant que service "d'utilité publique" pour les systèmes à caractéristiques multiples, le moteur de fusion est un bon candidat comme composant réutilisable d'un squelette d'application. Comme nous l'avons annoncé, nous avons cependant à approfondir le mécanisme de fusion et ses critères. Notre travail futur s'oriente vers l'intégration d'un réel modèle du dialogue au sein de l'architecture logicielle et l'exploitation plus importante du contexte de l'interaction comme critère supplémentaire de fusion. Notre support pour ce travail sera le système MATIS.

Nous avons présenté notre modèle d'architecture PAC-Amodeus au chapitre VI. Dans ce chapitre, nous venons de montrer son application au cas des systèmes à caractéristiques multiples. Nous avons choisi de concrétiser ce cadre conceptuel avec la réalisation de deux systèmes, MATIS et NoteBook, dont nous détaillons la mise en œuvre logicielle au chapitre VIII suivant.

Chapitre VIII

Du modèle à la réalité : les systèmes NoteBook et MATIS

1. Introduction
2. Description statique selon M^{2}LD de NoteBook et MATIS
3. Le système NoteBook
4. Le système MATIS
5. Conclusion

1. INTRODUCTION

Nous avons développé à Carnegie Melon University (Etats-Unis) deux systèmes à caractéristiques multiples : NoteBook et MATIS. L'objectif de ces réalisations est l'application et la vérification de nos résultats. Néanmoins NoteBook notre première réalisation constitue aussi un travail exploratoire pour la mise en évidence des problèmes. Nous avons sans nul doute exploité les leçons tirées de la réalisation de NoteBook pour la mise en œuvre de MATIS.

Ces deux systèmes ont été utilisés comme exemples pour illustrer nos travaux tout au long de ce mémoire. En conséquence les spécifications externes de NoteBook et MATIS ont été exposées au Chapitre I pour situer leurs domaines d'application.

Nos travaux, centrés sur les interfaces en entrée, nous ont conduits à réaliser ces deux systèmes dont les interfaces sont de type différent [Romary 93]:

- NoteBook comporte un noyau fonctionnel presque réduit à néant si ce n'est la gestion des fichiers des notes. La tâche principale réalisée avec NoteBook est la manipulation du bloc-notes affiché à l'écran. Le dialogue est donc orienté par les objets à l'écran que l'utilisateur manipule. Nous caractérisons cette interface de type *Faire* [Romary 93]. L'utilisateur *fait les actions* directement sur le bloc-notes.

- Au contraire MATIS comporte un Noyau Fonctionnel substantiel que constitue la base de données accessibles par le langage S.Q.L. (System Query Language). La tâche de l'utilisateur est la spécification de requête afin de planifier un voyage. Le dialogue est alors orienté par la tâche de planification. L'interface de MATIS est de type *Faire faire* [Romary 93] ; elle constitue un intermédiaire pour *faire rechercher* des informations par le Noyau Fonctionnel.

L'objet de ce chapitre est de montrer l'application de nos résultats à la description et à la réalisation d'un système. Nous présentons tout d'abord une description globale selon M^2LD de NoteBook et MATIS au paragraphe 2. Puis nous allons dans le détail de leurs utilisations et de leurs architectures logicielles aux paragraphes 3 et 4.

2. DESCRIPTION STATIQUE SELON M2LD DE NOTEBOOK ET MATIS

Nous présentons une description globale de MATIS et NoteBook en les classant dans l'espace M^2LD. Par conséquent nous considérons le nombre de dispositifs physiques et de langages d'interaction qui constituent les passerelles logiques et physiques entre l'utilisateur et le système.

2.1. Les dispositifs physiques d'entrée et sortie

Tous deux offrent :

- trois dispositifs physiques d'entrée (de = 3),
  - le clavier,
  - la souris,
  - le microphone,

- et un dispositif physique de sortie (ds = 1),
  - l'écran graphique de la machine NeXT.

Ils sont donc multidispositif en entrée et monodispositif en sortie.

2.2. Les langages d'interaction en entrée et en sortie

Tous deux offrent :

- plusieurs langages d'interaction d'entrée ($le_{NoteBook} = 2$ et $le_{MATIS} = 3$),
  - la manipulation directe sur les objets graphiques,
  - le langage naturel,
  - un langage de commande dans le cas de MATIS uniquement,

- deux langages d'interaction en sortie (ls = 2),
  - une présentation graphique, et
  - le langage naturel.

Ils sont donc multilangage en entrée et multilangage en sortie.

2.3. Positions dans l'espace M2LD

Comme le montre la figure 8.1, la classification M2LD ne permet pas de distinguer les deux systèmes. Tous deux appartiennent aux classes :

- Multidispositif et multilangage pour leurs interfaces en entrée et
- Monodispositif et multilangage pour leurs interfaces en sortie.

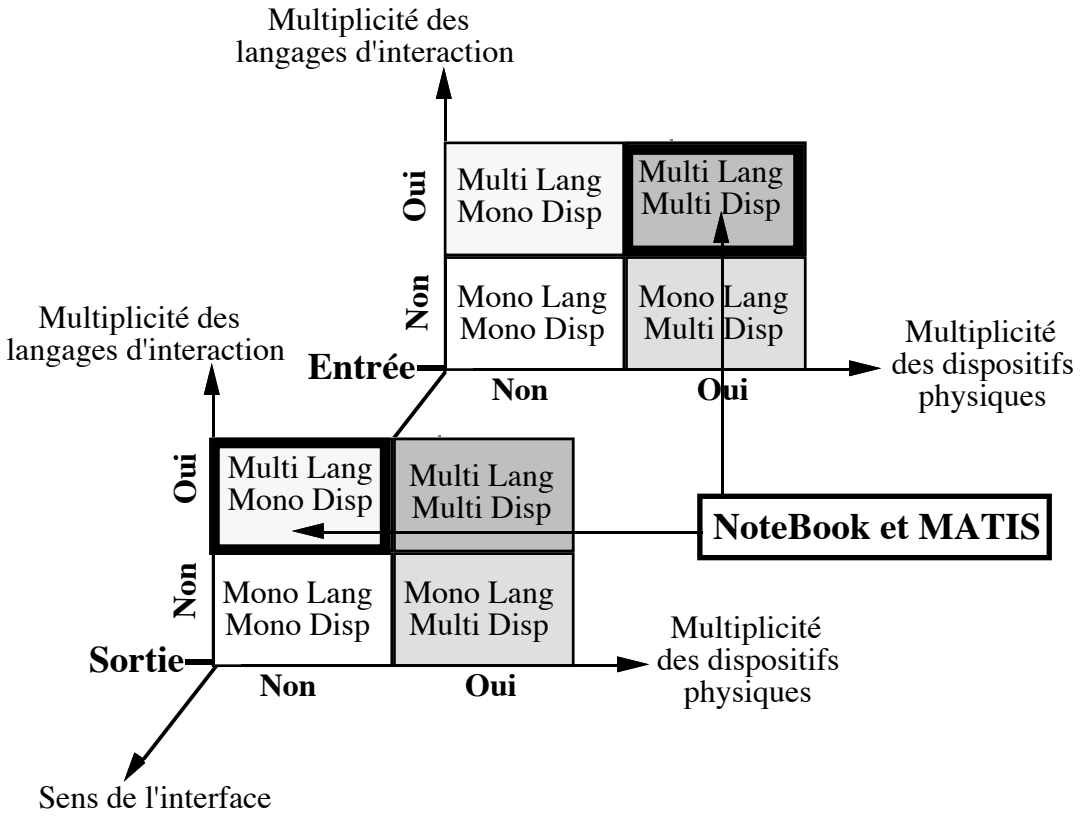


Figure 8.1 : NoteBook et MATIS situés dans l'espace M2LD.

Bien qu'ils appartiennent aux mêmes classes M2LD, nous verrons que leurs utilisations sont différentes : l'application de O2LD et ULD aux paragraphes suivants va donc permettre de les différencier.

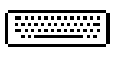
3. LE SYSTÈME NOTEBOOK

NoteBook [Nigay 91a] est un bloc-notes électronique. Il constitue notre première réalisation et un travail exploratoire. L'objectif initial est d'appliquer PAC-Amodeus à un système offrant plusieurs dispositifs et langages en entrée. Nous avons ensuite étudié l'usage alterné des langages et dispositifs en ajoutant la possibilité d'insérer une note en articulant la phrase "*Insert a new note*" puis en sélectionnant le lieu d'insertion.

3.1. Appliquons UOM à NoteBook

Au paragraphe précédent, l'application de M^2LD, première étape de la méthode UOM, a permis de caractériser NoteBook de multidispositif et multilangage pour son interface en entrée et de monodispositif et multilangage pour son interface en sortie. Nous étudions maintenant les choix offerts par NoteBook (O^2LD).

En entrée, le langage naturel est assigné au microphone ou au clavier. En Annexe C, nous décrivons le langage naturel dans le contexte de NoteBook en présentant la liste des blocs syntaxiques reconnus. La manipulation directe est assignée à la souris. Pour étudier les choix offerts à l'utilisateur, nous présentons une table récapitulative des commandes avec leurs langages associés.

	Manipulation directe 	**Langage naturel**  **ou** 
Changer de note		
Changement direct (numéro de la note)	**Oui**	**Oui**
Changement sémantique (sujet de la note)	***Oui***	***Oui***
Changement relatif (feuilleter le bloc-notes)	**Oui**	**Oui**
Supprimer une note		
Suppression directe (numéro de la note)	**Oui**	**Oui**
Suppression sémantique (sujet de la note)	***Non***	***Oui***
Suppression relative (suppression de la note précédente ou suivante)	**Non**	**Oui**
Ajouter une note		
Ajout en fin de bloc-notes	**Oui**	**Oui**
Ajout après la note	**Non**	**Oui**
Créer un nouveau bloc-notes	**Oui**	**Oui**

Les cases écrites en italique correspondent à des commandes non réalisées.

Figure 8.2 : Classification des commandes de NoteBook.

A la lumière de ce tableau, nous constatons que l'utilisateur a le choix du langage pour la plupart des commandes. Si le langage naturel est choisi, alors le choix subsiste pour le dispositif. Au contraire la manipulation directe étant assignée à la souris, son choix implique l'usage de la souris. Ces constatations nous conduisent à positionner NoteBook dans l'espace O^2LD au point noté "NoteBook e" de la figure 8.3.

En sortie le dispositif, l'écran, est unique. De plus, bien que NoteBook utilise deux langages en sortie, il n'effectue pas réellement un choix. Si le langage d'interaction en entrée est le langage naturel alors celui-ci est aussi utilisé en sortie en complément de la présentation graphique. Dans les autres cas, seule une présentation graphique est produite. Ceci justifie la position du point "NoteBook s" de la figure 8.3, proche du point remarquable "Oblig. L et Oblig. D".

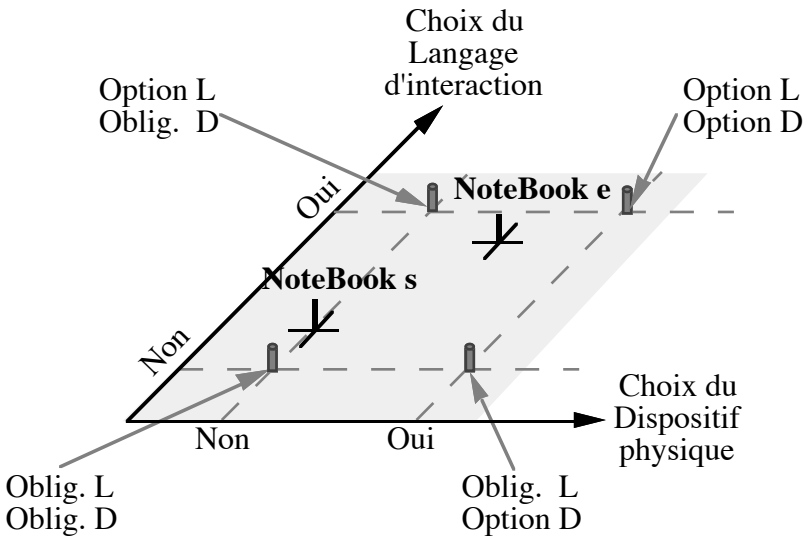


Figure 8.3 : Positions de NoteBook (Interface en entrée et en sortie) dans l'espace O^2LD.

Bien que NoteBook offre le choix des langages et des dispositifs en entrée, leur usage est indépendant (ULD). Seule la commande "*Insert a new note*" offre un usage combiné et séquentiel (noté usage alterné) des langages et des dispositifs.

Les dispositifs et langages peuvent être utilisés en parallèle. Cependant la notion de temps n'étant pas exploitée dans NoteBook, il est possible que l'ordre de spécification des commandes ne soit pas respecté. Par exemple si l'utilisateur sélectionne une note avec la souris tout en disant "*Remove the note*" : la note supprimée sera celle nouvellement sélectionnée avec la souris si la sélection est traitée avant la phrase articulée. Dans le cas contraire la note supprimée sera la note courante avant sélection.

En résumé NoteBook offre principalement un usage indépendant et parallèle des langages et des dispositifs d'entrée. Il est donc de la classe Concurrent dans la classification ULD.

En sortie NoteBook utilise les deux langages de façon parallèle mais non combinée. L'usage des langages de sortie est donc concurrent.

Nous venons de caractériser NoteBook en appliquant la méthode UOM. Nous présentons maintenant sa mise en œuvre logicielle selon le modèle PAC-Amodeus.

3.2. Architecture logicielle de NoteBook

Nous présentons l'architecture logiciel de NoteBook à la figure 8.4.

3.2.1. Les composants logiciels

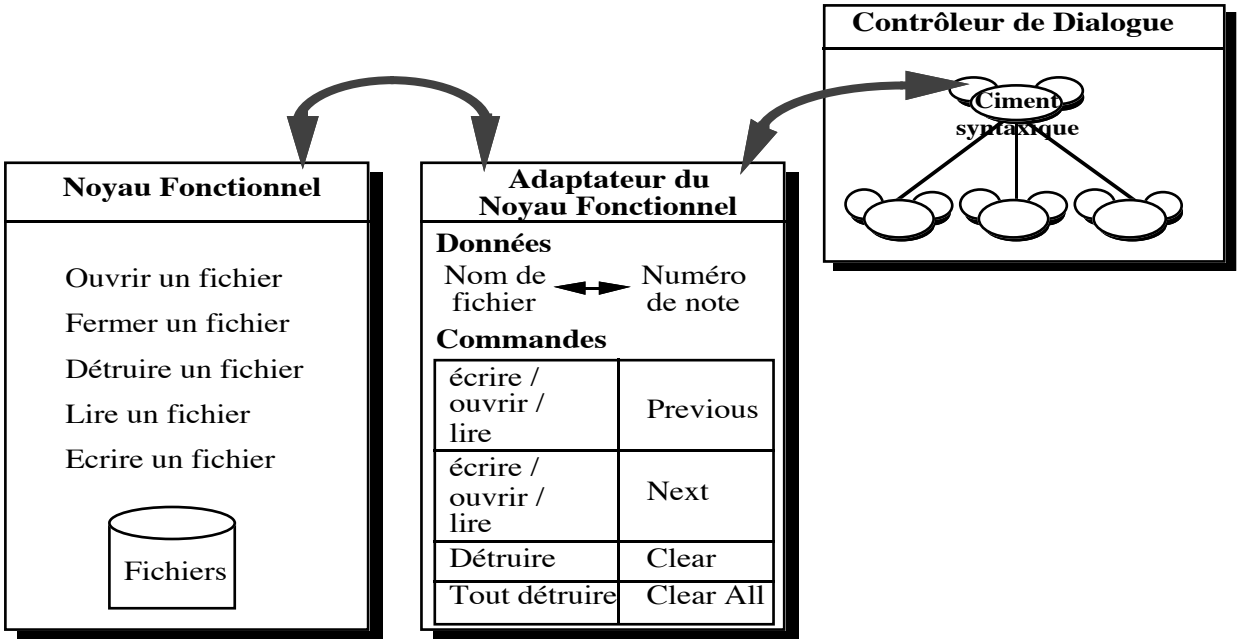
Le Noyau Fonctionnel gère les fichiers des notes. Il y a un ficher de type texte par note du bloc-notes.

L'Adaptateur du Noyau Fonctionnel traduit les commandes issues par l'utilisateur en termes compréhensibles par le Noyau Fonctionnel. Pour cela il maintient une table de correspondance entre les numéros de note manipulés par le Contrôleur de Dialogue et les noms des fichiers gérés par le Noyau Fonctionnel.

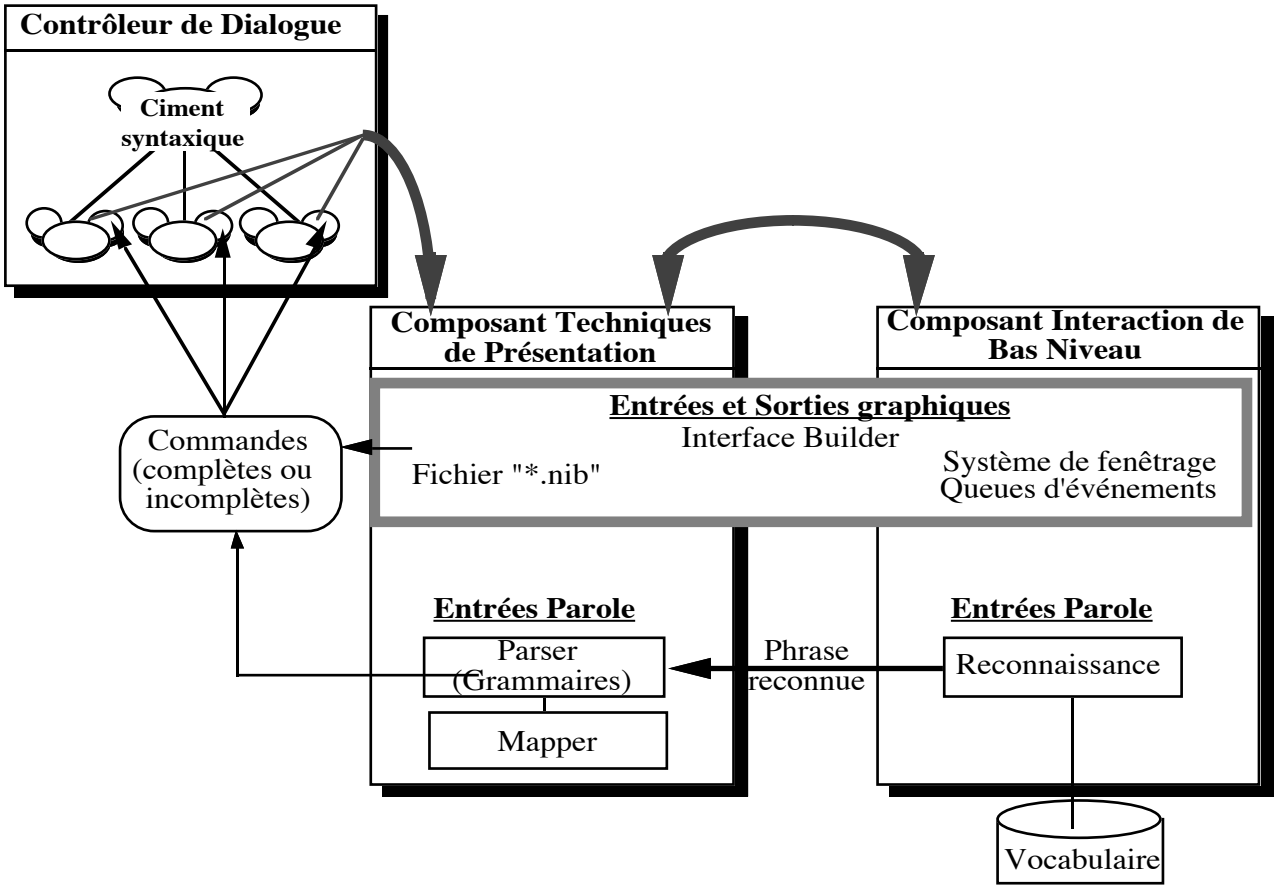
En sens inverse, le contenu des notes fourni par le Noyau Fonctionnel est envoyé sous forme textuelle au Contrôleur de Dialogue.

Organisé en une hiérarchie d'agents PAC, le Contrôleur de Dialogue est le composant principal. Avant de détailler la hiérarchie d'agents au paragraphe 3.2.2., nous devons rappeler les trois responsabilités de ce composant : enchaînement des tâches, transformation de formalisme et mécanisme de correspondance.

Le Composant Techniques de Présentation définit les règles de correspondance entre objets de présentation et objets d'interaction. En sens inverse, ce composant abstrait les informations spécifiées par l'utilisateur dans le formalisme du Contrôleur de Dialogue. Ce composant constitue le plus haut niveau d'abstraction où interviennent les langages d'interaction et les dispositifs physiques. Dans le cas de l'implémentation de NoteBook, ce composant est scindé en deux parties : l'une est dédiée à la présentation graphique et l'interaction par manipulation directe sur les objets graphiques (partie englobée par Interface Builder), l'autre est dédiée à l'analyse des phrases en langage naturel (écrites ou parlées).



1- Les composants de gauche de l'arche.



2- Les composants de droite de l'arche.

Figure 8.4 : Composants logiciels du système NoteBook.

Le Composant Bas Niveau d'Interaction désigne la plate-forme d'accueil logicielle et matérielle. Ce niveau regroupe les services d'acquisition, d'estampille et de répartition des événements mais aussi les objets d'interaction des boîtes à outils et les systèmes de reconnaissance spécialisés. Par exemple, un message vocal est transmis au Composant Techniques de Présentation sous forme d'un événement estampillé dont le contenu est une chaîne de caractères.

3.2.2. Le Contrôleur de Dialogue organisé en une hiérarchie d'agents PAC

La hiérarchie d'agents PAC présentée à la figure 8.5 structure le Contrôleur de Dialogue. Elle contient deux niveaux.

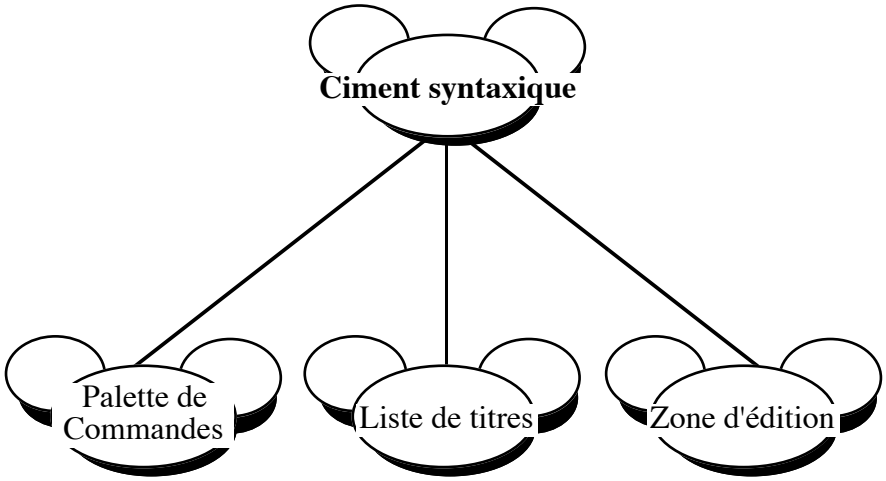


Figure 8.5 : Hiérarchie d'agents PAC structurant le Contrôleur de Dialogue dans NoteBook.

Les feuilles de la hiérarchie sont les agents suivants :

• L'agent *Palette* gère les commandes, spécifiées en langage naturel ou par manipulation directe. Sa présentation affiche la palette des boutons en bas de la fenêtre de NoteBook.

• L'agent *Liste de titres* correspond à la liste des titres des notes. Sa présentation affiche la liste, et reçoit les commandes de sélection. Son abstraction maintient la liste des titres sous forme textuelle.

• L'agent *Zone d'édition* correspond à la zone de saisie du contenu d'une note. Le contenu de la note est maintenu par l'abstraction.

L'agent *Ciment syntaxique*, racine de la hiérarchie, permet de cimenter les actions de l'utilisateur reçues et traitées par ses agents fils et communique avec le composant Adaptateur de Noyau Fonctionnel. Par exemple lorsque l'utilisateur saisit au clavier le contenu d'une nouvelle note, la première ligne constitue aussi le titre de la note. La modification de cette première ligne, reçue par l'agent *Zone d'édition*, provoque l'envoi d'un message à l'agent père, *Ciment syntaxique*, qui répercute alors la modification à l'agent *Liste de titres*.

3.2.3. Réalisation de fusion pour la commande d'insertion d'une note

A la figure 8.6, nous montrons comment la fusion est réalisée au sein de la hiérarchie d'agents dans le cas de l'énoncé suivant : l'utilisateur articule la phrase "*Insert a new note*" puis sélectionne une note comme lieu d'insertion. Une nouvelle note est alors créée après la note sélectionnée. La fusion est effectuée au niveau de l'agent *Ciment syntaxique*. Ce dernier reçoit de l'agent *Palette* la commande d'insertion et de l'agent *Liste de titre* le lieu d'insertion. La figure 8.6 explicite les messages entre agents et l'état maintenu par l'abstraction de l'agent Ciment syntaxique. Les numéros traduisent l'ordre des messages.

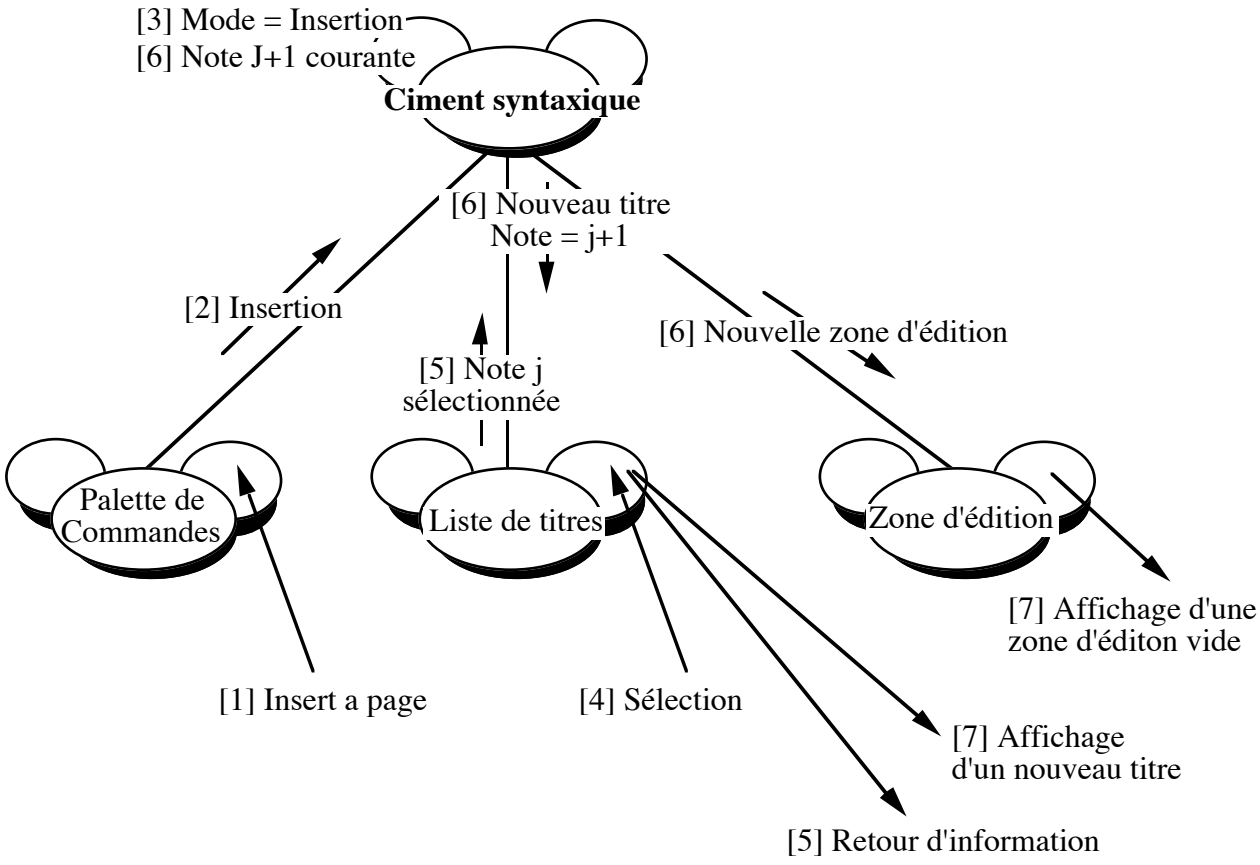


Figure 8.6 : Fusion réalisée au sein de la hiérarchie d'agents.

3.3. Conclusion

La réalisation de NoteBook a permis une étude sur l'architecture logicielle des systèmes qui offrent plusieurs langages et dispositifs. Nous en avons conclu :

- l'indépendance du Contrôleur de Dialogue vis-à-vis des langages et dispositifs,
- l'exploitation des langages d'interaction au niveau du Composant Techniques de Présentation,
- et la gestion des dispositifs physiques à la charge du Composant Interaction de Bas Niveau.

Cette répartition permet de rajouter des langages ou des dispositifs à moindre coût en localisant les modifications et sans remaniement du Contrôleur de Dialogue.

L'aspect temporel des informations spécifiées par l'utilisateur nous a semblé primordial pour offrir l'usage concurrent et synergique des dispositifs et des langages. Aussi, suite à cette première expérience, nous avons entrepris la réalisation de MATIS avec comme objectif l'étude de la fusion des informations et la prise en compte du temps. Pour cela MATIS est une plate-forme idéale : il permet la spécification de requête avec de nombreux paramètres et par là même offre un champ d'investigation pour la fusion d'informations. De plus le domaine d'application justifie la spécification de plusieurs requêtes en parallèle, et donc l'usage concurrent des langages et dispositifs.

4. LE SYSTÈME MATIS

Nous rappelons que MATIS [Nigay 93c] (Multimodal Airline Travel Information System) est un système d'information sur les transports aériens. Il fournit, en réponse à des requêtes de l'utilisateur, des informations sur les vols entre deux villes.

4.1. Appliquons UOM à MATIS

Au paragraphe 2, l'application de M^2LD, première étape de la méthode UOM, a conduit à caractériser MATIS de multidispositif et multilangage pour son interface en entrée et de Monodispositif et multilangage pour son interface en sortie.

Au chapitre IV, nous avons illustré la classification O^2LD en positionnant MATIS dans l'espace correspondant. Nous rappelons ici les résultats de cette analyse. Pour son interface en entrée, l'utilisateur a effectivement le choix parmi trois langages et dispositifs pour spécifier une requête. Au contraire la manipulation des objets graphiques se fait uniquement avec la souris par manipulation directe. Par exemple, il n'est pas possible de fermer une fenêtre de résultats en utilisant le langage naturel qu'il soit écrit (clavier) ou oral (microphone). Ainsi comme le montre la figure 8.7, MATIS quant à son interface en entrée a été placé à équidistance entre les quatre points remarquables de l'espace. Au contraire MATIS pour son interface en sortie est placée très proche du point "Oblig. L et Oblig. D" à la figure 8.7. Comme le système NoteBook, le dispositif physique de sortie, l'écran, est unique. D'autre part bien que les langages de sortie soient au nombre de deux, MATIS n'effectue pas réellement de choix : le langage de sortie est directement lié au langage utilisé en entrée.

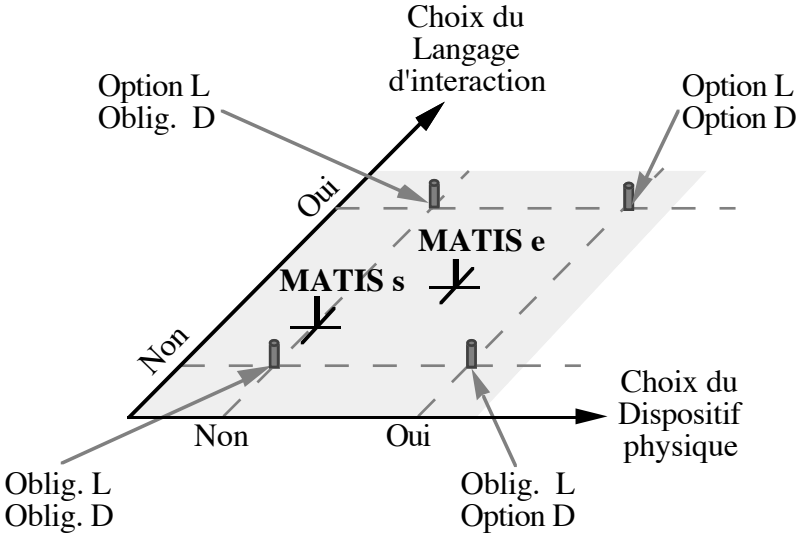


Figure 8.7 : Rappel des positions de MATIS (Interface en entrée et en sortie) dans l'espace O^2LD.

L'application d'ULD à MATIS a explicité plusieurs utilisations possibles que nous illustrons par des exemples d'énoncé.

En entrée MATIS offre :

1- L'usage synergique des langages et dispositifs que nous illustrons par l'exemple suivant : la phrase orale "*Flight from Pittsburgh to this city*" accompagnée de la sélection de "*Boston*" avec la souris.

2- L'usage synergique des dispositifs uniquement, comme l'exemple suivant : la phrase orale "*Flight from Pittsburgh*" accompagnée de la saisie au clavier de la phrase "*serving a meal and arriving in the afternoon*".

3- L'usage alterné des langages et dispositifs : ceci correspond au cas 1 avec séquencement des actions. Par exemple l'utilisateur articule une phrase puis sélectionne avec la souris.

4- L'usage alterné des dispositifs : ceci correspond au cas 2 avec séquencement des actions. Par exemple l'utilisateur commence par articuler le début d'une phrase et la finit par saisie au clavier.

5- L'usage concurrent des langages et dispositifs, comme l'exemple suivant : l'utilisateur spécifie une requête en la dictant en langage naturel, tout en complétant une autre requête par saisie directe dans le formulaire. L'usage concurrent permet la spécification de plusieurs requêtes en parallèle.

6- L'usage concurrent des dispositifs uniquement, comme l'exemple suivant : l'utilisateur spécifie une requête en la dictant en langage naturel, tout en saisissant au clavier une autre requête en langage naturel.

7- L'usage exclusif des dispositifs et langages : typiquement MATIS peut être utilisé comme un système vocal uniquement.

Au travers de ces exemples, nous montrons que MATIS a été conçu de façon à offrir le plus de choix possibles à l'utilisateur. A l'opposé son interface en sortie est très pauvre et supporte :

• L'usage concurrent des deux langages : par exemple MATIS affiche en langage naturel "*Flight from Pittsburgh*" tout en affichant "PIT" dans le formulaire de la requête courante (champ intitulé "from").

• L'usage exclusif des deux langages : c'est le cas lorsque MATIS affiche une fenêtre de résultats d'une requête.

En appliquant la méthode UOM à MATIS, nous avons identifié ses caractéristiques. Nous présentons maintenant sa réalisation logicielle selon le modèle appliqué PAC-Amodeus.

4.2. Architecture logicielle de MATIS

La figure 8.8 présente l'architecture logicielle de MATIS. L'utilisateur et le Noyau Fonctionnel produisent et consomment des informations par l'intermédiaire du Contrôleur de Dialogue, le composant clé de voûte de l'architecture.

4.2.1. Les composants logiciels

Le Noyau Fonctionnel correspond à la base de données contenant les informations sur les transports aériens. L'accès à ces informations se fait par l'intermédiaire de requêtes en langage S.Q.L.

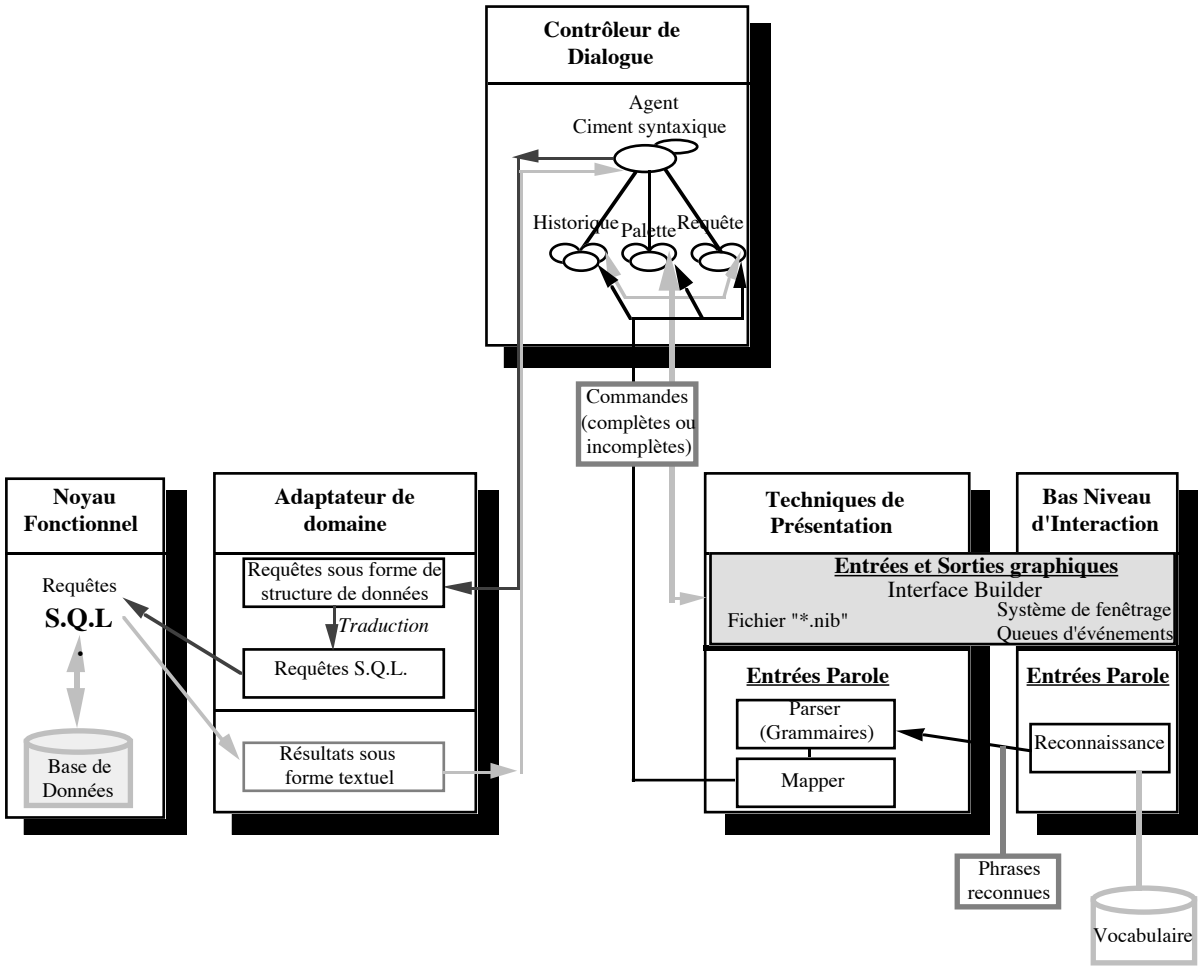


Figure 8.8 : Composants logiciels du système MATIS.

Le composant Adaptateur de Noyau Fonctionnel a un rôle de traducteur. Il traduit le résultat des requêtes soumises à la base de données sous une forme textuelle compréhensible par le Contrôleur de Dialogue. Ce dernier en produit une présentation graphique perceptible par

l'utilisateur. Symétriquement, l'Adaptateur Noyau Fonctionnel traduit les requêtes exprimées dans le formalisme du Contrôleur de Dialogue en une requête S.Q.L. Ce composant sert donc d'interface conçue pour absorber les changements entre ses voisins directs. Par exemple, le changement de la base de données provoquerait uniquement une modification du composant Adaptateur de Domaine.

Le Contrôleur de Dialogue est organisé en une hiérarchie d'agents PAC qui forme un pont entre l'Adaptateur du Noyau Fonctionnel et le Composant Techniques de Présentation. Nous détaillons la hiérarchie d'agents au paragraphe 4.2.2.

Les composants Techniques de Présentation et Interaction de Bas Niveau correspondent à ceux de NoteBook dans leurs organisations et rôles. Les mêmes langages et dispositifs sont utilisés sur une plate-forme logicielle et matérielle identique.

4.2.2. Le Contrôleur de Dialogue organisé en une hiérarchie d'agents PAC

La hiérarchie d'agents composant le Contrôleur de Dialogue est présentée à la figure 8.9. Pour sa conception, nous avons appliqué les règles heuristiques du chapitre VI permettant d'identifier les agents PAC en fonction de l'interface à gérer .

La hiérarchie contient deux niveaux. Les feuilles de la hiérarchie sont les agents suivants :

- Les agents *Requête* correspondent aux requêtes en cours de spécification.

- Les agents *Résultat de requête* correspondent aux fenêtres affichant les réponses à une requête (fenêtre intitulée "Results of request").

- L'agent *Retour d'Information* assure l'affichage d'un retour d'information sur tous les actes de parole (analyse syntaxique de la phrase dictée) ; cette information est affichée dans la fenêtre "Speech Panel".

- L'agent *Bloc-notes* gère un bloc-notes où l'utilisateur peut noter des informations relatives à son plan de voyage (fenêtre intitulée "Notepad").

- L'agent *Historique* gère l'historique des requêtes émises (cet historique peut être édité dans la fenêtre "Requests History").

• L'agent *Palette* correspond à la palette d'outils offerts à l'utilisateur pour spécifier une requête (fenêtre intitulée "Requests Tools") comme la liste des compagnies aériennes et la liste des villes.

Enfin l'agent *Ciment syntaxique*, racine de la hiérarchie, permet de cimenter les actions de l'utilisateur reçues et traitées par ses agents fils et communique avec le composant Adaptateur de Noyau Fonctionnel lors de l'envoi d'une requête.

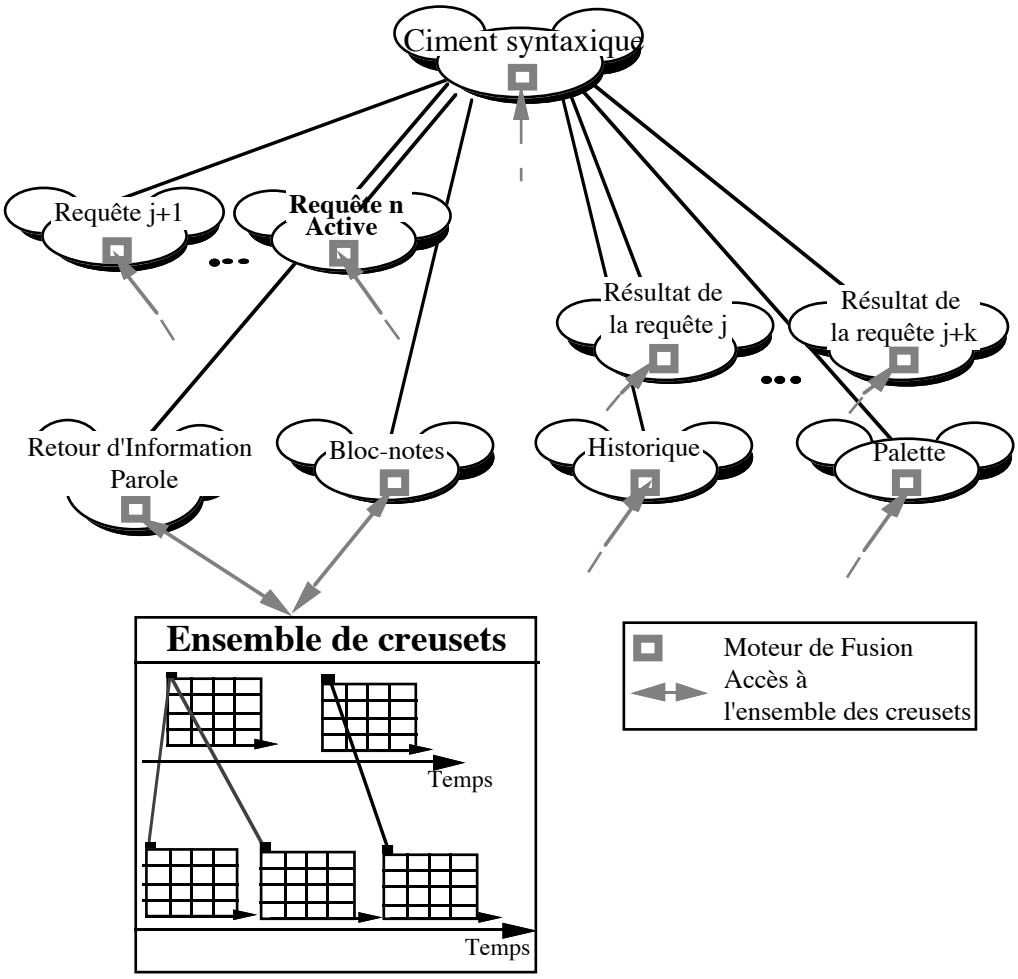


Figure 8.9 : Hiérarchie d'agents PAC structurant le Contrôleur de Dialogue dans MATIS.

La partie Contrôle de chaque agent appelle le moteur de fusion qui manipule les creusets regroupés en un seul ensemble. Le paragraphe suivant décrit le moteur de fusion ainsi que la représentation des informations au sein d'un creuset.

4.2.3. Le mécanisme de fusion

Dans MATIS, le moteur de fusion exploite les trois niveaux de fusion présentés au chapitre VII.

• La microfusion s'appuie sur le temps et combine des creusets ayant des intervalles de temps qui s'entrelacent (parallélisme ou pseudo parallélisme).

• La macrofusion s'appuie sur le temps et combine des creusets qui appartiennent à une fenêtre temporelle (proximité temporelle).

• La fusion contextuelle utilise uniquement le contexte courant de l'interaction. Dans MATIS, ce contexte est réduit à la requête courante, qui correspond à un agent PAC (figure 8.9).

Les composants structurels d'un creuset correspondent à tous les champs d'une requête. La fusion s'effectue donc uniquement sur des champs de requêtes. Un creuset est présenté à la figure 8.10.

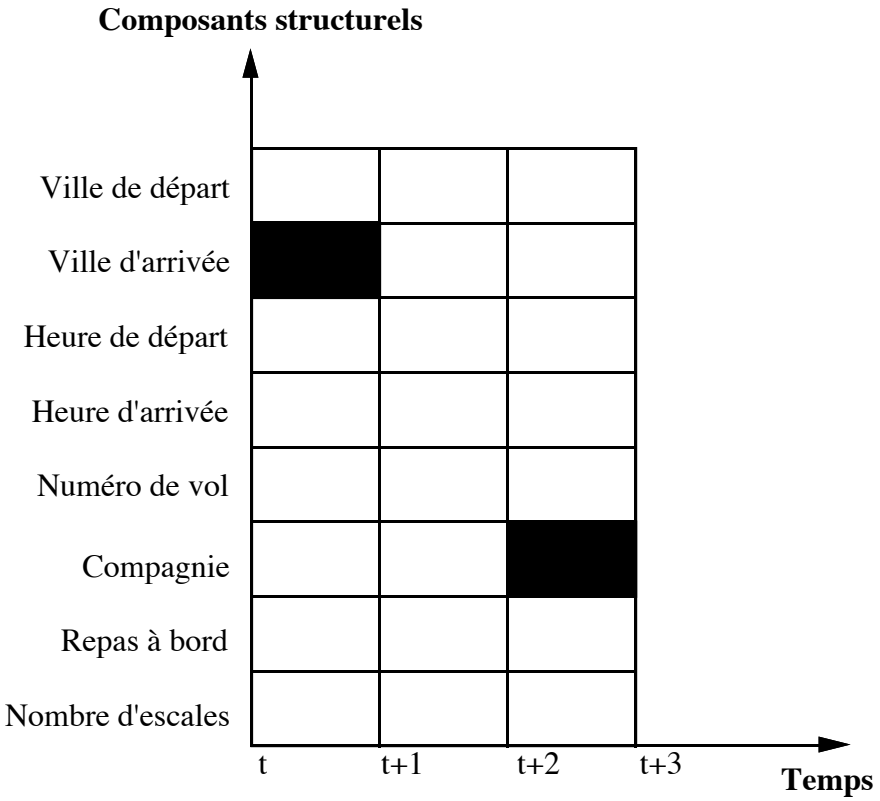


Figure 8.10 : Structure d'un creuset dans MATIS.

Nous illustrons le mécanisme de fusion en corrélation avec la hiérarchie d'agents PAC par l'exemple suivant : l'utilisateur prononce la phrase "*USAir flight from Pittsburgh*", tout en sélectionnant Boston dans une fenêtre de résultats avec la souris. Comme le montre la figure 8.11, deux creusets sont fusionnés par microfusion par le moteur.

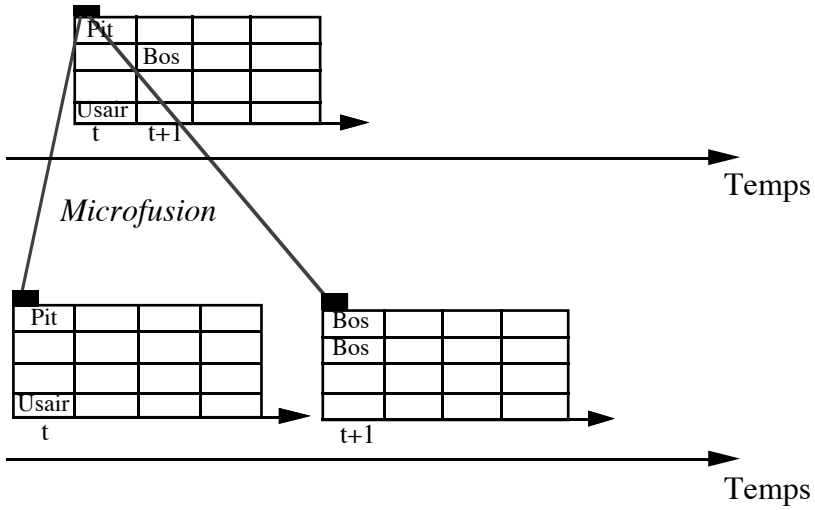


Figure 8.11 : Microfusion effectuée.

La phrase articulée et la sélection avec la souris sont acquises en parallèle par le Composant de Bas Niveau d'Interaction. La sélection provoque l'appel d'une fonction associée au bouton qui a été sélectionné. Cette fonction du Composant Techniques de Présentation crée alors un creuset qui contient "Boston" dans les deux premiers composants. "Boston" correspond au titre du bouton. Ce creuset est transmis à l'agent PAC *Résultat de requête* , qui produit un retour d'information immédiat traduisant la sélection et fait ensuite appel au moteur de fusion. Aucune fusion n'est effectuée pour l'instant. L'agent envoie le creuset incomplet à son père direct qui va à son tour transmettre le creuset à l'agent *Requête J*. Ce dernier correspond à la requête active et ne contient aucun champ rempli. Celui-ci produit un retour d'information immédiat en affichant Boston dans les deux champs du formulaire de la requête. Son abstraction maintient les valeurs des champs spécifiés.

En parallèle, mais nécessitant des traitements plus longs, la phrase acquise est traitée par Sphinx dans le Composant de Bas Niveau d'Interaction pour la transformer en une chaîne de caractères. Celle-ci est analysée dans le Composant Techniques de Présentation pour créer un creuset contenant les informations "PIT" et "USAir". Chaque composant du creuset correspond à un bloc syntaxique de la phrase, identifié par la grammaire du langage naturel. Nous fournissons l'ensemble des blocs syntaxiques pour le langage naturel de MATIS en Annexe D. Le nouveau creuset ainsi formé est transmis à l'agent *Retour d'Information* qui affiche les blocs syntaxiques reconnus de la phrase avec leurs valeurs, comme le montre la figure 8.12.

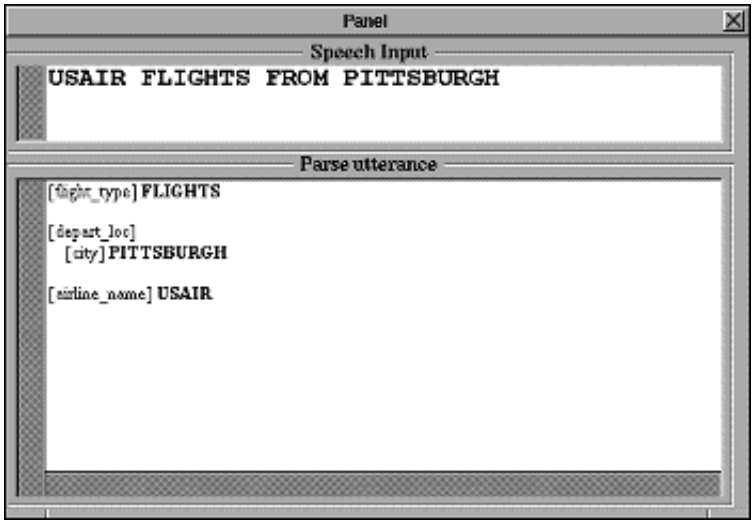


Figure 8.12 : Retour d'information de niveau syntaxique de la phrase en langage naturel.

L'agent *Retour d'Information* fait alors appel au moteur de fusion, qui va effectuer une microfusion avec le creuset précédent (figure 8.11). Le creuset résultat est envoyé au père commun des deux agents, en l'occurrence la racine : l'agent *Ciment syntaxique*. Le moteur spécifie à l'agent qu'il s'agit d'une modification d'une requête. L'agent *Ciment syntaxique* envoie le creuset résultat à l'agent *Requête J* actif, qui va alors afficher les valeurs du creuset dans son formulaire. La figure 8.13 présente une copie d'écran du formulaire de la requête ainsi obtenue.

Figure 8.13 : Formulaire de la requête courante.

L'agent Ciment syntaxique maintient associé un agent *Requête* à chaque creuset identifié par le moteur. Ainsi lors de dé-fusion ou de modification, il est en mesure de transmettre les creusets envoyés par le moteur à l'agent correspondant.

Nous envisageons maintenant que l'utilisateur décide de connaître les vols de Boston à Baltimore avant de finir de spécifier la requête courante. Pour cela il articule la phrase "*Continental flight from Boston to Baltimore serving a meal*". Cette phrase, après avoir été reconnue par Sphinx et analysée pour en extraire les blocs syntaxiques, va provoquer la création du creuset présenté à la figure 8.14.

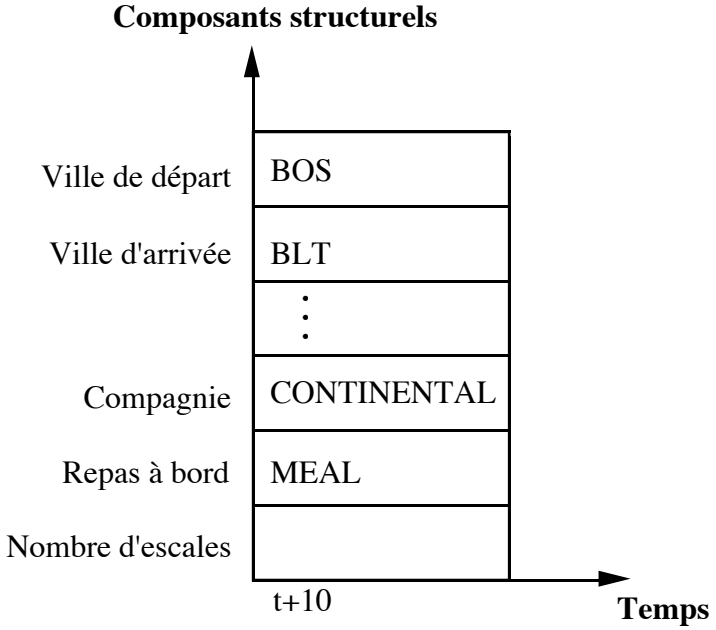


Figure 8.14 : Le creuset correspondant à la phrase "*Continental flight from Boston to Baltimore serving a meal*".

Reçu par l'agent *Retour d'Information* , il va être transmis au moteur de fusion. Aucune fusion n'est possible car les creusets ne sont pas structurellement compatibles. L'agent *Retour d'Information* transmet alors le creuset à son père l'agent *Ciment syntaxique*. Le creuset étant nouveau, il crée dynamiquement un agent *Requête J+1* qui correspond à la nouvelle requête active. Ce dernier présente les informations du creuset dans un nouveau formulaire. Comme le montre la figure 8.15, deux formulaires de requête sont alors affichés à l'écran.

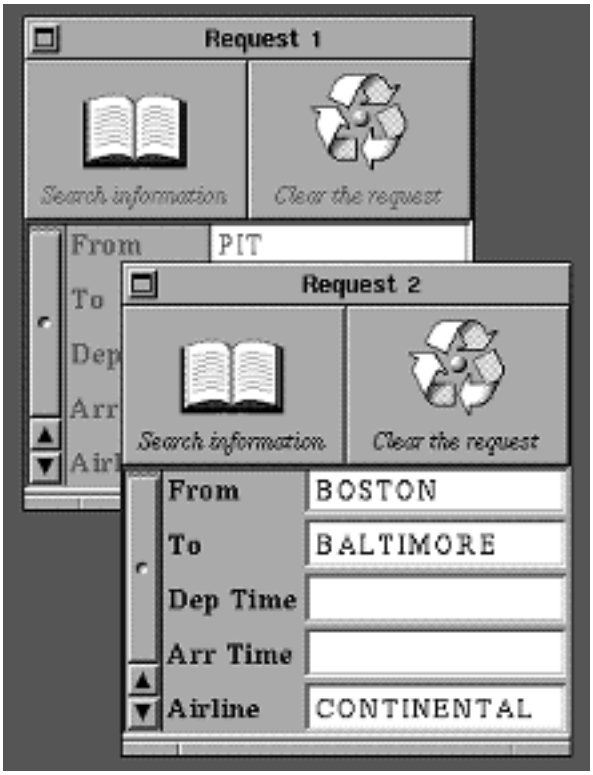


Figure 8.15 : Deux formulaires de requête à l'écran.

5. CONCLUSION

Les systèmes NoteBook et MATIS ont permis une étude sur l'architecture logicielle des systèmes à caractéristiques multiples ainsi que sur l'intégration des langages et dispositifs (mécanisme de fusion des creusets). Notre première réalisation, NoteBook, nous a permis de localiser les traitements liés aux différents dispositifs et langages au sein de PAC-Amodeus tandis que MATIS a été le siège de notre mécanisme de fusion.

Avec MATIS, nous avons montré que notre modèle d'architecture PAC-Amodeus fournit un cadre de réalisation pour le traitement en parallèle et la fusion de données en entrée. La fusion des données est effectuée au niveau du composant Contrôleur du Dialogue et s'appuie sur la hiérarchie d'agents PAC qui organise ce composant. Le moteur de fusion manipule des informations représentées dans un formalisme unique, le creuset. Par le biais d'exemples, nous avons montré que notre modèle offre aussi un support intéressant pour fournir un retour d'information immédiat. En effet un retour d'information peut être généré par chaque agent dans la hiérarchie. Ainsi il est possible de définir des retours d'information partiels et spécifiques au niveau d'abstraction, avant même que le mécanisme de fusion soit fini, pour une requête donnée.

MATIS offre de nombreux choix à l'utilisateur quant à l'usage des langages et des dispositifs (ULD). Nous souhaitons mener une expérimentation pour évaluer ces différents usages. Le scénario de l'expérimentation est fourni en Annexe A. Avant de procéder à l'expérimentation, nous envisageons de rajouter à l'écran une carte des Etats-Unis contenant les villes connues de la base de données. Cette carte sera gérée par un nouvel agent PAC, feuille de la hiérarchie. L'ajout de cette carte nous semble nécessaire pour inciter l'utilisateur à sélectionner les villes avec la souris et ainsi à spécifier des références déictiques.

A plus long terme, comme nous l'avons annoncé au chapitre VII, notre travail s'oriente vers l'intégration d'un réel modèle du dialogue au sein de l'architecture logicielle et vers les interfaces en sortie. Notre support pour ce travail sera le système MATIS. Par exemple, selon les stratégies de dialogue, les réponses de MATIS seront différentes (différents langages et dispositifs utilisés) et le comportement du moteur de fusion sera modifié

Chapitre IX

Conclusion

"*Ce que nous comprenons nous appartient.*
Notre histoire commence là... "
- Jean-Jacques Servan-Schreiber -

Contribution de la thèse

Ce travail contribue au domaine de l'ingénierie logicielle des interfaces utilisateur sous trois formes complémentaires : un espace de conception qui inclut le système et l'utilisateur ; une méthode de raisonnement et de classification des systèmes selon leurs utilisations ; un modèle d'architecture logiciel pour la réalisation des systèmes à caractéristiques multiples.

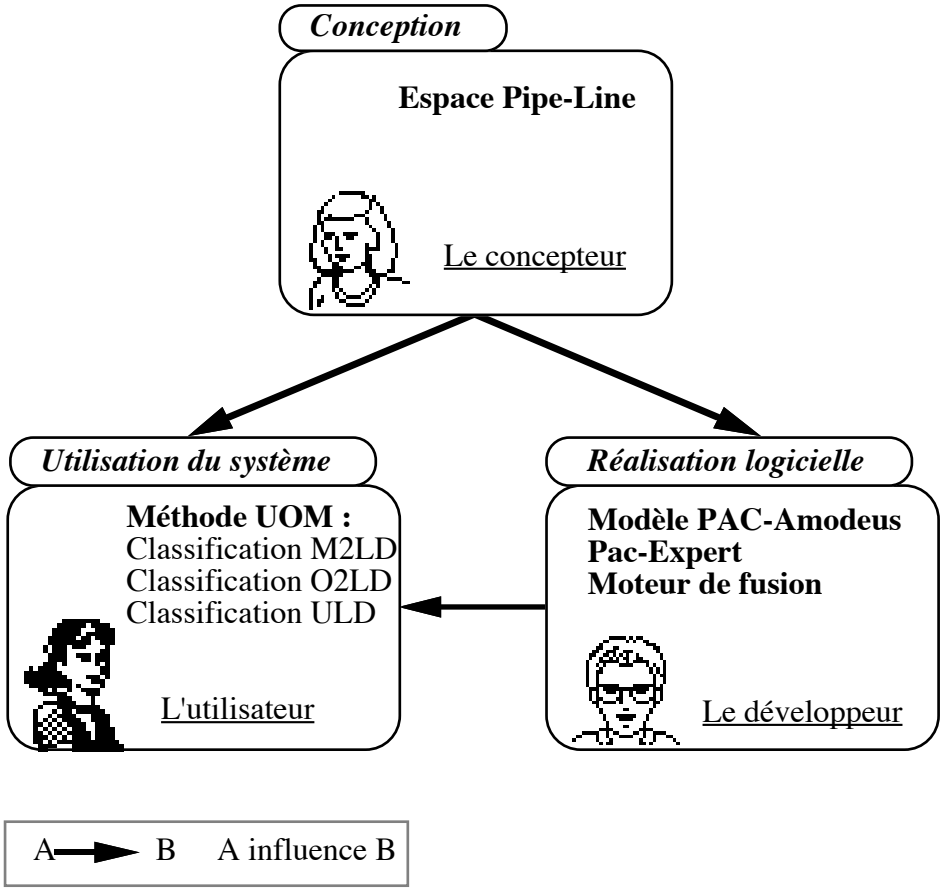
L'approche adoptée s'appuie sur une analyse conceptuelle des systèmes à caractéristiques multiples (Chapitres II et III). L'originalité de cette analyse tient à sa capacité à prendre en compte à la fois le système et l'utilisateur. Cette étude nous conduit à distinguer des étapes dans les mécanismes d'abstraction et de concrétisation et à montrer l'impact du contexte dans ces processus. Nous affinons la notion de contexte en le structurant en classes d'information. Au total, deux concepts caractérisant des interfaces émergent de cette analyse : les dispositifs physiques et les langages d'interaction. Nous organisons ces étapes et ces concepts en un canevas commun : l'espace pipe-line.

De l'espace pipe-line, canevas intégrateur des activités de l'utilisateur et du système, nous dérivons une méthode de raisonnement et de classification des systèmes en fonction de leurs utilisations. L'application de la méthode UOM (chapitre IV) permet de raisonner sur les caractéristiques de ces systèmes. La caractérisation d'un système selon UOM englobe à la fois des propriétés globales et statiques au système et des propriétés éphémères considérant un système à un instant particulier de son utilisation. Cet effort de conceptualisation et de classification des systèmes n'est pas à sous-estimer dans un domaine nouveau, en expansion rapide et dont la terminologie n'est pas encore fixée.

En adoptant comme fondement l'espace pipe-line, nous identifions les besoins pour la réalisation logicielle. Nous proposons un modèle d'architecture logicielle, PAC-Amodeus (Chapitres VI et VII) qui répond à ces besoins. Ce modèle est complété d'un guide d'application constitué d'un ensemble de règles heuristiques que nous avons concrétisées par un système expert PAC-Expert. En symbiose avec PAC-Amodeus, nous proposons un moteur de fusion des informations spécifiées par l'utilisateur. De part ses critères et sa technique de représentation des informations, le moteur est générique. PAC-Amodeus corrélé au moteur de fusion constitue une plate-forme pour la réalisation logicielle de systèmes à caractéristiques multiples. Bien que la plate-forme soit incomplète, nos travaux sont néanmoins novateurs.

Enfin nous avons prouvé la validité de nos résultats par la conception et la réalisation de deux systèmes : MATIS et NoteBook (au chapitre VIII).

L'approche de travail adoptée et les résultats se résument par le schéma suivant. Il comporte trois parties structurantes qui sont corrélées et dédiées aux trois intervenants du cycle de vie d'un système : le concepteur, le développeur et l'utilisateur.



Perspectives de développement

Un premier travail envisagé à court terme est le développement d'un système de démonstration de PAC-Amodeus et de son moteur de fusion : ce travail s'intègre dans le projet AMODEUS (Esprit BRA), dont le "A" signifie "Assey" ou transfert de connaissance. Nous envisageons de développer ce système sur Macintosh à l'aide du logiciel MacroMind Director.

Outre ce développement, nous entrevoyons pour nos travaux de multiples perspectives. Là encore, la dualité, conception et réalisation d'un système, apparaît dans nos propositions. Ces dernières s'organisent en deux parties : les extensions et les prolongements à plus long terme.

Les extensions

Au-delà des compléments évoqués au chapitre VII à propos du moteur de fusion, plusieurs travaux sont envisageables. L'objectif est de fournir une plate-forme pour la réalisation des systèmes à caractéristiques multiples qui réponde à tous les besoins de réalisation logicielle identifiée dans notre espace pipe-line:

Nouvelles règles heuristiques de construction de la hiérarchie d'agents

Les systèmes à caractéristiques multiples nous semblent générateurs de nouvelles règles de construction de la hiérarchie des agents PAC du Contrôleur de Dialogue. Cette étude est fortement corrélée à celle des interfaces en sortie. En effet les règles actuelles sont liées aux spécifications externes ou description de l'interface en sortie du système. Chaque règle identifiée implique une modification de PAC-Expert. D'autre part, il nous semble important de rendre PAC-Expert interactif en développant une interface approriée. Cette extension permettra d'accroître son utilisation, PAC-Expert étant pour l'instant un prototype utilisé par l'équipe uniquement. Pour cela, nous envisageons de remplacer la description textuelle par une description graphique des données en entrée.

Étude des interfaces en sortie

L'identification de nouvelles règles de construction doit passer par une analyse des interfaces en sortie. Notre étude actuelle sur les interfaces en sortie est restée à un niveau conceptuel et doit être approfondie. La réalisation de telles interfaces doit être abordée en prenant comme fondements l'espace pipe-line et la méthode UOM. En particulier le choix des langages et des dispositifs pour déterminer les informations en sortie (classification O2LD) nous semble particulièrement intéressant : les critères de choix, et sa réalisation au sein du modèle PAC-Amodeus. Nous estimons que nos résultats sur les interfaces en entrée peuvent être appliquées à celles en sortie : par exemple la structure de creuset peut être le véhicule dans le Contrôleur de Dialogue des concepts à présenter. Un moteur de fission peut alors déterminer plusieurs creusets à présenter. Le Composant Technique de Présentation choisit dynamiquement le ou les langages d'interaction et le Composant d'Interaction de Bas Niveau détermine le ou les dispositifs de sortie à piloter pour restituer le contenu des creusets.

Les prolongements

Nos perspectives s'organisent selon deux axes complémentaires : le premier axe de recherche concerne la formalisation des interfaces tandis que le deuxième concerne les modèles d'architecture et en particulier PAC-Amodeus.

Formalisation des interfaces

Une description en langage formel a pour objectif de vérifier des propriétés de conception. Nous avons identifié un ensemble de propriétés, telles que l'assignation, l'équivalence et la redondance, que nous souhaitons formaliser. Ayant établi le lien entre ces propriétés et la réalisation logicielle, une vérification formelle de ces propriétés nous permettrait de garantir des propriétés sur l'interface effective. Outre la vérification, une formalisation des propriétés évite les ambiguïtés d'une description en langue naturelle.

Modèles d'architecture

Plusieurs travaux sont envisagés autour du modèle PAC-Amodeus. En tout premier, nous souhaitons étudier le lien entre une description de la tâche (arbre de tâches) et la hiérarchie d'agents. Des études ont conduit à générer automatiquement l'interface d'un système à partir de l'analyse de la tâche. Or nous disposons d'un ensemble de règles qui permet de produire automatiquement la hiérarchie d'agents à partir d'une description de l'interface. En cela nous justifions l'approche qui consiste à produire la hiérarchie d'agents à partir de l'analyse de la tâche, en esquivant l'étape des spécifications externes. Établir ce lien entre l'analyse de la tâche et la réalisation logicielle en agents nous ouvrirait des perspectives interessantes : distinction totale entre les tâches et les langages et dispositifs qui permettent de réaliser ces tâches, et la garantie d'une interface orientée vers la tâche (interface cognitivement transparente).

Axé sur le modèle PAC-Amodeus, nous souhaitons étudier l'incorporation des modèles utilisateur et du dialogue. Aucune expérimentation n'a été encore conduite jusqu'ici : comment les intégrer au modèle, quelles sont leurs influences dans les mécanismes d'abstraction et de concrétisation.

Enfin et lié aux projets de l'équipe, nous souhaitons approfondir l'architecture des systèmes distribués et de réalité augmentée : dans le cadre du projet AMODEUS, nous avons abordé le problème de l'application de PAC-Amodeus à des systèmes distribués, tels qu'un éditeur de texte distribué et le média-space RAVE d'EuroPARC.

Annexe A

Scénario d'utilisation de MATIS

You are a famous author traveling to different cities to promote your newest book.

a) First, you will travel from Pittsburgh to Boston for a writers convention where you will give the keynote address at 12 am . You would prefer not to sleep in Dallas. A long trip is waiting for you.

b) On the same day you will fly to Dallas. You are due in Dallas, for an autograph session at the major book store from 8am to 12am.

c) Then the following day you are off to San Francisco to appear for a local talk show. The show airs at 9pm. So you want to make sure you get there on time. The show will finish around 12am.

d) Then you will return home to Pittsburgh from San Francisco as soon as possible to be relax after this exhausting trip !

Since you are a member of the USAir frequent traveler program, you will prefer to travel with USAir.

Fast facts about the Data Base:

Cities:	**Information:**
Atlanta	Flight numbers
Baltimore	Airlines
Boston	Inflight Meals
Dallas/Fort Worth	Diretc/non direct flights
Denver	Depart/Arrival time
Oakland Calif.	
Philadelphia	
Pittsburgh	
San Francisco	
Washington	

Annexe B

Spécification de MATIS en langage UAN

A titre indicatif, nous fournissons dans cette annexe un extrait de la spécification de MATIS en langage UAN [Hartson 92]. MATIS est complètement spécifié dans [Coutaz et al. 93e]. Sont présentés successivement l'arbre des tâches dérivé de la spécification et les trois tâches de planification d'un voyage. Celles-ci traduisent les différents langages et dispositifs mis à la disposition de l'utilisateur pour la définition d'une requête complète. Conséquemment les trois tâches décrites traduisent les choix offerts à l'utilisateur.

1. L'ARBRE DES TÂCHES

L'arbre de la Figure B-1 organise les tâches spécifiées en UAN. Chaque nœud correspond à une tâche. Les fils d'un nœud, noté N, sont les tâches apparaissant dans la colonne "Actions de l'utilisateur" ou la colonne "Retour d'information du système" de la spécification de N. Cet arbre est utile à la lisibilité et à la compréhension de la spécification . Il traduit la structure logique globale de la spécification.

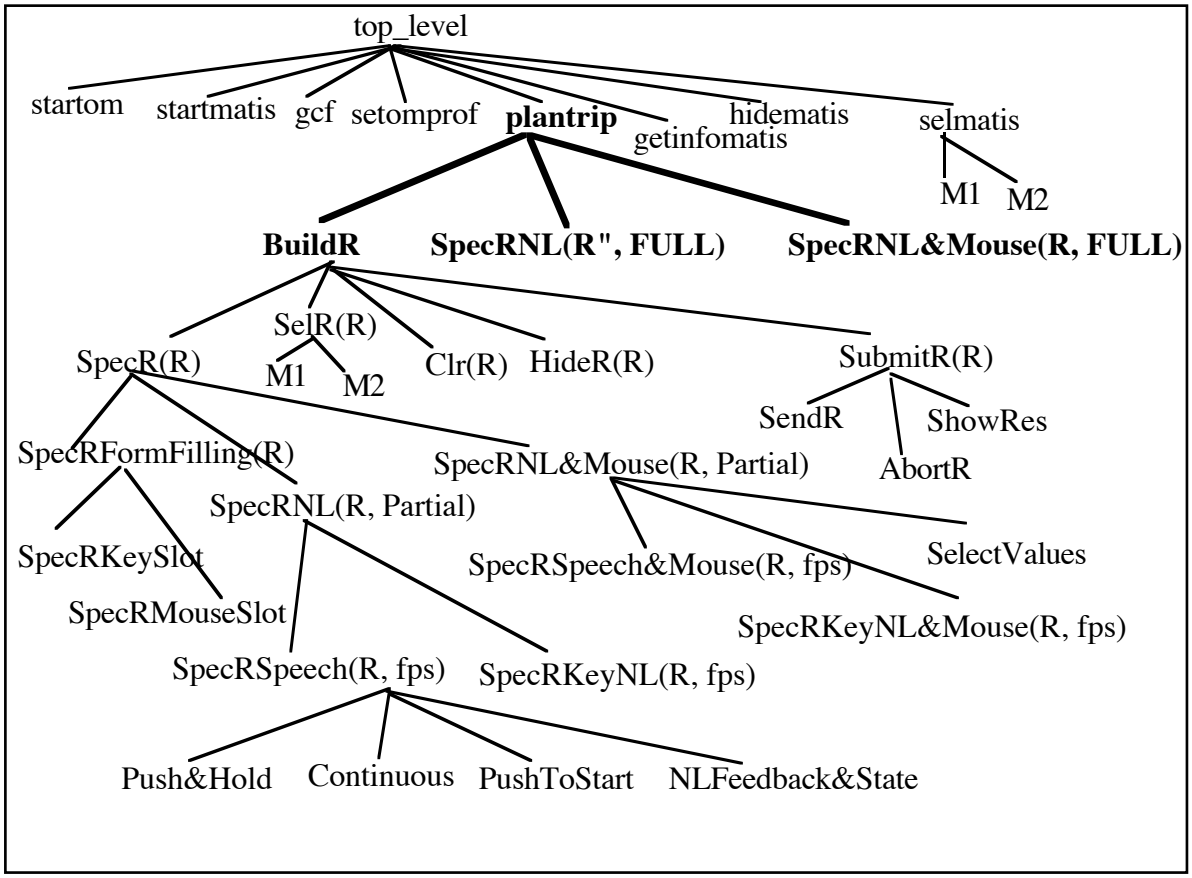


Figure B-1 : Arbre des tâches dérivé de la spécification UAN de MATIS.

Le paragraphe suivant contient la spécification en UAN de la partie en caractères gras de l'arbre de la Figure B-1.

2. SPÉCIFICATION UAN DE LA PLANIFICATION D'UN VOYAGE

Pour chaque tâche, la première colonne traduit les actions de l'utilisateur. La deuxième correspond aux retours d'information du système, conséquences des actions utilisateur. Enfin la dernière colonne décrit les états du système. Nous introduisons quelques opérateurs UAN utiles à la compréhension de la spécification des tâches ci-après, et en particulier les constructeurs de relations temporelles :

A B :	Séquence, la tâche B est exécutée après la tâche A.
A (t >n)B :	Délai, la tâche B est exécutée après un délai de n unités à partir de la fin de la tâche A.
(A|B)* :	Choix et répétition, choix de la tâche A ou B répété 0 ou n fois.
A & B :	Indépendance par rapport à l'ordre, les tâches A et B peuvent s'exécuter dans n'importe quel ordre.
A-->B :	Interruption dans un sens, la tâche A peut interrompre la tâche B.
A<-->B :	Interruption mutuelle, la tâche A peut interrompre la tâche B et vice versa.
A÷B :	Parallélisme, les tâches A et B peuvent s'exécuter en parallèle.

2.1 La tâche de planification d'un voyage

Task: PlanTrip

User Action	Interface Feedback	Interface State
SelAppli = MatisAppli : (SpecRNL (R, FULL) | SpecRNL&Mouse (R, FULL)) * → (PlanTripCT * ↔ (BuildR (R') 0-1 ↔ BuildR (R") 0-1)*)		

La spécification de requêtes en vue de planifier un voyage implique que MATIS soit l'application active (selAppli = MatisAppli).

Pendant la phase de construction d'une requête, toute tâche d'usage général (PlanTripCT, "*Convenience Tasks*") peut être exécutée, comme les services du Copier/Coller. Ces tâches peuvent être éventuellement exécutées ou exécutées plusieurs fois et se faisant de façon entrelacée avec la construction d'une requête. Ceci justifie la spécification (PlanTripCT * ↔).

Plusieurs requêtes peuvent éventuellement être construites en parallèle : (BuildR(R)$^{0\text{-}1}$ ↔ BuildR(R')$^{0\text{-}1}$). Et le nombre de requêtes spécifiées lors d'une même session n'est pas limité : (BuildR(R)$^{0\text{-}1}$ ↔ BuildR(R')$^{0\text{-}1}$)*.

Enfin la construction d'une ou plusieurs requêtes peut être interrompue par l'une des tâches atomiques suivantes : SpecRNL(R, FULL) ou SpecRNL&Mouse(R, FULL). SpecRNL(R, FULL) dénote la définition d'une requête complète en langage naturel écrit ou oral sans référence déictique. Au contraire SpecRNL&Mouse(R, FULL) désigne la définition d'une requête complète en langage naturel contenant des références déictiques spécifiées avec la souris.

2.2. La tâche de construction d'une requête en étapes successives

Task: BuildR (R)		
User Action	Interface Feedback	Interface State
((SelR (R) | ClrR (R) | HideR(R))* → SpecR (R)$^{+}$) SubmitR(R)		

BuildR implique la définition d'au moins une requête (SpecR (R)$^{+}$). Cette tâche peut être éventuellement interrompue par l'une des tâches atomiques suivantes : la sélection d'une autre requête, la mise à blanc de la requête ou la fermeture (icône) du formulaire de la requête ((SelR (R) | ClrR (R) | HideR(R))* →).

Une fois la requête spécifiée, elle peut être soumise à la base de données (Submit(R)).

2.3. La tâche de spécification d'une requête complète en langage naturel sans références déictiques

Task: SpecRNL (R, fps)		
User Action	Interface Feedback	Interface State
SpecRSpeech (R, fps) | SpecRKeyNL (R, fps)		

Le langage naturel peut être oral (SpecRSpeech) ou écrit (SpecRKeyNL). Le paramètre (fps) désigne une phrase de définition d'une requête complète ou incomplète, ou une phrase d'envoi d'une requête ("*Full, Partial, Show*"). Ces phrases ne sont pas accompagnées de références déictiques.

A titre d'exemple, nous décrivons la tâche (SpecRSpeech) :

Task: SpecRSpeech (R, fps) **is atomic**		
User Action	Interface Feedback	Interface State
SelDictionary = MatisDictionary: SelAppli = MatisAppli ∧ SelR = R: case		
SpeechMode = Push&Hold ~[x,y in OMEarW]∨ ProduceNonDeicticSentence (sentence, fps, 👄) ∧	OMEarW! *(Listening)* NLFeedback&State(sentence, 👄)	
SpeechMode = Continuous ProduceNonDeicticSentence (sentence, fps, 👄) ‖ TaskT*: TaskT ∈ Tasks ∧ ¬👄	 ($t > t_{silence}$) NLFeedback&State(sentence, 👄)	

SpeechMode = PushToStart ~[x,y in OMEarW]∨∧ ProduceNonDeicticSentence (sentence, fps, 👄) ‖ TaskT*: TaskT ∈ Tasks ∧ ¬👄 ~[x,y in OMEarW]∨∧ endcase		NLFeedback&State(sentence, 👄)

Pour spécifier une requête en langage naturel oral, MATIS doit être l'application active et le dictionnaire courant du système de reconnaissance doit être celui de MATIS (SelDictionary = MatisDictionary: SelAppli = MatisAppli).

Selon le mode d'utilisation du système de reconnaissance de la parole, la spécification du début et de la fin de la phrase articulée diffère. En correspondance avec les trois modes, trois cas sont spécifiés dans la première colonne.

Lorsque le mode est Push&Hold, l'utilisateur articule une phrase tout en laissant appuyé le bouton de la souris positionnée sur l'icône OMEarW. La sélection de cette icône en début de phrase provoque la modification de celle-ci qui contient alors une oreille (OMEarW! *(Listening))*. Cette nouvelle icône traduit le fait que le système de reconnaissance est en écoute.

Au contraire dans les deux autres modes, Continuous et PushToStart, une autre tâche peut être exécutée en parallèle de l'articulation d'une phrase. Cette tâche doit néanmoins être exécutée sans utiliser le microphone (TaskT*: TaskT ∈ Tasks ∧ ¬👄). Dans le mode Continuous, le système de reconnaissance se met à l'écoute après un silence ($t > t_{silence}$). Le silence constitue alors le marqueur de début et de fin de phrase. Dans les deux autres modes, la fin de la phrase est spécifiée par le relâchement du bouton de la souris.

NLFeedback&State(sentence, 👄) traduit le retour d'information pendant que le système de reconnaissance est en écoute, en cours d'analyse et à la fin de son traitement. Pendant l'analyse, le curseur de la souris pivote, l'icône du système de reconnaissance contient une montre et le texte "busy". A la fin du traitement la phrase est traduite en termes de champs dans la requête. La requête courante est complétée ou alors une nouvelle requête est créée : ceci modifie l'état du système.

2.4. La tâche de spécification d'une requête complète en langage naturel et références déictiques

Task: SpecRNL&Mouse (R, fps)		
User Action	Interface Feedback	Interface State
SpecRSpeech&Mouse (R, fps) | SpecRKeyNL&Mouse (R, fps)		

La tâche SpecRNL&Mouse décrit la spécification d'une requête en langage naturel accompagnée de références déictiques. Le langage naturel est oral (SpecRSpeech&Mouse) ou écrit (SpecRKeyNL&Mouse) tandis que les références déictiques sont spécifiées par manipulation directe en utilisant la souris. Nous décrivons par exemple la spécification d'une requête en langage naturel oral :

Task: SpecRSpeech&Mouse (R, fps) **is atomic**		
User Action	Interface Feedback	Interface State
SelAppli = MatisAppli ∧ SelR = R: case SpeechMode = Push&Hold ~[x,y in OMEarW]∨ ProduceDeicticSentence (sentence, fps, 👄) ∧ SelectValues	 OMEarW! *(Listening)* NLFeedback&State(sentence, 👄)	
SpeechMode = Continuous ProduceDeicticSentence (sentence, fps, 👄) ‖	 ($t > t_{silence}$) NLFeedback&State(sentence, 👄)	

SelectValues		
SpeechMode = PushToStart ~[x,y in OMEarW]∨∧ ProduceDeicticSentence (sentence, fps, 👄) ‖ SelectValues ~[x,y in OMEarW]∨∧	NLFeedback&State(sentence, 👄)	
endcase		

La tâche SpecRSpeech&Mouse est similaire à celle SpecRSpeech, mais implique au moins une référence déictique qui accompagne la phrase articulée.

Quand le mode d'utilisation est Push&Hold, la souris est utilisée à spécifier le début et la fin de phrase. Aussi la référence déictique à l'aide de la souris ne peut être spécifiée qu'après la fin de la phrase.

Au contraire dans les deux autres modes d'utilisation, la souris peut être utilisée en parallèle de l'articulation de la phrase pour spécifier une référence déictique. Il y a alors vrai parallélisme au niveau des actions de l'utilisateur (opérateur ||).

Annexe C

Langage naturel dans le contexte de NoteBook

Nous présentons les blocs syntaxiques recherchés lors de l'analyse d'une phrase reconnue en langage naturel dans NoteBook. L'Annexe C ne contient que le premier niveau dans l'arbre syntaxique. Chaque symbole non terminal présenté entre crochets est ensuite décrit jusqu'à l'obtention des symboles terminaux correspondant aux mots du dictionnaire (77 mots définis).

```
# ----------------------------------------
# Application : Notebook
# Subject :     slot-level networks
# Date :        22 June 1991
# Author :      Laurence Nigay
# ----------------------------------------

# only one task manager command is possible...
[START]
        ([OM_REQUEST])
        ([DIRECTGOTO])
        ([RELATIVEGOTO])
        ([SEMANTICGOTO])
        ([CLEARALL])
        ([DIRECTCLEAR])
        ([RELATIVECLEAR])
        ([SEMANTICCLEAR])
        ([NEW])

#----------------
[OM_REQUEST]
        (standby)

# -------------------------
# Absolute GOTO note
[DIRECTGOTO]
        ([cmdgo] [absolute_note])

[absolute_note]
        (THENOTE number [cardinal])
        (number [cardinal])
        (NOTE [cardinal])
```

```
        ([cardinal])
        (RANK NOTE)
        (RANK)

THENOTE
        (THE NOTE)
        (NOTE)

RANK
        (THE [ordinal])

# ----------------------------------------------------
# Relative GOTO note -> Forward or Previous
[RELATIVEGOTO]
        ([cmdmvt] [relative_depl])

[relative_depl]
        ([cardinal] NOTE)
        ([cardinal])

# ------------------------------------------
# Semantic GOTO note
# examples :    "Goto the note about a subject
[SEMANTICGOTO]

        ([cmdgo] NOTE about [subject_note])
        (what about [subject_note])

[subject_note]
        ([subject])

# ---------------------
# Clear all the notes
[CLEARALL]
        ([cmdclear_all])

# --------------------------
# Absolute CLEAR note
[DIRECTCLEAR]
```

```
        ([cmdclear] [absolute_note])

# ---------------------------------------------
# Relative CLEAR note
# examples :    "Clear the previous note"
#               "Clear the second previous note"
[RELATIVECLEAR]
        ([cmdclear] [relative_note])

[relative_note]
        (THE [ordinal] [way] NOTE)
        (THE [ordinal] [way])
        (THE  [way] NOTE)

# ---------------------------------------------
# Semantic Clear note
# examples :    "Clear the note about a subject

[SEMANTICCLEAR]
        ([cmdclear] NOTE about [subject_note])
        ([cmdclear] THE NOTE about [subject_note])

#------------------
# Create a new note
[NEW]
        ([create_a_note] NOTE)

#------------------
THE
        (the)

#------------------
NOTE
        (note)
        (page)
```

Annexe D

Langage naturel dans le contexte de MATIS

Nous présentons les blocs syntaxiques recherchés lors de l'analyse d'une phrase reconnue en langage naturel dans MATIS. Comme dans l'Annexe C, l'Annexe D ne contient que le premier niveau dans l'arbre syntaxique. Chaque symbole non terminal présenté entre crochets est ensuite décrit jusqu'à l'obtention des symboles terminaux correspondant aux mots du dictionnaire (581 mots définis).

```
# 0 explain fare_basis codes
   FUNCTION: ExplainFareCode
   NETS:
         [explain]
         [describe_fare_basis_code]
;

# 1 explain specified codes
   FUNCTION: ExplainCode
   NETS:
         [explain]
         [describe_code]
         [describe_heading]
;

# 2 list flight information
   FUNCTION: FlightInfo
   NETS:
         [list]
         [flight_fields]
         [flight_type]
         [ROUTE]
         [ARRIVE]
         [ARRIVE_DATE_RANGE]
         [ARRIVE_LOC]
         [ARRIVE_TIME_RANGE]
         [DEPART]
         [DEPART_LOC]
         [DEPART_TIME_RANGE]
         [DEPART_DATE_RANGE]
         [FLIGHT_NUM]
         [CLASS]
```

[NUM_STOPS]
[ONE_WAY]
[CHEAPEST]
[EARLIEST]
[duration]
[flight_anaph_pl]
[flight_anaph_sing]
[for_persons]
[on_aircraft]
[on_airline]
[price_range]
[with_connection]
[with_meal]
[with_restriction]
[with_stop]
;

 # 3 make a reservation
FUNCTION: Book
NETS:
[book]
[check_avail]
[flight_type]
[ROUTE]
[ARRIVE]
[ARRIVE_DATE_RANGE]
[ARRIVE_LOC]
[ARRIVE_TIME_RANGE]
[DEPART]
[DEPART_LOC]
[DEPART_TIME_RANGE]
[DEPART_DATE_RANGE]
[FLIGHT_NUM]
[CLASS]
[NUM_STOPS]
[ONE_WAY]
[CHEAPEST]
[EARLIEST]
[duration]

```
        [flight_anaph_pl]
        [flight_anaph_sing]
        [for_persons]
        [on_aircraft]
        [on_airline]
        [price_range]
        [with_connection]
        [with_meal]
        [with_restriction]
        [with_stop]
;

# 4 give distance between city and airport
  FUNCTION: Distance
  NETS:
        [distance]
        [city_airport]
;

# 5 show ground transportation
  FUNCTION: ListTransport
  NETS:
        [list]
        [transportation]
        [city_airport]
        [FARE]
        [DEPART_DATE_RANGE]
        [DEPART_TIME_RANGE]
;

# 6 show food service for flights
  FUNCTION: ListFoodService
  NETS:
        [list]
        [meal_served]
        [FLIGHTS]
        [DEPART_LOC]
        [ARRIVE_LOC]
;
```

```
# 7 give specs for aircraft
  FUNCTION: ShowAircraftSpec
  NETS:
       [list]
       [AIRCRAFT_SPEC]
       [FLIGHTS]
       [FLIGHT_NUM]
       [DEPART_LOC]
       [ARRIVE_LOC]
       [flight_anaph_sing]
       [flight_anaph_pl]
;

# 8 list airports for cities
  FUNCTION: AirportService
  NETS:
       [list]
       [airport_service]
;

# 9 list airport names
  FUNCTION: NameAirport
  NETS:
       [list]
       [name_airport]
       [FLIGHT_NUM]
       [land_in]
;

# 10 meals served on a flight
  FUNCTION: MealAvail
  NETS:
       [meal_avail]
       [FLIGHT_NUM]
       [DEPART_LOC]
       [DEPART_TIME_RANGE]
;
```

```
# 11 type of aircraft for flight
#   FUNCTION: AircraftForFlight
#   NETS:
#        [what_aircraft]
#        [FLIGHT_NUM]
#;

# 12 reset constraints
   FUNCTION: ResetConstraints
   NETS:
        [restart]
;

# 13 help for reservations
   FUNCTION: Help
   NETS:
        [help_reserve]
;

# 14 show cost of ground transportation
   FUNCTION: TransportCost
   NETS:
        [list]
        [ground_fare]
        [transportation]
        [DEPART_DATE_RANGE]
        [DEPART_TIME_RANGE]
;

# 15 time zone for city
   FUNCTION: TimeZone
   NETS:
        [list]
        [time_zone]
;
```

```
# 16 show stops for flights
   FUNCTION: ShowStops
   NETS:
        [list]
        [show_stops]
        [FLIGHTS]
        [DEPART_LOC]
        [ARRIVE_LOC]
;

# 17 aircraft_with_specs
   FUNCTION: WhichAircraft
   NETS:
        [which_aircraft]
        [aircraft_field]
;

# 18 list cities for airports
   FUNCTION: CityService
   NETS:
        [list]
        [city_service]
;

# 19 describe contents of table
   FUNCTION: ExplainTable
   NETS:
        [explain_table]
;

# 20 show restrictions for fares
   FUNCTION: ListRestrictions
   NETS:
        [list]
        [RESTRICTION]
        [FLIGHT_NUM]
        [DEPART_LOC]
        [ARRIVE_LOC]
;
```

```
# 21 list the days that a flight flies
   FUNCTION: FlightDays
   NETS:
        [list_days]
        [for_flight]
        [DEPART_LOC]
        [ARRIVE_LOC]
;

# 0 explain codes used in a table (under a heading)
   FUNCTION: explain_codes
   NETS:
        [explain]
        [explain_codes]
        [FLIGHT_NUM]
        [DEPART_LOC]
        [ARRIVE_LOC]
;
```

Références bibliographiques

[Abowd 92]
G. Abowd, J. Coutaz and L. Nigay, Structuring the Space of Interactive System Properties, Prooceedings of the IFIP TC2/WG2.7 Working Conference on Engineering for Human-Computer Interaction, édité par J. Larson et C Unger, Ellivuori, Finlande, (10-14 Août 1992), pp. 113-128.

[Alty 91]
J. L. Alty, Multimedia -- What is It and How do we Exploit It?, People and Computers VI, Proceedings of the HCI'91 Conference, édité par D. Diaper et N. Hammond, (20-23 Août 1991), pp. 31-44.

[Anderson 90]
D. P. Anderson, R. Govindan and G. Homsy, Abstractions for Continuous MEdia in a network window system, Technical report UCB/CSD 90/596, Computer Science Division (EECS), University of California, Berkeley, (Sept. 1990).

[André 93]
E. André, W. Finkler, W. Graf, T. Rist, A. Schauder, W. Wahlster, WIP: The Automatic Synthesis of Multimodal Presentations, Workshop ERCIM on Multimodal Human-Computer Interaction, Working Material, Nancy, France, (2-4 Novembre 1993).

[Apple 92]
Apple Computer Inc., Inside MAcintosh, Macintosh Toolbox Essentials, Apple Technical Library, Addison-Wesley Publ., (October 1992).

[Arens 93]
Y.Arens, E. Hovy, S. van Mulken, Structure and Rules in Automated Multimedia Presentation Planning, Proceedings IJCAI'93, Chambéry, France, (1993), pp. 1253-1259.

[Austin 62]
J. L. Austin, How to do thing with words, Oxford : Clarendon Press, (1962).

[Azémard 93]
F. Azémard, Interface multimodale : des concepts et des règles pour une interprétation contextuelle, actes de la conférence InforMatique'93, l'interface des mondes réels & virtuels, 2èmes journées internationales, Montpellier, France, (22-26 Mars 1993), pp.85-96.

[Baecker 87]
R.M. Baecker & W.A.S. Buxton, Readings in human computer interaction, Los Altos, California: Morgan Kaufmann Inc, (1987).

[Barnard 87]
P. Barnard, Cognitive Resources and the Learning of Computer Dialogs, in Interfacing Thought, Cognitive aspects of Human Computer Interaction, J.M. Carroll Ed., MIT Press Publ., pp. 112-158.

[Barnard 93]
P. Barnard, J. May, Real time blending of data streams: a key problem for the cognitive modelling of user behaviour with multimodal systems, UM/WP10, User Modelling, Working Paper 10, The Amodeus Project, Esprit Basic research Action 7040, (9 Juin 1993).

[Barthet 88]
M.F. Barthet, Logiciels interactifs et ergonomie, Ed. Dunod informatique, (1988).

[Bass 88]
L. Bass, E. Hardy, K. Hoyt, R. Little, R. Seacord, The Serpent run time architecture and dialogue model, Technical Report Carnegie Mellon University, CMU/SEI-88-TR-6, (Janvier 1988).

[Bass 90]
L. Bass, J. Coutaz, Developing Software for the User Interface, SEI Series in Software engineering, Addison-Wesley Publ., (1990), 256 pages.

[Beaudouin-Lafon 93]
M. Beaudoin-Lafon, L'usage de capteurs de contexte dans les systèmes interactifs, actes de IHM'93 Cinquièmes journées sur l'ingénierie des interfaces homme-machine, Lyon, (19-20 Oct. 1993), pp. 165-170.

[Bellik 92]
Y. Bellik, D. Teil, Multimodal Dialogue Interface, Proceedings of WWDU'92, Berlin, (1-4 Sept. 1992).

[Ben Amara 91]
H. Ben Amara, B. Peroche, H. Chappel et M. D. Wilson, Graphical interaction in a multimodal interface, Proceedings of the Annual Esprit Conference, Esprit'91, Bruxelles, (25-29 Novembre 1992), pp 303-321.

[Bernsen 93]
N. O. Bernsen, Taxonomy of HCI Systems: State of the Art, ESPRIT BR GRACE, deliverable 2.1, (1993).

[Blattner 90]
M. M. Blattner, R. B. Dannenberg, CHI'90 Workshop on multimedia and multimodal interface design, SIGCHI Bulletin,Volume 22, Number 2, (Octobre 1990), pp. 54-58.

[Blattner 92]
Editeurs M. M. Blattner, R. B. Dannenberg, Multimedia Interface Design, ACM Press, Frontier Series, Addison-Wesley Publ., (1992), 438 pages.

[Boehm 81]
B.W. Boehm, Sotfware Engineering Economics, Ed. Prentice-Hall, (1981).

[Bordegoni 93]
M. Bordegoni, M. Hemmje, An Interaction Model Based on Hand Gestures for 3D User Interfaces, Workshop ERCIM on Multimodal Human-Computer Interaction, Working Material, Nancy, France, (2-4 Novembre 1993).

[Bourguet 92]
M. L. Bourguet, Conception et réalisation d'une interface de dialogue personne-machine multimodale, thèse de l'Institut National Polytechnique de Grenoble, (1992), 206 pages.

[Bourguet 92 b]
M. L. Bourguet, ICPplan : dialogue multimodal pour la conception de plans architecturaux, 19e J.E.P., Bruxelles, (1992), pp 369-374.

[Brooke 80]
J. B. Brooke & K. D. Duncan, Flowcharts versus lists as aides to program debugging, Ergonomics, 23, (1980), pp. 387-399.

[Buxton 83a]
W.Buxton, Lexical and pragmatic considerations of input strcutures, Computer Graphics 17 (1), (1983), pp. 31-37.

[Buxton 83b]
W.Buxton, There's more to Interaction than meets the eye: Some Issues in Manual Input, D. A. Norman & Draper Ed., User Centered System Design, Lawrence Erlbaum, (1983), pp. 319-337.

[Buxton 90]
B. Buxton, Smoke and Mirrors, Byte, (Juillet 1990), pp. 202-251.

[Cadoz 92]
C. Cadoz, Le Geste de communication homme/machine la communication "Intrumentale", Rapport de Recherche ACROE RR 92-103, L.I.F.I.A., (Mars 1992), 36 pages.

[Card 83]
S. K. Card, T. P. Moran, A. Newell, The Psychology of Human-Computer Interaction, Chapter 2. The Human Information-Processor, Hillsdale, New Jersey : Lawrence Erlbaum Associates, (1986), pp. 23-97.

[Card 90]
S. K. Card, J.D. Mackinlay, G.G. Robertson, The Design Space of Input Devices, Proceedings of CHI'90, Seattle, Whashington, (Avril 1990), pp. 117-124.

[Cardelli 88]
L. Cardelli, Building User Interfaces by Direct Manipulation, Proc. of the ACM SIGGRAPH Symposium on User Interface Software, Banff, alberta, (Octobre 1988), pp. 152-166.

[Carlsen 91]
N. V. Carlsen, Modelling User Interface Software, thèse de l'Université Technique du Danemarque, département de la communication graphique, (1991), Vol. 1, 276 pages.

[Chabert 89a]
A. Chabert, N. Martin, L. Nigay, T. Peuzin, Architecture logiciel, Projet Mithra, version 1, DESS Génie Informatique, Université Joseph Fourier, (14 Avril 1989).

[Chabert 89b]
A. Chabert et L. Nigay, Expert data-base for drug interactions in Anesthesiology, Software Architecture Document, Release 1, University of Melbourne, School of Information Technology & Electrical Engineering (Juillet 1989).

[Cockton 91]
G. Cockton, The Architectural Bases of Design Re-Use, in Management and Design, Editeurs D. A. Duce, M. R. Gomes, F. R. A. Hopgood et J. R. Lee, Springer-Verlag Ed., (1991).

[Coutaz 85]
J. Coutaz, A Layout Abstraction for User System Design, ACM SIGCHI, (Janvier 1985), pp. 18-24.

[Coutaz 87]
J. Coutaz, PAC: an Implementation Model for Dialog Design, Proceedings of Interact'87, H-J. Bullinger, B. Shackel ed., North Holland, Stuttgart, (Sept. 1987), pp. 431-436.

[Coutaz 90]
J. Coutaz, Interface homme-ordinateur : conception et réalisation, Ed. Dunod Informatique, (1990), 455 pages.

[Coutaz 91]
J. Coutaz, S. Balbo, Applications: A Dimension Space for User Interface Management Systems, Proceedings of CHI'91, New Orleans, Louisiana, (27 Avril -2 Mai 1991), pp. 27-32.

[Coutaz et al. 91b]
J. Coutaz, L. Nigay, M. Harrison, G. Abowd, Design Scenarios for M1.5, RP2/WP11, Research Package 2, Working Paper 11, The Amodeus Project, Esprit Basic research Action 3066, (Mars 1991), 20 pages.

[Coutaz&Nigay 91]
J. Coutaz et L. Nigay, Seeheim et Architecture Multi-agent, actes de IHM'91 Troisièmes journées sur l'ingénierie des interfaces homme-machine, Dourdan, (11-13 Déc. 1991), pp. 89-98.

[Coutaz 92]
J. Coutaz,, Software Architecture Modeling for User Interfaces, Encyclopedia of Software Engineering, (1992), 33 pages.

[Coutaz 93a]
J. Coutaz, L. Nigay, D. Salber, The MSM Framework for Multi-Sensory-Motor Systems, SM/WP4, System Modelling, Working Paper 4, The Amodeus Project, Esprit Basic research Action 7040, (Janvier 1993).

[Coutaz 93b]
J. Coutaz, L. Nigay, D. Salber, Taxonomic Issues for Multimodal and Multimedia Interactive Systems, Workshop ERCIM on Multimodal Human-Computer Interaction, Working Material, Nancy, France, (2-4 Novembre 1993).

[Coutaz 93c]
J. Coutaz, L. Nigay, D. Salber, The MSM framework: A Design Space for Multi-Sensori-Motor-Systems, Lecture Notes in Computer Science, L. Bass, J. Gornostaev, C. Under Editors, Human-Computer Interaction, EWCHI'93, East/West Human Computer Interaction, Selected Papers, Springer-Verlag, Moscou, (Août 1993), pp. 231-241.

[Coutaz 93d]
J. Coutaz, F. Paterno, G. Faconti, L. Nigay, A Comparison of Approaches for Specifying MultiModal Interactive Systems, Workshop ERCIM on Multimodal Human-Computer Interaction, Working Material, Nancy, France, (2-4 Novembre 1993).

[Coutaz 93e]
J. Coutaz, G. Faconti, L. Nigay, F. Paterno, D. Salber, MATIS: a UAN Description and Lesson Learned, System, SM/WP14, System Modelling, Working Paper 10, The Amodeus Project, Esprit Basic research Action 7040, (28 Juin 1993), 35 pages.

[Coutaz 93f]
J. Coutaz, D. Salber, S. Balbo, Towards Automatic Evaluation of Multimodal User Interfaces, Journal on Knowledge-Based Systems, special issue on intelligent user interfaces, (1993).

[Crowley 94]
J. L. Crowley et J. M. Bedrune, Integration and Control of Reactive Visual Processes, 1994 European Conference on Computer Vision, (ECCV-'94), Stockholm, (mai 1994).

[Cypher 91]
A. Cypher, Video Presentation EAGER: Progamming Repetitive Tasks by Example, Prooceedings of CHI'91 Human Factors on Computing Systems, News Orleans, ACM Press, (Avril 27-Mai 2 1991), pp. 445-446.

[DeMarconnay 93]
P. De Marconnay, J. L. Crowley, D. Salber, Visual Interpretation of Faces in the NEIMO Multi-Modal HCI Test-bed, Proceedings of IJCAI'93, Chambéry, (Septembre 1993).

[Demazeau 92]
Y. Demazeau et J. Ferber, Systèmes multi-agents, Quatrièmes Journées Nationales du PRC GDR Intelligence Artificielle, Marseille, (19-21 Octobre 1992).

[Dix 91]
A. Dix, Formal Methods for Interactive Systems, Academic Press Publ., (1991).

[Dix 93]
A. Dix, J. Finlay, G. Abowd and R. Beale, Human-Computer Interaction, Prentice Hall Publ., (1993), 570 pages.

[Duce 90]
Editeurs D.A. Duce, M.R. Gomes, F.R.A. Hopgood, J.R. Lee, User Interface Management and Design, Proceedings of the Workshop on User Interface Management Systems and Environments, Springer, (4-6 Juin 1990).

[Duke 92]
D. Duke, M. Harrison, Abstract Models for Interaction Objects, SM/WP1, System Modelling, Working Paper 1, The Amodeus Project, Esprit Basic research Action 7040, (Novembre 1992).

[Egeria 90]
Egeria Manual, BULL, Centre de recherche de Sophia Antipolis, (1990).

[Faconti 93]
G.P. Faconti, Towards the Concept of Interactor, SM/WP8, System Modelling, Working Paper 8, The Amodeus Project, Esprit Basic research Action 7040, (17 Février 1993), 23 pages.

[Faure 93]
C Faure et L. Julia, Interaction homme-machine par la parole et le geste pour l'édition de documents : TAPAGE, actes de la conférence InforMatique'93, l'interface des mondes réels & virtuels, 2èmes journées internationales, Montpellier, France, (22-26 Mars 1993), pp.171-180.

[Farallon 87]
Farallon Computing Inc., le système Timbuktu, (1987).

[Feiner 91]
S.K. Feiner, K.R. McKeown, COMET: Generating Coordinated Multimedia Explanations, Prooceedings of CHI'91 Human Factors on Computing Systems, News Orleans, ACM Press, (Avril 27-Mai 2 1991), pp. 449-450.

[Foley 84]
J.D. Foley, V.L. Wallace, P.Chan, The Human Factors of computer Graphics interaction techniques, IEEE computer Graphics and Applications, 4(11), (1984), pp. 13-48

[Foley 84]
J.D. Foley, C. Gibbs, W. C. Kim, S. Kovacevic, A Knowledge-based User Interface Management System, Proceedings of CHI'88, New York, U.S.A., (1988), pp. 67-72.

[Frohlich 91]
D. M. Frohlich, The Design Space of Interfaces, Multimedia Systems, Interaction and Applications, Proceedings of 1st Eurographics Workshop, Stockholm, Sweden (Avril 18/19,1991), Springer Verlag, pp. 53-69.

[Gargan 88]
R. A. Gargan, J. W. Sullivan & S. W. Tyler, Multimodal resposne planning: an adaptative rule based approach, Proceedings of CHI'88, New York, U.S.A., (1988), pp. 229-234.

[Gaver 86]
W. W. Gaver, Auditory icons: Using sound in computer interfaces, Human Computer Interaction, 2, (1986), pp. 167-177.

[Gaver 89]
W. W. Gaver, The SonicFinder, An Interface That Uses Auditory Icons, Human Computer Interaction, 4,1, (1989).

[Giarratano 88]
J. C. Giarratano, CLIPS User's Guide, Version 4.2 of CLIPS, Volume 2 of 3, Artificial Intelligence Section, Lyndon B. Johnson Space Center, (3 Juin 1988), 170 pages.

[Goldberg 84]
A. Goldberg, Smalltalk-80, The Interactive Programming Environment, Ed. Addison-Wesley, (1984).

[Gourdol 91]
A. Gourdol, Architecture des Interface Homme-Machine multimodales, mémoire de DEA Informatique, IMAG, Grenoble, (Juin 1991).

[Gourdol 92]
A. Gourdol, L. Nigay, D. Salber, J. Coutaz, Two case Studies of Software Architecture for Multimodal Interactive Systems: VoicePaint and a Voice-enabled Graphical Notebook, Prooceedings of the IFIP TC2/WG2.7 Working Conference on Engineering for Human-Computer Interaction, édité par J. Larson et C Unger, Ellivuori, Finlande, (10-14 Août 1992), pp. 271-283.

[Guastello 89]
S. J. Guastello & M. Traut, Verbal versus pictorial representations of objects in a human-computer interface, International Journal Man-machine Studies, 31, (1989), pp.99-120.

[Harrison 90]
Editeurs M. D. Harrison and H. W. Thimbleby, Formal Methods in Human-Computer Interaction, Cambridge University Press, (1990).

[Hartson 92]
H. R. Hartson et P.D. Gray, Temporal aspects of tasks in the User Action Notation, Human-Computer Interaction, Vol. 7, (1992), pp. 1-45.

[Hemslev 47]
L. Hemslev, Structural Analysis of language, Studia Phonetica, vol. 1, pp. 69-78.

[Henry 90]
T. R. Henry, S. E. Hudson, G. L. Newell, Integrating gesture and Snapping into a User Interface Toolkit, Technical Report, TR 90-15, University of Arizona, Tucson Arizona, Department of computer science, (Mai 1990), 12 pages.

[Hill 92]
R.D. Hill, The Abstraction-Link-View Paradigm: Using Constraints to Connect User Interfaces to Applications, Proceedings of CHI'92, Monterey, California, (3-7 Mai 1991), pp. 335-342.

[Hoare 85]
C. A. R. Hoare, Communicating Sequential Processes, Prentice Hall International Publ., (1985).

[Hutchins 86]
E. L. Hutchins, J. D. Hollan and D. A. Norman, Direct Manipulation Interfaces, User Centered System Design, New Perspectives on Computer Interaction, édité par D. A. Norman, S.W. Draper, Hillsdale, New Jersey : Lawrence Erlbaum Associates, (1986), pp. 87-124.

[IHMM 91]
Pôle Interface Homme-Machine Multimodale du PRC Communication Homme-Machine, Ed. J. Caelen et J. Coutaz, (Janvier 1991).

[IHM 92]
Atelier IHM'92, ENST, Interfaces Multimodales et Architecture Logicielle, (30 Nov.-2 Déc. 1992).

[Jacob 91]
R. J. K. Jacob, The use of eye movements in human-computer interaction techniques: What you look at is what you get, ACM Transactions on Information Systems, 9, 2, (Avril 1991), pp. 152-169.

[Jambon 94]
F. Jambon, J. Coutaz, Retours d'Informations sensori-moteurs des dispositifs physiques : Tâche de sélection avec une souris sans clic, proposition de communication affichée pour IEA'94, Toronto.

[Jaulent 89]
P. Jaulent, SADT. Un langage pour communiquer, Ed. Eyrolles Informatique, (1989), 344 pages.

[Johnson 91]
H. Johnson & P. Johnson, Task Knowledge Structures : Psychological basis and integration into system design, Acta Psychologica, (1991), pp. 3-26.

[Johnson 93]
P. Johnson, S. Wilson, P. Markopoulos et J. Pycok, ADEPT - Advanced Design Environment for Prototyping with Task Models, Proceedings of InterCHI'93, Amsterdam, The Netherlands, (24-29 Avril 1993), pp. 56.

[Kantowitz 85]
B. H. Kantowitz, Channels and stages in human information processing, Journal of Mathematical Psychology, 29, (1985), pp. 135-174.

[Kass 89]
R. Kass, Student Modeling in Intelligent Tutoring Systems - Implications for User Modeling, User Models in Dialog Systems, A. Kobsa, W. Wahlster Eds, Springer Verlag, (1989), pp. 386-410.

[Kobsa 89]
A. Kobsa, A Taxonomy of Beliefs and Goals for User Models in Dialog Systems, User Models in Dialog Systems, A. Kobsa, W. Wahlster Eds, Springer Verlag, (1989), pp. 52-68.

[Krasner 88]
G. Krasner, S. Pope, A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80, Journal of Object Oriented Programming, (Août/Sept. 1988), pp. 26-49.

[Krueger 91]
M.W. Krueger, Artificial Reality II, Addison Wesley, (1991).

[Kuijpers 92]
E. Kuijpers, M. D. Wilson, A multi-modal interface for man machine interaction with knowledge based systems-MMI, Proceedings of the EWHCI'92, East-West International conference on Human-Computer Interaction, St. Petersburg, Russie, (4-8 Août 1992), pp. 373-378.

[Kuutti 93]
K. Kuutti and L. J. Bannon, Searching for unity among diversity: exploring the "interface" concept, Proceedings of INTERCHI'93, Amsterdam, The Netherlands, (24-29 Avril 1993), pp. 263-268.

[Laurel 86]
B. K. Laurel, Interface as Mimesis, User Centered System Design, New Perspectives on Computer Interaction, édité par D. A. Norman, S.W. Draper, Hillsdale, New Jersey : Lawrence Erlbaum Associates, (1986), pp. 67-85.

[Lefebvre 92]
P. Lefebvre, F. Poirier, G. Duncan, A Multimodal Approach to Man-Machine Dialogue with the Unix Operating System, Prooceedings of the IFIP TC2/WG2.7 Working Conference on Engineering for Human-Computer Interaction, édité par J. Larson et C Unger, Ellivuori, Finlande, (10-14 Août 1992), pp. 285-294.

[Lunati 91]
J. M. Lunati and A. I. Rudnicky, Spoken Language Interfaces: The OM system, Prooceedings of CHI'91 Human Factors on Computing Systems, News Orleans, ACM Press, (Avril 27-Mai 2 1991), pp. 453-454.

[Luzzati 89]
D. Luzzati, Recherches sur le dialogue homme-machine, modèles linguistiques et traitements automatiques, thèse d'état de l'Université de la Sorbonne Nouvelle, (1989).

[Mackinlay 90]
J.Mackinlay, S. Card, G. Robertson, A Semantic Analysis of the Design Space of Input Devices, Human Computer Interaction, Lawrence Erlbaum, Vol. 5, No 2 & 3, (1990), pp. 145-190.

[Mann 88]
W.C. Mann, S.A. Thomson, Rhetorical Structure Theory: towards a Functional Theory of Text organization, Text 8(3), (1988), pp. 243-281.

[Marmolin 91]
H. Marmolin, Multimedia from the Perspectives of Psychology, Multimedia systems and applications, First Eurographics workshop, Stockholm, Suède, (18-19 Avril 1991), Springer-Verlag, chapitre 4, pp. 39-52.

[Martin 89]
N. Martin, A help system for Solo, rapport de stage DESS, Herriot-Watt University, Edinburg, (Juin-Août 1989).

[Martin 92]
J.C. Martin et D. Béroule, Eléments de modélisation connexionniste pour l'interaction homme-machine multimodale, actes de IHM'92 Quatrièmes journées sur l'ingénierie des interfaces homme-machine, Paris, (30 Nov., 1 et 2 Déc. 1992), pp. 43-48.

[Martin 93]
J.C. Martin et D. Béroule, Types et buts de coopérations entre modalités, actes de IHM'93 Cinquièmes journées sur l'ingénierie des interfaces homme-machine, Lyon, (19-20 Oct. 1993), pp. 17-22.

[Mayes 90]
J. T. Mayes, The M-World: Multimedia Interfaces and their Role in Interactive Learning Systems, Multimedia Interfaces for education, édité par A. Edwards et S. Holland, Italy, Proccedings of the NATO Advanced Research Workshop, (1990).

[McCall 77]
J. McCALL, Factors in Software Quality, General Electric Ed., (1977).

[McDermid 84]
J. McDermid & K. Ripkin, Life Cycle Support in the ADA environment, Cambridge University Press, (1984).

[Mignot 93]
C. Mignot, C. Valot, N. Carbonell, An Experimental Study of Future 'Natural' Multimodal Human-Computer Interaction, Adjunct Proceedings of InterCHI'93, Amsterdam, The Netherlands, (24-29 Avril 1993), pp. 67-68.

[Moran 81]
T.P. Moran, The Command Language Grammar: a representation for the user interface of interactive computer systems, Internal Journal of Man-Machine Studies, No 15, (1981), pp. 3-50.

[Myers 88]
B. A. Myers, Creating User Interfaces by Demonstration, Academic Press, Boston, 1988.

[Myers 89]
B. A. Myers, Tools for Creating User Interfaces: An Introduction and Survey, IEEE Software 6 (1), et CMU-CS-89-107 (1989), pp. 15-23.

[Myers 90]
B. A. Myers, A New Model for handling Input, ACM Transactions on Information Systems, Vol. 8. Np. 3, (Juillet 1990), pp. 289-320.

[Nanard 84]
J. et M. Nanard, Manipulation Interactive de Documents, Techniques et Science Informatiques, vol.3, 6, (1984), pp. 443-451.

[Nanard 90]
J. Nanard, La manipulation directe en interface homme-machine, thèse de Doctorat d'Etat, Université de Montpellier II, (Déc. 1990).

[Neal 88]
J. G. Neal, Z. Dobes, K. E. Bettinger et J. S. Byoun, Multi-Modal References in Human-Computer Dialogue, The Seventh National Conference on Artificial Intelligence, Saint Paul, Minesota, vol. 2, (Août 1988).

[Neal 91]
J. G. Neal et S. C. Shapiro, Intelligent Multi-Media Interface Technology, Intelligent User Interfaces, édité par J. W. Sullivan et S. W. Tyler, ACM Press, Frontier Series, Addison-Wesley Publ., (1991), pp. 11-43.

[NeXTstep 91]
NeXTstep Reference, Addison-Wesley Publ., (1991).

[Nielsen 93]
J. Nielsen, Noncommand user interfaces, Communications of the ACM, Vol. 36, No 4, (Avril 1993), pp. 83-99.

[Nigay 90]
L. Nigay, Modélisation des architectures logicielles des systèmes interactifs, Rapport de DEA d'Informatique de l'Université Joseph Fourier, Grenoble, (27 Juin 1990), 127 pages.

[Nigay 90b]
L. Nigay et J. Coutaz, Poste de Composition, Cahier des charges, Version 2.2, (19 Juillet 1990), 52 pages.

[Nigay 91a]
L. Nigay, An example of Multimodal Interative System: a Voice Enabled Note Book, rapport technique, Octobre 1991, 31 pages.

[Nigay 91b]
L. Nigay, J. Coutaz, Building User Interfaces: Organizing Software Agents, Proceedings of the Annual Esprit Conference, ESPRIT'91, (25-29 Nov. 1991), Bruxelles, pp. 707-719.

[Nigay 92a]
L. Nigay et J. Coutaz, Interfaces multimodales et architecture : Fusion et parallélisme, actes de IHM'92 Quatrièmes journées sur l'ingénierie des interfaces homme-machine, Paris, (30 Nov., 1 et 2 Déc. 1992), pp. 29-36.

[Nigay 92b]
L. Nigay et J. Coutaz, Projet Palas-X, Rapport d'activité de consulting auprès de la société Sextant Avionique, , (Juin 1992), 10 pages.

[Nigay 92c]
L. Nigay et J. Coutaz, PAC-Expert: Towards an Automatic Generation of Dialogue Controllers, Rapport Technique LGI, Institut IMAG, RT 83, (Mai 1992), 26 pages.

[Nigay 92d]
L. Nigay et J. Coutaz, PAC-Expert: Towards an Automatic Generation of Dialogue Controllers, Research Package 2, Working Paper 17, The Amodeus Project, Esprit Basic research Action 3066, (Février 1992), 24 pages.

[Nigay 93a]
L. Nigay, J. Coutaz, Espace problème, fusion et parallélisme dans les interfaces multimodales, actes de la conférence InforMatique'93, l'interface des mondes réels & virtuels, 2èmes journées internationales, Montpellier, France, (22-26 Mars 1993), pp.67-76.

[Nigay 93b]
L. Nigay, J. Coutaz, A design space for multimodal interfaces: concurrent processing and data fusion, Proceedings of InterCHI'93, Amsterdam, The Netherlands, (24-29 Avril 1993), pp. 172-178.

[Nigay 93c]
L. Nigay, MATIS un système multimodal d'information sur les transports aériens, Journée du pôle Interfaces Multimodales du PRC Communication Homme-Machine, Lyon, (18 Octobre 1993).

[Nigay 93d]
L. Nigay, J. Coutaz, S. Salber, MATIS: A Multimodal Airline Travel Information System, SM/WP10, System Modelling, Working Paper 10, The Amodeus Project, Esprit Basic research Action 7040, (23 Février 1993), 9 pages.

[Norman 86]
D. A. Norman, Cognitive Engineering, User Centered System Design, New Perspectives on Computer Interaction, édité par D. A. Norman, S.W. Draper, Hillsdale, New Jersey : Lawrence Erlbaum Associates, (1986), pp. 31-61.

[Normand 92]
V. Normand, Le modèle SIROCO : de la spécification conceptuelle des interfaces utlisateur à leur conception, thèse de l'Université Joseph Fourier de Grenoble, (1992), 258 pages.

[Olsen 87]
D. R. Olsen, et al., ACM SIGGRAPH workshop on software tools for user interface management, Computer Graphics 21(2), (1987), p. 71-147.

[Olsen 89]
D. R. Olsen, A Programming Language Basis for User Interface Generator, Proceedings of CHI'89, special issue of the SIGCHI Bulletin, ACM Press, (1989), pp. 171-176.

[OSF/MOTIF 90]
OSF/MOTIF, Programmer's guide, Revision 1.0, Open Software Foundation, Prentice Hall Publ., (1990).

[Palanque 92]
P. Palanque, Modélisation par Objects Coopératifs Interactifs d'interfaces homme-machine dirigées par l'utlisateur, thèse de l'Université Toulouse 1, (1992), 357 pages.

[Palmiter 1991]
S. Palmiter, J. Elkerton and P. Baggett, Animated demonstrations verse written instructions for learning procedural tasks, International Journal of Man-Machine Studies, 34, pp. 687-701.

[Payne 86]
S. J. Payne et T. R. G. Green, Task-Action Grammars: a model of the mental representation of task languages, Human-Computer Interaction, Vol. 2, No. 2, Lawrence Erlbaum Associates Pub., (1986).

[Petoud 89]
I. Petoud, Y.Pigneur, Des Spécifications fonctionnelles à l'interface homme-utlisateur sans programmation, Premier Colloque sur l'Ingénierie des Interfaces Homme-Machine, Sophia Antipolis, (Mai 1989).

[Petzold 93]
C. Petzold, Programmer sous windows 3.1, Collection "Programmation", Microsoft Press, (1993), 1008 pages.

[Pfaff 85]
User Interface Management Systems, G.E. Pfaff ed., Eurographics Seminars, Springer Verlag, 1985.

[Pittman 91]
J. A. Pittman, Recognizing Handwritten Text, Proceedings of CHI'91, New Orleans, Louisiana, (27 Avril -2 Mai 1991), pp. 271-275.

[Plossu 93]
S. Plossu, Modèles d'utilisateur pour interfaces homme-machine multimodales, Rapport de DEA d'Informatique de l'Université Joseph Fourier, Grenoble, (23 Juin 1993), 179 pages.

[Poirier 93]
F. Poirier et P. Lefebvre, Une interface multimodale de dialogue homme-machine sous Unix, actes de la conférence InforMatique'93, l'interface des mondes réels & virtuels, 2èmes journées internationales, Montpellier, France, (22-26 Mars 1993), pp.161-170.

[Pountain 93]
D. Pountain, Track People With Active Badges, Byte, (Décembre 1993), pp. 57-64.

[Rasmussen 86]
J. Rasmussen, Information processing and human-machine interaction : an approch to cognitive engineering, North-Holland series in system science and engineering, A.P. Sage Ed., (1986).

[Rich 89]
Rich, Stereotypes and User Modeling, User Models in Dialog Systems, A. Kobsa, W. Wahlster Eds, Springer Verlag, (1989), pp. 35-51.

[Romary 93]
L. Romary, B. Gaiffe et J-M. Pierrel, Multimodality: why and how?, Workshop ERCIM on Multimodal Human-Computer Interaction, Working Material, Nancy, France, (2-4 Novembre 1993).

[Royce 70]
W.W. Royce, Managing the development of large software systems, Proceedings WESTCON, Californie, USA, (1970).

[Senach 90]
B. Senach, Evaluation de l'ergonomie des interfaces homme-machine : du prototypage aux systèmes experts, actes de la conférence ERGO.IA'90, Biarritz, France, (19-21 Septembre 1990), pp. 85-106.

[Sellen 92]
A. Sellen, G.P. Kurtenbach, W. Buxton, The Prevention of Mode errors Through Sensory Feedback, Human Computer Interaction, Lawrence Erlbaum, Vol.7, No 2, (1992), pp. 141-164.

[Smart 93]
W. D. Smart, A. Cobley, I. W. Ricketts, A. Y. Cairns, Pratical Multimodality, Workshop ERCIM on Multimodal Human-Computer Interaction, Working Material, Nancy, France, (2-4 Novembre 1993).

[Schmucker 86]
K. Schmucker, MacApp, An Application Framework, Byte, Vol. 11, No. 8, (1986), pp. 189-193.

[Salber 93]
D. Salber, J. Coutaz, A Wizard of Oz Platform for the Study of Multimodal Systems, Adjunct Proceedings of InterCHI'93, Amsterdam, The Netherlands, (24-29 Avril 1993), pp. 95-96.

[Scapin 90]
D. L. Scapin, C. Pierret-Golbreich, Towards A Method For Task Description: MAD, Work with Display Units 89, L. Berlinguet and D. Berthelette Editors, Elsevier Science Publishers B.V. (North Holland), (1990).

[Sutcliffe 93]
A. Sutcliffe et P. Faraday, Designing Multimedia Interfaces, Lecture Notes in Computer Science, L. Bass, J. Gornostaev, C. Under Editors, Human-Computer Interaction, EWCHI'93, East/West Human Computer Interaction, Selected Papers, Springer-Verlag, Moscou, (Août 1993), pp. 104-114.

[Takebayashi 92]
Y. Takebayashi, Integration of understanding and synthesis functions for multimedia interfaces, in Multimedia Interface Design, M. Balttner and B. Dannenberg Ed., ACM Press, (1992), pp. 233-256.

[Tanner 83]
P. Tanner, W. Buxton, Some Issues in Future User Interface Management Systems (UIMS) Development, IFIP Working Group 5.2 Workshop on User Interface Management, Seeheim, (novembre 1983).

[Tarby 93]
J.C. Tarby, Gestion Qutomatique du Diqlogue Homme-Machine à partir de Spécifications Conceptuelles, thèse de l'Université TOULOUSE I, (1993), 279 pages.

[Tauber 90]
M.J. Tauber, ETAG: Extended Task-Action Grammar - A language for the description of the user's task language, in Proceedings of INTERACT'90, D. Diaper et al. Ed., Elsevier Science Publishers, North-Holland, (1990).

[Thimbleby 90]
H. Thimbleby, User Interface Design, ACM Press, Frontier Series, Addison-Wesley Publ., (1990), 470 pages.

[Tschumy 90]
B. Tschumy and P. Birns, Guided Tour, NeXT application, Release 2.0 (v9), NeXT Computer, (1990).

[UIMS 92]
The UIMS Workshop Tool Developers : A Metamodel for the Runtime Architecture of an Interactive System, SIGCHI Bulletin, 24, 1, (Janvier 1992), pp. 32-37.

[Vainio-Larsson 90]
A. Vainio-Larsson, Evaluating the usability of user interfaces: research in practice, Proceedings of INTERACT'90, Elsevier Science, Amsterdam, (27-30 August 1990), pp. 323-328.

[Valdès 89]
R. Valdès, A Virtual Toolkit for Windows and the Mac, Byte, vol. 14, 3, (1989), pp. 209-216.

[Vigouroux 92]
N. Vigouroux, V. Gaildrat, R. Caubet, G. Pérennou, Une architecture d'interface pour l'interprétation d'informations multimodales, actes de IHM'92 Quatrièmes journées sur l'ingénierie des interfaces homme-machine, Paris, (30 Nov., 1 et 2 Déc. 1992), pp. 49-56.

[Webster 89]
B. F. Webster, The NeXT Book, Addison Wesley, New York, (1989).

[Wellner 93]
P. Wellner, Interacting with paper on the digitaldesk, Communications of ACM, Vol. 36, No. 7, (July 1993), pp. 87-97.

[Wiecha 89]
C. Wiecha, W. Bennett, S. Boies, J. Gould, Generating Highly Interactive User Interfaces, Proceedings of CHI'89, special issue of the SIGCHI Bulletin, ACM Press, (1989), pp. 277-282.

[Wilson 91]
M. D. Wilson, D. Sedlock, J.-L. Binot, P. Falzon, An Architecture for Multimodal Dialogue, Proceedings of the second Venona Workshop on Multi-Modal Dialogue, M.M. Taylor, F. Neel & D.G. Bouwhuis Eds., (1991).

[Yamamoto 90]
B. Yamamoto, Mail, NeXT application, Release 2.0 (v63), NeXT Computer, (1990).

Glossaire

Action physique :
Actions effectuées par l'utilisateur sur un ou plusieurs dispositifs physiques d'entrée, ou actions produites par le système en pilotant un ou plusieurs dispositifs physiques de sortie.

Action système :
Unité de traitement atomique : elle est indivisible. Elle ne peut donc être interrompue.

Adaptateur de noyau fonctionnel :
Composant logiciel conçu pour absorber les différences de modélisation des objets conceptuels entre le Noyau fonctionnel et le Contrôleur de dialogue.

Agent PAC :
Agent logiciel qui adopte les trois perspectives complémentaires : Présentation, Abstraction et Contrôle. La Présentation définit le comportement perceptible de l'agent pour un agent humain. L'Abstraction définit la compétence de l'agent indépendamment des considérations de présentation. Le Contrôle sert de pont entre les facettes Présentation et Abstraction de l'agent.

Agent logiciel :
Système de traitement de l'information : il se distingue par un jeu d'opérations, des mécanismes d'entrée/sortie et une capacité à représenter un état que l'on appelle vecteur d'état. Un agent est un acteur communicant : il assure un rôle et se manifeste par un comportement observable via l'acquisition et la production d'informations (il communique).

Arch :
Modèle d'architecture logiciel qui préconise une décomposition fonctionnelle en cinq composants organisés en arche : Noyau fonctionnel, Adaptateur de noyau fonctionnel, Contrôleur de dialogue, Techniques de Présentation, Interaction de bas niveau.

Architecture multi-agent :
Société d'agents dont la nature et l'organisation définissent ce qu'on appelle l'architecture du système. Cette architecture répond, le plus souvent, à un modèle. (cf. Modèle d'architecture.)

Assignation :
Relation qui se définit entre une classe d'unités informationnelles et un langage d'interaction ou entre un dispositif physique et un langage d'interaction.
Assignation entre une classe d'unités informationelles et un langage :
Il y a assignation entre une classe d'unités informationelles et un langage lorsque cette classe ne peut s'exprimer que dans ce langage.
Assignation entre un langage et un dispositif :
Il y a assignation entre un langage et un dispositif lorsque les expressions de ce langage ne peuvent être spécifiées qu'au moyen de ce dispositif.

Etat interne d'un système :
Valeur des informations contenues dans le contexte dynamique du système à l'instant t. C'est la photographie du contexte dynamique pratiquée à l'instant t.

Boîte à outils :
Bibliothèque de composants logiciels accessibles au programme client via des appels procéduraux.

Canal digital :
Canal de communication du système.

Canal humain :
Canal de communication de l'utilisateur.

Canal de communication d'entrée (inversement, de sortie) :
Groupe de dispositifs physiques d'entrée ou capteurs (inversement, de sortie ou effecteurs) capables de recevoir (d'émettre) des informations de types donnés sous le contrôle d'une capacité computationnelle ou processus.

Complémentarité :
Relation entre plusieurs langages d'interaction ou entre plusieurs dispositifs physiques.
Complémentarité entre langages :
La complémentarité entre deux langages d'un système est totale (ou partielle) lorsque toutes les (ou un sous-ensemble des) unités informationnelles de ce système utilisent, pour leur construction ou leur restitution, des expressions non équivalentes dans ces langages. Typiquement, les expressions coréférencielles entre deux langages sont complémentaires.
Complémentarité entre dispositifs :
La complémentarité entre deux dispositifs pour un langage donné est totale (ou partielle) lorsque toutes les (ou un sous-ensemble des) expressions de ce langage s'expriment au moyen d'actions non équivalentes avec ces dispositifs.

Contexte :
Pour un système donné, ensemble d'informations maintenu par ce système et indispensable aux traitements des informations échangées avec l'environnement. Un contexte se compose d'une partie statique dont le système possède une description immuable, et d'une partie dynamique qui varie au cours de l'utilisation du système.

Contrôleur de dialogue :
Composant logiciel qui gère l'enchaînement des tâches ainsi que les liens entre les objets regroupés dans les deux composants : Adaptateur de noyau fonctionnel et Techniques de présentation.

Creuset :
Représentation commune aux unités informationnelles émises par le composant Techniques de Présentation et aux unités informationnelles qui résultent du processus de fusion. Ainsi de la fusion de deux creusets, résulte un creuset. (cf. microfusion, macrofusion et fusion contextuelle.).

Dispositif physique :
Transducteur de propriétés physiques.

Dispositif physique d'entrée :
Capteur des mouvements de l'environnement (et notamment les actions de l'utilisateur).

Dispositif physique de sortie :
Initiateur des mouvements perceptibles par l'environnement (et notamment par l'utilisateur).

Effet d'une action système :
Différence entre l'état interne initial (c.-à-d. l'état interne avant l'exécution de l'action) et l'état interne final (c.-à-d. l'état interne après l'exécution de l'action).

Equivalence :
Relation entre plusieurs langages d'interaction ou entre plusieurs dispositifs physiques.
Equivalence entre langages :
L'équivalence entre plusieurs langages d'un système est totale lorsque toutes les unités informationnelles de ce système peuvent s'exprimer dans chacun des langages. Elle est partielle lorsqu'un sous-ensemble seulement des unités informationnelles sont exprimables dans chacun des langages.
Equivalence entre dispositifs :
Plusieurs dispositifs sont totalement équivalents pour un langage donné, si toutes les expressions permises par ce langage sont spécifiables avec chacun d'entre eux. L'équivalence est partielle si l'équivalence s'applique à un sous-ensemble des expressions du langage.

Fonction d'interprétation :
Processus interne au système qui transforme l'information acquise par les canaux d'entrée digitaux.

Fonction de restitution ou de concrétisation :
Processus interne au système qui transforme l'information représentée par le système à un haut niveau d'abstraction en information représentée au niveau d'abstraction des canaux digitaux de sortie.

Fusion lexicale :
Synchronisation entre des actions physiques, effectuées par l'utilisateur ou produites par le système informatique. La fusion lexicale est réalisée dans le composant Interaction de Bas Niveau.

Fusion sémantique :
Combinaison de commandes pour aboutir à une nouvelle fonction. La fusion sémantique est effectuée dans le Contrôleur de Dialogue ou l'Adaptateur du Noyau Fonctionnel.

Fusion syntaxique :
Combinaison d'unités informationnelles pour obtenir la description complète d'une commande. La fusion syntaxique revient au Contrôleur de Dialogue.

Fusion contextuelle :
Traitement effectué par le système qui combine des informations spécifiées par l'utilisateur avec des informations contextuelles maintenues par le système.

Grappe de tâches :
Un ensemble de tâches élémentaires permettant à l'utilisateur d'accomplir une tâche composée.

Interface dans un système informatique :
Un intermédiaire matériel et logiciel entre l'utilisateur et le noyau fonctionnel.

Interface en entrée (du point de vue système) :
Ensemble moyens de communication artificiels mis à la disposition de l'utilisateur pour modifier l'état du système. (cf. Fonction d'interprétation.)

Interface en sortie (du point de vue système):
Ensemble des moyens de communication qui rendent l'état du système perceptible à l'utilisateur. (cf. Fonction de restitution ou de concrétisation.)

Interaction de bas niveau :
Composant qui désigne la plate-forme d'accueil à la fois logicielle et matérielle. Il comprend par exemple le système de fenêtrage tel X window et ses boîtes à outils.

Langage d'interaction :
Système conventionnel structuré de signes qui assure une fonction de communication entre un système informatique et un utilisateur.

M^2LD (Multiplicité des Langages et Dispositifs) :
Taxonomie des systèmes interactifs selon la multiplicité des langages d'interaction et des dispositifs physiques.

MATIS (Multimodal Airline Travel Information System) :
Système à caractéristiques multiples d'information sur les transports aériens.

Microfusion :
Traitement effectué par le système qui combine des informations spécifiées par l'utilisateur à des instants très proches. Les intervalles de temps qui caractérisent les informations doivent s'entrelacer (parallélisme ou pseudo-parallélisme) pour que la microfusion soit déclenchée.

Macrofusion :
Traitement effectué par le système qui combine des informations spécifiées par l'utilisateur à des instants proches. Les limites des intervalles de temps, qui caractérisent les informations, doivent être proches (proximité temporelle) pour que la macrofusion soit déclenchée.

Modèle d'architecture :
Eléments directeurs d'aide à l'organisation modulaire du logiciel de l'interface d'un système informatique. Un modèle d'architecture :

- préconise la séparation entre les services du noyau fonctionnel et ceux de l'interface,
- définit une stratégie de répartition des services de l'interface qui se traduit par un ensemble de constituants logiciels,
- définit le protocole d'échange entre les constituants logiciels qu'il a permis d'identifier.

Multimodal versus multimédia :
Soient :

- C_e, l'ensemble des canaux digitaux d'entrée du système $|C_e| \geq 2$,
- C_s, l'ensemble des canaux digitaux de sortie du système $|C_s| \geq 2$,
- n_e, le nombre de canaux digitaux de C_e dont la fonction d'interprétation a un fort pouvoir d'abstraction,
- n_s, le nombre de canaux digitaux de C_s dont la fonction d'interprétation a un fort pouvoir d'abstraction,

alors, le système est :

- multimodal en entrée si $n_e \geq 2$, sinon il est multimédia en entrée,
- multimodal en sortie si $n_s \geq 2$, sinon il est multimédia en sortie.

NoteBook :
Système à caractéristiques multiples qui correspond à un bloc-notes électronique.

Noyau fonctionnel :
Le logiciel modeleur des concepts de la tâche informatisée.

O^2LD (caractère Obligatoire ou Optionnel des Langages et Dispositifs) :
Taxonomie des systèmes interactifs selon les choix temporels.

Objet conceptuel de présentation :
Interacteur abstrait, idéalisé, qui traduit un besoin communicationnel.

PAC-Amodeus:
Modèle d'architecture logiciel hybride qui reprend les composants du modèle ARCH dont il affine le contrôleur de dialogue, composant principal, en termes d'agents PAC.

PAC-Expert :
Générateur d'architecture logicielle incluant une méthode de conception d'architecture logicielle répondant au modèle PAC-Amodeus. A partir de la description du comportement externe d'un système, PAC-Expert produit l'architecture PAC-Amodeus correspondante.

Pipe-Lines :
Espace conceptuel, dont l'objectif est de couvrir à la fois les entrées/sorties et les points de vue système et utilisateur. Pipe-Lines explicite les traitements effectués à la fois par l'utilisateur et par le système et exacerbe trois niveaux d'abstraction : les actions physiques, les unités informationnelles et les effets.

Redondance :
Relation entre plusieurs langages d'interaction ou entre plusieurs dispositifs physiques.
Redondance entre langages :
Il y a redondance totale/partielle entre deux langages d'un système si ces langages sont totalement/partiellement équivalents, s'ils peuvent être utilisés de manière concurrente et si tout couple d'expressions équivalentes correspond à un exemplaire unique d'unité informationnelle.
Redondance entre dispositifs :
Il y a redondance totale/partielle entre deux dispositifs pour un langage s'ils sont totalement/partiellement équivalents, s'ils peuvent être utilisés de manière concurrente et si tout couple d'actions équivalentes correspond à un exemplaire d'expression.

Système à caractéristiques multiples :
Système informatique qui offre plusieurs langages d'interaction et/ou dispositifs physiques en entrée comme en sortie. (cf. Multimodal versus multimédia.)

Système informatique :
Réalisation logicielle qui comprend une interface et un noyau fonctionnel.

Squelette d'application :
Ensemble de fonctions générales d'une interface sous forme d'un logiciel réutilisable et extensible. L'utilisation d'un squelette consiste à greffer des composants spécifiques au système à implémenter.

Tâche élémentaire :
Toute tâche que l'utilisateur peut accomplir avec le système via une et une seule commande.

Techniques de Présentation :
Composant logiciel qui définit le comportement perceptible conceptuel du système en termes d'objets conceptuels de présentation. Ce composant est une couche logicielle qui assure la portabilité du Contrôleur de dialogue sur différentes plates-formes d'accueil.

ULD (Usage des Langages et Dispositifs) :
Taxonomie des systèmes interactifs selon les usages des langages d'interaction et des dispositifs physiques.

Unité informationnelle :
Unité conceptuelle qui regroupe un ensemble indivisible d'informations qui, dans le système, modélise un concept du domaine de la tâche. Parce qu'elle fait sens, une unité informationnelle est qualifiée de "représentation du niveau d'abstraction le plus haut".

UOM (Usage, Option, Multiplicité) :
Méthode de classification des systèmes interactifs qui s'appuie sur les langages d'interaction et les dispositifs physiques, deux éléments de contact entre l'utilisateur et le système. (cf. M^2LD, O^2LD, ULD.)

Table des figures

Table des matières