CONSERVATOIRE NATIONAL DES ARTS ET MÉTIERS
CENTRE AGRÉÉ DE GRENOBLE (C.U.E.F.A.)

MÉMOIRE

présenté en vue d'obtenir
le Diplôme d'Ingénieur C.N.A.M.
en informatique

CONCEPTION ET RÉALISATION D'UN OUTIL D'AIDE À L'ANALYSE DE DONNÉES COMPORTEMENTALES

par Nathalie LISCHETTI

Soutenu le 30 mars 1995 devant le jury composé de :

Présidente : Mme Véronique Donzeau-Gouge
Membres : Mme Pascale Anas
Mme Maria-Virginia Aponte
Mme Joëlle Coutaz
M. Jacques Courtin
M. Bruno Guillaumont

- Qui êtes-vous ?
- Je suis Oz, le Grand et le Redoutable, dit le petit homme d'une voix tremblante.
- Je croyais qu'Oz était une grande tête, dit Dorothée.
- Et moi, une belle dame, dit l'Épouvantail.
- Et moi, une terrible bête sauvage, dit le Bûcheron.
- Et moi, une boule de feu, dit le Lion.
- Non ! vous vous trompiez, avoua humblement le petit homme. Je vous l'ai seulement fait croire. Je ne suis qu'un homme ordinaire.
- Oh ! vous êtes plus que cela ! dit l'Épouvantail, vous êtes un charlatan.
- Mais, j'ai honte, j'ai très honte d'être un pareil charlatan, dit le petit homme tristement. Hélas, je n'avais pas le choix. Je vais vous raconter mon histoire.

Extrait du livre "The Wonderful Wizard of Oz" écrit par Lyman Frank Baum en 1900 aux États-Unis.

Le titre de ce conte a été repris pour désigner la technique du Magicien d'Oz, employée dans l'étude de l'utilisabilité des interfaces homme-machine.

Un jeune homme alla rencontrer le plus sage des hommes pour apprendre le secret du bonheur. Le sage lui remit une petite cuiller dans laquelle il versa deux gouttes d'huile et il lui suggéra de faire une promenade dans le palais en veillant à ne pas renverser l'huile.

A la fin de sa promenade, le jeune homme, confus, dut avouer au sage qu'il n'avait rien vu du palais. Son seul souci avait été de ne pas renverser les deux gouttes d'huile. "Eh, bien, retourne faire connaissance avec les merveilles de mon palais", lui dit le sage.

De retour auprès du sage, le jeune homme relata de façon détaillée tout ce qu'il avait vu. " Mais où sont les deux gouttes d'huile que je t'avais confiées? Eh bien, dit alors le sage, c'est le seul conseil que j'aie à te donner : le secret du bonheur est de regarder toutes les merveilles du monde mais sans jamais oublier les deux gouttes d'huile dans la cuiller."

Extrait de "L'Alchimiste" écrit par Paulo Coehlo en 1988 au Brésil.

Remerciements

Ce travail revient à ma famille, Jacqueline, Paul, Serge, François, Aurélien et à mes amis les plus chers. Je les remercie vivement de m'avoir fait confiance, soutenu et aidé jusqu'à l'aboutissement de ce travail.

Je tiens aussi à remercier les nombreuses personnes qui ont participé directement ou indirectement à la réalisation de ce mémoire, et plus particulièrement :

Madame Véronique Donzeau-Gouge, Professeur au CNAM de Paris, qui préside le Jury de soutenance de ce mémoire,

Madame Maria-Virginia Aponte, Maître de Conférences au CNAM de Paris, pour l'attention qu'elle manifeste à ce travail en participant au Jury,

Monsieur Jacques Courtin, Professeur à l'Université Pierre Mendès-France de Grenoble, Responsable du cycle ingénieur CNAM en informatique de Grenoble et membre du Jury,

Madame Joëlle Coutaz, Professeur à l'Université Joseph Fourier de Grenoble, qui m'a accueillie dans son équipe, m'a confié ce travail et m'a ainsi permis de découvrir ce domaine de recherche qui m'attirait,

Madame Pascale Anas et Monsieur Bruno Guillaumont, responsables d'une équipe informatique de la société Schneider Electric qui me font l'honneur de participer au Jury,

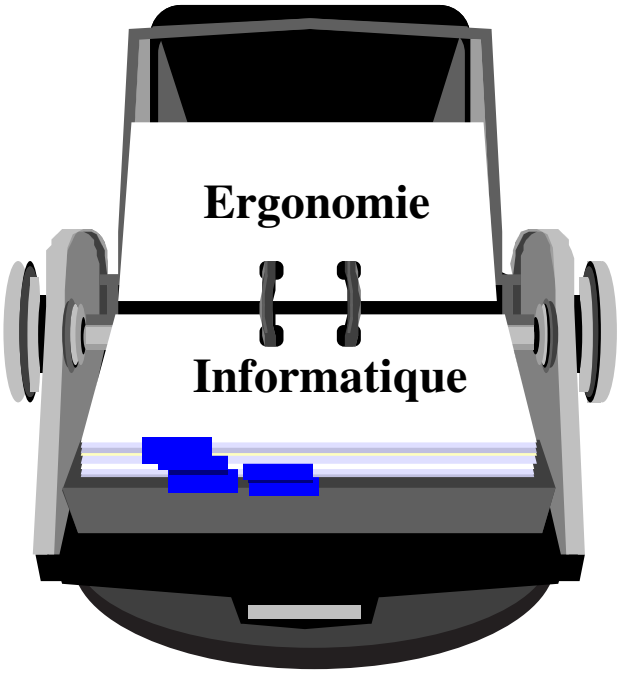
Les quatre étudiants en DESS avec qui j'ai collaboré quelques mois, et surtout, Stéphane Liska et Stéphane Megy qui m'ont apporté leur aide précieuse et leur soutien,

Toute l'équipe Interface Homme-Machine, thésards, étudiants, et ingénieurs qui contribuent à l'ambiance de travail et à la renommée de l'équipe, et tout particulièrement François Bérard pour le temps qu'il m'a consacré lors de l'apprentissage de MacApp, Gilles Ambone pour ses conseils techniques, Francis Jambon pour avoir partagé avec moi son bureau et même parfois son Macintosh, et enfin Daniel, Laurence et Sandrine pour leur appui lors de la rédaction,

Sans oublier, Philippe pour sa bonne humeur et ses petits en-cas lors de mes fringales nocturnes et Pascal qui tout en préparant lui aussi son mémoire a toujours fait preuve de patience et de disponibilité,

Et dans le désordre, tous ceux que j'ai rencontrés et qui m'ont accompagnée sur ce chemin, Bernard, Yves, Gilles, Pascale, Jeanine, Pierre, Monique, Stéphane et Bruno.

Table des matières



Introduction

Le contexte et les motivations..... 2
L'organisation du mémoire..... 3

Chapitre 1 : Le contexte de l'étude

1.1. La technique du "Magicien d'Oz"..... 6
1.2. La plate-forme NEIMO : principe..... 8
1.3. Architecture logicielle de NEIMO et ses services réutilisables 9
1.4. NEIMO appliqué au cas de l'étude : SUPRATEL ★.....13
1.5. Les limites de ma contribution 17

Chapitre 2 : Le domaine de recherche

2.1. La conception d'interfaces homme-machine et l'apport des sciences cognitives 20
2.2. Evaluation des interfaces homme-machine : un état de l'art.....21
2.2.1. Les méthodes prédictives.....23
2.2.2. Les méthodes expérimentales 25
2.3. L'évaluation ergonomique automatique 27

Chapitre 3 : La démarche de développement

3.1. Le cycle de vie du logiciel.....30
3.1.1. Les modèles en cascade et en V.....30
3.1.2. Le modèle en spirale 33
3.1.3. NEIMO : un développement en spirale.....34
3.1.4. Mon modèle de développement.....35
3.2. Les outils et méthodes de spécification employés.....37
3.2.1. OMT, une méthode intervenant dès la phase d'analyse des besoins 38
3.2.2. UAN, un langage de spécification conceptuelle 40

3.3. Les modèles d'architecture 43
3.3.1. Définition, intérêts et classes de modèles 44
3.3.2. Le modèle PAC : une architecture multi-agents 45
3.4. Les outils de développement 49
3.4.1. Les boîtes à outils 49
3.4.2. Les squelettes d'application 49
3.4.3. L'environnement de programmation de Symantec 51
3.4.4. Les générateurs d'interfaces homme-machine 53

Chapitre 4 : Réalisation de SupraAnalyse, un outil d'aide à l'analyse de données comportementales

4.1. Analyse des besoins des ergonomes : un premier pas vers l'évaluation automatique d'une interface homme-machine 56
4.1.1. La démarche d'évaluation ergonomique du CCETT 56
4.1.2. Le déroulement d'une session d'expérimentation 58
4.2. La démarche d'évaluation proposée avec SUPRATEL ★ 61
4.2.1. Des propositions pour la phase de capture et d'analyse 61
4.2.2. Le fichier historique d'expérimentation 62
4.2.3. L'outil SupraAnalyse et ses services 64
4.2.3.1. La gestion des fichiers 65
4.2.3.2. La visualisation d'une session d'expérimentation 66
4.2.3.3. L'annotation a posteriori d'une session d'expérimentation 67
4.2.3.4. L'analyse d'une session d'expérimentation 67
4.3. Spécifications externes de l'outil réalisé 68
4.3.1. La fenêtre de visualisation de l'interaction 70
4.3.2. La fenêtre de pré-analyse 75
4.3.3. La fenêtre d'analyse 76
4.3.4. La barre de menu 79
4.4. Architecture globale de l'outil réalisé 80
4.4.1. L'architecture PAC de SupraAnalyse 80
4.4.2. La conception orientée objet et le modèle PAC 84

Conclusion

Bilan du travail réalisé 88
Perspectives 91

Annexes

Annexe 1 : QuickTime A-3
Annexe 2 : Les Apple Events A-4
Annexe 3 : Le mécanisme des vues A-5
Annexe 4 : L'interface homme-machine de SupraAnalyse A-6

Bibliographie

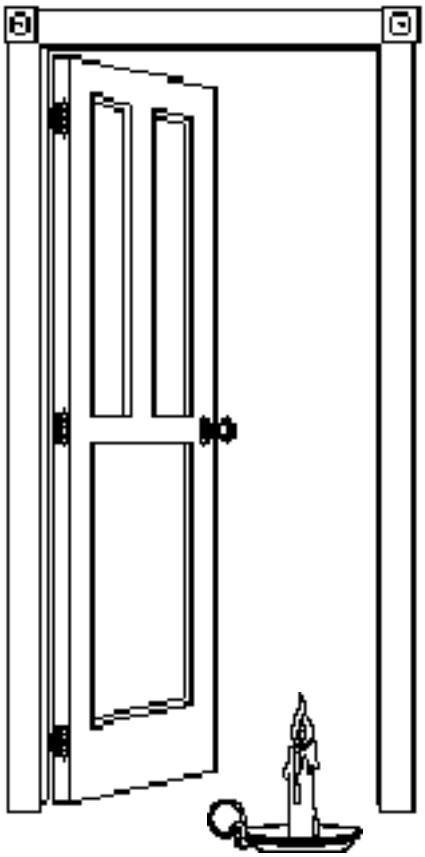
Références bibliographiques B-2
Références par thèmes B-14

Table des figures

Figure 1.1 : Une session d'expérimentation selon la technique du Magicien d'Oz. 6
Figure 1.2 : L'analyse des sessions d'expérimentation. 7
Figure 1.3 : Le dispositif expérimental NEIMO. 8
Figure 1.4 : NEIMO et ses bibliothèques pour une session d'expérimentation. 11
Figure 1.5 : NEIMO et ses bibliothèques pour une session d'analyse. 12
Figure 1.6 : La structure d'une application (compère, sujet ou analyste). 13
Figure 1.7 : La structure des applications sujet et compères dans SUPRATEL ★ 14
Figure 1.8 : La structure d'une application analyste dans SUPRATEL ★. 14
Figure 1.9 : Ma contribution sur l'ensemble du projet SUPRATEL ★ 16
Figure 2.1 : Une classification des méthodes d'évaluation des interfaces homme-machine. 22
Figure 2.2 : Une classification de l'évaluation automatique des interfaces homme-machine. 27
Figure 3.1 : Le modèle en cascade initial. 30
Figure 3.2 : Le modèle en cascade affiné. 31
Figure 3.3 : Le modèle en V. 32
Figure 3.4 : Le modèle en spirale. 33
Figure 3.5 : La notation UAN sous forme d'un tableau à trois colonnes. 40
Figure 3.6 : Les relations temporelles dans UAN. 41
Figure 3.7 : La description UAN de la sélection d'un objet. 42
Figure 3.8 : Un système informatique et son utilisateur. 44
Figure 3.9 : Le modèle d'architecture de base. 44
Figure 3.10 : Un objet interactif PAC vu comme un agent tri-processeur. 46
Figure 3.11 : La structure d'un système interactif en objets PAC élémentaires. 47
Figure 4.1 : La capture des données avec NEIMO. 62
Figure 4.2 : La décomposition fonctionnelle de l'application SupraAnalyse. 64
Figure 4.3 : Un extrait du diagramme OMT de SupraAnalyse. 65
Figure 4.4 : Les trois fenêtres de l'application SupraAnalyse. 70
Figure 4.5 : Le film qui reproduit l'écran du sujet lors de la session d'expérimentation. 71
Figure 4.6 : L'outil de pilotage du magnétoscope. 72
Figure 4.7 : La tâche "Rejouer" en UAN. 73
Figure 4.8 : Le "magnétoscope" à l'arrêt (haut) et en état de marche (bas). 74
Figure 4.9 : La sous-tâche "Jouer" en UAN. 75
Figure 4.10 : La sous-tâche "Avancer image par image" en UAN. 76
Figure 4.11 : Les annotations des compères. 76
Figure 4.12 : La fenêtre d'analyse. 77

Figure 4.13 : La liste des scénarios de la session d'expérimentation. 78
Figure 4.14 : L'analyse des tâches effectuées par le sujet..... 79
Figure 4.15 : La barre de menu. 80
Figure 4.16 : La modélisation de SupraAnalyse à l'aide du modèle PAC..... 82
Figure 4.17. : Les communications autorisées par le modèle PAC. 86
Figure 5.1 : Les outils et méthodes employés dans le cycle en V. 89
Figure 5.2 : Le Perspective Wall. 93
Figure A.1 : Le format movie. A-3
Figure A.2. : La structure interne d'un Apple Event..... A-4
Figure A.3. : Le mécanisme des vues. A-5
Figure A.4. : L'interface homme-machine de SupraAnalyse. A-6

Introduction



Le contexte et les motivations

Ce travail relève de l'Interaction Homme-Machine (IHM). L'étude de l'Interaction Homme-Machine fait aujourd'hui l'objet d'attentions particulières. Les approches en recherche diffèrent mais visent un objectif commun : garantir l'utilisabilité des systèmes et ainsi, améliorer le confort, augmenter le plaisir, la productivité et la sécurité des utilisateurs. L'utilisabilité, qui participe à la Qualité, constitue un point d'ancrage privilégié entre l'ergonomie et les pratiques du Génie Logiciel.

Le Génie Logiciel a su développer des modèles, des méthodes et des outils pour valider ou vérifier la Qualité. En particulier, les normes ISO rendent explicite le rôle de l'utilisabilité dans la Qualité. En pratique, les concepteurs de logiciel sous-estiment ce facteur, le traitent de manière superficielle et réductrice, ou l'ignorent délibérément. Les raisons à cela sont multiples : manque de compétences de l'équipe de développement, absence de conviction sur l'intérêt d'une telle pratique ou encore un rapport prédictif "coûts/bénéfices" non probant.

Et pourtant, la technologie des interfaces homme-machine s'est enrichie récemment d'outils de développement efficaces qui participent à leur manière au progrès de l'utilisabilité. Parmi ces outils, nous relevons les générateurs d'interfaces, tels Interface Builder [Webster 89] et XFaceMaker [Non Standard Logic], qui permettent à moindre coût de créer des maquettes d'interface. Ces maquettes sont censées faire l'objet d'évaluations ergonomiques avec des utilisateurs représentatifs, être modifiées de manière itérative jusqu'à ce que les critères d'utilisabilité soient satisfaits. Cette évaluation fondée sur l'observation d'utilisateurs est en vérité une tâche longue qui requiert l'enregistrement et l'analyse d'un grand volume de données comportementales. S'il est vrai que l'ingénierie des logiciels a développé des outils pour la mise en œuvre des interfaces, aucun d'entre eux ne vise directement l'aide à l'évaluation des interfaces. A l'évidence, cette lacune doit être comblée. Mon travail s'inscrit dans cet objectif.

Le thème central de mon étude est la conception et la réalisation d'un outil informatique d'aide à l'évaluation d'Interfaces Homme-Machine. L'application pilote retenue pour tester cet outil est l'interface utilisateur d'un terminal de télécommunication multiservice. Ce terminal permet la Communication Homme-Homme par visiophonie, fax, téléphone, etc. La conception de l'interface de ce terminal fait également partie de mon travail.

L'outil d'aide à l'évaluation s'inscrit dans le projet NEIMO (Nouvelle Evaluation des Interfaces par le Magicien d'Oz) développé par l'équipe Interface Homme-Machine du Laboratoire de Génie Informatique de l'IMAG (LGI-IMAG[1]). NEIMO est un noyau d'accueil logiciel et matériel disponible sur station Macintosh qui offre les mécanismes de base pour la capture et la relecture de données comportementales [Ambone 92a et 92b]. L'outil que je dois développer habille le noyau NEIMO de services adaptés à l'évaluation d'une interface homme-machine particulière : celle d'un terminal multiservice de télécommunication simulé sur station Macintosh. La conception de cette interface et son évaluation via mon outil font l'objet d'un contrat dans lequel le CCETT (Centre Commun d'Etudes de Télédiffusion et Télécommunications) est le client du LGI-IMAG. Les ergonomes du CCETT sont les utilisateurs potentiels de l'outil d'évaluation tandis que le terminal multiservice est destiné à du personnel de bureau.

Pour réaliser ce travail, j'ai été accueillie par l'équipe Interface Homme-Machine dans les locaux du Laboratoire de Génie Informatique. Le responsable de l'équipe, Joëlle Coutaz, Professeur à l'Université Joseph Fourier, a encadré mon projet.

L'organisation du mémoire

J'ai choisi d'organiser ce mémoire en quatre chapitres.

Le chapitre 1 précise le contexte du travail, ma contribution et les acteurs du projet. J'y décris la plate-forme d'accueil NEIMO et, au-dessus de NEIMO, les développements que j'ai réalisés et ceux auxquels j'ai contribué en collaboration avec d'autres étudiants.

L'état de l'art du chapitre 2 cerne le domaine de recherche dans lequel s'inscrit mon travail. Il s'appuie sur une étude bibliographique qui montre la place de l'évaluation ergonomique dans la conception des interfaces. Une classification des différentes méthodes d'évaluation permet de situer celle adoptée avec la plate-forme à Magicien d'Oz NEIMO.

Au chapitre 3, nous considérons la démarche de développement. La conduite de projet que j'ai retenue est décrite en regard des différents cycles de vie du logiciel. La description des méthodes et des outils de spécification et de conception des IHM conduit à préciser mes choix mais aussi, au vu de l'expérience, à émettre une opinion critique sur les outils employés.

[1]LGI-IMAG : Laboratoire de Génie Informatique de l'Institut de Mathémathiques Appliquées de Grenoble.

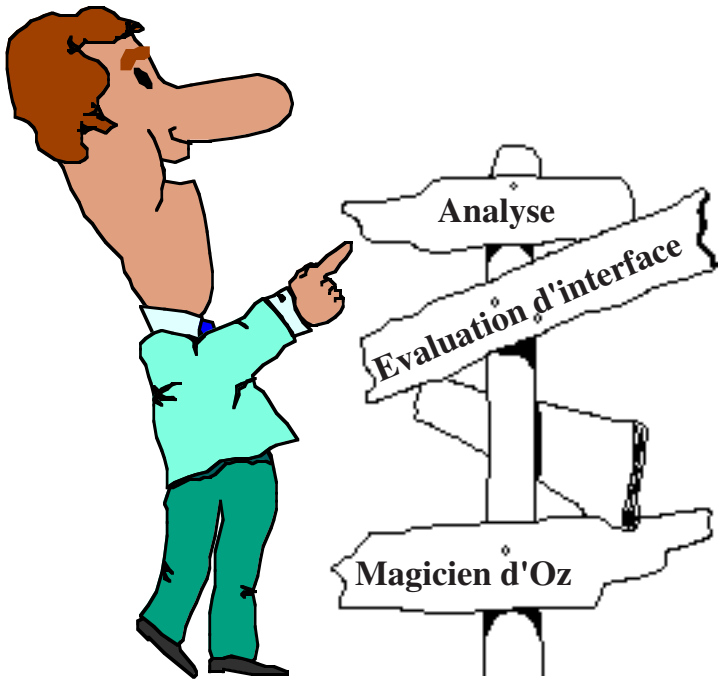
Le chapitre 4 aborde la réalisation proprement dite en reprenant les différentes fournitures du cycle de vie. Tout d'abord, l'analyse des besoins des ergonomes. Celle-ci s'appuie sur une étude bibliographique mais aussi sur la visite du laboratoire d'ergonomie de notre client (le CCETT). Ensuite, je présente la démarche d'évaluation que nous avons proposée aux ergonomes et qui exploite NEIMO. Les spécifications externes traitent uniquement de l'outil SupraAnalyse et décrivent les formes de présentation retenues pour les services d'analyse des données comportementales. Enfin, je détaille l'architecture logicielle choisie et précise sa mise en œuvre dans une conception orientée objet.

En conclusion, je présente le bilan de l'expérience et les perspectives envisagées.

On trouvera en annexe des détails techniques sur l'environnement de développement et notamment en annexe 1 et 2 une description de QuickTime et le principe des Apple Events. En annexe 3, des précisions sur le mécanisme des vues mis en œuvre dans la bibliothèque historique de NEIMO. Enfin, l'annexe 4 présente sur une double page l'interface homme-machine de l'outil SupraAnalyse que j'ai réalisé.

Chapitre 1

Le contexte de l'étude



Ce chapitre présente le contexte de l'étude : l'existant, le logiciel qui doit y être greffé et les acteurs du projet. NEIMO, nous l'avons vu en introduction, définit le cœur de l'existant. Cette plate-forme permet de conduire des expérimentations de type Magicien d'Oz. J'en présente le principe au paragraphe 1.1 et 1.2. (Au chapitre 2 consacré à l'état de l'art, on situera l'approche du Magicien d'Oz parmi les méthodes possibles.) En 1.3, on trouvera l'architecture matérielle et logicielle de NEIMO et ses composants réutilisables. On montre en 1.4 comment enrichir NEIMO de composants logiciels pour une expérimentation ciblée comme la nôtre. Après avoir explicité ma contribution, j'énonce au paragraphe 1.5, en les justifiant, les limites de mon projet.

1.1. La technique du "Magicien d'Oz"

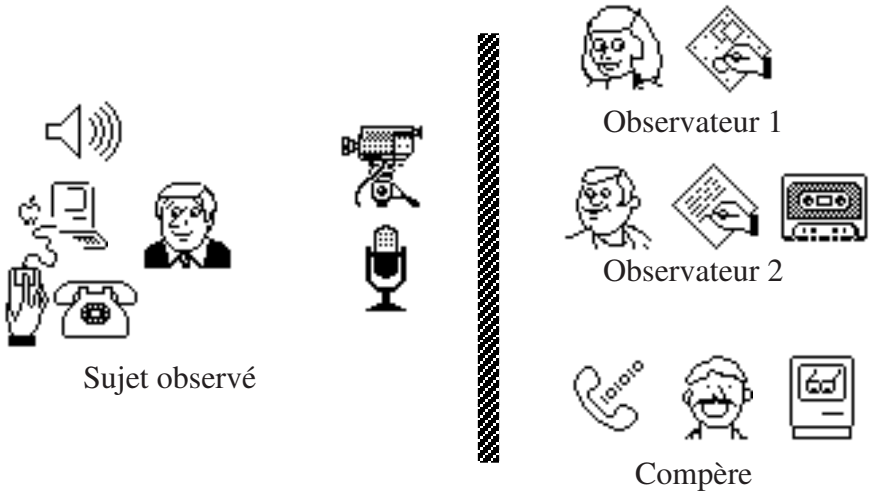


Figure 1.1 : Une session d'expérimentation selon la technique du Magicien d'Oz.

Comme le montre la figure 1.1, la technique du Magicien d'Oz consiste à faire exécuter par un compère humain dissimulé les fonctions manquantes d'un système utilisé par un sujet représentatif. Lors d'une session d'expérimentation, le sujet réalise un scénario (par exemple téléphoner à l'hôtel Bellevue pour réserver une chambre) via l'interface dont on souhaite évaluer l'utilisabilité. Tandis que le compère simule les services non implémentés (par exemple, la reconnaissance de la parole), d'autres spécialistes enregistrent le comportement du sujet sur bande audio et vidéo et/ou prennent des notes manuscrites. Ce matériau de données comportementales est ensuite analysé en vue de dégager des problèmes d'utilisabilité, d'en comprendre l'origine et de définir les actions correctives. La figure 1.2 illustre l'organisation de ce processus : des sessions d'expérimentation suivies d'une étape d'analyse.

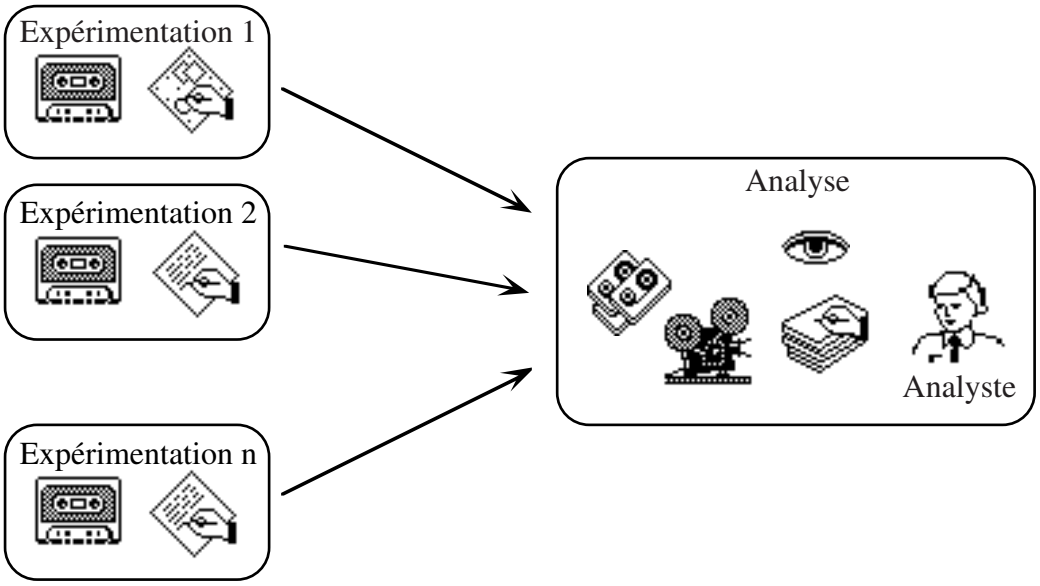


Figure 1.2 : L'analyse des sessions d'expérimentation.

En général, l'existence du compère-simulateur n'est pas dévoilée au sujet mais l'éthique exige que le sujet soit informé avant l'expérimentation que des spécialistes ou des instruments l'observent.

En l'absence de théorie prédictive opérationnelle, une observation expérimentale comme celle du Magicien d'Oz, permet de dégager des principes ou, en tout état de cause, de tester en laboratoire des interfaces complexes avant leur réalisation complète. L'inconvénient majeur de cette technique est le dépouillement des données comportementales dont le volume devient vite explosif et dont l'inspection est malaisée (par exemple, retrouver un incident particulier sur une bande vidéo). La plate-forme NEIMO, qui permet la capture informatique du comportement, vise à combler les difficultés liées à l'exploitation des données.

1.2. La plate-forme NEIMO : principe

NEIMO est une plate-forme multiprocesseur extensible permettant de conduire des expérimentations de type Magicien d'Oz [Ambone 92a et 92b]. La figure 1.3 montre le dispositif expérimental.

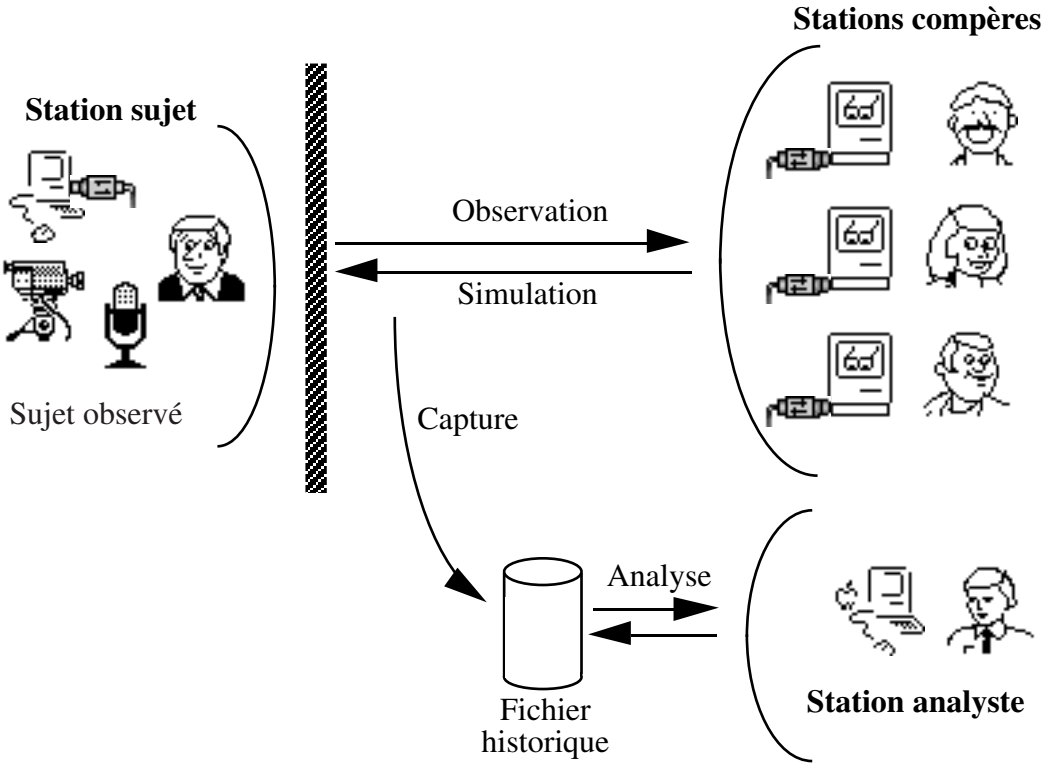


Figure 1.3 : Le dispositif expérimental NEIMO.

Pour les sessions d'expérimentation, le sujet et les compères disposent chacun d'une station de travail. Dans NEIMO, le terme "compère" désigne aussi bien un individu qui simule une fonction manquante qu'un observateur spécialiste dont le rôle est de détecter des problèmes d'utilisabilité pendant la session. Les stations sont reliées par un réseau local Ethernet. Le comportement du sujet et les annotations des compères sont enregistrés sur un support informatique, le fichier "historique" :

- Le comportement du sujet est capturé à différents niveaux d'abstraction : actions physiques de bas niveau tel un clic souris, tâches élémentaires comme la sélection d'un objet ou encore l'activation d'une commande.

- Les annotations des compères incluent des informations générales (comme l'échec ou la réussite du scénario par le sujet) ou l'identification d'erreurs locales (comme la difficulté pour le sujet de retrouver une information sur l'écran).

Pour la phase d'analyse qui suit les sessions d'expérimentation, le spécialiste ergonome visualise le contenu du fichier historique sur une station dédiée. Les données comportementales numérisées de ce fichier font l'objet de traitements automatiques qui facilitent la tâche de dépouillement, d'inspection, de détection de problèmes d'utilisabilité concernant l'interface homme-machine évaluée, etc. Je reviendrai sur ce point, objet central de ma contribution.

En résumé, NEIMO comprend une station sujet, des stations compères et une station analyste. Sur chaque station s'exécute une application. On parle d'*application compère* pour les logiciels qui s'exécutent sur les stations compères, d'*application sujet* pour désigner le logiciel qui s'exécute sur une station sujet, et d'*application analyste* pour la station analyste. Le *fichier historique* sert de recueil commun de données à ces applications. Les applications compères et sujet déposent de l'information dans l'historique tandis que l'application analyste la lit et éventuellement la modifie (édition, ajout, retrait d'annotations). Les premières font des accès concurrents à l'historique, la seconde l'utilise seul.

Ayant présenté les principes généraux de la plate-forme d'accueil, nous étudions maintenant son architecture logicielle pour montrer ensuite la nature des travaux à réaliser et leur intégration dans NEIMO.

1.3. Architecture logicielle de NEIMO et ses services réutilisables

L'architecture de NEIMO est composée d'un noyau de communication, NeimoCom, auquel s'ajoutent des bibliothèques d'interface. NeimoCom est le serveur de communication entre les applications. Notons que pour des raisons de performance, le processus NeimoCom utilise une station dédiée ; mais il a été conçu pour fonctionner sur n'importe quelle station sujet ou compère. Il existe trois sortes de bibliothèques :

- La *bibliothèque compère* permet à un développeur d'application compère de connecter son logiciel à NeimoCom. Elle offre les services suivants : initialisation et terminaison d'une session, enregistrement d'un message dans l'historique, abonnement dynamique à un type de message donné pour autoriser une reconfiguration[2] des stations compères

[2]Un compère donné n'est pas forcément utile pendant toute la session, et les compères doivent éventuellement pouvoir changer de rôle en cours de session.

en cours de session, envoi et réception de messages émis par l'application sujet ou les applications compères (les compères peuvent échanger des messages entre eux).

- La *bibliothèque sujet* permet au développeur d'une application sujet de la relier à NeimoCom : envoi de message en direction d'une application compère donnée, enregistrement dans l'historique (via NeimoCom) d'une information ou d'un message destiné aux compères, réception de messages en provenance des stations compères.

- La *bibliothèque historique* sert uniquement au développeur d'application analyste. Elle offre les services suivants : accès aux informations de l'historique via des fonctions de positionnement (début/fin de l'historique, positionnement absolu et relatif par rapport à la position courante), lecture sélective des données grâce à un mécanisme de tri et de filtrage sur critères (date de l'information, provenance sujet ou compère, type d'information, etc.). On trouvera en annexe 3 des précisions sur ce mécanisme.

En s'appuyant sur ces bibliothèques de liaison spécialisées, les applications sujet et compères échangent des messages structurés uniquement via le noyau de communication NeimoCom. Les messages sont destinés à être transmis à une autre application et/ou destinés à être sauvegardés dans le fichier historique par NeimoCom à la demande des applications sujet et compère ; NeimoCom les réceptionne et les aiguille vers le destinataire. NeimoCom se charge uniquement de la structure des messages, auxquels il ajoute la date, la durée et l'origine (émetteur). Du fait de sa position centrale, NeimoCom définit une base de temps universelle pour l'estampillage des messages. NeimoCom n'a pas connaissance de la sémantique des informations échangées : celle-ci est à la charge du concepteur des applications émettrices. Ces dernières complètent l'information de base avec un type de leur choix, par exemple la nature de l'information (clic, son, image), ou la modalité (geste, parole) et l'information brute peuvent être de n'importe quel niveau d'abstraction. En conséquence, la bibliothèque historique de NEIMO propose à l'application analyste des services lui permettant de s'affranchir du codage des informations enregistrées dans le fichier.

NEIMO a été développé au LGI par l'équipe IHM en C et C++ avec MPW[3] [Apple 90] et MacApp[4] [Schmucker 86a]. Le protocole de communication de NeimoCom utilise le standard de communication d'Apple basé sur les Apple Events[5] [Apple 91a]. L'historique est un fichier

[3]Macintosh Programmer's Workshop (MPW) est un environnement de programmation.
[4]MacApp est un squelette d'application qui propose un ensemble de bibliothèques à objet pour Macintosh.
[5]Apple Computer Inc. a développé un protocole de communication inter-applications nommé Apple Event Interprocess Messaging Protocol (AEIMP) pour le système d'exploitation System 7. Toute communication inter-applications s'appuie sur les événements appelés Apple Events.

au format QuickTime [Apple 92c]. QuickTime est une extension du système d'exploitation System 7 d'Apple qui permet le stockage et la manipulation de données estampillées issues de différents médias[6]. On trouvera en annexes 1 et 2, une description du principe des Apple Events et de QuickTime.

NeimoCom et les bibliothèques constituent le noyau réutilisable de la plate-forme NEIMO. Les figures 1.4 et 1.5 montrent les composants qu'il convient de développer ou de réutiliser pour obtenir une plate-forme d'expérimentation ciblée.

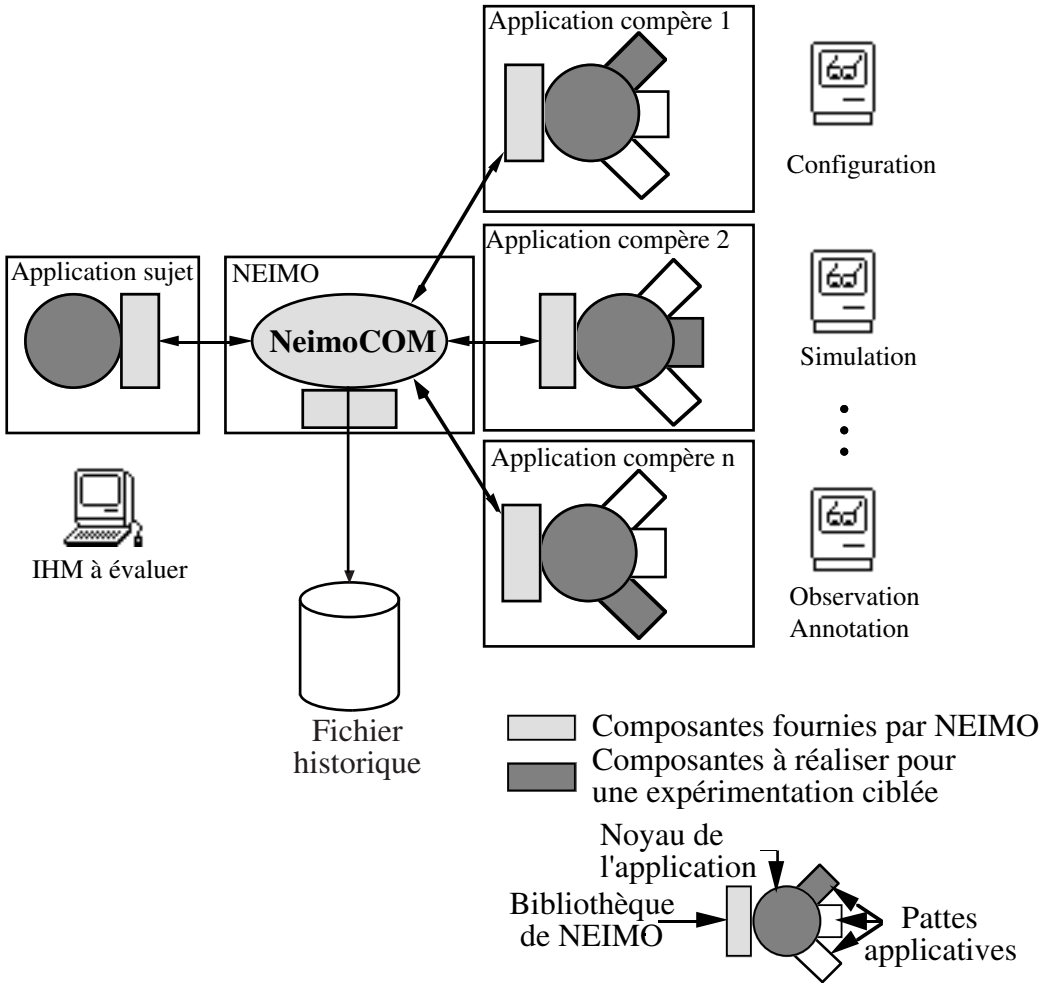


Figure 1.4 : NEIMO et ses bibliothèques pour une session d'expérimentation.

[6]Un média est un support technique de l'information. En l'occurrence, QuickTime permet de sauvegarder du son et de l'image.

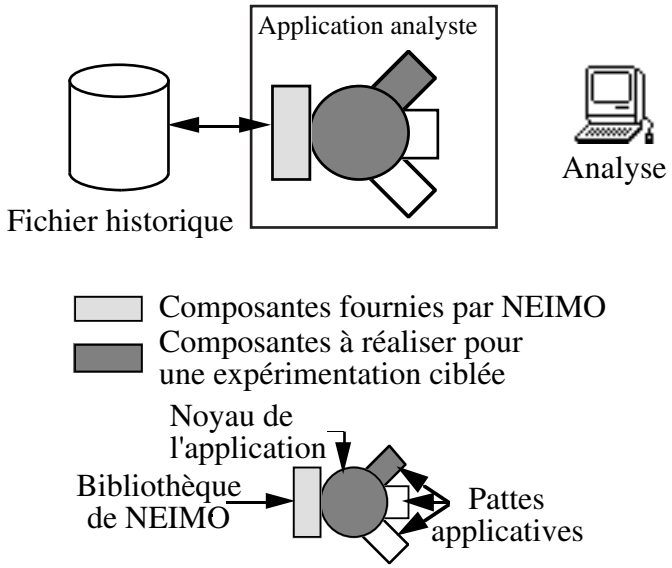


Figure 1.5 : NEIMO et ses bibliothèques pour une session d'analyse.

Dans ces figures, on distingue en gris clair les composants réutilisables du noyau NEIMO :

- le serveur de communication NeimoCom,
- les bibliothèques de connexion aux applications sujet et compères,
- la bibliothèque de sauvegarde vers le fichier historique,
- la bibliothèque de connexion à l'application analyste.

Les parties gris foncé dénotent les composants qui doivent être spécialisés ou écrits entièrement pour une expérimentation ciblée :

- l'interface homme-machine de l'application sujet à tester,
- le logiciel qui permet aux compères de configurer la session d'expérimentation (localisation des stations, répartition des rôles entre les compères, etc.),
- le logiciel qui permet aux compères de simuler les fonctions manquantes,
- le logiciel qui permet aux compères d'observer le sujet et de faire des annotations,
- le logiciel qui permet à l'analyste d'exploiter le contenu du fichier historique.

A l'évidence, l'interface homme-machine du sujet concerne l'application sujet. Les autres services logiciels (configuration, simulation, observation) font partie des applications compères. Certains, comme les services de configuration, risquent d'être utiles à plusieurs domaines de tests. Dans ces conditions, il convient de les concevoir de manière à permettre leur réutilisation. Pour cette raison, il est conseillé de structurer une application (compère, sujet, ou analyste) selon le schéma de la figure 1.6 : un noyau qui s'interface avec une bibliothèque pour la connexion à NeimoCom et des "pattes applicatives" qui se branchent sur ce noyau et qui incluent une interface avec un utilisateur (sujet, compère, analyste). Certaines de ces pattes, notées en blanc dans les figures 1.4, 1.5 et 1.6 , sont des services de haut niveau réutilisables.

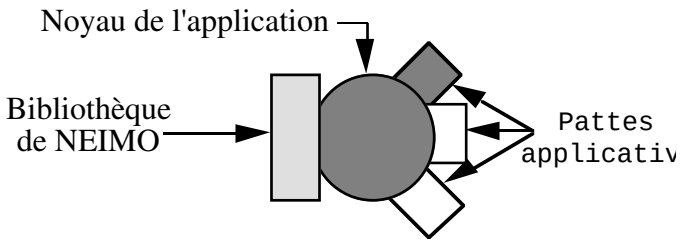


Figure 1.6 : La structure d'une application (compère, sujet ou analyste).

Dans ce cadre général, nous situons maintenant le travail à réaliser.

1.4. NEIMO appliqué au cas de l'étude : SUPRATEL ★

Nous appelons SUPRATEL ★ la plate-forme réutilisable NEIMO augmentée des composants logiciels pour tester l'interface utilisateur d'un terminal multiservice de télécommunication. Ce terminal permet l'émission et la réception d'appels téléphoniques et visiophoniques, l'émission et la réception de fax, la consultation d'un répertoire et d'un classeur. Je précise que le nom du projet et en particulier le symbole ★ nous a été inspiré par le logo du CCETT.

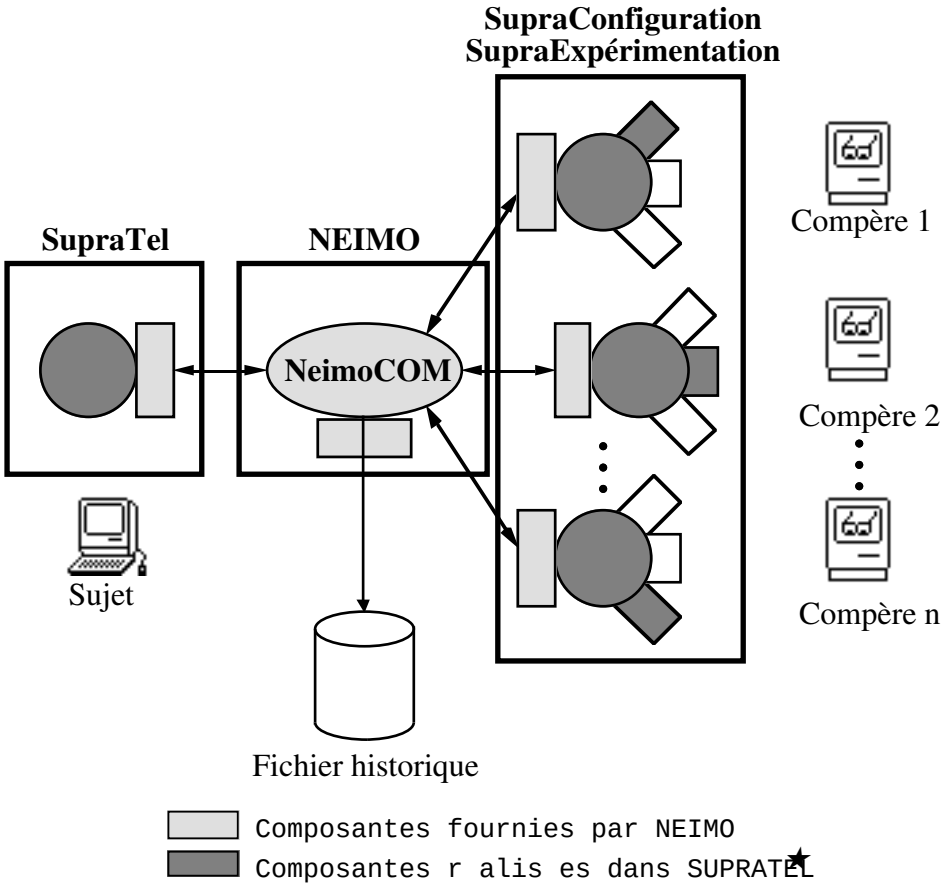


Figure 1.7 : La structure des applications sujet et compères dans SUPRATEL ★.

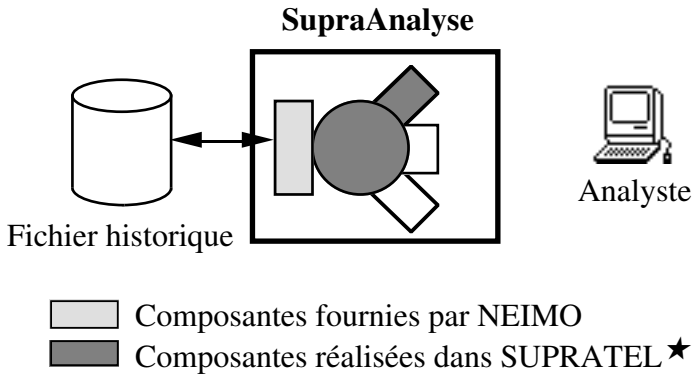


Figure 1.8 : La structure d'une application analyste dans SUPRATEL ★.

Comme le montrent les figures 1.7 et 1.8, le projet SUPRATEL ★ se décompose en quatre applications complémentaires qu'il faut développer pour remplir les termes du contrat qui lie le LGI-IMAG et le CCETT [Lischetti 94a] :

- L'*application SupraTel* comprend l'interface homme-machine à évaluer, une simulation des services de communication et une connexion à NEIMO. Sur le plan matériel, la station multimédia[7] du sujet comprend un écran, un clavier, une souris, un microphone, un haut-parleur et une caméra. La modalité[8] observée se limite au geste en deux dimensions, résultant de l'utilisation du clavier et de la souris.

- L'*application SupraConfiguration* permet la configuration et la conduite de la session d'expérimentation sous le contrôle du supercompère. Cet acteur se distingue des autres compères en raison de son rôle particulier ; il n'est pas exclu qu'il assume aussi le rôle classique d'un compère.

- L'*application SupraExpérimentation* permet la simulation des services manquants à SupraTel, l'observation du sujet par les compères et l'annotation des sessions d'expérimentations. Chaque compère se voit attribuer par le supercompère un ou plusieurs rôles de simulation, d'observation ou d'annotation, par exemple la réception d'un appel téléphonique, ou encore le contrôle du respect du scénario imposé au sujet.

- L'*application SupraAnalyse* permet à l'analyste d'exploiter les données comportementales du sujet et les observations des compères enregistrées lors de la session d'expérimentation.

[7]Le terme multimédia indique que le système exploite différents supports techniques de l'information.
[8]Une modalité désigne une catégorie de qualités sensorielles.

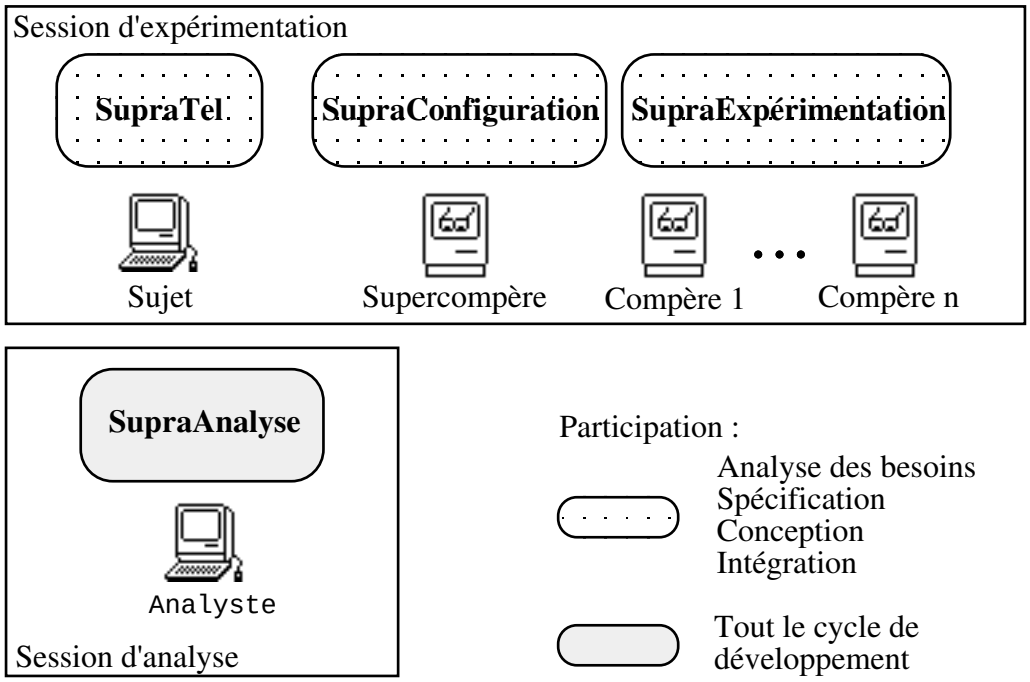


Figure 1.9 : Ma contribution sur l'ensemble du projet SUPRATEL ★.

La figure 1.9 précise ma contribution à ce projet. Concernant les applications SupraTel, SupraConfiguration et SupraExpérimentation (qui sont nécessaires à la session d'expérimentation), j'ai participé à :

- l'analyse des besoins,
- la définition du cahier des charges,
- les spécifications externes des interfaces homme-machine du sujet et des compères,
- la conception globale de l'architecture,
- aux étapes de tests (test du système et test d'acceptation).

Ce travail a été réalisé en liaison avec un groupe de quatre étudiants en DESS-GI[9] dans le cadre de leur projet de fin d'études. Je n'ai pas participé au codage de ces applications.

Concernant SupraAnalyse, j'ai eu à charge tout le développement de l'application, avec le fichier historique comme point-clé de dépendance avec les étudiants de DESS-GI. C'est pourquoi, j'ai défini conjointement avec les étudiants de DESS-GI, une spécification précise de l'interface logicielle pour les données enregistrées. Il est à noter que j'ai porté beaucoup d'attention à cette première étude car les informations choisies (type, granularité, etc.) conditionnent les possibilités offertes a posteriori par l'analyse automatique. Il est important aussi de mettre en œuvre des méthodes de capture efficaces pour réduire la quantité de données

[9]Diplôme d'Etudes Supérieures Spécialisées Option Génie Informatique.

à stocker et à analyser. Notre démarche coopérative s'est révélée avantageuse pour notre client (le CCETT) car le fait d'aborder de pair les deux parties (session d'expérimentation et session d'analyse) permet l'élaboration d'un cahier des charges commun pour l'ensemble du projet. Cette étude nous a permis de proposer une offre globale pour l'évaluation d'une interface utilisateur.

1.5. Les limites de ma contribution

L'automatisation complète des activités d'un ergonome, qui exige la modélisation des tâches d'analyse, n'est pas l'objet de mon travail. La modélisation constitue un exercice délicat, complexe et relativement prospectif. Actuellement, les ergonomes mettent en œuvre une forte activité mentale et s'appuient sur leur savoir-faire expert pour apprécier et qualifier le comportement de l'utilisateur face à l'interface homme-machine. Forte de cette constatation, j'ai choisi de conduire le projet en adoptant une vue assez large lors de l'étude de leurs besoins.

Ce travail m'a permis d'élaborer des propositions pour un premier outil d'aide à l'analyse des données comportementales. En l'absence de possibilité de comparaison, mes suggestions ont servi à établir le dialogue avec les ergonomes du CCETT. J'ai ensuite modéré la portée de mon intervention dans la phase de conception en restreignant mon étude et en optant pour une réalisation plus en profondeur. En conséquence, je n'ai retenu que certaines fonctions et certaines solutions parmi toutes celles envisagées. Il est clair que le temps m'aurait manqué pour mettre en œuvre tous les services proposés dans le cahier des charges. Dans ces conditions, il convenait de ne réaliser complètement que certains services de base et de concevoir un noyau logiciel capable d'accueillir facilement des services complémentaires.

Dans ce cadre, SupraAnalyse se positionne comme un outil informatique dont les services sont susceptibles d'évoluer pour s'adapter aux besoins des ergonomes. Mes suggestions initiales ainsi que les solutions retenues en phase de conception et de réalisation sont exposées dans le chapitre 4.

Le système NEIMO a été conçu dans la perspective d'offrir aux concepteurs d'interfaces homme-machine une extension du Magicien d'Oz à l'étude de l'interaction multimodale. On trouvera des définitions précises relatives à la multimodalité dans [Nigay 94]. L'interaction multimodale se caractérise par la possibilité pour un utilisateur d'interagir avec un système informatique selon les modalités de la communication humaine : usage simultané du geste, de la langue naturelle orale et écrite, de la vision, etc. L'utilisateur est donc à même d'exploiter différentes voies de communication. Le système doit alors être capable de représenter et de manipuler des informations de différents niveaux d'abstraction [Balbo 93b], [Gourdol 92]. Il

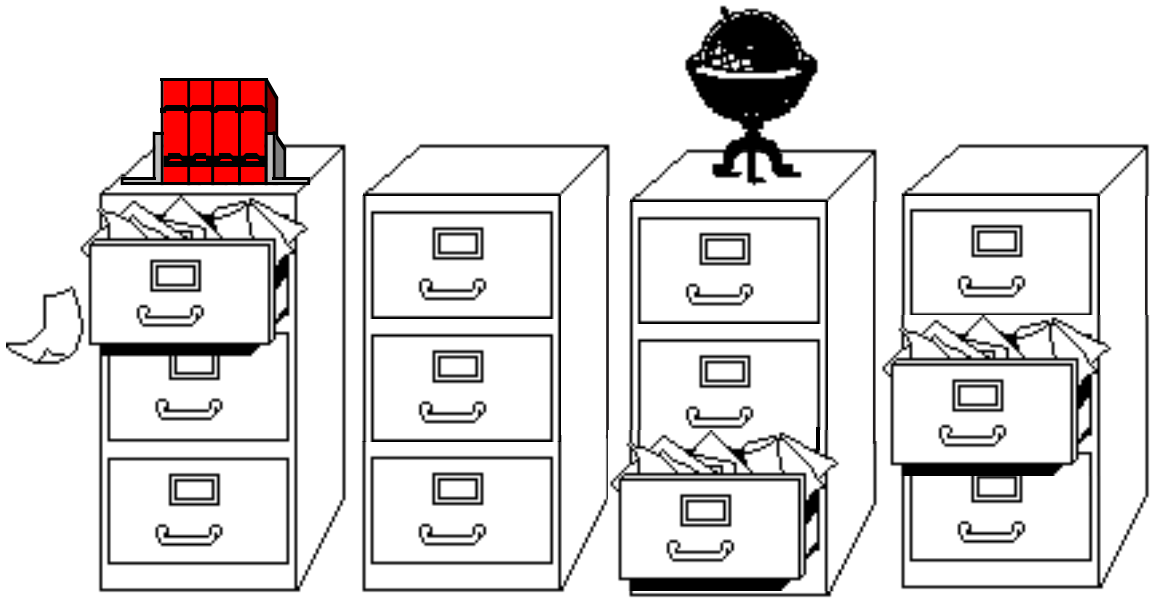
doit aussi être équipé du matériel et des logiciels adaptés à l'acquisition, à la restitution et à la compréhension d'énoncés multimodaux (microphones, caméras, synthétiseur de sons, ..). Il est à noter que la plate-forme n'a jamais été appliquée pour évaluer des interfaces multimodales. En raison des difficultés techniques spécifiques à la multimodalité et de son aspect novateur, le projet SUPRATEL ★ n'aborde que quelques points liés aux multiples voies de la communication humaine. Nous précisons dès à présent l'usage des modalités retenues.

Nous rappelons que l'interface homme-machine évaluée est celle d'un terminal de télécommunication multiservice. En conséquence, l'utilisateur emploie les modalités de la communication humaine car il utilise des services de télécommunication (téléphone, visiophone). Quant aux données vues par le système complet (NEIMO et les quatre applications), nous pouvons considérer qu'elles sont multimédias et hétérogènes. En effet, les informations traitées par les diverses applications concernent des structures de données spécifiques aux applications (par exemple, les Apple Events, le format QuickTime). Au sens propre du terme et du point de vue de l'interface homme-machine évaluée, l'interaction est strictement monomodale car il s'agit pour l'utilisateur d'employer la manipulation directe sur une interface purement graphique. Il en est de même pour toutes les interfaces homme-machine des compères et de l'analyste.

Nous venons de décrire le contexte applicatif, nous allons maintenant voir comment il s'inscrit dans le domaine de recherche.

Chapitre 2

Le domaine de recherche



Ce chapitre dresse l'état de l'art dans le domaine de la conception des interfaces homme-machine, qui concerne à la fois les ergonomes et les informaticiens. Je présente au paragraphe 2.1 l'apport des sciences cognitives dans ce domaine. Nous verrons que l'évaluation ergonomique des interfaces homme-machine devient une étape incontournable. Les différentes méthodes sont exposées en 2.2. Le paragraphe 2.3 présente un état de l'art de l'automatisation de l'évaluation ergonomique dans laquelle NEIMO s'inscrit.

2.1. La conception d'interfaces homme-machine et l'apport des sciences cognitives

Au sens large, une interface est un dispositif qui sert de limite commune à plusieurs entités communicantes. Les trois entités concernées dans le cas de la Communication Homme-Machine sont l'utilisateur, le système et leur environnement commun coopérant dans l'exécution d'une tâche donnée. La conception d'une interface homme-machine suppose donc d'aborder deux problèmes distincts : celui des techniques logicielles et celui des sciences cognitives, qui intéressent d'une part les informaticiens et d'autre part les ergonomes et les psychologues. Chez les informaticiens, l'interface homme-machine est considérée comme un assemblage de composants logiciels et matériels qui permet l'accomplissement de tâches avec le concours d'une machine informatique. Les ergonomes et les psychologues définissent l'interaction homme-machine comme l'ensemble des phénomènes physiques et cognitifs intervenant dans la réalisation de tâches informatisées.

Il est important de considérer les deux points de vue, sciences cognitives et techniques informatiques, et de concevoir l'interface homme-machine sous le double aspect logiciel et ergonomie. Malheureusement, jusqu'à présent, les équipes de recherche ont travaillé de façon cloisonnée. C'est pourquoi, parmi les différents ouvrages qui traitent de l'ergonomie des logiciels [Scapin 86, Barthet 88, Séguret-Geye 89, Coutaz 90, Meinadier 91], chacun apporte un éclairage particulier lié au public visé (ergonomes, informaticiens, ...) et à leur sujet d'intérêt (interfaces de dialogue, méthodes, architecture, etc.). Cependant, il ressort de l'ensemble un consensus certain sur les recommandations ergonomiques pour la définition des interfaces de dialogue et sur la nécessaire participation active des utilisateurs dans le processus

de conception. Selon J. Coutaz, “l'objectif du concepteur de l'interface utilisateur se doit de déterminer les limites du chemin mental de l'utilisateur en s'appuyant sur le domaine des sciences cognitives pour que l'informatique se plie aux exigences de l'utilisateur et non l'inverse car c'est le seul gage d'une interaction avec succès. Pour les informaticiens, il est nécessaire de s'ouvrir aux concepts de l'ergonomie cognitive pour construire des systèmes logiciels d'interaction capables de répondre aux principes de l'ergonomie” [Coutaz 90]. Plus généralement, on parle maintenant de conception d'interface homme-machine centrée sur l'utilisateur[10] car l'intégration de l'utilisateur doit constamment servir de principe directeur à la conception.

Les disciplines informatiques et cognitives tentent aujourd'hui de coordonner leurs actions pour servir de fondement à la conception des interfaces utilisateur. Mais, leurs théories et leurs modèles sont encore trop réducteurs pour aboutir à une conception d'interface homme-machine sans défaut dès la première tentative. En conséquence, dans l’état actuel des connaissances, le schéma “conception-réalisation-évaluation” prôné en génie logiciel, constitue la seule approche réaliste au développement des interfaces utilisateur. Les auteurs des ouvrages cités ci-dessus proposent eux aussi la mise en place d'un processus de conception itérative des interfaces homme-machine, dans lequel alternent des phases de développement et des phases d'évaluation (sur une maquette, sur des réalisations partielles, lors d'une recette ou sur un site pilote). Le résultat d'une évaluation peut conduire à des retours arrière dans le cycle de développement du logiciel (en V, en spirale, etc.). Dans tous les cas, l'évaluation de la qualité ergonomique ou l’utilisabilité d’un logiciel devient une étape incontournable et doit être régie par des techniques et des outils appropriés.

2.2. Evaluation des interfaces homme-machine : un état de l'art

L'évaluation de la qualité ergonomique d'un système consiste à estimer la conformité des performances effectives avec les performances désirées du système [Dowell 89]. Les performances observées encapsulent celles de l’utilisateur “augmenté” du système informatique matériel et logiciel.

[10]En anglais, on parle de “user-centered”.

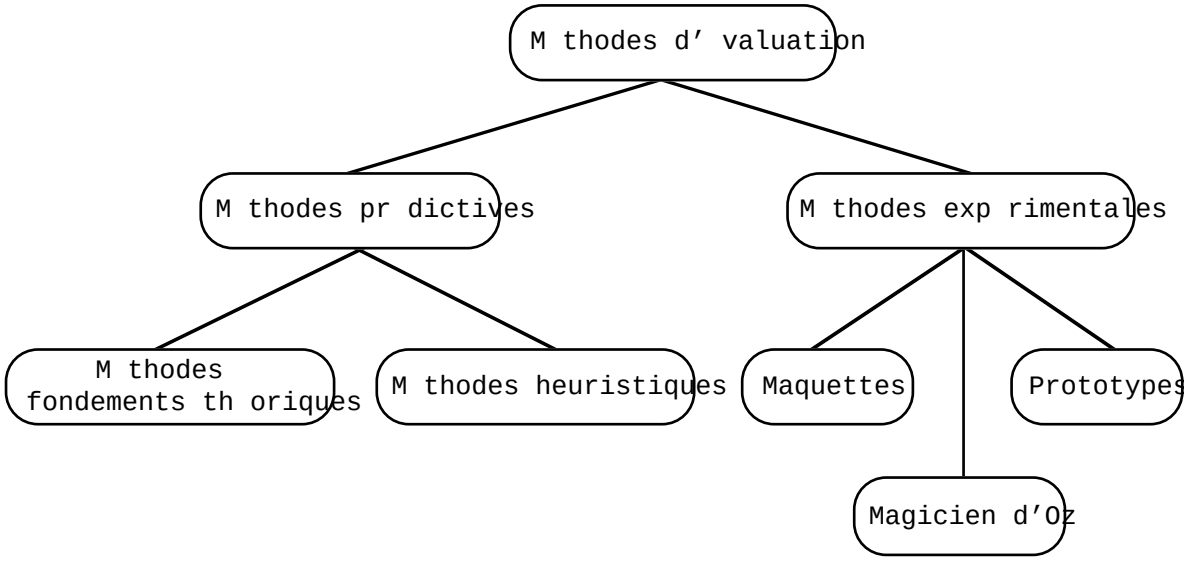


Figure 2.1 : Une classification des méthodes d’évaluation des interfaces homme-machine.

Comme le montre la figure 2.1, l’évaluation des interfaces utilisateur se pratique selon deux approches complémentaires : la simulation qui s’appuie sur des techniques prédictives et l’observation qui a recours à l’expérimentation.

- Les méthodes prédictives (ou analytiques) qui s’appuient sur les théories fondamentales de la psychologie cognitive ne nécessitent ni la présence de l'utilisateur final, ni la réalisation, même partielle, du système. Elles conviennent donc aux toutes premières étapes du cycle de vie d'un logiciel : l'analyse des besoins et la conception.

- Les méthodes expérimentales exploitent des données observées sur le cas réel. Elles imposent la présence d'utilisateurs et la réalisation ou la simulation de tout ou partie du système. Ce type d'évaluation semble tout destiné à intervenir en fin de cycle dans un modèle en V, ou bien en fin de sous-cycle dans un développement en spirale. Bien que nécessitant la réalisation partielle du système, l'application des méthodes expérimentales peut être étendue aux premières étapes du cycle de vie du logiciel. Par exemple, le support de l'évaluation peut prendre la forme d’enchaînement d’écrans esquissés sur papier (“scénarimage”) ou produits automatiquement par un générateur d’interfaces de type “présentation” [Nigay 94].

De manière générale, les méthodes prédictives et expérimentales se complètent et peuvent se pratiquer tout au long du processus de développement. La pratique itérative "tests-corrections" définit le fondement de l'évaluation "formative" [Hix 93] : chaque évaluation fournit de nouveaux enseignements qui, après intégration, conduisent à une nouvelle version du produit. La conception itérative du produit convient au processus de développement en spirale mais elle s'inscrit également dans un cycle de vie en V avec retours arrière. Ce cycle de vie du logiciel est décrit au chapitre 3. A l'évaluation formative, on oppose l'évaluation "summative" pratiquée une seule fois en fin de mise au point ou pour comparer des logiciels existants [Hix 93]. Nous développons plus en détail les deux techniques, prédictives et expérimentales, dans les paragraphes 2.2.1 et 2.2.2 qui suivent.

2.2.1. Les méthodes prédictives

Les méthodes prédictives permettent, à partir d'une description du système, voire de l'utilisateur, d'identifier des problèmes potentiels d'utilisabilité. Selon que les modèles sont heuristiques ou théoriques, les descriptions utilisées sont informelles, semi-formelles ou formelles ; l'évaluation est dite heuristique ou à fondements théoriques.

L'évaluation heuristique d'une interface homme-machine s'appuie sur le savoir expert en ergonomie cognitive de l'évaluateur. A partir de la description semi-formelle ou informelle du système, l'évaluateur recherche directement dans l'interface les propriétés qu'il sait par expérience mettre en défaut les principes ergonomiques reconnus. Il estime ainsi les performances du système en s'appuyant sur des guides [Smith 86][11] ou des ensembles de règles simplifiées comme celles de [Scapin 86], [Barthet 88] et [Nielsen 90]. Par exemple : "La gestion des erreurs doit réduire les occasions d'erreur et toute erreur doit être détectable dès son occurrence et doit pouvoir être corrigée. Elle concerne tous les moyens permettant d'une part d'éviter ou de réduire les erreurs, et d'autre part de les corriger lorsqu'elles surviennent". La méthode dite "cognitive walkthrough" [Lewis 90] structure l'usage de ces guides en proposant une liste de questions-types. Par exemple : "Pour cette situation, quelle action atomique l'utilisateur doit-il faire ?". Malgré tous ces guides, l'évaluation heuristique reste un exercice difficile que seuls les spécialistes en ergonomie sont capables de mener à bien : les règles sont trop nombreuses, parfois contradictoires, ou encore trop généralisées pour être applicables directement.

[11] Le rapport [Smith 86] contient 944 recommandations consignées dans un document de 480 pages!

L'évaluation à fondements théoriques met en application des modèles qui s'appuient sur une représentation formelle et explicite de l'utilisateur ou de l'interface, ou encore de la tâche. Pour illustrer notre présentation, nous décrivons brièvement trois exemples : le modèle théorique ICS (Interacting Cognitive Subsystems) [Barnard 87], le modèle Programmable User Models (PUM) [Young 90] et la "Théorie de la Complexité Cognitive" ou CCT (Cognitive Complexity Theory) [Kieras 85].

ICS axe son étude sur le comportement cognitif. Il prédit l'utilisabilité d'un système en évaluant la charge cognitive de l'individu lors de l'accomplissement d'une tâche. ICS modélise les activités humaines sous forme d'une architecture multiprocesseurs composée de sous-systèmes reliés sur un bus commun de communication. Ces sous-systèmes effectuent leurs traitements en parallèle sur des informations locales. ICS distingue huit sous-systèmes spécialisés chargés d'un traitement particulier (le son, la vue, le langage, les relations spatiales, etc.). ICS et les principes cognitifs sous-jacents ont fait l'objet d'une mise en œuvre sous forme d'un système expert. A partir de la description de la tâche à réaliser au moyen d'une interface homme-machine donnée, le système ICS identifie les sous-systèmes et les échanges d'information impliqués dans l'accomplissement de cette tâche. Il est alors capable de détecter les cas potentiels de surcharge cognitive.

PUM restreint son domaine d'étude à la planification avec un modèle dynamique du plan de résolution[12] de l'utilisateur. Pour ce faire, le concepteur décrit, à l'aide d'un langage approprié, les connaissances que l'utilisateur doit posséder pour accomplir une classe de tâches représentatives. Cette description est compilée sous forme de règles exécutables qui représentent l'aptitude supposée de l'utilisateur à accomplir les tâches données. L'exécution des règles fournit une simulation du comportement cognitif de l'utilisateur, c'est-à-dire de son plan de résolution. Une planification infructueuse doit attirer l'attention du concepteur sur un défaut potentiel d'utilisabilité ou une difficulté d'apprentissage.

Contrairement à PUM qui construit dynamiquement un plan de résolution, GOMS [Card 83] et CCT sont basées sur une décomposition explicite et statique des tâches de l'utilisateur en sous-tâches. Cette représentation hiérarchique, censée représenter l'approche de résolution de l'utilisateur, suppose d'avoir préalablement modélisé les activités de l'utilisateur accomplissant une série de tâches sur le système considéré. Le GOMS original est un modèle purement prédictif qui ne tient pas compte des erreurs humaines. Plusieurs études ont cherché à étendre GOMS à la prédiction des erreurs, notamment celles liées à la surcharge cognitive [Lerch 89].

[12]Le plan de résolution est le schéma mental développé par l'utilisateur pour accomplir une suite de tâches donnée.

CCT permet de comparer les mérites relatifs de plusieurs conceptions d'une même interface homme-machine en termes de transfert de connaissances et de difficulté d'apprentissage. A partir d'un modèle de la tâche et d'un modèle du processus mental de l'utilisateur décrits sous formes de règles, la complexité de l'interface est évaluée en fonction de la quantité, du contenu et de la structure des connaissances requises pour utiliser efficacement l'interface. L'estimation s'appuie sur des "métriques" tels que le nombre total de règles nécessaires à la modélisation de la tâche, etc.

L'intérêt majeur des techniques prédictives est leur capacité à intervenir très tôt dans le cycle de développement. Cette caractéristique permet de détecter des anomalies de conception de l'interface utilisateur avant même sa mise en œuvre. D'un autre côté, l'établissement et l'interprétation de tels modèles sont souvent complexes et peuvent s'avérer aussi coûteux en temps qu'une réalisation de l'interface. De plus, [Pollier 91] et [Nielsen 90] affirment qu'une évaluation heuristique est incomplète : au moins trois évaluateurs sont nécessaires pour déceler seulement la moitié des problèmes d'utilisabilité potentiels. En outre, les données nécessaires au modèle reposent sur des hypothèses théoriques et non sur des données réelles. Comme dans d'autres domaines, les approches formelles sont réductrices. Les méthodes expérimentales tentent d'apporter une solution à ces limites en raisonnant directement sur des données observées.

2.2.2. Les méthodes expérimentales

Les approches expérimentales basées sur l'observation constituent la seconde stratégie d'évaluation en interface homme-machine. Contrairement aux méthodes prédictives, elles requièrent la présence d'une population représentative du futur utilisateur et la réalisation ou la simulation du système à évaluer. En échange, elles fournissent des données réelles centrées sur le comportement perceptible de l'utilisateur lorsqu'il accomplit des tâches déterminées et interagit avec l'interface homme-machine à évaluer. Les hypothèses de l'approche prédictive sont alors remplacées par des faits.

Le système faisant l'objet d'une évaluation expérimentale peut prendre diverses formes : maquette papier, maquette informatique, prototype, Magicien d'Oz.

Les premiers systèmes à Magicien d'Oz ont été conçus pour l'étude des interfaces utilisateur exploitant le langage naturel dans des systèmes de recherche d'informations. Des services d'informations téléphoniques tels que des annuaires, des services de renseignements et de réservation de trains ou d'avions ont été les premiers champs d'exploration [Fraser 92], [Richards 84]. Le principe est assez simple : le compère répond aux appels téléphoniques et sa

voix déguisée fait croire à son correspondant qu'il parle à un système automatique d'information. Les dialogues sont enregistrés sur une bande audio et subissent une analyse a posteriori qui permet de rassembler des corpus de vocabulaire pour améliorer un reconnaisseur de langage naturel écrit ou parlé. Ces systèmes ont permis de tirer les premiers enseignements sur les utilisateurs et sur les compères.

Dans la plupart des cas, les données comportementales de l'utilisateur observé sont enregistrées sur support audiovisuel. Bien qu'elles ne concernent généralement qu'une seule modalité de communication (le graphique dans le cas d'une interface homme-machine ou la parole dans le cas d'un système de renseignements téléphonique), ces données représentent déjà un volume d'informations très important. C'est pourquoi la tâche la plus lourde de l'évaluation est le dépouillement manuel du très gros volume de données ; ceci requiert de la part des ergonomes patience et compétence. Lors des tests auxquels j'ai assisté dans le laboratoire d'ergonomie du CCETT, un seul utilisateur a produit un enregistrement audiovisuel de trois heures et huit expérimentations ont eu lieu pour évaluer le produit dans la phase de conception. Deux ergonomes et une lexicologue ont participé aux tests pendant trois jours.

Il est clair qu'à l'heure actuelle, l'une des caractéristiques principales de ce type d'évaluation est l'importance du rôle joué par l'évaluateur, généralement un expert ergonome. Celui-ci porte un jugement sur l'interface homme-machine en exploitant des données qui renseignent sur le comportement des utilisateurs en situation d'interaction. Le recueil des données comportementales s'effectue sur support audiovisuel, et/ou par capture automatique d'actions[13], sur des interviews et des questionnaires. Les données observées font ensuite l'objet d'une analyse. Celle-ci porte généralement sur les chemins suivis et les procédures adoptées par les utilisateurs, les temps d'exécution, la nature, la cause et la fréquence des incidents [Valentin 93]. Elle conduit à une synthèse qui indique notamment le degré de gravité des incidents et si possible, des recommandations.

En dehors de la complexité du domaine, on se heurte à des contraintes non scientifiques. Le cas du CCETT montre en particulier l'investissement humain nécessaire : la présence inévitable de spécialistes, le temps consacré et la compétence requise lors de la capture et de l'analyse des données. Toutes ces contraintes rendent les techniques expérimentales itératives assez longues et coûteuses. L'automatisation, même partielle, d'une évaluation laisse présager des gains considérables. Les systèmes à Magicien d'Oz sont susceptibles d'évoluer dans ce sens. Malheureusement, aujourd'hui, peu d'entre eux effectuent une capture entièrement électronique des actes comportementaux et encore moins progressent dans l'aide à l'analyse. Dans tous les

[13]En anglais, on parle de "monitoring".

cas, les outils d'aide à l'analyse des données comportementales sont limités et sont rarement intégrés dès le début dans la plate-forme Magicien d'Oz. De plus, la plupart des systèmes actuels ont été développés pour répondre à un besoin d'étude particulier et ne sont pas réutilisables. Nous allons voir comment l'automatisation de l'évaluation est abordée en général.

2.3. L'évaluation ergonomique automatique

Dans la littérature, les classifications pour les méthodes d'évaluation abordent peu la dimension d'automatisation des processus d'évaluation. [Balbo 92a] a choisi de les classer selon les niveaux d'automatisation offerts par les outils d'aide à l'évaluation. Le tableau 2.2 présente les quatre types différents proposés.

Type d'évaluation	Capture	Détection	Correction/ Explication
non automatique			
à capture automatique	✓		
à analyse automatique	✓	✓	
à critique automatique	✓	✓	✓

Figure 2.2 : Une classification de l'évaluation automatique des interfaces homme-machine.

Les techniques non automatisées utilisent les méthodes heuristiques que nous avons vues au paragraphe 2.2.1 ; elles reposent entièrement sur la compétence des évaluateurs.

Les techniques avec capture automatique des actions de l'utilisateur utilisent un dispositif matériel ou logiciel. Ce dernier est parfois doublé d'un enregistrement audiovisuel analogique qui permet d'enregistrer, dater et reproduire la session de travail originale. S'appuyant sur cette technique, MRP ou Multiple Repeating Pattern détecte les répétitions de schémas d'actions de l'utilisateur qui servent ensuite de support d'étude à un évaluateur [Siochi 91]. Les techniques audiovisuelles de [Hammontree 92] laissent toute la liberté à l'évaluateur de revoir la séquence d'actions de l'utilisateur autant de fois que nécessaire. Malheureusement, ces systèmes sont généralement conçus pour des applications spécifiques et leurs domaines d'étude sont trop restreints. De plus, les données brutes issues des enregistrements audiovisuels et/ou de la capture numérique sont trop abondantes et par conséquent, difficiles à analyser manuellement. Par exemple, Human Interface Monitoring System ou HIMS [Theaker 89], destiné aux interfaces graphiques, enregistre directement les signaux vidéo envoyés à la console (non pas l'image de l'écran). Il capture ainsi jusqu'à 200 MO de données à la seconde [Balbo 92a].

Les techniques avec analyse automatique cherchent à préciser et localiser les problèmes d'utilisabilité. Elles reposent soit sur l'utilisation de méthodes formelles théoriques, basées sur une représentation de l'activité cognitive de l'utilisateur, soit sur l'approche expérimentale avec une capture logicielle des données d'observation. Dans le cas des méthodes théoriques, elles permettent de relever certains problèmes ou de comparer différentes conceptions d'une même interface, mais elles n'apportent aucune solution pour résoudre les problèmes rencontrés. Pour ce qui est de l'analyse automatique basée sur des données expérimentales, les techniques à Magicien d'Oz sont une voie de recherche très prometteuse.

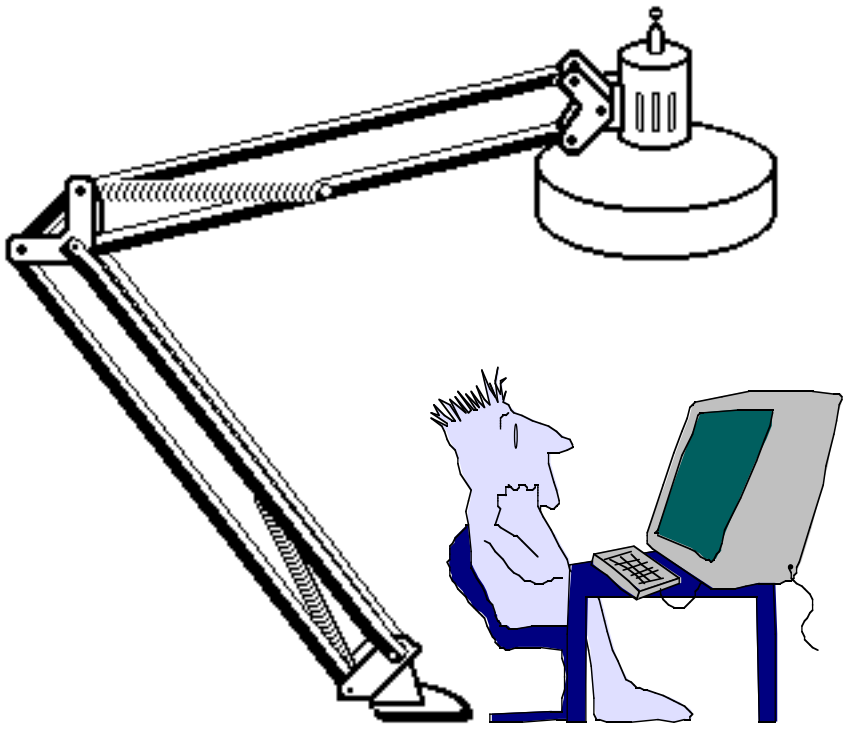
Les techniques capables d'effectuer la critique automatique d'une interface homme-machine sont encore peu développées. Les techniques prédictives employées par les systèmes experts KRI [Löwgren 90] et SYNOP [Kolski 89] sont peu satisfaisantes, car la portée des services offerts est limitée et la modélisation s'appuie sur des hypothèses réductrices. Dans le cas de KRI, qui présente une critique sous forme de commentaires et propose des améliorations, les indications fournies relèvent de l'aspect syntaxique de l'interaction. La modélisation de la tâche, et plus généralement toute activité cognitive d'ordre supérieur, sont ignorées et surtout, KRI exclut totalement l'utilisateur de son analyse.

Un système avec critique fondé sur les observations réelles produites par une évaluation expérimentale est en cours d'élaboration par l'équipe Interface Homme-Machine du LGI-IMAG [Balbo 92a]. Ce critique s'appuie sur un modèle de tâche qui définit la référence. Les actions physiques de l'utilisateur (touches du clavier, souris) permettent de construire le graphe de navigation réel. La comparaison entre le graphe de référence et la navigation effective tente de faire apparaître les anomalies de conception de l'interface utilisateur. Ce système ouvre des portes et montre la faisabilité de l'automatisation, mais sa portée reste limitée.

Consciente de l'apport que peut avoir une évaluation prolongée d'outils automatiques par rapport à la simple approche expérimentale, l'équipe interface homme-machine de J. Coutaz a développé NEIMO [Ambone 92a et 92b] en vue d'offrir la capture et l'analyse automatique et informatique des données d'observation pour des interfaces homme-machine multimodales.

Chapitre 3

La démarche de développement



L'objectif de ce chapitre est de présenter la démarche de développement retenue. Tout d'abord en 3.1, j'indique le cycle de vie du logiciel que j'ai adopté. Ma décision fait suite à une étude bibliographique dans le domaine du Génie Logiciel qui montre les caractéristiques des différents modèles. L'ensemble des documents produits est détaillé et les méthodes employées sont précisées. Les parties suivantes traitent des méthodes et des outils de spécification (3.2) et de conception des interfaces homme-machine (3.3 et 3.4). Pour chacun, je présente un descriptif qui conduit à exposer mes choix et j'émets une opinion critique basée sur mon expérience.

3.1. Le cycle de vie du logiciel

Le modèle en cascade initialement proposé par [Royce 70] a été affiné sous diverses formes avec notamment le modèle en V [McDermid 84] et plus récemment, le modèle en spirale [Boehm 88]. Ce paragraphe expose les avantages et les inconvénients de ces modèles.

3.1.1. Les modèles en cascade et en V

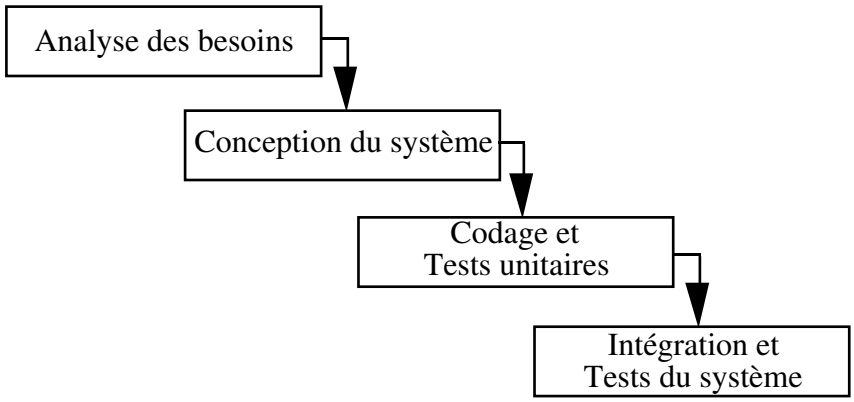


Figure 3.1 : Le modèle en cascade initial.

Dans sa première version, la décomposition du modèle en cascade représentée par la figure 3.1 comporte quatre étapes distinctes :

- L'analyse des besoins, faite en accord avec le client, permet d'établir les services requis du système et les contraintes de développement.
- La phase de spécification et de conception du système consiste à définir une solution matérielle et logicielle qui répond à l'analyse des besoins et aux contraintes.
- Le codage traduit les spécifications issues de l'étape de conception en code exécutable. Lors des tests unitaires, chaque module est testé séparément en vue de vérifier sa conformité avec les spécifications.
- Dans la dernière phase, les modules sont intégrés et testés ensemble pour constituer un système livré au client.

Dans la figure 3.1, les flèches traduisent le cheminement entre les étapes ; celles-ci s'exécutent en séquence.

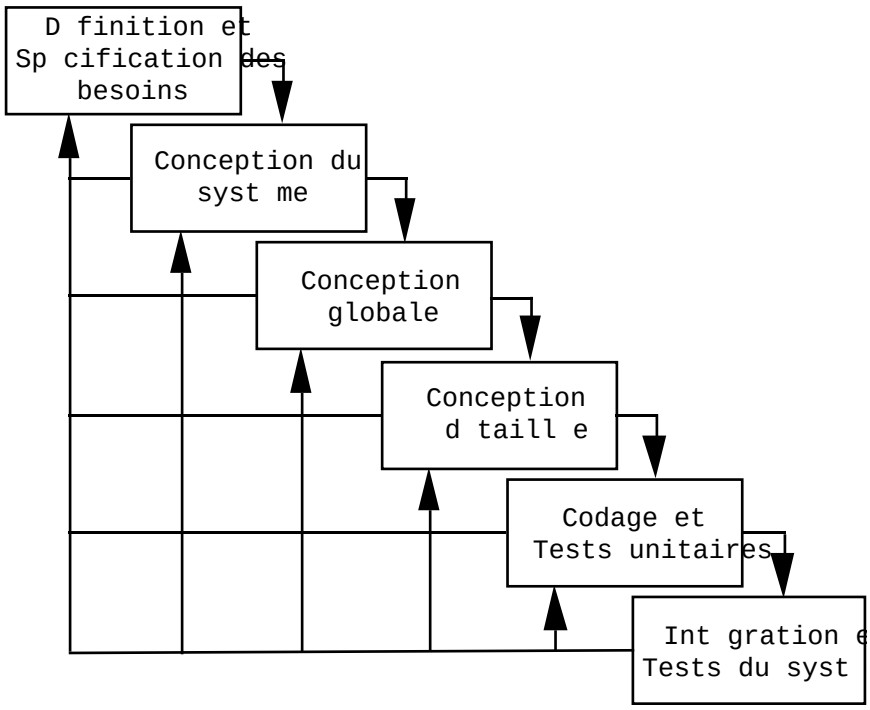


Figure 3.2 : Le modèle en cascade affiné.

Cette décomposition du modèle en cascade initial s'est révélée insuffisante. Comme le montre la figure 3.2, on a complété le modèle avec des étapes supplémentaires et rendu explicites les retours arrière. Les étapes complémentaires de conception globale et de conception détaillée ont été ajoutées pour préciser les tâches de conception logicielle qui précèdent l'activité de codage et éviter ainsi de traduire directement en langage exécutable les spécifications externes produites dans l'étape de conception. Les retours arrière permettent de revenir sur les résultats des étapes antérieures, autorisant ainsi un processus de développement itératif. Les flèches sur la figure 3.2 traduisent les cheminements possibles en avant et en arrière. Par exemple, depuis l'étape d'intégration et de tests du système, il est possible de revenir à l'étape d'implémentation mais également à l'une quelconque des étapes précédentes.

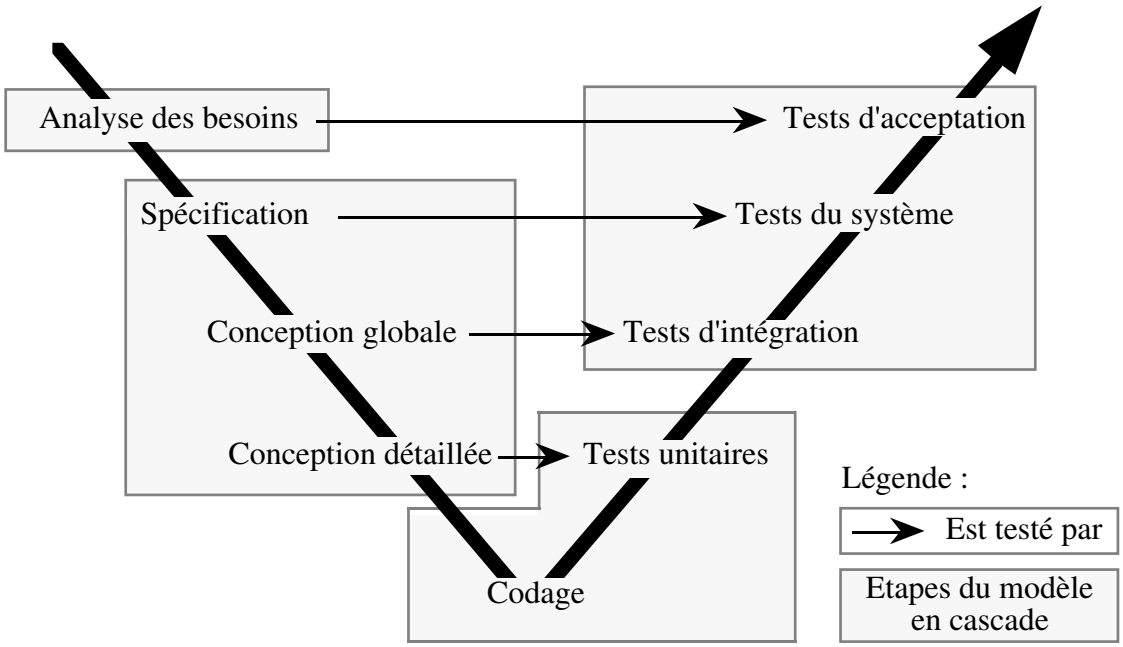


Figure 3.3 : Le modèle en V.

Le modèle en V présenté dans la figure 3.3 complète le modèle en cascade en rendant explicites les étapes de test [McDermid 84]. La flèche en trait épais et en forme de V traduit la séquence temporelle des étapes. La pente descendante du V reprend les étapes d'affinement progressif du modèle en cascade (carrés grisés de la figure 3.3) tandis que la pente ascendante leur associe un ensemble de tests. Ces derniers permettent de vérifier la conformité du produit par rapport à une description de référence ou de valider l'adéquation du logiciel produit par rapport à la demande[14]. Les tests sont effectués en séquence une fois le codage réalisé ou lors de la descente, avant même le codage. Dans tous les cas, ils doivent être définis en même temps que leur correspondant dans la pente descendante.

Ce canevas du modèle "dirigé par les documents" [Boehm 88] convient au développement de projets bien cernés ayant des objectifs clairement définis. En effet, la définition des besoins et les spécifications externes sont figées dès les toutes premières étapes du processus de développement. Le critère de transition autorisant le passage d'une étape à l'autre est la mise à disposition d'un document complètement élaboré. Et la portée des retours arrière est limitée pour contrôler l'ensemble du processus de développement.

[14] La validation n'est pas une caractéristique mesurable et relève d'un jugement subjectif [Boehm 81].

Dans le cas de systèmes à risques (objectifs mal définis comme en recherche, sécurité essentielle comme en contrôle de processus, etc.), il est préférable d'éviter l'emploi du modèle en V et d'adopter la technique du prototypage ou une approche exploratoire. Le modèle en spirale permet de faire une validation progressive et itérative du produit.

3.1.2. Le modèle en spirale

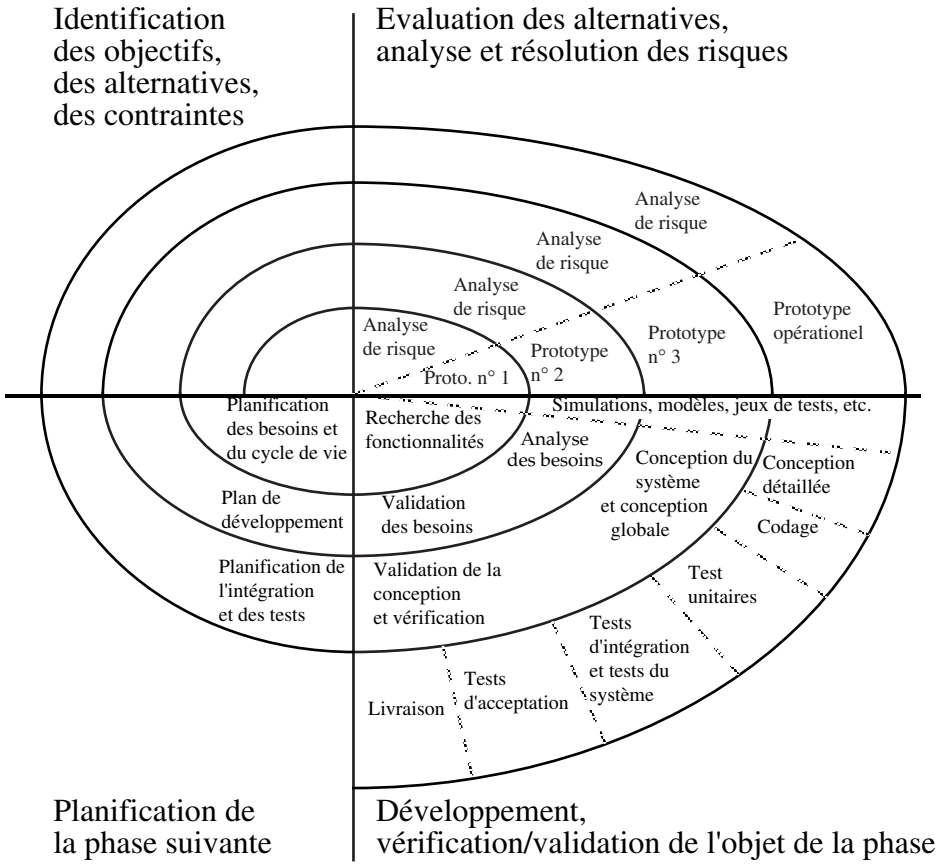


Figure 3.4 : Le modèle en spirale.

Comme le montre la figure 3.4, pour chaque cycle de spirale (ou phase) le modèle rend explicites quatre activités fondamentales :

- l'identification des objectifs de la phase, l'alternative pour atteindre ces objectifs et leurs contraintes,
- l'analyse et la résolution des risques,
- le développement et la vérification/validation de l'objet de la phase,
- la planification de la phase suivante.

Ce modèle est bien adapté aux projets pour lesquels les aboutissants sont peu connus ou mal maîtrisés, car il permet l'analyse explicite des facteurs de risques et des solutions. En tant que métamodèle, il a l'avantage d'être adaptable aux spécificités de chaque projet [Boehm 88]. C'est précisément le cas du projet de recherche dans lequel s'inscrit NEIMO.

3.1.3. NEIMO : un développement en spirale

Dans le projet NEIMO, chaque tour de spirale est constitué d'un cycle en V d'une durée d'une année au terme de laquelle est présenté un démonstrateur incrémental. Les propriétés du métamodèle du cycle en spirale sont ainsi mises en pratique. L'objectif visé lors du choix des cycles du projet NEIMO est d'allier les bénéfices des deux processus de développement en spirale et en V.

L'approche itérative adoptée avec le cycle en spirale permet de ne pas réaliser la spécification complète du produit trop tôt dans le cycle de vie. L'évolution du projet procède par phases lors desquelles seulement certains composants sont développés à chaque tour de boucle. Le développement est incrémental et les hypothèses se vérifient au fur et à mesure de celui-ci. Dans le cas de NEIMO, chaque tour de boucle bien maîtrisé avec le cycle en V définit une étape en direction du produit final (enrichissement et affinement) et délivre un état d'avancement du projet. Sa nature dépend du cas à traiter.

Le projet NEIMO a déjà subi deux tours de spirale. Le premier a permis la réalisation du noyau NeimoCom et des bibliothèques de communication et leur mise en œuvre avec une première application de laboratoire[15] [Gandriau 93]. Faute de temps, cette dernière a abordé l'analyse des données sans l'approfondir. Elle a permis cependant de tester la connexion à la bibliothèque historique fournie par NEIMO.

Avec un deuxième tour de boucle, l'objectif est d'aller plus loin :

- éprouver la validité et la robustesse du noyau NeimoCom,
- faire un premier pas significatif vers l'analyse automatique des données recueillies.

L'application choisie est cette fois industrielle : il s'agit de notre projet SUPRATEL ★ lors duquel deux cycles en V sont menés en parallèle :

- l'observation et la capture des données comportementales pour un terminal de télécommunication,
- l'aide à l'analyse automatique de données comportementales, qui me concerne directement et que je détaille en 3.1.4.

[15] AppliLevelt, la première application test de laboratoire a été réalisée pour les psychologues et les ergonomes des équipes de Jean Caelen (ICP-INPG) et d'André Bisseret (INRIA-IMAG).

Dans un troisième tour de spirale, on envisage de :

- définir un protocole qui détermine les conditions d'accueil des logiciels à évaluer,
- doter NEIMO d'outils qui permettent aux expérimentateurs de paramétrer les conditions de l'expérience (scénarios),
- faire une réelle mise en œuvre de la plate-forme pour des interfaces multimodales car, nous l'avons vu, SUPRATEL ★ se limite à l'évaluation d'une interface multimédia (graphique et vidéo).

3.1.4. Mon modèle de développement

En ce qui concerne la partie analyse automatique, l'approche adoptée est incrémentale. Dans le cadre du projet SUPRATEL ★, on envisage uniquement le niveau fouineur, marqueur et synthétiseur des informations brutes stockées dans le fichier historique et la présentation des informations sous une forme adaptée à la tâche d'interprétation. Les autres étapes orientées vers l'automatisation de l'analyse (détection automatique de "pattern" de comportement, détection automatique de problèmes d'utilisabilité, explication et proposition d'actions correctives) requièrent la présence de compétences en psychologie et en ergonomie et nécessitent une collaboration avec d'autres équipes de recherche.

Conformément à ce cadre, mon projet d'aide à l'analyse des données comportementales suit le modèle en V présenté précédemment. Je reprends ici les différentes phases du cycle de vie du logiciel pour décrire ma propre démarche et dresser la liste des documents établis dans chacune des étapes. Il est précisé le rôle des documents, leur contenu et les méthodes complémentaires que j'ai utilisées.

Lors de la première étape du cycle en V, l'analyse des besoins donne lieu à l'élaboration d'un cahier des charges (CdC) [Lischetti 94a], document à valeur contractuelle entre le maître d'œuvre (J. Coutaz, responsable du projet) et le client (CCETT). La structure du cahier des charges du projet que j'ai adoptée suit la norme [IEEE 84] et regroupe à la fois la définition et la spécification des besoins. Le document détaille aussi bien les besoins fonctionnels qui décrivent les services attendus du système et les tâches qu'il doit permettre d'effectuer, que les besoins non fonctionnels qui définissent les contraintes matérielles ou logicielles et les qualités requises du système.

Contrairement à un cahier des charges qui se contente d'énoncer les besoins de manière informelle en langue naturelle, j'ai complété la spécification avec des schémas et des diagrammes formels en utilisant la méthode Object Modeling Technique (OMT) [Rumbaugh 91]. Afin de ne pas négliger l'utilisateur dans cette première phase, j'ai ajouté une description de l'espace des tâches proposé à l'utilisateur. J'ai utilisé deux méthodes complémentaires pour construire cette analyse de tâches. La première est basée sur une représentation graphique qui montre la décomposition des tâches sous la forme d'un arbre. Pour compléter cette vision statique, j'ai utilisé une notation qui permet de décrire la dynamique de l'activité. Il s'agit de User Action Notation (UAN), un langage habituellement utilisé pour la spécification des interfaces homme-machine [Hartson 92]. Nous verrons en détail ces méthodes dans le paragraphe 3.2 qui traite des outils de spécification et nous verrons leur apport dès la phase d'analyse des besoins.

Le document de spécifications externes (DSE) [Lischetti 94b] qui traduit la conception du système décrit le système tel qu'il sera perçu par l'utilisateur. Comme pour le cahier des charges, la description de l'interface homme-machine est généralement informelle, exprimée en langue naturelle et dans certains cas complétée de schémas d'écran. En ce qui me concerne, pour une meilleure compréhension entre le client et le concepteur, j'ai choisi d'utiliser comme dans l'étape précédente le langage de spécification User Action Notation (UAN). Dans le paragraphe 3.2.2 réservé à UAN, je précise ce qui a motivé mon choix, ainsi que les limites et les avantages d'une telle notation.

La conception globale constitue l'étape préliminaire orientée vers la mise en œuvre du logiciel. Elle précise l'architecture générale retenue. Quant à la conception détaillée, elle 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 utilisés. Pour ma partie, j'ai regroupé dans un seul document la conception globale et détaillée [Lischetti 94c] et j'ai basé ma réflexion sur la réutilisation logicielle et la modularité. Réalisant un logiciel interactif, mon architecture se base sur le principe de séparation modulaire entre le noyau fonctionnel qui réalise les services du domaine, et l'interface homme-machine qui gère les échanges avec l'utilisateur. Le modèle conceptuel d'architecture PAC [Coutaz 90] que j'ai retenu favorise l'application et le respect de ce principe de base. Il est décrit en détail dans le paragraphe 3.3 traitant des modèles d'architecture.

Le codage constitue la dernière étape de la pente descendante du modèle en V. Son résultat est le code exécutable du système. Pour s'interfacer avec l'environnement logiciel existant et pour les qualités propres à ce type de langage évolué, mon choix s'est porté sur le langage orienté objet C++. Le paragraphe 3.4 qui traite des outils de développement relate notre expérience avec le générateur d'interfaces MacroMind Director [MacroMind 91] et le squelette d'application MacApp [Schmucker 86a]. Il précise aussi l'environnement de programmation que j'ai finalement employé.

Sur la pente ascendante, un ensemble de tests mis en regard de chacune des étapes permet de vérifier et de valider celles-ci par rapport au code produit. Les tests unitaires qui correspondent à la phase de conception détaillée permettent de vérifier que les composants modulaires du système répondent à leurs spécifications. Les tests d'intégration liés à la phase de conception globale assurent que les modules réalisés indépendamment interagissent correctement. Les tests du système associés à la phase de conception contrôlent que tous les éléments de la solution exprimés dans le dossier de spécifications externes sont traités. Enfin, les tests d'acceptation correspondent à la phase de définition et de spécification des besoins. Ils 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.

Ayant présenté les étapes du processus de développement, il convient maintenant de préciser pour chacune d'elles les méthodes et outils employés.

3.2. Les outils et méthodes de spécification employés

Dans la phase d'analyse de besoins, je me suis appuyée sur OMT. J'ai utilisé UAN pour la formalisation conceptuelle et les spécifications externes. Le modèle d'architecture PAC m'a été utile pour la conception globale et la mise en œuvre s'est faite dans l'environnement de développement Symantec. Je vais reprendre chacun de ces points avec une analyse critique tirée de l'expérience.

3.2.1. OMT, une méthode intervenant dès la phase d'analyse des besoins

Présentation

Object Modeling Technique (OMT) est une méthode de développement logiciel qui exploite les concepts orientés objets [Rumbaugh 91]. Elle utilise une notation graphique indépendante du langage de programmation. Les caractéristiques de l'approche objet sont respectées : les classes, les instances de classe, le polymorphisme[16] et l'héritage[17] simple ou multiple.

L'approche autorise l'utilisation des mêmes concepts et des mêmes notations tout au long du cycle de développement qu'elle organise en 4 étapes : analyse ou spécification des objets, conception du système, conception des objets et enfin programmation. Ainsi, la méthode OMT se propose de guider le concepteur à travers les différentes phases du cycle de vie du logiciel en lui évitant la traduction de notations entre les différentes phases et en assurant la documentation de toutes les phases. Pour ce faire, OMT repose sur les mêmes concepts pour toutes les étapes du développement et suit trois schémas complémentaires : le schéma d'objets, le schéma dynamique et le schéma fonctionnel.

Le schéma d'objets présente les aspects statiques du système. Il modélise les concepts du domaine applicatif et inclut une description des propriétés et des comportements de ces concepts. Une classe définit un modèle d'objets et regroupe l'ensemble des objets correspondant à ce modèle. L'héritage entre les classes est possible. L'association entre classes est représentée par un modèle de liens entre objets. Une clé regroupe un ensemble d'attributs (l'ensemble des propriétés propres d'une classe) ; elle identifie chaque objet d'une classe ou chaque lien d'association. Des contraintes (sous forme d'équation ou en langue naturelle) peuvent être ajoutées pour restreindre les valeurs des attributs d'une entité ou d'un lien. L'agrégation permet d'élaborer des objets composés à partir d'autres objets ; un agrégat peut être traité comme un objet unique sans se soucier de sa composition. Un module est un constructeur logique pour regrouper des classes, des associations et des hiérarchies selon un point de vue particulier sur le système.

[16]Une même opération peut avoir des comportements différents pour des classes différentes.
[17]L'héritage permet le partage d'attributs et de méthodes entre des classes qui ont une relation hiérarchique.

Le modèle dynamique décrit l'interaction entre les objets. Élaboré à partir des notions d'événements et d'états, il décrit les séquences d'opérations réalisées en réponse aux stimuli extérieurs. L'événement intervient à un instant donné mais n'a pas de durée. Les événements sont regroupés en classes d'événements décrivant la structure et le comportement d'une catégorie d'événements. Un état est défini par son nom, la séquence d'événements qui le produit, la condition qui le caractérise et les événements acceptés dans cet état. L'opération déclenchée par chaque événement est précisée, ainsi que l'état atteint après l'exécution de cette opération. Pour chaque classe, on peut définir un diagramme d'état structuré selon des relations d'agrégation ou de spécialisation. Un diagramme d'état est un graphe dont les nœuds sont les états, les arcs les transitions et les événements des étiquettes portées par les arcs. Le schéma dynamique est en fait une collection de diagrammes d'états interagissant par des événements partagés.

Le modèle fonctionnel décrit les transformations des données à partir des relations existantes entre les données d'entrée et celles de sortie, en s'aidant des diagrammes de flots de données. Un tel diagramme est construit à partir de processus, de flots de données, d'acteurs et de mémoires de données. Les processus transforment les données, les acteurs sont des objets actifs qui consomment et produisent des données, les mémoires de données sont des objets passifs qui servent au stockage.

Critique

En conclusion, OMT est une méthode complète et riche qui applique le modèle à objet à tout le cycle de développement du logiciel : le schéma d'objet présente un modèle statique enrichi et affiné sous la forme des modèles dynamique et fonctionnel. Il reste ensuite à coder ce schéma dans le langage de programmation. La continuité de la démarche orientée objet lors des différentes phases du développement évite les sauts de notation qui engendrent fréquemment des pertes d'informations. Néanmoins, la richesse de la modélisation par objets et le degré d'abstraction des concepts (événements, diagrammes d'états, etc.) rend OMT plus particulièrement adapté à la phase d'analyse.

Comme dans la plupart des méthodes de spécification, l'utilisateur est souvent négligé à ce stade. C'est le cas par exemple du formalisme SADT associé à la méthode MERISE [Tardieu 83]. Pour ce qui est de OMT, la méthode ne prend pas en compte le dialogue homme-machine, caractéristique importante de mon projet. En conséquence, je n'ai utilisé qu'une partie de la méthode OMT dans l'analyse des besoins et j'ai défini le modèle à objet, sans décrire les modèles dynamique et fonctionnel préconisés par la méthode. J'ai complété le schéma d'objet défini avec OMT avec une description de l'espace des tâches proposé à l'utilisateur au moyen de la notation UAN. L'association de OMT et de UAN permet de prendre en compte le système mais aussi l'utilisateur. En cela, la modélisation des tâches que j'ai réalisée dès l'analyse des besoins est originale.

3.2.2. UAN, un langage de spécification conceptuelle

Présentation

Le langage User Action Notation (UAN) [Hartson 92] est un support à la spécification du comportement d'un système perceptible par son utilisateur.

La notation s'appuie sur une représentation tabulaire à trois colonnes (figure 3.5) qui permet de spécifier les actions physiques de l'utilisateur et du système. Ces actions sont regroupées en tâches. Une tâche peut se voir comme un objectif associé à un ensemble d'opérations organisées par des relations temporelles et logiques ; son exécution permet d'atteindre un état visé [Normand 92a]. Une tâche comprend trois facettes et chaque facette occupe une colonne. La colonne de gauche spécifie la suite temporelle des actions physiques que l'utilisateur doit effectuer pour accomplir la tâche. La colonne centrale décrit, en regard des actions utilisateur, les retours d'information fournis par le système. La colonne de droite sert à noter le changement d'état de variables pertinentes du système pour l'interface homme-machine.

Tâche :		
Action de l'utilisateur	Retour de l'interface	Etat de l'interface

Figure 3.5 : La notation UAN sous forme d'un tableau à trois colonnes.

UAN autorise la description de tâches élémentaires ou composées : la colonne de gauche d'une description UAN peut en effet comprendre des actions physiques et/ou des noms de tâche. Dans les deux cas, actions et tâches sont liées par des relations temporelles. Le tableau de la figure 3.6 présente les différentes possibilités. La description d'une tâche peut être assortie de préconditions. Les sous-tâches peuvent être décrites par décomposition successive jusqu'au niveau des actions physiques.

Relation temporelle	**Symbole UAN**	**Signification**
Séquence	A B	B ne peut être entreprise que si A est terminée.
Attente	A (t>n) B	L'exécution de B n'a lieu que si A est terminée depuis un temps t>n.
Disjonction répétée	(A | B)*	L'utilisateur a le choix entre l'exécution de A ou de B.
Indépendance d'ordre	A & B	L'utilisateur doit exécuter A et B mais dans l'ordre qui lui convient.
Entrelacement mono-directionnel	A <- B	L'exécution de la tâche A peut être interrompue par B, puis reprendre une fois B achevée.
Entrelacement bidirectionnel	A <-> B	A et B peuvent s'interrompre mutuellement.
Parallélisme	A || B	A et B peuvent s'exécuter simultanément.

Figure 3.6 : Les relations temporelles dans UAN.

UAN offre au concepteur la possibilité d'utiliser les niveaux de granularité voulus depuis les fonctions du système jusqu'à la description d'actions physiques. Une action physique se définit le plus souvent comme une opération indivisible que l'utilisateur applique à un dispositif d'entrée physique et qui a pour effet immédiat de modifier l'état du dispositif. Par exemple, enfoncer et relâcher une touche du clavier sont deux actions distinctes. Par extension, une action physique émanant du système est une opération indivisible sur le changement d'état perceptible du système.

Dans l'exemple de la figure 3.7, la tâche de sélection d'un objet est notée Sélectionner(objet) ; "Sélectionner" désigne le nom de la tâche et "objet" en est le paramètre. Pour accomplir cette tâche, l'utilisateur doit déplacer le curseur de la souris sur la représentation iconique de l'objet (~[icône(objet)]) puis enfoncer le bouton de la souris (M$^{\vee}$). Alors, le système réagit en mettant en évidence l'icône de l'objet (icône(objet)!) et, s'il existe un objet objet' dont l'icône est en évidence (" icône(objet')!:), alors celle-ci est éteinte (icône(objet')-!). Lorsque l'utilisateur relâche la souris (M$^{\wedge}$), le système ne fournit pas de retour d'information mais note dans une variable d'état (ObjetSélectionné) que "objet" est maintenant sélectionné.

Tâche : Sélectionner(objet)		
Action de l'utilisateur	Retour de l'interface	Etat de l'interface
~[icône(objet)] M$^{\vee}$ M$^{\wedge}$	icône(objet)!, $\forall$ icône(objet')!: icône(objet')-!	 ObjetSélectionné = objet

Figure 3.7 : La description UAN de la sélection d'un objet.

La notation permet de représenter les actions de type manipulation directe. Le symbole ~ de la figure 3.7 précédente en est un exemple. L'expression des formes élastiques et l'asservissement de fantômes aux déplacements de la souris, typiques de la manipulation directe, sont également prévus dans le langage.

Critique et solution

Il est important d'analyser UAN par rapport à sa puissance d'expression. Les niveaux de description permettent un affinement suffisant pour décrire les plans d'actions possibles en s'appuyant sur la décomposition hiérarchique des tâches. Les relations temporelles entre les tâches d'une même colonne sont clairement exprimées, mais il subsiste une ambiguïté pour les instructions situées dans des colonnes distinctes. Une nouvelle notation XUAN (Extended UAN) vise à combler ce manque de rigueur sémantique [Gray 94].

Les inconvénients généralement reprochés à tels formalismes sont les coûts d'apprentissage. L'expérience prouve que ce coût est acceptable en regard du gain qu'il est possible de dégager lors de son utilisation. En effet, l'approche systématique que UAN induit, amène le concepteur à la précision et à la mise en place d'hypothèses réalistes lors des spécifications externes. J'ai choisi d'étendre son utilisation à la phase d'analyse des besoins pour que les deux premières étapes du cycle de développement exploitent les mêmes concepts et les mêmes notations. Ma démarche se déroule en deux étapes. Tout d'abord, une première description est faite selon le niveau d'abstraction nécessaire à la phase d'analyse des besoins. Il s'agit d'une analyse basée sur une décomposition hiérarchique et conceptuelle des tâches en sous-tâches : la hiérarchie traduit l'organisation structurelle de l'espace de travail et l'analyse conceptuelle décrit la nature des concepts manipulés dans le domaine. Le graphe orienté de tâches obtenu peut ensuite être considéré et affiné selon une analyse descendante pour la définition des spécifications externes. Les tâches sont alors décomposées jusqu'au niveau des actions physiques.

Cette démarche donne de l'avance sur les spécifications externes et évalue très tôt la conformité de l'espace des tâches par rapport à celui attendu par le client. Il est vrai que grâce à sa notation lisible et partageable par le client et le concepteur, la communication entre les différents intervenants est facilitée sur une partie du cycle de vie du logiciel. De plus, UAN est d'autant plus profitable lors de la spécification de certaines interfaces homme-machine que ce formalisme est tout particulièrement consacré aux interfaces graphiques. Ainsi, il permet d'intégrer une approche ergonomique dans le développement des logiciels et préconise une modélisation centrée sur l'utilisateur [Norman 86].

3.3. Les modèles d'architecture

Une fois les spécifications terminées, la phase de conception logicielle peut démarrer. En ce qui concerne la mise en œuvre des interfaces homme-machine, la modularité est un facteur déterminant pour une conception itérative. Les principes de cette technique sont reconnus. Cependant, leur application est difficile à maîtriser en l'absence de modèle de décomposition. Le paragraphe 3.3.1 présente les modèles d'architecture des systèmes interactifs. PAC, le modèle multi-agent que j'ai retenu, est présenté en 3.3.2.

3.3.1. Définition, classes de modèles et leurs intérêts

Nous rappelons qu'une interface homme-machine (figure 3.8) se trouve à la jonction de deux composants :

- un noyau fonctionnel qui est le logiciel modeleur des concepts de la tâche informatisée,
- un utilisateur qui ne veut pas se contenter de tâches séquentielles.

L'interface homme-machine est donc pilotée par des événements en provenance du noyau fonctionnel et de l'utilisateur.

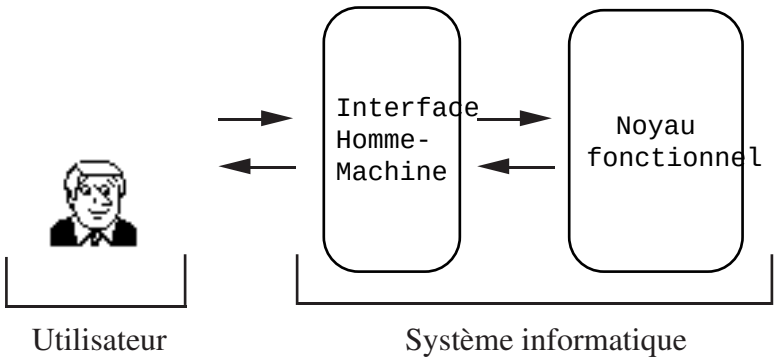


Figure 3.8 : Un système informatique et son utilisateur.

Il est important que la mise au point des interfaces homme-machine suive une stratégie itérative autorisant la modification sans mettre en cause la fiabilité et sans atteindre des coûts prohibitifs. C'est le rôle d'un modèle d'architecture de systèmes interactifs. Pour cela, il doit définir d'une part les éléments directeurs qui aident à l'organisation modulaire du logiciel. D'autre part, il doit s'appuyer sur l'indépendance du noyau fonctionnel et de l'interface utilisateur tout en maintenant les échanges d'information.

Figure 3.9 : Le modèle d'architecture de base.

Le modèle d'architecture de base représenté par la figure 3.9 illustre la séparation entre le noyau fonctionnel et l'interface utilisateur. Le noyau fonctionnel regroupe les concepts propres au domaine sans considération de leur présentation. Pour ce faire, il modélise le domaine à l'aide d'objets sémantiques liés par des relations. La présentation revient à l'interface organisée en un ensemble d'objets interactifs qui définissent la partie perceptible du système et restituent à l'utilisateur les concepts du domaine. Un troisième composant, le contrôleur de dialogue est

responsable de la gestion des relations et des échanges entre le noyau fonctionnel et le composant de présentation : enchaînement des tâches, mise en correspondance entre concepts du domaine et objets interactifs. En conséquence, le noyau fonctionnel et la présentation ne communiquent pas directement. Dans la figure 3.9, les flèches précisent la communication : une flèche à double sens traduit l'existence d'échanges d'informations bidirectionels entre deux composants.

Un modèle d'architecture définit aussi la stratégie de répartition des services de l'interface homme-machine sous la forme d'un ensemble de constituants logiciels modulaires. Enfin, il décrit précisément les relations et le protocole d'échange entre les constituants logiciels identifiés.

Niels Carlsen [Carlsen 91] identifie trois types de modèles qu'il classe en fonction de la structure du logiciel préconisée par le modèle :

- La structure en composants hétérogènes implique une organisation selon un ensemble de composants qui ne remplissent pas les mêmes services. Seeheim en est le modèle-type [Pfaff 85].
- 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". PAC-Amodeus est un exemple de cette approche [Nigay 94].
- La structure en espace d'objets homogènes organise le logiciel en une collection d'agents de mêmes caractéristiques et de même rôle fonctionnel. Le modèle PAC décrit dans le paragraphe suivant en est un exemple [Coutaz 90].

3.3.2. Le modèle PAC : une architecture multi-agents

Présentation du modèle PAC

Le modèle PAC structure un système interactif en un ensemble d'agents coopérants[18]. Un agent est un système de traitement de l'information. Son rôle est de réagir à des événements externes déterminés (des stimuli). Le résultat du traitement d'un événement se traduit généralement par un changement d'état de l'agent et par l'émission de nouveaux événements. L'état d'un agent fournit à tout instant une image de son évolution. Chaque agent comporte trois facettes (Présentation, Abstraction, Contrôle).

[18]Les agents coopèrent dans la conduite de l'interaction avec l'utilisateur, d'où leur nom d'objets interactifs.

- La présentation définit l'image de l'agent, c'est-à-dire son comportement perceptible en sortie (image de l'agent présentée à l'utilisateur) et en entrée (action de l'utilisateur sur l'agent). Ainsi, la présentation interprète les événements résultant des actions physiques que l'utilisateur applique sur l'agent via des dispositifs d'entrée et engendre des événements qui traduisent des actions physiques sur les dispositifs de sortie.

- L'abstraction désigne les concepts et les fonctions de l'agent, c'est-à-dire sa compétence interne. Les éléments de l'abstraction ignorent la façon dont ils sont représentés et manipulés par l'utilisateur.

- Le contrôle maintient la cohérence entre les facettes présentation et abstraction : il gère les liens entre les côtés abstrait et présentation qui ne sont pas autorisés à communiquer directement. Il est l'arbitre et le centre de décisions. En particulier, il détermine les techniques de traduction entre les formes abstraites et concrètes de l'agent, il ordonne la synchronisation et la coordination entre les facettes. Il gère les relations avec d'autres agents PAC car deux agents PAC communiquent uniquement via leur contrôle.

Un objet interactif peut être simple (agent élémentaire) ou composé. Un objet interactif élémentaire (figure 3.10) est une unité de compétence et d'exécution répartie sur trois processeurs (micro-parallélisme) : le contrôle C (moteur central), l'abstraction A et la présentation P (périphériques spécialisés). Ils sont dotés chacun d'un répertoire d'instructions et d'une mémoire privée pour l'état de contrôle, l'état d'abstraction et l'état graphique. Dans la figure 3.10, les flèches traduisent le flux des échanges. Les traits en pointillé indiquent l'échange avec l'extérieur de l'agent ; seul le contrôle permet la communication entre les agents.

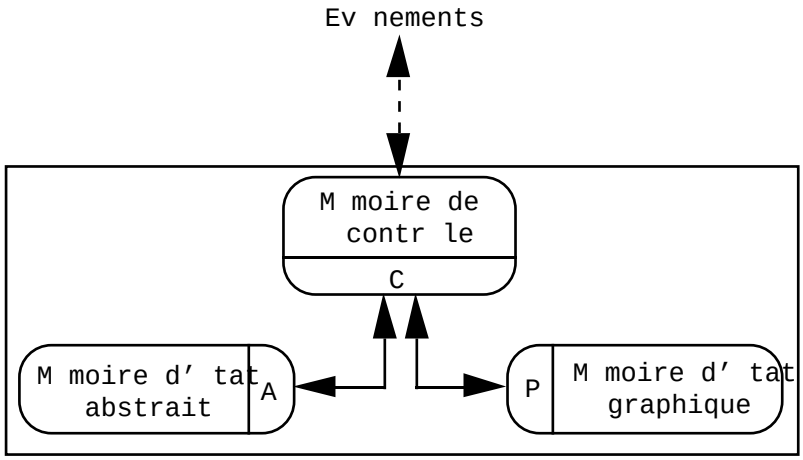


Figure 3.10 : Un objet interactif PAC vu comme un agent tri-processeur.

Un objet composé est créé à partir d'agents élémentaires qui travaillent en parallèle : c'est un agent d'interaction multiprocesseur structuré. Son comportement dépend de lui-même et des objets qui le constituent. Sa présentation hérite des propriétés de présentation de ses constituants et ajoute ses propres caractéristiques. Son abstraction lui est spécifique. Son contrôle se comporte comme un reconnaisseur d'événements : il vérifie la cohérence entre les côtés abstrait et présentation et gère, en plus, la collaboration ou la dépendance entre les objets composants.

Par application des techniques d'affinement et/ou de composition (décomposition récursive), le modèle PAC structure l'architecture d'un système interactif sous la forme d'un arbre d'agents. Le système interactif présenté par la figure 3.11 est donc un objet PAC composé. Les traits pleins figurent des relations entre les 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 (encapsulation et affinement.)

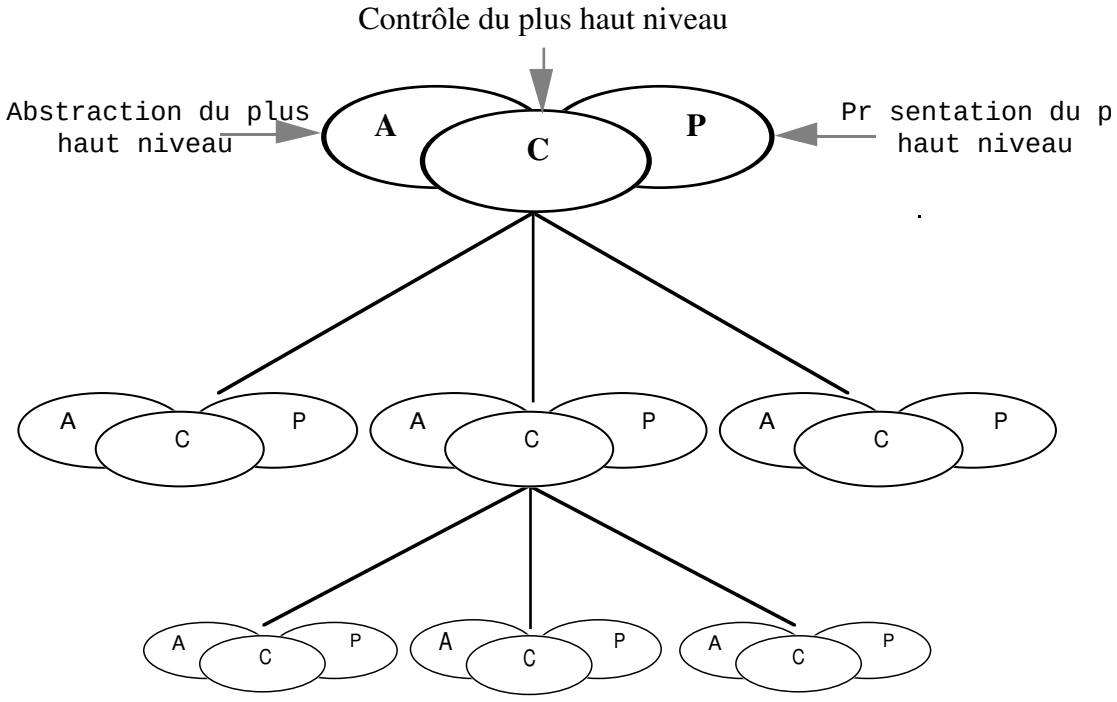


Figure 3.11 : La structure d'un système interactif en objets PAC élémentaires.

L'abstraction du sommet de la hiérarchie (racine) est le noyau fonctionnel qui implémente les concepts du domaine. Via le contrôle du plus haut niveau, le noyau fonctionnel est relié au monde de l'interface homme-machine par des intermédiaires adaptés à l'interaction avec l'utilisateur. Ce contrôle met en correspondance les concepts du noyau fonctionnel avec des abstractions d'agents. L'image du système est définie par la présentation du plus haut niveau. Comme pour tout objet composé, cette présentation hérite des présentations des objets

composants auxquelles s'ajoutent des propriétés intrinsèques. Ainsi, lorsque l'application modifie un concept, le contrôle traduit le changement dans les termes propres aux abstractions des agents PAC composants qui sont concernés par la modification. Les contrôles de ces agents prennent le relais jusqu'à ce que l'effet se traduise dans les présentations des agents élémentaires alors actualisés. Inversement, les actions de l'utilisateur sont interprétées par les présentations des agents manipulés par l'utilisateur (plus bas niveau). Elles sont reflétées dans les abstractions puis remontées dans la hiérarchie jusqu'à ce que des abstractions d'agents soient en liaison directe avec des concepts de l'application.

Intérêt du modèle PAC

Les parties abstraction et présentation d'un même agent ne communiquent jamais directement. Le mécanisme de référence indirecte assuré par le contrôle, permet de conserver l'indépendance des traitements et des modélisations entre les fonctions abstraites, les services de contrôle et les techniques de présentation. On garantit ainsi la séparation modulaire des rôles entre l'abstrait dirigé par le fonctionnement interne du logiciel (la sémantique), et le concret orienté vers l'utilisateur (la syntaxe). Les modifications logicielles d'une classe de compétences sont donc applicables à tous les niveaux d'abstraction sans mettre en cause le fonctionnement de l'ensemble. En particulier, il est aisé de remplacer la présentation d'un concept par d'autres agents ou d'ajuster localement l'image d'un agent élémentaire.

En résumé, le modèle d'architecture PAC multi-agent facilite le travail du réalisateur au bénéfice de l'utilisateur. PAC s'appuie sur la décomposition des traitements en niveaux lexical, syntaxique et sémantique ; il exploite la modularité, le parallélisme et une communication par événements. Cette approche facilite la conception itérative de l'interface homme-machine et elle est adaptée au comportement opportuniste de l'utilisateur [Hayes-Roth 79]. Elle contribue donc à une meilleure prise en compte de l'utilisateur, en particulier les dialogues à plusieurs fils d'activité[19] (ensemble d'acteurs coopérants en parallèle avec une répartition des charges). Le parallélisme ouvre la voie à des interfaces plus souples dirigées par l'utilisateur. Applicable à tous les niveaux d'abstraction d'un système interactif, PAC fournit un cadre pour la conception d'architectures qui respecte les techniques de structuration modulaire sans imposer les méthodes d'analyse ascendante et descendante. Le plan de résolution de problèmes suit l'heuristique habituelle du concepteur. PAC s'adapte bien à la conception orientée objet : les agents sont à rapprocher des objets au sens usuel de la programmation par objets. En ce sens, PAC et OMT forment un ensemble cohérent.

[19]Les dialogues à plusieurs fils d'activité se caractérisent par une interaction de l'utilisateur avec des actions d'entrée et de sortie simultanées. En d'autres termes, l'utilisateur a le droit de changer de centre d'intérêt au gré de son plan d'actions, il peut agir simultanément sur la présentation de plusieurs concepts, et plusieurs présentations de concept peuvent s'exprimer simultanément.

Dans le chapitre 4, je présente l'architecture retenue pour le projet, basée sur le modèle PAC. Elle est commentée et je précise comment le modèle PAC s'intègre dans la conception orientée objet.

3.4. Les outils de développement

Les outils de développement sont destinés à la phase de codage uniquement. Nous en distinguons deux classes particulières selon leur utilisation. Les environnements de programmation proposent un cadre de travail plus ou moins riche. Ils sont spécialisés pour assister le concepteur lors du codage, alors que les squelettes et les boîtes à outils sont des outils de construction d'interfaces.

3.4.1. Les boîtes à outils

Une boîte à outils est une bibliothèque de composants logiciels. Les boîtes à outils pour le développement d'interfaces homme-machine offrent des services à plusieurs niveaux d'abstraction depuis les services graphiques élémentaires (dessiner droite, dessiner cercle) jusqu'à des objets de présentation composés (formulaires, menus). La mise à disposition de composants de haut niveau d'abstraction permet de développer une interface utilisateur et définit un style d'interface homme-machine et d'interaction, mais elle rend plus difficile le développement d'une interface utilisateur qui ne correspond pas au style d'interface homme-machine de la boîte à outils.

La Toolbox du Macintosh [Apple 92a] est une boîte à outils de bas niveau d'abstraction sur laquelle s'appuie l'environnement de programmation Symantec que j'ai utilisé pour le développement.

3.4.2. Les squelettes d'application

Au contraire des boîtes à outils, les squelettes d'application réalisent les fonctions générales et usuelles d'une interface utilisateur (noyau d'exécution et services étendus) sous la forme d'un logiciel réutilisable et extensible. La tâche du programmeur consiste alors à ajouter au squelette des composants spécifiques à son application. En général, un squelette est construit en corrélation avec une boîte à outils. MacApp [Schmucker 86a] en est un exemple développé en Object Pascal[20] au-dessus de la Toolbox Macintosh [Apple 92a]. Exploitant l'environnement à objets, MacApp offre un ensemble de classes d'objets prédéfinies, surchargeables et extensibles par spécialisation. Par exemple, la classe d'objet Tapplication fournit la boucle d'acquisition et

[20]Langage orienté objet dérivé du langage Pascal.

de traitement des événements utilisateur et constitue le programme principal de l'interface homme-machine.

Avec les squelettes d'application, le programmeur hérite d'une architecture logicielle saine qui facilite et encourage la mise au point itérative de l'interface utilisateur et contribue ainsi à améliorer la qualité de l'interface. Le squelette précédemment développé par des spécialistes, fournit un assemblage de briques logicielles prêt à l'emploi. Le programmeur reçoit au-delà d'une architecture, un ensemble de services dont le niveau d'abstraction se rapproche de ses besoins. Son concepteur annonce dans [Schmucker 86b] que MacApp peut réduire le temps de développement d'un système jusqu'à un facteur 5!

Si la réutilisation de logiciels est un remède à la duplication d'efforts de programmation, elle requiert inévitablement une étape d'apprentissage. Ma tentative pour utiliser MacApp n'a pas échappé à cette règle. Comme pour la plupart des environnements à objets, je lui reproche la difficulté pour le programmeur d'identifier les services disponibles (classes et méthodes) et de dégager les surcharges et les extensions à faire pour répondre à ses besoins. En effet, pour surcharger et étendre les fonctions, le programmeur retombe inévitablement sur le service de la boîte à outils qu'il doit nécessairement bien connaître, même si les squelettes tentent d'éviter ce niveau de programmation jugé trop bas. Mon peu d'expérience en environnement Macintosh a été pénalisant et m'a obligée à faire un double apprentissage, celui de MacApp et celui de la Toolbox. Quant à l'héritage, il impose à une application une grande quantité de fonctions dont certaines sont inutiles pour les besoins du programmeur. Pour un débutant, et avec le peu de documentation fournie par Apple sur MacApp, l'effort pour maîtriser l'ensemble des classes est très important (au moins trois mois d'apprentissage).

MacApp s'appuie sur l'environnement de programmation Macintosh Programmer's Workshop (MPW) [Apple 90a] qui s'inspire de la philosophie UNIX (par exemple, le shell, le langage de commande). Mon apprentissage de cet outil réputé puissant auprès des spécialistes a été long et semé d'embûches. En particulier, le fichier d'aide à la compilation (make ou build) doit être écrit directement par le programmeur sans soutien de l'outil. La gestion des modifications n'est pas toujours assurée et j'ai dû parfois recompiler toute l'application pour résoudre des problèmes de lien (link). Un autre exemple concerne l'éditeur de ressources ViewEdit [Apple 92h] qui utilise pour les menus une description textuelle saisie par le programmeur. Pour éviter ce travail fastidieux, j'ai décrit les ressources avec l'éditeur graphique ResEdit [Apple 91b], puis je les ai compilées et modifiées manuellement jusqu'à obtenir la compatibilité voulue. Après avoir dépassé le délai d'apprentissage que je m'étais fixé, avec l'accord du responsable et sur les conseils de certains programmeurs de l'équipe, j'ai

abandonné MacApp et MPW et j'ai choisi l'environnement de programmation Symantec pour développer toute l'application.

3.4.3. L'environnement de programmation de Symantec

L'environnement de programmation Think Class Library (TCL) de Symantec pour Macintosh offre au programmeur un cadre de développement performant et complet [Symantec 94a]. Il permet de construire des applications pour Macintosh, des drivers, des accessoires de bureau, et toutes sortes de code utilisant les ressources. Il comprend un grand nombre d'outils complémentaires :

- Project Manager,
- Visual Architect,
- un ensemble de bibliothèques,
- un "debugger" intégré,
- un constructeur d'objet,
- un inspecteur d'objet ("browser").

L'outil Project Manager comprend l'ensemble des composants utiles au développement d'une application :

- un éditeur de texte,
- un compilateur C++ et C ANSI,
- un outil d'optimisation de code,
- un éditeur de liens ("linker"),
- un outil de compilation ("make").

De plus, Project Manager inclut un gestionnaire des fichiers de l'application développée (organisation, mapping, recompilation, etc.). Il permet aussi de passer du "debugger", à l'édition des fichiers sources et vice versa pour contrôler le déroulement du programme écrit.

Visual Architect est un outil de développement utilisé conjointement avec les bibliothèques pour construire en partie l'interface utilisateur à développer. Le programmeur définit des classes dérivées directement avec cet outil interactif sous forme graphique et est dispensé d'écrire du code. Visual Architect est composé d'un outil de prototypage d'interface homme-machine, d'un générateur de code, d'un éditeur de ressource et d'un outil d'explication pour les bibliothèques. Il peut donc être utilisé à n'importe quelle phase durant le développement de l'application. En qualité d'outil de prototypage d'interface homme-machine, il aide à développer la partie interface utilisateur de l'application : fenêtres, boîtes de dialogue, boutons, menus, etc. Le programmeur peut tester et modifier les éléments séparément du reste de l'application. En tant que générateur de code, il crée automatiquement des fichiers sources à partir de la définition de l'interface

utilisateur. Il les ajoute automatiquement au reste de l'application et crée un squelette d'application. Le programmeur est autorisé à modifier les classes d'objets ainsi produites. L'éditeur de ressources gère et crée des fichiers de ressources qui sont ajoutés à l'application. Son rôle d'outil d'explication de classes consiste à commenter le code généré. Il permet donc au programmeur d'apprendre la structure des classes des bibliothèques.

Il existe trois types de bibliothèques :

- la bibliothèque C standard qui contient des fonctions spécifiques au C ANSI et quelques fonctions additionnelles du système Unix,
- la bibliothèque C++ contient une bibliothèque de classes extensibles pour les opérations d'entrée-sortie et les opérations mathématiques,
- la bibliothèque de classes Think ou TCL (Think Class Library).

La bibliothèque de classes Think est une collection de classes C++ qui implémentent une application Macintosh standard. Ces classes se chargent des fonctions de base, telles que la gestion des menus et de leurs commandes, la mise à jour des fenêtres, le contrôle des événements, la collaboration avec le système d'exploitation, la gestion de la mémoire, l'impression, etc. La bibliothèque libérant le programmeur de ces tâches, celui-ci ne développe plus directement à partir de la Toolbox Macintosh et du système d'exploitation, mais il s'appuie sur les classes existantes dans la bibliothèque de classes Think déjà connectées à ces interfaces de bas niveau.

La bibliothèque de classes Think est organisée selon trois structures distinctes et interconnectées :

- la hiérarchie de classes composée de l'ensemble de toutes les classes qui constituent la bibliothèque de classes Think (les relations sont l'héritage, une forêt ou un graphe acyclique direct),
- la hiérarchie Visual qui décrit l'organisation de toutes les entités visibles, par exemple les fenêtres, les boutons d'une application,
- la chaîne de commande qui spécifie quels objets de l'application gèrent les commandes et dans quel ordre.

La hiérarchie de classes est statique car les relations entre les classes sont fixes pour toutes les applications et ne changent pas au cours de leur exécution. Au contraire, la hiérarchie Visual et la chaîne de commande sont des structures dynamiques : elles sont composées d'objets qui existent uniquement pendant l'exécution de certaines parties de programme.

La bibliothèque de classes Think convertit les événements Macintosh en appels de fonctions définies par des classes de la hiérarchie de classes (le “switchboard” appelle des fonctions de la Toolbox comme WaitNextEvent ou GetNextEvent). Certaines fonctions manipulent des

événements qui affectent la hiérarchie Visual. Les événements Visual sont les clics souris, les événements d'activation et les événements de mise à jour. Certaines fonctions manipulent des commandes directes pour demander à un objet de réaliser une action.

Créer une application qui utilise la bibliothèque de classes Think consiste à utiliser de nouvelles classes dérivées à partir de celles fournies par la bibliothèque. Les nouvelles classes implémentent les seules parties de code qui sont spécifiques à l'application ; le comportement générique de l'application est déjà pris en charge par les classes de base de la bibliothèque. Le programmeur doit aussi définir la structure du menu qui contient les commandes pour l'application et les liens entre les items de menu et les commandes. Et finalement, il lui faut définir les ressources spécifiques à l'application.

J'ai apprécié la souplesse et la performance de cet environnement faisant suite à mon expérience de l'environnement MPW et du squelette d'application MacApp [Schmucker 86a]. Les boîtes à outils et les squelettes constituent le premier niveau des outils de construction des interfaces homme-machine. Les générateurs visent plus loin : ils proposent au concepteur la génération automatique des interfaces.

3.4.4. Les générateurs d'interfaces homme-machine

Les générateurs d'interfaces utilisateur (UIMS ou User Interface Management Systems) sont classés en trois catégories :

- Les générateurs de la présentation permettent de décrire les écrans et leur contenu en termes d'objets de présentation appartenant à la boîte à outils. Il en existe des centaines. Par exemple, Visual Architect de Symantec en fait partie, ainsi que les éditeurs de ressources ResEdit et ViewEdit pour MacApp. Ces générateurs semblent très séduisants, d'autant plus lorsqu'ils sont interactifs. Cependant, il est souvent nécessaire de revenir au niveau de la boîte à outils pour des besoins spécifiques de l'interface. La partie statique de l'interface graphique est spécifiée par manipulation directe (comme les fenêtres, leur taille, leur position etc.), mais la partie dynamique de l'interface (comme les liens d'ouverture entre deux composants graphiques), nécessite de programmer.

- Les générateurs de la présentation et du contrôleur de dialogue permettent de spécifier la dynamique de l'interface homme-machine, et notamment les enchaînements d'écrans. Hypercard [Harvey 88], Interface Builder [Webster 89], XFaceMaker [Non Standard Logic] et Macromind Director [Macromind 91] en sont des exemples. Pour développer l'interface utilisateur du sujet (terminal de télécommunication), nous avons tenté d'exploiter les possibilités de Macromind, mais il s'est avéré que celui-ci ne permettait pas d'utiliser les Apple Events nécessaires à la communication entre les applications. De plus, la structure du code généré nous a donné à penser qu'il faut considérer MacroMind Director comme un bon outil de prototypage mais il faut garder à l'esprit la différence entre un prototype et le système livré.

- Les générateurs conceptuels produisent automatiquement le code du contrôleur de dialogue et de la présentation à partir de la spécification formelle du fonctionnement du noyau fonctionnel. Ils procèdent de manière descendante depuis les concepts jusqu'aux objets de présentation. Utilisables dès l'analyse des besoins, ils 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. En raison de la génération automatique, les tests de nature informatique sont superflus. Mais cette vision idéale de l'automaticité s'accorde avec les 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 la réversibilité. Actuellement, ces générateurs relèvent du domaine de la recherche. On peut citer UIDE [Foley 88], SIROCO [Normand 92] et HUMANOID [Szekely 93].

Ce chapitre a précisé et expliqué mes choix en matière de démarche de développement. Nous avons abordé le cycle de vie et discuté des outils et des méthodes utilisables. Nous allons voir à présent leur mise en œuvre dans la réalisation de SupraAnalyse.

Chapitre 4

Réalisation de SupraAnalyse, un outil d'aide à l'analyse de données comportementales

Ce chapitre présente le détail de ma réalisation. Sa structure s'appuie sur le cycle de développement du logiciel que j'ai suivi pour le projet et présenté au chapitre 3. J'y aborde quatre phases : l'analyse et la spécification des besoins, la spécification de l'interface homme-machine et la conception. Dans chacune de ces parties, je donne un exemple de mise en œuvre des méthodes exposées au chapitre précédent. La première partie du chapitre présente l'analyse des besoins des ergonomes pour mener une évaluation ergonomique expérimentale avec la technique du Magicien d'Oz. Ensuite, j'expose la démarche d'évaluation ergonomique proposée dans le cahier des charges de SUPRATEL ★. La troisième partie reprend les spécifications externes de l'outil d'analyse et montre l'interface homme-machine proposée aux ergonomes. La dernière partie décrit et commente l'architecture globale retenue. Elle précise comment PAC s'intègre dans une conception orientée objet.

4.1. Analyse des besoins des ergonomes : un premier pas vers l'évaluation automatique d'une interface homme-machine

La partie la plus importante de mon travail consiste à définir les besoins d'un analyste en ergonomie. Pour ce faire, j'ai organisé à Grenoble des réunions de travail avec des ergonomes du CCETT. Je les ai complétées par une visite de leur laboratoire d'ergonomie à Rennes, à l'occasion de séances de tests pour une interface homme-machine similaire à celle conçue dans le cadre de SUPRATEL ★. Cela m'a donné l'occasion d'observer les ergonomes dans une situation réelle d'évaluation avec Magicien d'Oz et de tirer aussi des leçons sur les compères et la conduite de session.

4.1.1. La démarche d'évaluation ergonomique du CCETT

La démarche du CCETT implique une participation active et une étroite collaboration des informaticiens, des utilisateurs et des ergonomes. Les informaticiens mettent l'accent sur les fonctions du logiciel, les contraintes des outils de développement et la faisabilité des évolutions demandées. Les utilisateurs sont concernés directement par le produit qui doit être adapté à leur travail. Et les ergonomes sont des spécialistes dans la réalisation des évaluations et l'observation de l'activité.

Une démarche complète d'évaluation ergonomique est organisée en cinq phases principales :

- La prise de connaissance du logiciel permet à l'ergonome de découvrir le produit, de réaliser un pré-diagnostic sur l'interface de dialogue et de repérer des points à étudier

plus particulièrement avec les utilisateurs. Cette première phase permet de faire un contrôle a priori des caractéristiques ergonomiques de l'interface homme-machine.

- La définition de la population et des scénarios se fait avec le concours des concepteurs du logiciel et des utilisateurs expérimentés dans le travail. L'objectif est double. Il s'agit tout d'abord de cibler la population concernée et de constituer un échantillon représentatif d'utilisateurs débutants et expérimentés. Les ergonomes doivent aussi déterminer les conditions d'utilisation et définir les scénarios de tâches qui simulent la situation de travail réelle avec le logiciel. Un scénario de tâches est une suite d'instructions qui décrit les objectifs de travail. Cette définition consiste à choisir les tâches les plus importantes dans le cadre de l'utilisation du logiciel, en fonction de leur fréquence et/ou de leur incidence sur l'activité de travail. [Valentin 93] note que seules environ 20% des fonctions sont étudiées, mais judicieusement sélectionnées, celles-ci correspondent en fait à environ 80% du temps d'utilisation du logiciel.

- Le recueil des données sur l'utilisation du logiciel s'effectue en faisant réaliser les scénarios par la population définie et en observant individuellement les activités cognitives des utilisateurs mises en œuvre dans les situations de travail. Les éléments relevés concernent les stratégies suivies (chemins dans les écrans, informations utilisées, etc.), les incidents (hésitations, difficultés, etc.) et les commentaires des utilisateurs. L'ergonome tente de détecter les écarts entre le travail prescrit (prévu et demandé) et le travail réellement effectué par l'utilisateur.

- L'analyse des données recueillies porte sur deux aspects principaux qui permettent d'apprécier la facilité d'apprentissage et d'utilisation du logiciel : d'une part la dynamique de l'activité, les chemins suivis et les procédures adoptées par les utilisateurs, d'autre part la nature, la cause et la fréquence des incidents.

- La synthèse des résultats de l'analyse inclut des recommandations qui portent sur l'interface de dialogue, l'adéquation des fonctions aux tâches étudiées, la documentation, l'aide en ligne, la formation et l'organisation du travail.

4.1.2. Le déroulement d'une session d'expérimentation

Dans les cinq phases principales qui composent la démarche d'évaluation ergonomique d'une interface homme-machine au CCETT, mon intervention est très ciblée et se situe dans la phase de recueil des données caractéristiques de l'utilisation du logiciel et dans la phase d'analyse de ces mêmes données. Précisément, la visite faite au laboratoire du CCETT à Rennes m'a donné l'occasion d'assister à la phase de capture des données et à leur pré-analyse. Dans le local d'observation où se regroupent les ergonomes et les compères, j'agissais comme un observateur supplémentaire et je focalisais toute mon attention sur l'activité des ergonomes et sur leurs conditions de travail. J'expose ici le déroulement d'une session d'expérimentation et les remarques qu'elle a suscitées.

Une expérimentation a lieu dans deux pièces distinctes, séparées par une glace sans tain pour que les observateurs puissent voir ce qui se passe dans la pièce des utilisateurs. Dans la pièce observée se trouvent des caméras et microphones qui captent et enregistrent tout ce qui concerne les utilisateurs : l'interaction avec l'interface homme-machine, leur attitude face au produit, et toutes leurs verbalisations. Installés dans l'autre pièce, les observateurs et les compères peuvent surveiller et suivre le déroulement de l'expérimentation en utilisant la glace sans tain et des écrans vidéo de retransmission.

Après une mise au point de l'évaluation avec au moins un premier utilisateur, les séances de tests démarrent avec des utilisateurs qui travaillent en couple pour être incités à verbaliser plus naturellement. Certains ergonomes préfèrent une situation avec un sujet unique qui a pour consigne de dire tout haut ce qu'il pense tout bas. Dans les deux cas, ce dialogue est précieux pour les ergonomes qui disposent ainsi d'informations sur le plan de résolution[21] du sujet. Par exemple, les verbalisations ou commentaires peuvent aider les observateurs à comprendre ce qui guide les utilisateurs (objectifs, informations recherchées, etc.).

L'un des observateurs présente aux utilisateurs la structure du logiciel et ses objectifs, le déroulement prévu pour l'évaluation (consigne générale) et les résultats attendus. L'ensemble des observateurs et les compères se coordonnent avant de lancer la session d'expérimentation car chacun des intervenants a un rôle précis à jouer lors du déroulement des scénarios. Les utilisateurs sont avertis qu'ils sont observés.

[21]Le plan de résolution est le schéma mental développé par l'utilisateur pour accomplir une suite de tâches donnée.

Les compères sont dédiés à la simulation, c'est-à-dire qu'ils remplacent le système quand les fonctions appelées ne sont pas réalisées ou font appel à une intervention extérieure (par exemple un appel téléphonique). J'ai remarqué que les compères disposaient d'un support informatique, contrairement à la plupart des systèmes à Magiciens d'Oz qui, non informatisés, ne proposent aux compères qu'un scénario papier. Cet outil les assiste dans leur tâche en leur présentant l'enchaînement des scénarios et les rôles à jouer pour chacune des situations. Malgré la simplicité de l'outil, il n'en est pas moins utile, car il s'avère qu'avec un scénario papier, le compère est parfois débordé par la charge de travail.

Les observateurs procèdent à une pré-analyse en ligne. Pour cela, chacun adopte une méthode qui lui est propre et fondée sur l'expérience. Certains ont rédigé des grilles de notation qui leur permettent de classer leurs remarques selon les critères attendus. D'autres suivent le déroulement des scénarios et s'attachent à noter au fur et à mesure, de façon séquentielle, ce qu'ils repèrent comme distinctif et typique. D'autres encore notent tous les enchaînements d'écrans que l'utilisateur provoque et les compare aux enchaînements théoriques qu'ils avaient préparés pour le même scénario. Quelle que soit la méthode employée par les ergonomes, les points relevés cherchent à mettre en évidence les actions de l'utilisateur, les étapes suivies, les fonctions utilisées et les difficultés rencontrées (dysfonctionnements, erreurs, difficultés, tâtonnements, procédures non prévues dans les scénarios, etc.).

A la fin de l'évaluation, un observateur soumet à l'utilisateur un questionnaire plus général. Ce dernier permet de compléter les observations quantitatives et objectives par un recueil d'appréciations subjectives de l'utilisateur sur le logiciel. Il permet aussi de couvrir les fonctions non évaluées dans les scénarios.

Il faut une certaine homogénéité et une coordination entre les observateurs lors du recueil des données. En particulier, les compères ont un rôle prépondérant dans le bon déroulement des scénarios. J'ai relevé quelques consignes données aux observateurs et aux compères :

- ne pas donner d'information sur le produit testé ni faire de commentaire personnel,
- relever les différentes actions de l'utilisateur, solliciter des explications et noter les commentaires correspondants,
- noter à quoi correspondent les actions par rapport au logiciel (titre d'écran, message, touche fonction utilisée, etc.),
- en cas de chemin prévu non suivi, le laisser suivre sa procédure et recueillir la dynamique propre à ses actions,
- en cas de solution non immédiate, le laisser chercher et relever les difficultés rencontrées,

- en cas de blocage, ne pas intervenir personnellement, faire appel à une personne de l'équipe de conception et relever les explications fournies,
- se garder d'interpréter immédiatement la cause des difficultés et simplement noter le déroulement effectué.

Les notes prises en ligne par les observateurs pendant la session et l'enregistrement audiovisuel constituent l'essentiel des données recueillies pendant les tests. Elles sont complémentaires et indispensables à l'ergonome lors de la synthèse des résultats. Elles sont utilisées d'une part pour l'analyse des stratégies (les procédures suivies sont comparées aux procédures prévues), et d'autre part pour l'analyse des incidents survenus lors du déroulement des scénarios. Les commentaires des utilisateurs permettent de préciser les causes des événements observés et d'effectuer un regroupement par nature (événements ayant des causes communes) pour ensuite généraliser les solutions proposées. Les questionnaires des utilisateurs sont ensuite rapprochés et résumés. L'objectif est d'avoir une vision synthétique des appréciations, des similitudes ou de leurs écarts. Les données recueillies pendant l'observation sont pondérées et complétées, voire généralisées aux fonctions non traitées dans les scénarios.

A partir de ces premiers résultats d'analyse, l'ergonome suggère un compromis entre les différentes recommandations ergonomiques utilisables dans le domaine concerné, les contraintes de la situation de travail et les contraintes techniques liées aux outils de développement ou la structure du projet. Les recommandations portent sur tout ou partie des points suivants :

- les modes de dialogue (touches fonctions, vocabulaire, structure, etc.),
- la présentation des informations (densité, structure, etc.),
- les fonctions du produit (regroupements d'informations, enchaînement des écrans, etc.),
- la documentation et l'aide en ligne,
- la formation des utilisateurs et l'assistance à leur donner.

La présentation des résultats comprend l'analyse et la synthèse des écarts effectuées par l'ergonome. Elle regroupe tout d'abord des éléments généraux sur l'évaluation : l'identification de la demande, la description du produit, les conditions de l'évaluation, la définition des scénarios de tâches. Enfin, elle contient des résultats sur les séances de tests : le compte rendu des observations et des différents questionnaires, les propositions et la conclusion.

4.2. La démarche d'évaluation proposée avec SUPRATEL ★

L'observation des séances au laboratoire de Rennes et la consultation de comptes-rendus d'analyse ergonomique prêtés par le CCETT m'ont permis de dégager des fonctions qui méritent d'être assistées avec un outil informatique. Le paragraphe suivant montre mes propositions d'automatisation pour la capture des données comportementales et leur analyse dans le cadre du projet SUPRATEL ★. Nous rappelons que l'analyse de la session se fait en ligne par les observateurs et a posteriori par les analystes.

4.2.1. Des propositions pour la phase de capture et d'analyse

Pour la phase de recueil des données, les fonctions proposées sont les suivantes :

- une capture informatique et automatique des données comportementales du sujet en interaction avec l'interface homme-machine remplace l'enregistrement audiovisuel,
- les compères connectés à la station du sujet peuvent observer le sujet, interagir avec lui et simuler les fonctions manquantes du système,
- les observateurs disposent d'un outil d'annotation informatique qui leur permet de faire une pré-analyse en ligne du comportement du sujet.

Ce principe met les compères à forte contribution car il leur est demandé d'assurer des rôles de simulation, d'observation et d'annotation. Et, comme je l'ai relevé lors de ma visite au CCETT, la tâche d'un compère est déjà délicate et assez lourde. Par exemple, lors de la simulation, celui-ci doit suivre des consignes précises, et notamment respecter les temps de réponse et adopter un comportement cohérent et constant tout au long d'une même expérience. Malheureusement, bien que le dispositif expérimental du compère conditionne la réussite de l'expérience, l'aide à la tâche du compère, et notamment son interface homme-machine sont rarement étudiés en tant que tels. C'est pourquoi, avec SUPRATEL ★, dans le but d'aider les compères à accomplir leurs tâches, chaque rôle à assurer est pourvu d'une interface homme-machine spécialisée qui tient compte de la charge cognitive à mettre en œuvre lors des sessions. C'est le supercompère qui affecte les rôles aux différents compères lors de la configuration de l'expérimentation, en fonction de la répartition des tâches (simulation, observation et annotation). Les interfaces homme-machine de simulation des services manquants et celles d'annotation en ligne ont été conçues de façon à ce que l'intervention des compères soit aisée et rapide. En particulier, pour les annotations de l'expérimentation en ligne, les interfaces homme-machine contiennent des informations prédéfinies qui peuvent être complétées ou modifiées par l'ergonome avant le démarrage de la session d'expérimentation. Le détail des outils proposés aux compères est exposé dans les dossiers du projet SUPRATEL ★ [Lischetti 94a et 94b].

Dans la phase d'analyse, je propose d'aider l'ergonome à exploiter le contenu du fichier historique d'expérimentation en lui permettant de :

- visualiser le déroulement de la session d'expérimentation telle qu'elle s'est déroulée pour le sujet et avoir accès à des informations identiques à celles contenues dans un enregistrement audiovisuel (représentation et niveau d'abstraction),
- compléter l'annotation de la session d'expérimentation déjà renseignée dans la phase précédente,
- mener son analyse en s'appuyant sur une automatisation partielle de l'analyse de tâches.

Après cette présentation générale des services proposés, je détaille les informations mises à la disposition de l'ergonome car l'analyse s'appuie sur le contenu du fichier historique d'expérimentation, résultat de la capture.

4.2.2. Le fichier historique d'expérimentation

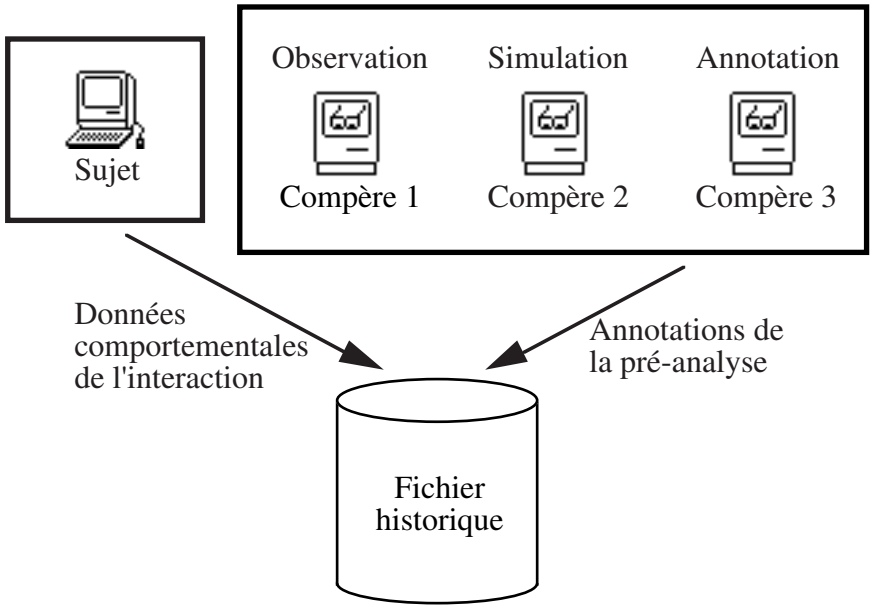


Figure 4.1 : La capture des données avec NEIMO.

La figure 4.1 montre que les informations recueillies lors de la capture ont deux origines :

- Les données comportementales sont issues de la station du sujet ; elles sont capturées automatiquement.
- Les annotations de la pré-analyse proviennent des compères ; elles sont sous leur responsabilité : ils les renseignent et provoquent leur enregistrement à la demande en utilisant une interface homme-machine spécifique.

Parmi les données comportementales qui reflètent l'interaction, on distingue :

- des informations de bas niveau d'abstraction qui sont semblables à celles que l'ergonome exploite couramment avec un enregistrement audiovisuel,
- des informations de plus haut niveau d'abstraction qui montrent la dynamique de l'activité du sujet ; chaque activation, terminaison ou annulation de service produite par l'utilisateur est enregistrée et datée.

En ce qui concerne les informations de bas niveau d'abstraction, le fichier historique d'expérimentation est relativement limité. En effet, il ne comporte pas les attitudes du sujet face au produit, puisque celles-ci ne sont pas filmées (la caméra est réservée au visiophone). Par contre, il contient les informations qui ont rapport avec l'écran du sujet et toute l'interaction visible sur cet écran. L'interface homme-machine évaluée étant une interface graphique à manipulation directe, on y trouve les enchaînements de tous les écrans et les actions de l'utilisateur qui les a produits, comme les déplacements de la souris, les sélections d'objets, les "drag and drop", etc. Quant aux verbalisations du sujet, elles ne sont pas accessibles car l'enregistrement du son en continu n'a pas été retenu : les problèmes techniques que nous avons rencontrés entraînaient des performances relativement médiocres.

Les annotations de la pré-analyse sont à la charge des compères. La possibilité de faire des annotations vocales n'a pas été retenue : le compère dispose uniquement du clavier et de la souris pour faire la pré-analyse, toujours pour des raisons de performance. Ces annotations montrent :

- tous les incidents rencontrés par les utilisateurs, datés et caractérisés selon quatre critères : le nom donné à l'incident, le type d'erreur, la tâche concernée et la gravité de l'incident,
- une appréciation sur la réussite et le respect du scénario par le sujet,
- des données temporelles sur le scénario (début, fin, durée).

En utilisant les services de NEIMO, toutes les données comportementales et les annotations de la pré-analyse sont numérisées, synchronisées et enregistrées sur un seul et même support informatique, le fichier historique d'expérimentation. Sa structure (format emprunté à QuickTime) est telle que les informations forment un ensemble cohérent et homogène au sens temporel et représentent toute la durée de l'expérimentation. Ce support de stockage offre la possibilité de prolonger la pré-analyse en ligne dans la phase d'analyse a posteriori. L'ergonome dispose alors d'une base de travail unique et n'a plus à consulter de multiples supports qui contiennent des informations asynchrones, comme les bandes audiovisuelles et les notes manuscrites. Nous allons nous intéresser plus précisément à l'analyse de ces données comportementales et détailler l'ensemble des services prévus pour couvrir les besoins en analyse de l'ergonome.

4.2.3. L'outil SupraAnalyse et ses services

Ce paragraphe expose les fonctions principales de l'outil SupraAnalyse : la gestion des différents fichiers à manipuler par l'ergonome, la visualisation, l'annotation et l'analyse d'une session d'expérimentation. La figure 4.2 montre la décomposition fonctionnelle de l'outil d'analyse dans le projet SUPRATEL ★. On remarque que l'application s'appuie sur trois types de fichiers différents dont il est nécessaire de prévoir la gestion.

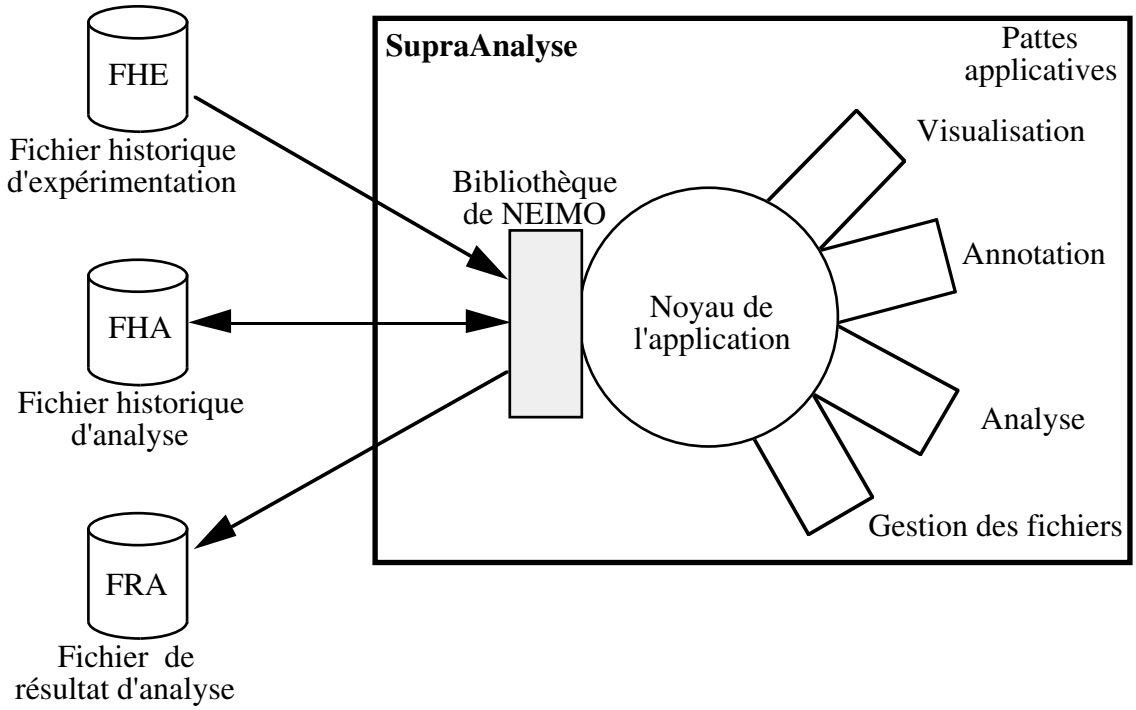


Figure 4.2 : La décomposition fonctionnelle de l'application SupraAnalyse.

4.2.3.1. La gestion des fichiers

La gestion des différents fichiers à manipuler par l'ergonome est une fonction classique dans toute application informatique qui exploite des fichiers. Je ne la détaille pas particulièrement mais je présente ici un extrait du schéma d'objet OMT (figure 4.3) qui montre les différents types de fichiers envisagés. On rappelle que ce diagramme décrit les aspects statiques du système sous forme d'objets. Dans le cas présent, les objets considérés sont les fichiers.

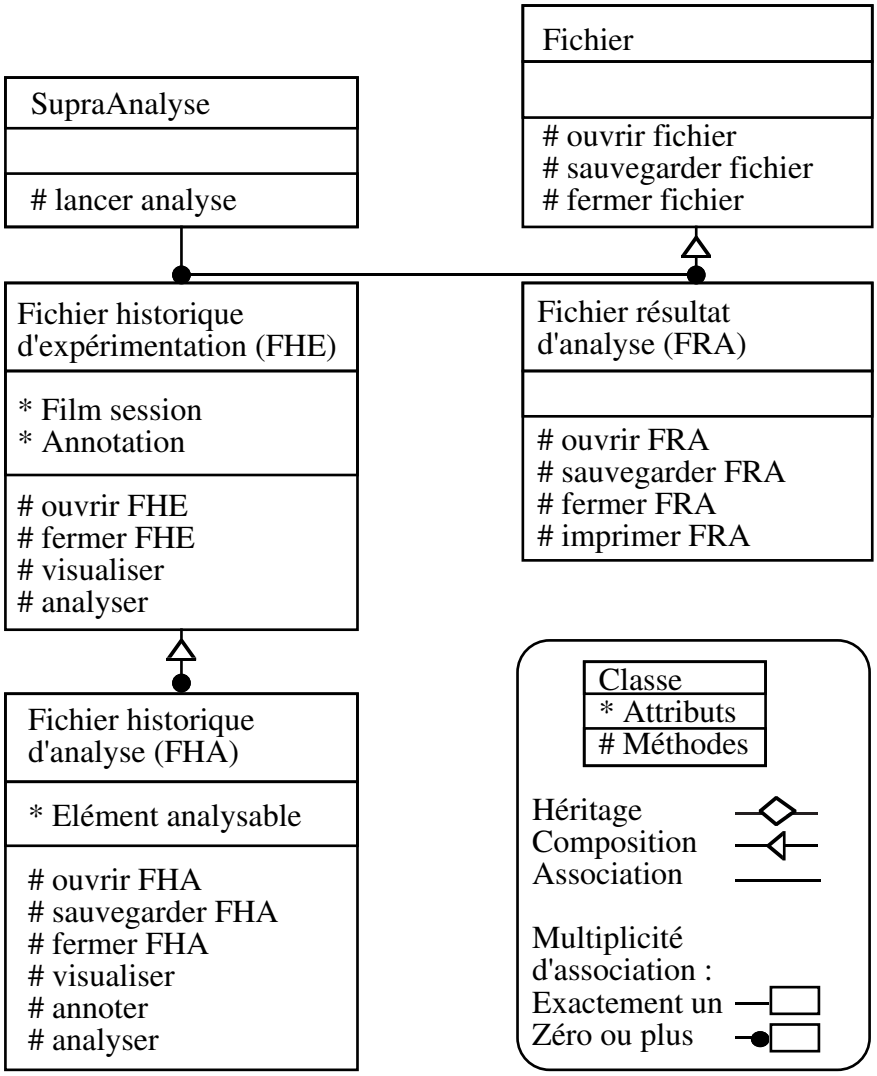


Figure 4.3 : Un extrait du diagramme OMT de SupraAnalyse.

- Le fichier historique d'expérimentation ou FHE contient toutes les données numérisées et enregistrées de façon synchrone lors de la capture. La visualisation d'une expérimentation exploite son contenu.

- Le fichier historique d'analyse ou FHA contient toutes les données du fichier historique d'expérimentation et les compléments ou les modifications apportées par l'ergonome. En effet, il est préférable que l'annotation a posteriori du fichier historique d'expérimentation n'entraîne pas sa modification afin de conserver les informations initiales. La double existence des fichiers historiques d'expérimentation et fichiers historiques d'analyse est transparente pour l'ergonome qui ne se soucie pas de la protection des différents fichiers.

- Les fichiers de résultats d'analyse ou FRA ne contiennent que des données choisies par l'ergonome et mises en forme pour l'impression et l'insertion dans d'autres documents informatiques, comme la synthèse des résultats. Ces fichiers sont créés à la demande de l'ergonome à partir d'un fichier historique d'expérimentation ou d'analyse.

Dans le Cahier des Charges, ce schéma d'objet OMT est accompagné d'une description des tâches de l'ergonome et d'une représentation de la décomposition hiérarchique des tâches. Le formalisme UAN étant tout particulièrement consacré à la spécification de l'interface homme-machine, il ne fait apparaître dans cette phase d'analyse de tâches que des éléments de haut niveau d'abstraction qui ne sont pas très représentatifs de l'interaction. C'est pourquoi, je préfère citer plus loin dans ce document des exemples qui décrivent une partie de l'interface homme-machine de SupraAnalyse.

4.2.3.2. La visualisation d'une session d'expérimentation

La visualisation d'une session d'expérimentation permet à l'ergonome de revoir la session d'expérimentation telle qu'elle s'est déroulée pour le sujet. Elle exploite les informations du fichier historique d'expérimentation (FHE) qui décrivent l'interaction visible sur l'écran du sujet. Nos échanges avec les ergonomes ont mis en évidence que ceux-ci tenaient à conserver toutes les informations qu'ils manipulaient habituellement, même si certaines informations étaient de très bas niveau d'abstraction et exigeaient de la patience pour être exploitées et interprétées manuellement. Les ergonomes proposant de doubler l'outil avec un enregistrement audiovisuel, j'ai été attentive à respecter leurs habitudes de travail et à faciliter le transfert de connaissances.

4.2.3.3. L'annotation a posteriori d'une session d'expérimentation

L'annotation a posteriori d'une session d'expérimentation a pour objectif de fournir à l'ergonome un support informatique qui le dispense de prendre des notes par écrit. Il peut alors appuyer son travail sur le fichier historique d'expérimentation qui contient déjà toutes les informations capturées pendant la session d'expérimentation, c'est-à-dire les annotations en ligne des compères et l'enregistrement de la session d'expérimentation. A partir de ces données, il peut créer un fichier historique d'analyse (FHA) qui apporte des compléments ou des modifications aux informations initiales. Dans un premier temps, l'outil SupraAnalyse permet uniquement la consultation du fichier historique d'expérimentation, la partie annotation a posteriori n'a pas été réalisée. Cependant, j'ai étudié le sujet, et en particulier, dans les spécifications externes, j'ai soumis aux ergonomes une proposition de palette d'outils d'annotation sous la forme de "scénarimage" (photos et enchaînements d'écrans).

4.2.3.4. L'analyse d'une session d'expérimentation

Je rappelle que l'outil se limite pour l'instant aux fonctions d'un fouineur et synthétiseur d'informations. En conséquence, j'ai centré mon étude sur les éléments exploités habituellement par les ergonomes dans leur analyse d'une session d'expérimentation et j'ai cherché pour ceux-ci des formes de présentation synthétique et lisible. L'objectif est de montrer à l'ergonome des informations de plus haut niveau d'abstraction que celles présentées lors de la visualisation. Pour cela, j'ai étudié des documents de synthèse prêtés par les ergonomes dans lesquels ils présentent des résultats de nature quantitative ou qualitative.

En ce qui concerne les informations qualitatives, le savoir-faire et l'expérience de l'ergonome sont, aujourd'hui, encore indispensables. Néanmoins, l'outil SupraAnalyse met à sa disposition les annotations renseignées par les compères et un graphe qui présente sous forme synthétique l'activité de l'utilisateur. Associées à l'enregistrement de l'interaction du sujet, ces informations lui donnent une vue globale de la session d'expérimentation.

Je propose aussi d'assister l'ergonome lors de la rédaction de la synthèse et lors de la mise en forme des résultats en complétant les services offerts par l'outil avec ceux d'autres logiciels couramment employés en bureautique. Par exemple, pour tous les calculs statistiques, graphiques ou autres, SupraAnalyse permet l'interopérabilité avec le tableur Excel. En conséquence, tous les fichiers de type résultat d'analyse (FRA) respectent le format Excel. Dans un premier temps, l'ergonome peut exporter vers Excel uniquement les informations

renseignées par les compères, c'est-à-dire celles qui concernent les scénarios (début, fin, durée et appréciations d'un scénario) et les annotations de pré-analyse en ligne (les quatre critères de caractérisation des incidents). Cette passerelle vers un tableur permet à l'ergonome d'automatiser des calculs algébriques, de faire des tris, des comparaisons et facilite la mise en forme de toutes sortes de graphique et leur impression, voire même leur exportation vers l'éditeur de texte qui sert à la rédaction de la synthèse. Les formes de représentation choisies pour les données contenues dans le fichier historique d'expérimentation sont décrites dans le paragraphe 4.3. Celui-ci expose les spécifications externes de l'interface homme-machine de SupraAnalyse que j'ai proposée aux ergonomes.

4.3. Spécifications externes de l'outil réalisé

Je me limite à présenter ici les spécifications externes de l'outil SupraAnalyse tel qu'il a été réalisé. En conséquence, ce paragraphe traite essentiellement des fonctions de visualisation et d'analyse et ne décrit pas mes propositions pour l'annotation.

SupraAnalyse est une application Macintosh traditionnelle qui permet d'ouvrir un fichier et de l'afficher dans une ou plusieurs fenêtres. En l'occurrence, le contenu du fichier historique d'expérimentation est affiché dans trois fenêtres différentes (figure 4.4) : la fenêtre de visualisation permet de rejouer l'interaction du sujet à l'aide d'un outil appelé "magnétoscope", la fenêtre d'annotation montre la pré-analyse en ligne des compères et la fenêtre d'analyse présente sous forme graphique une synthèse de l'activité du sujet. Les deux premières fenêtres font appel à la fonction de visualisation ; la dernière est liée à la fonction d'analyse, et les contenus de ces trois fenêtres sont fortement synchronisés. Les paragraphes suivants détaillent le contenu de ces fenêtres et de la barre de menu. La figure 4.4 montre les trois fenêtres de SupraAnalyse : la fenêtre de visualisation avec son outil de navigation, puis celle d'annotation et enfin celle d'analyse.

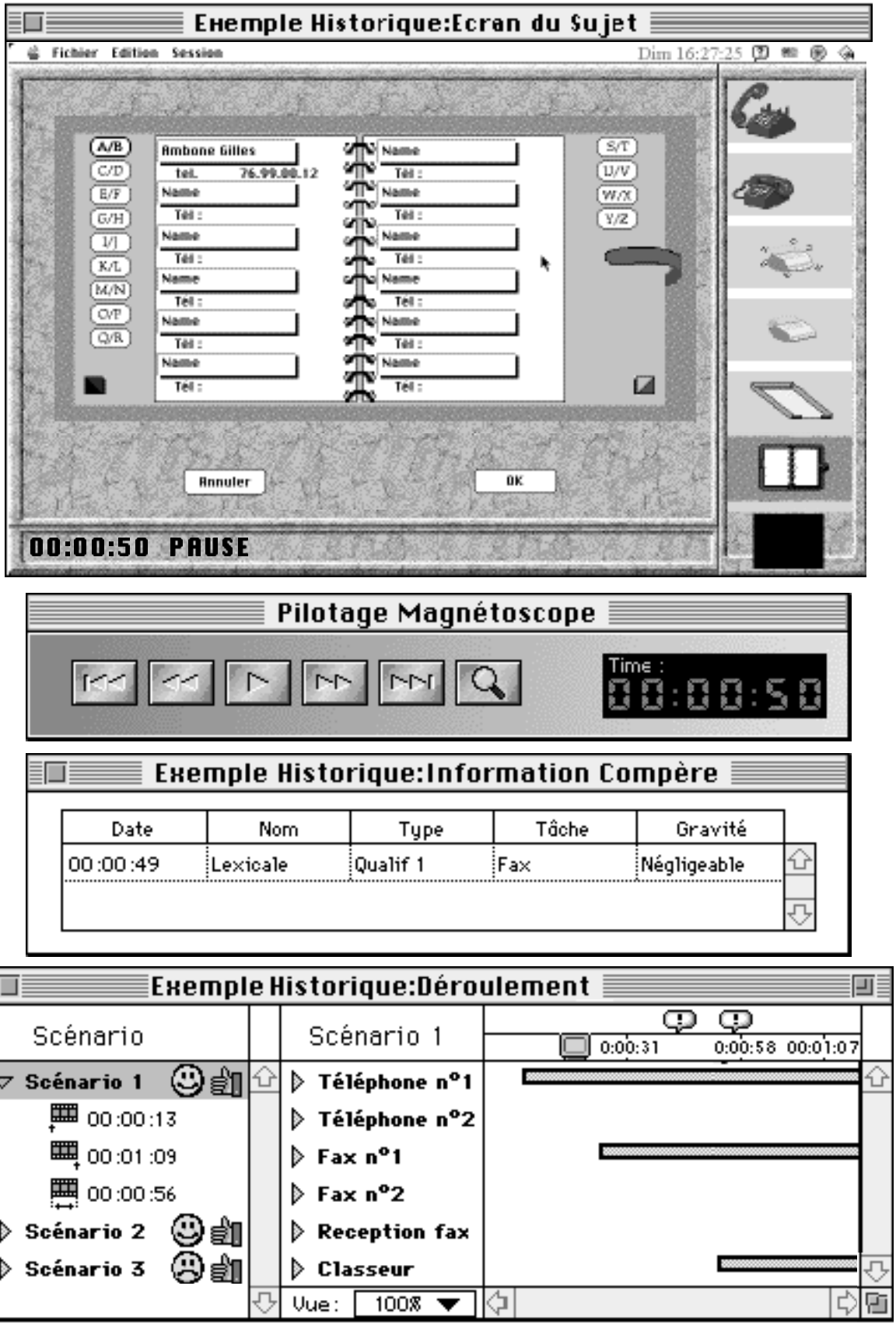


Figure 4.4 : Les trois fenêtres de l'application SupraAnalyse.

4.3.1. La fenêtre de visualisation de l'interaction

L'ergonome a la possibilité de visionner la session d'expérimentation comme un film audiovisuel et de piloter le déroulement de la visualisation avec un outil informatique similaire à un magnétoscope. Par analogie avec le monde audiovisuel, deux fenêtres ou “dialogues” complémentaires (écran et magnétoscope) permettent de rejouer l'interaction capturée pendant l'expérimentation.

Comme le montre la figure 4.5, la fenêtre “écran” est une reproduction fidèle de l’écran du sujet. Par exemple, l’utilisateur a décroché le téléphone (icône téléphone décroché en haut à droite) et il est en train de consulter le répertoire téléphonique (icône répertoire ouvert en bas à droite et répertoire en pleine page sur l’écran).

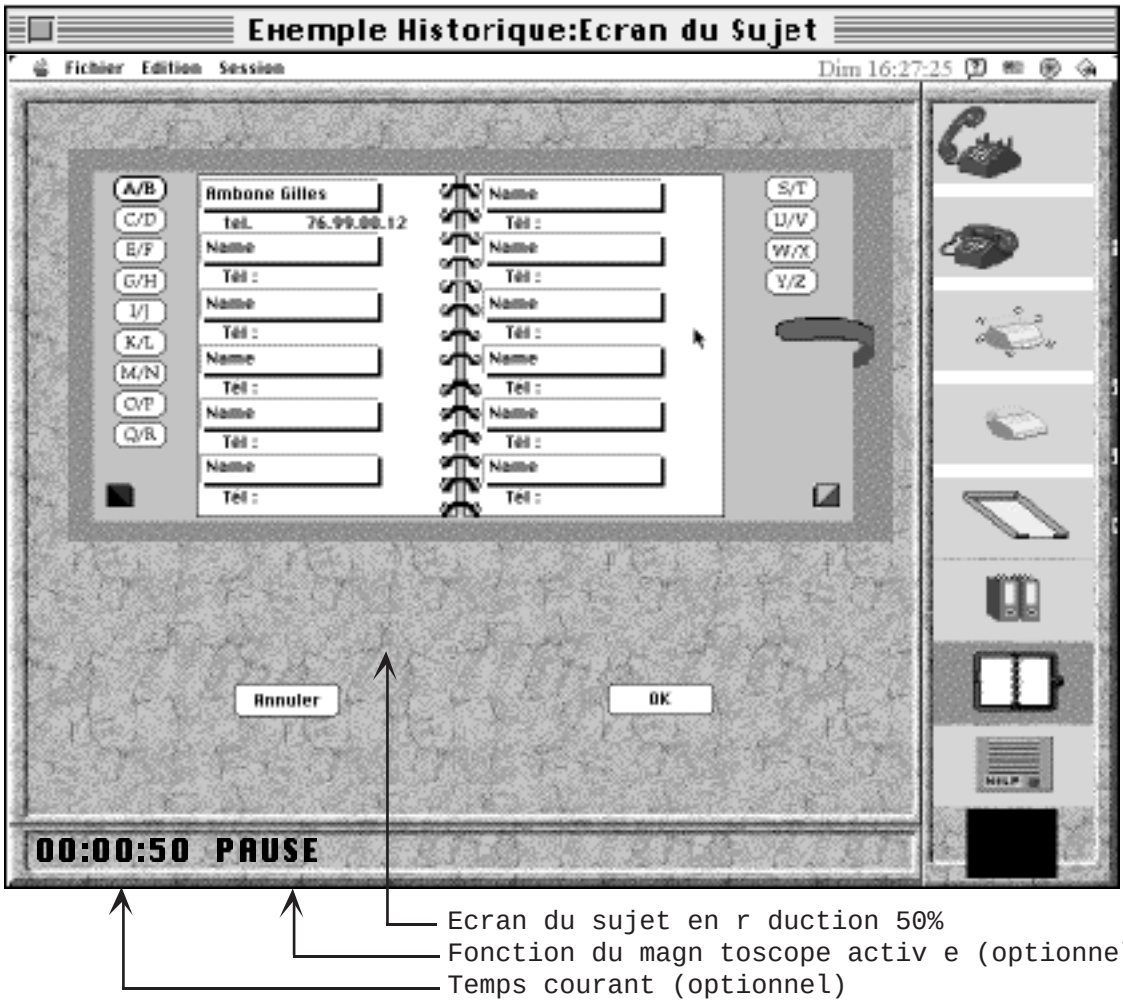


Figure 4.5 : Le film qui reproduit l'écran du sujet lors de la session d'expérimentation.

La fenêtre “magnétoscope” (figure 4.6) permet de piloter le déroulement de la visualisation observable sur la fenêtre “écran”. En plus des fonctions classiques (lecture, avance et retour rapides), on dispose de fonctions étendues par rapport à un magnétoscope traditionnel. En incrustation dans la fenêtre de l'écran du sujet (figure 4.5), on retrouve le mode de fonctionnement du magnétoscope et le temps courant de lecture (cet affichage est optionnel).

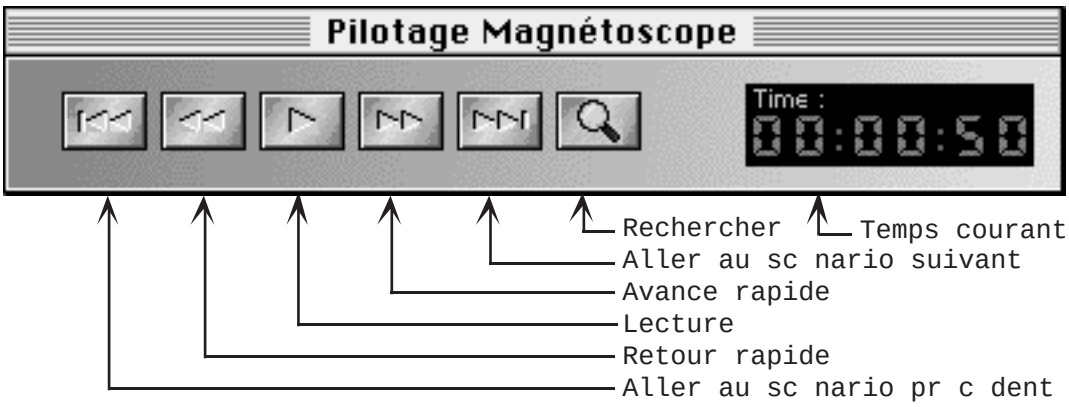


Figure 4.6 : L'outil de pilotage du magnétoscope.

Quelques extraits du dossier de spécifications externes me permettent d'exposer les fonctions offertes avec l'outil “magnétoscope”. Le tableau à trois colonnes de la notation UAN qui décrit chacune des tâches est toujours accompagné de quelques lignes de commentaires. Pour montrer les possibilités d'UAN, je décris les première et deuxième colonnes du tableau sur un exemple, en décomposant les tâches jusqu'au niveau d'abstraction le plus bas, décrivant ainsi les plus petites actions que peut faire l'utilisateur et le retour de l'interface qui leur est associé. Quant à la troisième colonne qui présente l'état de l'interface, je me contente d'un niveau d'abstraction élevé (celui sur lequel j'ai appuyé la conception).

Dès que l'utilisateur a ouvert un fichier historique d'expérimentation, il peut accéder aux fonctions suivantes :

- démarrer la visualisation (tâche “jouer”),
- l'arrêter (tâche “arrêter”),
- se déplacer en avant et en arrière, en pas à pas ou en continu (tâches “avancer ...” et “reculer ...”),
- se déplacer vers le scénario suivant ou précédant la position courante dans le fichier (tâche “aller à scénario ...”),

- se déplacer vers une date particulière en modifiant la date du compteur ou en se plaçant le curseur de position sur l'échelle de temps de la fenêtre d'analyse (tâche “aller à date …”),
- faire une recherche en fonction d'un critère dans l'ensemble du document à partir de la position courante pour trouver une annotation particulière renseignée lors de la pré-analyse des compères (tâches “chercher” et “poursuivre chercher”).

L'ensemble de cette description en langue naturelle constitue la tâche “Rejouer”. J'expose trois exemples de tâches exprimées en UAN : la tâche composée “Rejouer” (tableau 4.7) et deux de ses sous-tâches élémentaires “Jouer” (tableau 4.9) et “Avancer image par image” (tableau 4.10) qui décrivent les actions physiques de l'utilisateur.

Tâche : REJOUER		
Action de l'utilisateur	Retour de l'interface	Etat de l'interface
jouer		
| arrêter | avancer image par image | avancer rapidement | reculer image par image | reculer rapidement | aller à scénario précédent | aller à scénario suivant | aller à date direct | aller à date par échelle de temps | (chercher poursuivre chercher*)		Est fin_document : est-en-marche = faux bs-jouer = non disponible bs-arrêter = disponible remplacer bouton (bs-arrêter, bs-jouer)

Figure 4.7 : La tâche “Rejouer” en UAN.

Dans le tableau 4.7, le symbole “ | ” se lit “OU”. Il indique qu'il est possible de faire au choix l'une ou l'autre des sous-tâches. La séquence “(chercher poursuivre chercher)” précise que les deux tâches s'exécutent en séquence et le symbole “ * ” indique que la tâche peut être exécutée 0 ou n fois. En conséquence, “(chercher poursuivre chercher*)” signifie qu'une fois la tâche “chercher” exécutée, il est alors possible d'exécuter autant de fois que l'on souhaite la tâche “poursuivre chercher”.

La dernière ligne du tableau 4.7 montre que lorsque le compteur atteint la date de fin du document, le bouton jouer est remplacé par le bouton arrêt et le magnétoscope est à l'arrêt. Il est à noter que l'emplacement de ce bouton peut prendre deux significations différentes (marche et arrêt). Il en est de même dès la mise en marche du magnétoscope : le bouton “jouer” est remplacé par le bouton “arrêter” et le bouton “chercher” (recherche d'annotation) n'est plus disponible (figure 4.8). Une fois le magnétoscope en marche (en bas de la figure 4.8), le compteur évolue au rythme de la seconde, les images associées à cette date dans le fichier historique d'expérimentation sont affichées et un curseur de temps est positionné sur le graphe de la fenêtre d’analyse (voir la figure 4.14 page 8). Lors de l'arrêt du magnétoscope (en haut de la figure 4.8), la fenêtre qui contient l'écran du sujet se fige sur la dernière image affichée. Un nouveau démarrage du magnétoscope repart de cette image.

Figure 4.8 : Le “magnétoscope” à l'arrêt (haut) et en état de marche (bas).

La sous-tâche “Jouer” qui correspond à une partie de cette description est traduite en UAN dans le tableau 4.9. La séquence “Est condition : action” est une précondition ; elle se lit “si condition alors action”. Dans ce cas, elle exprime que le bouton “jouer” est disponible lorsqu'il est possible de démarrer la visualisation du document et qu'il suffit de le sélectionner (clic souris) pour lancer la visualisation et mettre le magnétoscope en marche. “~[bs-jouer] M$\vee\wedge$ ” indique un déplacement du curseur (“ ~ ”) vers le bouton “bs-jouer” puis l'appui (“ $\vee$ ”) sur le bouton de la souris (“ M ”) et enfin (“ $\wedge$ ”) le relâcher du bouton de la souris : c’est le clic souris.

Tâche : JOUER		
Action de l'utilisateur	Retour de l'interface	Etat de l'interface
Est bs-jouer disponible : ~[bs-jouer] M$\vee\wedge$	 erase (bs-jouer) display (bs-arrêter)	 sens = avant est-en-marche = vrai image = image-suivante rafraîchir (fenêtres-présentes) rafraîchir (compteur) rafraîchir (curseur) bs-jouer = non disponible bs-chercher = non disponible bs-arrêter = disponible remplacer bouton (bs-jouer, bs-arrêter)

Figure 4.9 : La sous-tâche “Jouer” en UAN.

La sous-tâche “Avancer image par image” est décrite par le tableau 4.10. L'avance image par image est activée lorsqu’un seul clic souris est appliqué sur le bouton avancer. L'expression “tâche1 (t>n)tâche 2” indique que la tâche2 est réalisée après un temps supérieur à n après la tâche1. Dans ce cas, il s'agit de calibrer un clic souris, pour différencier un clic simple d'un clic prolongé utilisé pour l'avance rapide (les images défilent tant que l'utilisateur appuie sur le bouton). La colonne “Retour de l’interface” indique que le bouton bp-avancer est mis en évidence (vidéo inverse) tant qu’il est sollicité par l’utilisateur.

Tâche : AVANCER IMAGE PAR IMAGE		
Action de l'utilisateur	Retour de l'interface	Etat de l'interface
~[bp-avancer] M∨ t(>t simple clic)M∧	bp-avancer! bp-avancer-!	 sens = avant image = image-suivante rafraîchir (fenêtres-présentes) rafraîchir (compteur) rafraîchir (curseur)

Figure 4.10 : La sous-tâche “Avancer image par image” en UAN.

4.3.2. La fenêtre de pré-analyse

La fenêtre de pré-analyse ou d’annotation (figure 4.11) présente sous la forme d'un tableau les quatre critères d'annotations renseignés par les compères. Ces informations sont utiles pour l'étude des incidents rencontrés par les utilisateurs au cours du déroulement des scénarios. Le contenu de cette fenêtre est mis à jour dynamiquement de manière synchronisée avec la visualisation de l'écran du sujet.

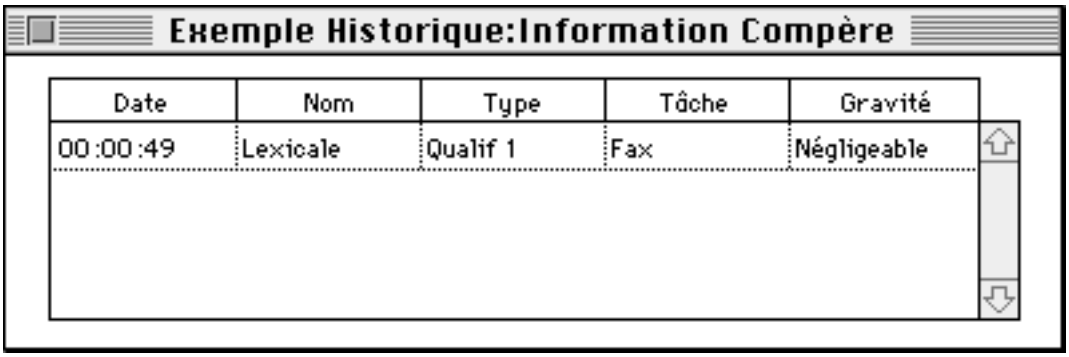


Figure 4.11 : Les annotations des compères.

Dans l’exemple de la figure 4.11, on relève une erreur mineure de type lexicale au temps 00:00:49 liée à la tâche de manipulation de fax. En consultant la fenêtre “écran sujet”, l'ergonome peut s’informer du contexte, faire des rapprochements entre l’incident et les annotations qui lui correspondent et juger de leur pertinence.

Une annotation peut aussi servir de critère de recherche. Par exemple, l’ergonome peut positionner la visualisation sur la prochaine erreur de type lexicale.

4.3.3. La fenêtre d'analyse

Cette fenêtre présente de manière synthétique et sous forme graphique l'intégralité de la session d'expérimentation. Elle comprend deux parties complémentaires présentées sur la figure 4.12 : la liste des scénarios (colonne de gauche) et le détail du scénario sélectionné (colonnes du centre et de droite de l'écran).

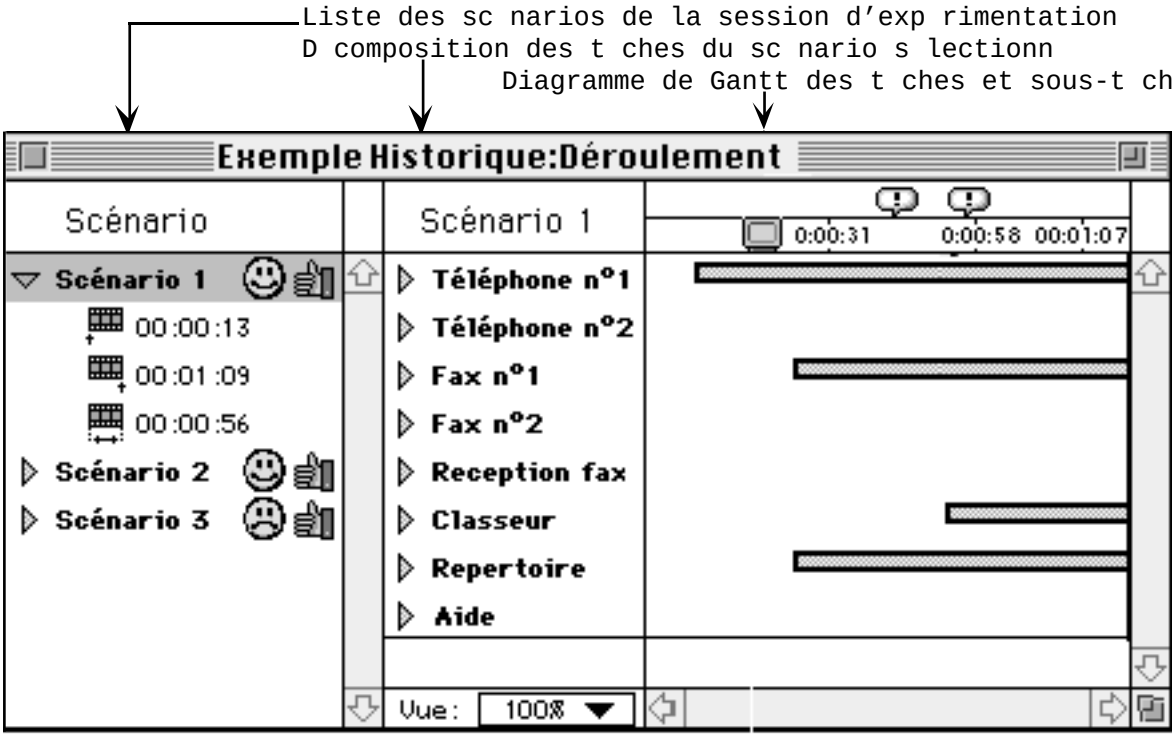


Figure 4.12 : La fenêtre d'analyse.

Dans la zone de gauche, une liste présente la suite des scénarios qui composent la session d'expérimentation. Des informations supplémentaires (figure 4.13), renseignées par les compères pendant la session d'expérimentation, sont associées à chacun des scénarios : la date de début, la date de fin et la durée du scénario. Ces différentes dates sont à l'échelle de l'ensemble de la session d'expérimentation, comme sur un enregistrement audiovisuel. Elles sont exprimées en heures, minutes, secondes et permettent de situer la position relative du scénario dans le fichier historique d'expérimentation. Pour des raisons de lisibilité, ces informations peuvent être apparentes ou pas. C'est le rôle du triangle "▷" à gauche du nom de scénario qui fonctionne de façon identique à la visualisation hiérarchique des fichiers sur le bureau Macintosh. Dans la figure 4.13, le "scénario 2" débute 1 minute 44 secondes après le

début de la session d'expérimentation, se termine 2 minutes 25 secondes après le début de la session d'expérimentation et dure 41 secondes.

Les deux petites icônes à gauche permettent d'apprécier l'écart entre le travail prescrit au sujet dans le scénario et le travail effectivement réalisé. Elles sont renseignées lors de la session d'expérimentation par un compère qui a pour rôle d'émettre son avis sur la réussite et le respect du scénario, en prenant en compte les deux points de vue : celui du sujet (“ ”) et le sien (“ ”). Dans le scénario 2, le sujet pense avoir mené à bien le scénario et le compère le confirme, alors que dans le scénario 3, le sujet pense avoir échoué.

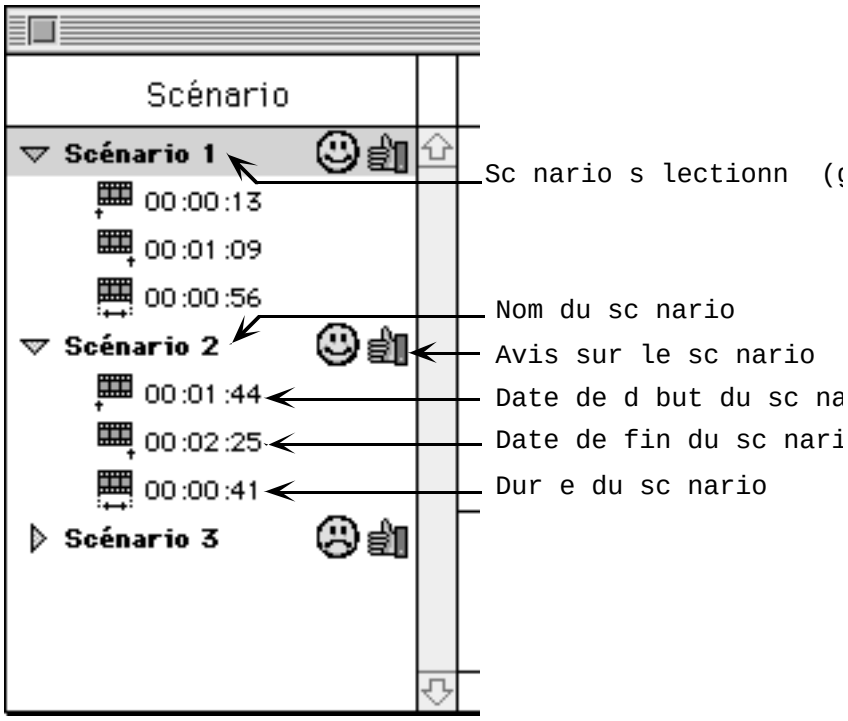


Figure 4.13 : La liste des scénarios de la session d'expérimentation.

En sélectionnant l'un des scénarios de la liste, l'ergonome peut faire correspondre à droite l'analyse des tâches effectives de l'utilisateur pour le scénario considéré. Cette représentation graphique qui s'appuie sur la capture automatique des tâches exécutées par l'utilisateur, décrit la dynamique de l'activité. J'ai choisi de l'exprimer avec une décomposition hiérarchique des tâches sous la forme d'un graphe de Gantt, diagramme couramment utilisé pour le suivi de planning (figure 4.14). L'ergonome a ainsi une vue globale de l'entrelacement des tâches effectuées par le sujet. Associé à ce graphe, l'enregistrement de l'interaction du sujet fournit des

informations complémentaires pour analyser les fonctions utilisées, les chemins suivis et les procédures adoptées par les utilisateurs. Par exemple, il est aisé de voir le chemin d'accès aux tâches et aux sous-tâches, l'ordre d'enchaînement des paramètres choisis par le sujet dans une sous-tâche (ordre de remplissage d'un formulaire, etc.). Dans la figure 4.14, on voit que l'utilisateur a entrelacé trois tâches différentes : il a utilisé le téléphone, puis le fax, et enfin il a consulté le répertoire téléphonique.

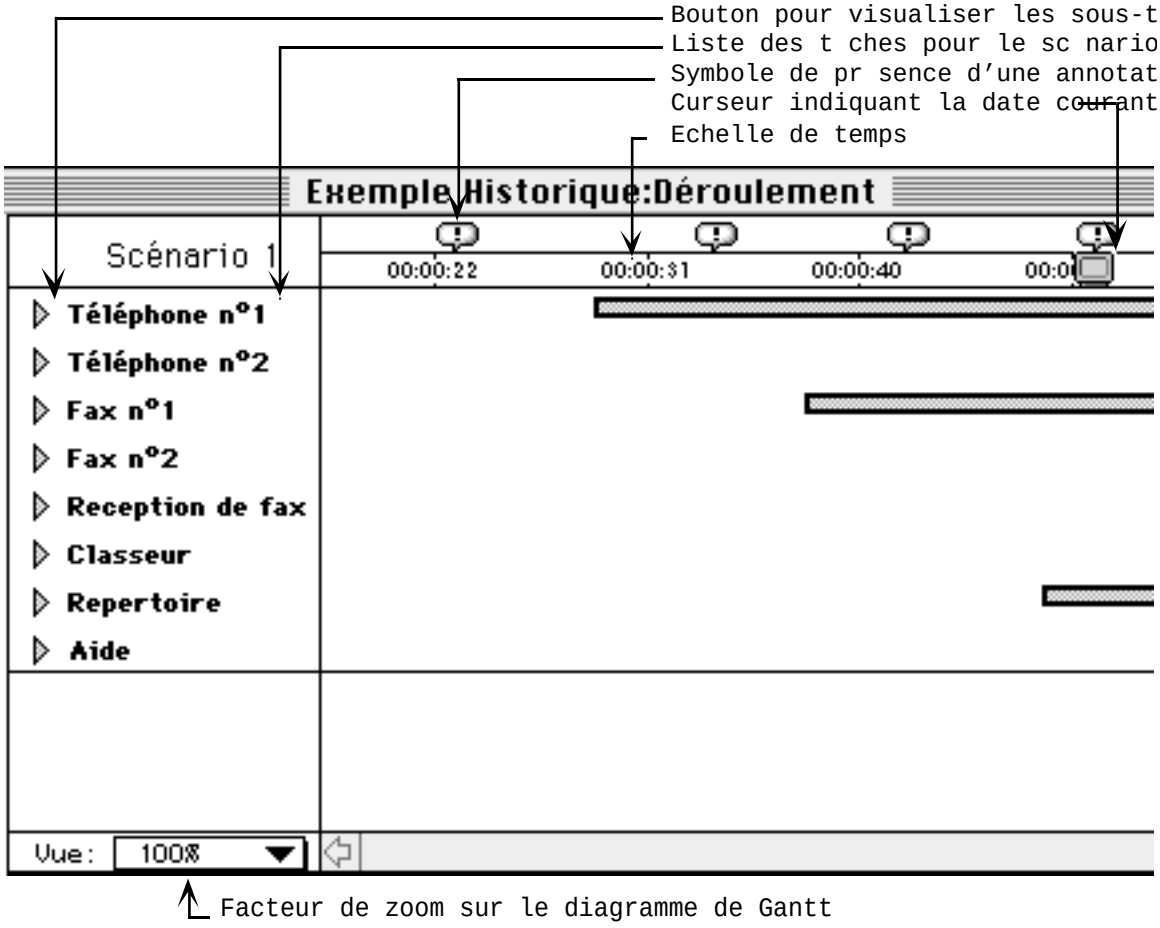


Figure 4.14 : L'analyse des tâches effectuées par le sujet.

La fenêtre d'analyse apporte quelques facilités d'emploi supplémentaires pour la navigation dans le fichier historique d'expérimentation et la consultation des annotations. Ainsi, en cliquant dans la zone de l'échelle des temps qui surplombe le diagramme de Gantt (figure 4.14), on se rend immédiatement à la date pointée et le curseur "" qui indique la date courante est mis à jour. Au-dessus de cette échelle, les annotations des compères représentées par l'icône "" sont situées par rapport à l'ensemble du scénario (leur affichage est optionnel). L'ergonome peut visualiser le contenu d'une annotation en cliquant sur son icône. Le menu déroulant intitulé

"Vue" permet d'agrandir ou de réduire le diagramme de Gantt : "100%" correspond à visualiser l'intégralité du scénario dans la fenêtre.

4.3.4. La barre de menu

Ce paragraphe présente les fonctions offertes par la barre de menu illustrée par la figure 4.15. Notons que certaines fonctions ("Rechercher", "Poursuivre la recherche", "Cacher/Montrer une fenêtre"…) sont disponibles à la fois dans les menus et les dialogues. L'utilisateur peut alors choisir à son gré entre un clic sur un bouton, la sélection d'un menu ou un raccourci clavier. Les fonctions disponibles uniquement dans le menu présentent une fréquence d'utilisation plus faible.

Figure 4.15 : La barre de menu.

Dans le menu "Fichier", on retrouve les fonctions de manipulation traditionnelles sur les fichiers (Ouvrir, Fermer, etc.), la fonction qui permet de quitter l'application, et une fonction de paramétrage "Préférences". Cette dernière permet en particulier d'afficher les annotations en avance par rapport au moment où elles ont été saisies par les compères. En effet, l'annotation enregistrée est postérieure au fait observé par les compères puisque c'est après avoir vu le sujet dans une situation particulière qu'ils ont pu déterminer les quatre critères de qualification de la situation. Il est donc possible de paramètrer en secondes le temps de recouvrement de ces deux informations. Par exemple, si le temps de recouvrement est de 4 secondes, alors la visualisation de la fenêtre "écran" à l'instant t force l'affichage dans la fenêtre "annotation" de toutes les annotations saisies entre t et t+4 secondes.

Le menu “” permet l’accès à la boite de dialogue habituelle “A propos de ...”. Le menu “Exportations” permet d'exporter des informations vers Excel. Le menu “Vues” sert à gérer l'affichage des différentes fenêtres (ouvrir et fermer les fenêtres).

En annexe 4, on trouve sur une double page, un exemple de consultation d'un fichier historique d'expérimentation.

4.4. Architecture globale de l’outil réalisé

Pour développer le logiciel, j'ai respecté les principes de conception logicielle appliqués au domaine des interfaces homme-machine énoncés dans le chapitre 3. L'architecture fait référence au modèle PAC (Présentation - Abstraction - Contrôle), basé sur la notion d'agents ou objets interactifs [Coutaz 90].

4.4.1. L’architecture PAC de SupraAnalyse

Je rappelle que le modèle PAC permet la construction d'un système informatique interactif sous la forme d'une structure composée d'une hiérarchie d'agents coopérants. Les agents se décomposent à divers niveaux d'abstraction selon trois facettes complémentaires : la Présentation ou partie perceptible de l'agent, l'Abstraction ou composante fonctionnelle de l'agent et le Contrôle qui joue le rôle de pont entre la représentation et l'abstraction et sert de lien de communication avec les autres agents du système.

La figure 4.16 présente l'architecture PAC retenue pour SupraAnalyse.

Il existe des règles heuristiques de mise en œuvre du modèle PAC qui s’appuient sur les spécifications externes du système [Nigay 94].

- Tout lieu d’interaction privilégié est un agent. En l’occurrence, une fenêtre qui sert de support de travail est modélisée par un agent.
- Lorsque le contenu d’une fenêtre est complexe, l’agent fenêtre a des sous-agents.
- Le maintien de la cohérence entre les vues multiples est assuré par un agent père “vues multiples”.

Ces trois règles s’appliquent au cas de SupraAnalyse. On retrouve dans l’architecture PAC de SupraAnalyse les trois fenêtres principales (sujet, compère et analyse) : SubjectView, WizardView, AnalyseView ainsi que la fenêtre “magnétoscope” appelée MoviePlayer. La fenêtre AnalyseView est un exemple d’agent qui se décompose à l’aide de quatre sous-agents. Dans notre cas, l’agent “vues multiples” est représenté par l’agent Document.

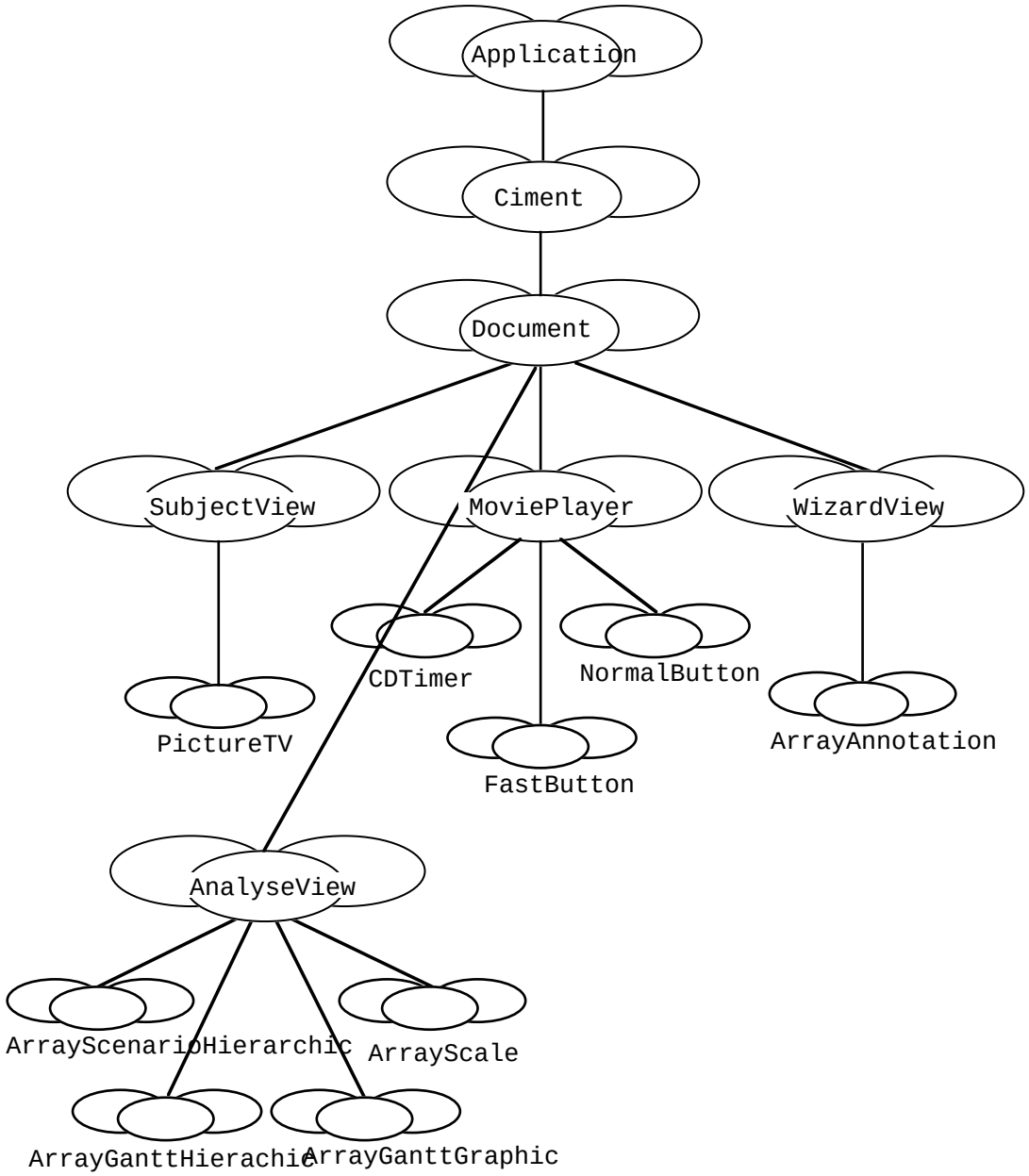


Figure 4.16 : La modélisation de SupraAnalyse à l'aide du modèle PAC.

L'application systématique des règles de [Nigay 94] peut conduire à une prolifération d'agents. Ce phénomène est en accord avec le principe de la modularité, mais il peut selon les plates-formes de mise en œuvre entraîner un excès de communication par messages. C'est pourquoi, il existe une règle de réduction visant à combiner un agent et son fils unique. Il est clair que je n'ai pas appliqué cette règle à tous les niveaux. En effet, dans le cas de SupraAnalyse, tous les agents qui ont un fils unique (Ciment, SubjectView, WizardView) sont susceptibles dans une version future de contrôler une seconde grappe d'agents.

Je vais maintenant décrire les agents identifiés dans l'architecture de SupraAnalyse. Remarquons que chaque agent est spécialisé et que les traitements sont répartis à différents niveaux d'abstraction tels que celui de l'application, du document, des fenêtres. Chaque agent PAC mémorise un état de l'interaction, laissant à l'utilisateur l'initiative de la conduite du dialogue.

L'agent Application correspond au plus haut niveau d'abstraction, celui de SupraAnalyse, en tant qu'application Macintosh traditionnelle. La présentation se contente de définir la vue initiale du système sous forme d'une icône. Une fois l'icône ouverte, l'image du système est entièrement assurée par les constituants de l'interface utilisateur.

Ciment est un "super-agent" qui cimente les actions distribuées ; il gère en particulier les fichiers. Actuellement, son existence est justifiée par le fait que SupraAnalyse ne doit pas se limiter à la consultation d'un fichier d'expérimentation, mais doit pouvoir étendre son champ d'action à des fichiers de types différents. Moyennant de lui adjoindre d'autres services tel que l'arbitrage, Ciment doit pouvoir gérer plusieurs fichiers en entrée et en sortie (de type expérimentation, analyse et résultat) et par là même, permettre la comparaison entre fichiers. Dans le cas où les fichiers en entrée sont des fichiers d'analyse et non seulement des fichiers d'expérimentation, l'agent Document doit être personnalisé pour offrir les fonctions adéquates. Pour ce qui est des fonctions évoluées de l'analyse automatique des données, il est nécessaire de lui ajouter un agent fils spécialisé dans ce type de tâche.

L'agent Document agit comme un contrôle de multi-fenêtrage capable de gérer toute la synchronisation et la cohérence entre les objets inférieurs. Il fait correspondre une image du système à un fichier d'expérimentation et coordonne toutes les actions réparties sur cette image. Pour ce faire, il contrôle l'état de l'interaction sur l'ensemble des fenêtres et se charge de gérer la correspondance entre les états de chacun des objets qui lui sont rattachés. Toute modification émise par l'un de ces objets est transmise au contrôle de Document. Celui-ci les répercute sur ses constituants (les fenêtres) via leur contrôle, qui se charge alors de la traduction des événements reçus. Le concept étant réparti sur l'ensemble des fenêtres et sur plusieurs niveaux d'abstraction (fils de chacune des fenêtres), les messages circulent entre les agents dans la hiérarchie des objets PAC grâce au mécanisme d'indirection aménagé dans les contrôles.

Toutes les fenêtres du système sont des entités PAC spécialisées. Elles s'appuient sur une organisation qui permet de réutiliser les services de l'environnement graphique. Le contrôle d'une fenêtre graphique gère l'état de la fenêtre et vérifie le droit d'interpréter un événement. Il agit avec l'abstraction comme un intermédiaire et un gestionnaire local de l'état de l'interaction. Voici la liste des fenêtres dont s'occupe l'agent Document et leur rôle :

- L'agent SubjectView rejoue l'interaction sur l'écran du sujet. Son seul agent fils est PictureTV qui présente le contenu de la fenêtre (partie graphique).
- L'agent WizardView contient les annotations des compères. Son unique fils est ArrayAnnotation qui présente le contenu de la fenêtre (partie graphique).
- L'agent AnalyseView montre l'analyse de tâches de l'expérimentation. Ses quatre fils sont ArrayScenarioHierarchic pour la décomposition des scénarios, ArrayGanttHierarchic pour la décomposition hiérarchique des tâches, ArrayGanttGraphic pour la représentation graphique des tâches et ArrayScale pour l'échelle de temps du diagramme des tâches.

L'agent MoviePlayer correspond au magnétoscope et permet d'agir sur l'ensemble des fenêtres. Ses fils sont les trois agents CDTimer qui gère la date courante de lecture sur le fichier d'entrée (timer du magnétoscope), NormalButton pour un bouton de magnétoscope simple (clic court) et FastButton pour un bouton de magnétoscope à deux comportements (clic court et clic prolongé). Il existe plusieurs instances de FastButton (avance et recul rapide) et de NormalButton (marche, arrêt, chercher, scénario suivant et précédent).

Bien que l'agent MoviePlayer ait besoin d'établir des communications avec les autres objets de même niveau dans la hiérarchie, il ignore l'existence de ses voisins. En effet, d'après les principes PAC, ceux-ci ne se désignent jamais directement et passent par une indirection gérée dans le contrôle de niveau supérieur ; en l'occurrence, toute référence s'effectue par l'intermédiaire de l'objet père Document. Ce principe augmente la réutilisabilité et diminue l'interdépendance entre les agents.

Les feuilles de la hiérarchie PAC sont des objets élémentaires qui tendent à reproduire le comportement intrinsèque d'un objet physique. Les objets élémentaires ne sont pas nécessairement des objets PAC au sens strict du terme. C'est le cas des agents NormalButton et FastButton qui sont hérités de l'environnement et ne possèdent pas de partie abstraction. Ils peuvent être considérés comme une extension de la boîte à outils.

L'étape suivante du développement a consisté à mettre en œuvre l'architecture choisie. Pour ce faire, nous avons opté pour l'environnement de développement Symantec qui s'appuie une bibliothèque de classes objet réalisée en C++ (Think Class Library). Nous allons voir comment le modèle PAC s'intègre dans la conception objet.

4.4.2. La conception orientée objet et le modèle PAC

PAC n'impose aucune méthode d'implémentation particulière, mais les agents PAC semblent tout naturellement destinés à être codés en utilisant les techniques de programmation orientée objet. Le modèle agent retrouve dans l'approche objet les thèmes de la modularité et de la communication. Un objet et un agent sont tous deux des entités aux traitements spécialisés. Ces entités décident de leur état ; celui-ci ne peut être manipulé directement par les autres entités mais résulte d'un traitement sollicité par autrui.

La méthodologie PAC de conception d'architecture pour une application interactive consiste à construire le diagramme des agents. L'identification des agents PAC et des objets graphiques est basée sur la description des besoins à satisfaire pour l'application (Cahier des Charges ou CdC) et sur l'étude des fonctions du système dans le respect de l'interface utilisateur (Dossier de Spécifications Externes ou DSE). A partir de là, on peut imaginer les classes d'implémentation des parties abstraction et présentation et déterminer la hiérarchie des classes. Certaines questions surgissent à ce point du développement : la présentation, l'abstraction et le contrôle doivent-ils être des classes séparées et comment doivent-ils communiquer? Ces décisions doivent être prises au moment de la conception détaillée des classes en approche orientée objet, en respectant l'architecture PAC.

Pour chaque agent PAC, j'ai choisi de considérer les trois facettes présentation, abstraction et contrôle dans une même classe. Les classes séparées offrent l'avantage d'une plus grande modularité et d'une plus grande rigueur dans la communication entre les agents et les facettes d'un même agent. Bien adapté aux projets importants (temps de développement, ressources humaines disponibles, etc.), ce principe se justifie moins dans mon projet. De plus, le logiciel développé doit être maintenable et trois objets impliquent trois fois plus de classes à prendre en compte pour la personne qui va devoir appréhender le logiciel. Aussi dans SupraAnalyse, chaque agent est un objet et chaque classe d'objets est codée dans un module séparé.

Il faut souligner que toutes les communications doivent être bidirectionnelles et que les facettes présentation et abstraction d'un agent ne sont pas autorisées à communiquer directement entre elles. La figure 4.17 montre la structure d'un agent PAC telle que je l'ai définie et ses relations avec les autres agents. Les flèches "In" indiquent les communications internes à l'agent, et les flèches "En" les communications externes. La définition des méthodes générales de communication est délicate car les données rentrant dans les communications sont hétérogènes. Une relation est établie pour autoriser la communication entre les classes réalisant chaque agent. Pour simplifier le codage, j'ai choisi d'implémenter les communications requises sous la forme d'appels de fonctions (méthodes publiques).

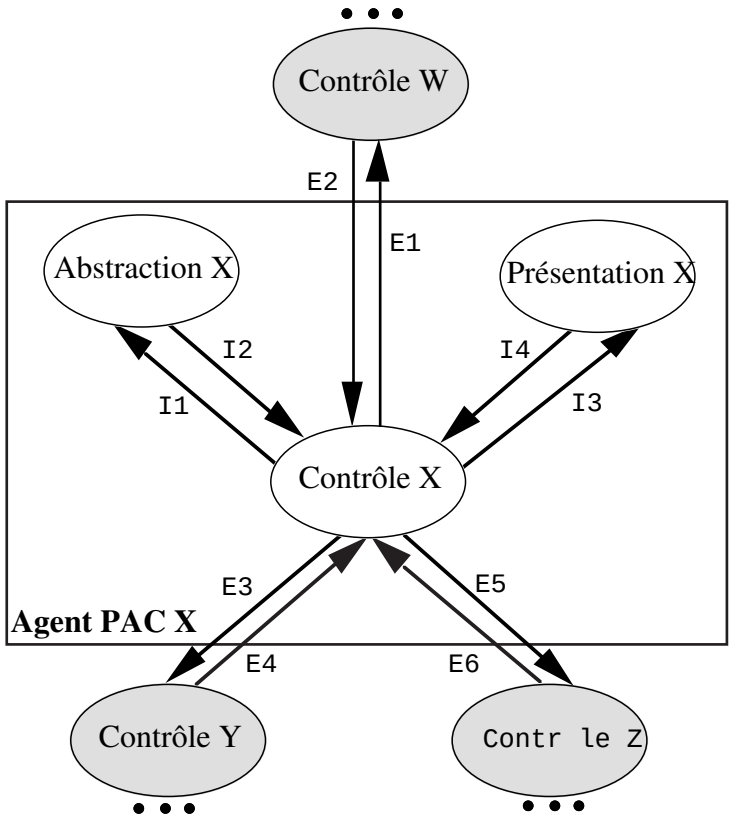
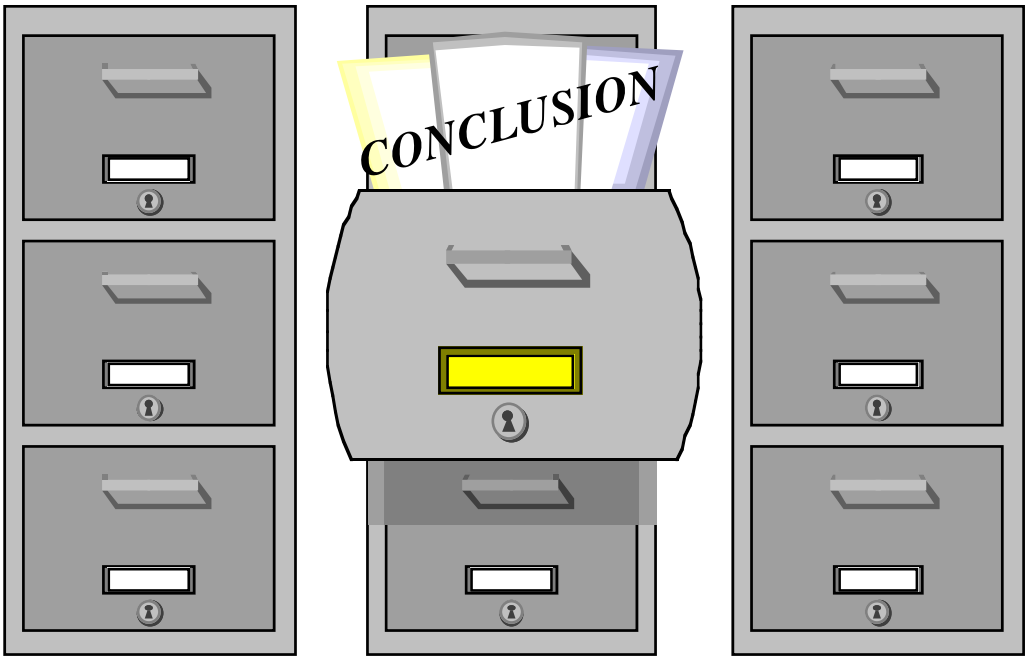


Figure 4.17 : Les communications autorisées par le modèle PAC.

Ce chapitre a présenté les éléments essentiels de SupraAnalyse. Pour plus de détail, le lecteur pourra consulter les documents élaborés pour le projet SUPRATEL ★ : le Cahier des Charges, le Dossier de Spécifications Externes, le Dossier de Conception Globale et le Manuel Utilisateur [Lischetti 94a, 94b, 94c, et 94d].

Conclusion



Bilan du travail réalisé

A l'issue du travail réalisé sur l'ensemble du projet SUPRATEL ★, notre client, le CCETT dispose à présent d'un laboratoire informatique adapté à l'étude de l'utilisabilité des interfaces homme-machine. NEIMO et SUPRATEL ★ constituent un outil d'évaluation ergonomique

complet pour lequel les services de capture et d'analyse ont été étudiés conjointement. Pour valider mon travail sur l'analyse des données comportementales recueillies pendant une expérimentation, l'interface homme-machine de SupraAnalyse a été confiée à des ergonomes (Michel Nael et Nathalie Portolan du CCETT, Jean Caelen et André Bisseret, chercheurs à l'Université de Grenoble). Leurs impressions sur l'outil ont été très positives et SupraAnalyse a été considéré comme un premier pas significatif vers l'analyse des données comportementales. SupraAnalyse va à présent servir de base de dialogue et de comparaison pour le CCETT, car celui-ci s'est doté de MUSiC (développé par la société DRUM), un logiciel permettant de mesurer l'utilisabilité d'un système.

Notons que l'équipe IHM était elle-même notre client vis à vis de NEIMO. En effet, la seconde mise en oeuvre de NEIMO dans le cadre du projet SUPRATEL★ a contribué à éprouver la validité et la robustesse de la plate-forme. L'analyse des données comportementales était un sujet relativement novateur pour l'équipe : la première application de laboratoire connectée à NEIMO a permis de valider la connexion à la bibliothèque historique mais l'analyse des données comportementales n'a pas pu être abordée, faute de temps. Pour cette raison, je comptais sur le savoir-faire expert des ergonomes, qui dans ce cadre, se sont comportés à la fois comme des clients et comme des partenaires. Leur apport a été primordial dans les phases d'analyse et de spécification des besoins, même si le dialogue s'est avéré assez difficile à établir. Il est vrai que les ergonomes n'avaient pas étudié leur activité et ils ont éprouvé des difficultés à décrire leurs besoins et partager leurs connaissances. Deux points importants m'ont permis de progresser dans la compréhension du problème : l'ouvrage de [Valentin 93] qui sert de support de travail aux ergonomes et la visite organisée pour assister à des tests dans le laboratoire d'ergonomie du CCETT. Cette dernière m'a fourni la possibilité d'étudier une mise en situation pour les différents participants à une expérimentation. Elle a en ce sens, été particulièrement riche d'enseignements à propos des besoins des ergonomes mais également à propos des compères. Cette visite m'a aussi permis de décrire la démarche d'évaluation menée par le CCETT et elle a guidé mes recherches et mes propositions pour les ergonomes.

Afin de vérifier si les services d'analyse proposés correspondaient aux besoins et aux attentes, j'ai organisé des rencontres avec des utilisateurs potentiels de l'outil dès les phases d'analyse des besoins et de définition des spécifications externes. Il est à noter qu'avant d'arriver au produit final, il a fallu de nombreux échanges pour faire préciser aux ergonomes ce qu'ils pensaient de chacune de mes propositions : comme tout utilisateur, ceux-ci étaient sensibles au fait que je modifiais leur tâche habituelle. J'ai compris la nécessité d'une conception itérative des interfaces.

La figure 5.1 positionne sur le cycle de développement les méthodes et les outils que j'ai employés ; ils couvrent une grande partie du cycle en V.

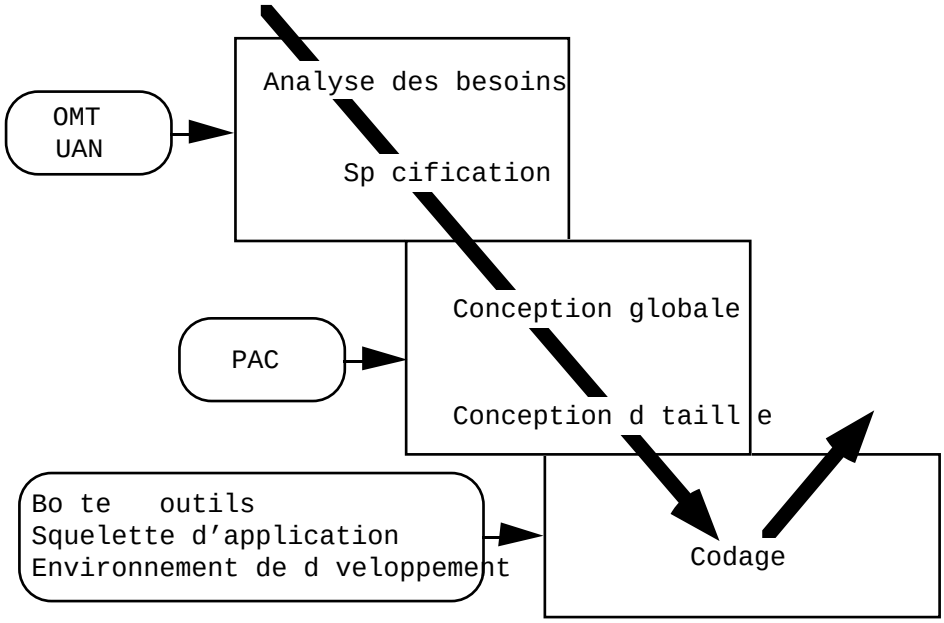


Figure 5.1 : Les outils et méthodes employés dans le cycle en V.

La méthode orientée objet OMT, particulièrement adaptée à la phase d'analyse des besoins, présente des lacunes dans le cas des systèmes interactifs : elle néglige l'utilisateur. Le langage de spécification conceptuelle UAN apporte à ce stade un complément indispensable : le système et l'utilisateur sont tous deux pris en compte. L'utilisation d'UAN dès l'analyse des besoins est relativement novatrice car UAN est généralement réservé à la phase de spécifications externes. Son application très tôt dans le cycle de développement m'a permis de faire une première évaluation prédictive de l'utilisabilité de l'interface homme-machine et a ainsi contribué à la robustesse de l'interaction. En effet, UAN m'a forcé à me poser de nombreuses questions, qui intervenant en amont de la conception m'ont permis d'éviter des anomalies.

L'application des mêmes méthodes UAN et OMT sur une partie du cycle en V a facilité le développement. Il y a une réelle continuité dans la réflexion entre les différentes étapes, puisque le concepteur ne repart pas d'une feuille blanche à chaque fois. Le principe de l'affinement entraîne la validation des étapes précédentes et on évite les sauts de notation qui engendrent fréquemment des pertes d'informations, voire des déformations. De plus, les méthodes OMT et UAN étaient partagées par les différents intervenants du projet (client, concepteurs et maître d'œuvre). Elles ont donc servi d'outil de dialogue entre deux acteurs du projet qui communiquent difficilement : le "client-utilisateur" et le "concepteur-informaticien". Même si leur apport s'était limité à cela, il aurait été justifié car cet échange est le point de départ du projet : il convient d'y porter beaucoup d'attention pour qu'en final le client soit satisfait.

L'architecture PAC prolonge efficacement OMT : les agents mis en œuvre dans le modèle sont tout destinés à être considérés comme des objets. En ce qui concerne OMT, faute d'expérience, j'ai eu des difficultés à penser en termes d'objets très tôt dans la conception et je n'ai pas su prolonger la méthode efficacement dans les phases suivantes. A trop vouloir considérer l'utilisateur, j'ai un peu négligé le système. J'ai néanmoins bâti la conception globale sur la première décomposition OMT. Mais telle que je l'avais faite, elle était incomplète et le passage de OMT à PAC m'a demandé des efforts qui auraient pu être moindres si j'avais suffisamment affiné le modèle d'objets OMT.

Quant aux outils de développement, leur choix est délicat et engage sur la suite du projet. Les squelettes d'application exploitant le langage objet sont bien appréciés pour leurs avantages, mais on néglige trop souvent la surcharge d'apprentissage qu'ils requièrent, surtout si le développeur débutant n'est pas assisté par un spécialiste et si le temps de codage prévu est court et limité. Les difficultés que nous avons rencontrées avec MacroMind Director ont attiré notre attention sur le fait que, malgré leur fort attrait, les générateurs d'interfaces ne sont pas toujours conçus en pensant interopérabilité, modularité et réutilisation. Dans le cas où les classes d'objets de présentation prêtes à l'emploi ne conviennent pas, il est nécessaire de descendre au niveau de la boîte à outils pour adapter le code généré. Mais dans ce cas, le code modifié manuellement pose des problèmes lors des modifications ultérieures faites automatiquement par l'outil : la vision idéale du "tout automatique" a ses limites.

Grâce à l'environnement de développement Symantec, le codage et le test ont été facilités. L'encapsulation de différents outils tels que la gestion des fichiers du projet, le squelette d'application, le générateur d'interface, le “debugger” offre une solution très confortable pour le programmeur. J'ai d'autant plus apprécié Symantec après mes expériences avec MPW, MacApp et MacroMind Director.

La réalisation faite dans le cadre de ce projet n'était pas destinée à un usage industriel. Le présent travail se veut être une démonstration de faisabilité. En conséquence, je n'ai pas rédigé de plan qualité formel et détaillé. Néanmoins, j'ai cherché à assurer les critères de qualité qui permettent de réutiliser et de compléter la maquette. Mon effort a surtout porté sur l'assurance qualité et j'ai mis en place des actions préventives réparties tout au long du développement. De plus, je me suis appuyée sur l'expérience de l'équipe qui a déjà beaucoup éprouvé l'efficacité des outils et des méthodes que j’ai employés. La collaboration étroite avec les quatre étudiants a permis de définir le projet et son découpage rapidement et d'un commun accord. Elle a favorisé la spécification précise des interfaces logicielles entre les différentes parties du projet. Elle a eu pour avantage immédiat de faciliter grandement l'intégration logicielle.

Perspectives

Je précise que l'interface homme-machine de SupraAnalyse n’a pas été évaluée avec la technique du Magicien d'Oz. Il serait souhaitable d'évaluer également les interfaces homme-machine des compères car la tâche de ceux-ci est difficile et une bonne interface utilisateur est le gage d'une bonne session d'expérimentation. Nous avons proposé aux ergonomes de les mettre en situation d'évaluation face à l'outil. Ceux-ci ont été très réticents à l'idée d'être observés, chose qu'ils pratiquent pourtant couramment dans leur travail.

Il subsiste quelques points qui restreignent l'utilisation de l'outil pour des tests en laboratoire. Certains sont liés aux limites actuelles de NEIMO, d'autres sont purement techniques et ont leur solution. Le point le plus contraignant pour le CCETT est lié au fait que NEIMO n'est pas une plate-forme hétérogène et n’a pas été conçue en permettant l’interopérabilité. Elle impose l’utilisation de stations Macintosh d'Apple (Quadra 700, Quadra 950 ou AV 840) reliées par le protocole de communication inter-applications basé sur les Apple Events, y compris pour la station qui accueille l'interface à évaluer. En conséquence, dans les cas où la dite interface n'a pas été conçue pour fonctionner sur Macintosh, il est nécessaire de simuler son fonctionnement pour permettre sa connexion à NEIMO.

Les évolutions possibles du produit SUPRATEL★ concernent bien sûr l'analyse des données comportementales mais aussi la conduite des sessions d'expérimentation et d'analyse. Il est clair que l'analyse s'appuie sur le contenu du fichier historique. C'est pourquoi, il faut non seulement augmenter le type des événements et des informations sauvegardés dans le fichier historique lors de la capture, mais également pouvoir les relire et les traiter lors de l'analyse.

En ce qui concerne les applications compères, les ergonomes proposent des outils d'aide à la construction de scénarios et des formulaires de renseignements sur la session. Par exemple, les points qui les intéressent concernent : l'identité du sujet, la description des scénarios, les observateurs présents et leurs rôles respectifs, la version de l'interface évaluée, etc. Pour faciliter et accélérer l'intervention des compères, nous avons imaginé la possibilité pour ceux-ci d'ajouter des annotations ou des commentaires vocaux à l'enregistrement de la session d'expérimentation.

Du côté de l'ergonome, dans l'état actuel du projet, SupraAnalyse n'autorise que la consultation du fichier historique. Il est à noter que son architecture a été conçue de façon à permettre des évolutions : les services de base sont greffés sur un noyau logiciel capable d'accueillir facilement des services complémentaires. En ce qui concerne le fichier historique, nous avons envisagé la modification et la sauvegarde de nouvelles informations, ainsi que la comparaison de fichiers. Parmi les évolutions possibles, nous suggérons de pouvoir :

- modifier et supprimer des annotations renseignées par les compères,
- ajouter des commentaires écrits ou vocaux,
- utiliser une palette d'outils qui fournit une aide à l'analyse du document,
- sélectionner et supprimer de l'analyse des périodes de temps (temps morts compressibles) tout en conservant une trace de leur emplacement et de leur durée pour les restituer à la demande,
- comparer des expérimentations du même scénario réalisées par différents sujets ou pour différentes versions de l'interface à évaluer.

Dans un premier temps, l'évolution significative vers un système vraiment multimodal consisterait à autoriser des annotations et des commentaires vocaux traduits : le son en langue naturelle serait converti en texte grâce à un logiciel de traduction automatique. Pour l'instant, il existe un seul logiciel capable de comprendre des phrases simples en anglais, mais il n'est pas suffisamment robuste.

Mais avant d'aller plus loin vers la multimodalité, il est nécessaire de trouver des solutions aux limites de NEIMO et aux difficultés techniques liées au son et à l'image. En effet, NEIMO utilise le réseau pour relier les différentes stations lors de la capture. Pour la transmission du son et de l'image dans les conditions de nos expérimentations, les performances du réseau sont insuffisantes (débit, temps de réponse). Il est incontournable d'optimiser le transfert et le traitement de ces données, d'une part pour assurer leur transport rapide sur le réseau et d'autre part pour diminuer la taille du fichier historique qui contient les données capturées. Actuellement, la variété et la taille des informations font que les fichiers historiques atteignent très vite de grandes dimensions : environ 4 MOctets pour une minute d'expérimentation. Des algorithmes de compression vont permettre de réduire ces tailles de façon très significative.

L'analyse de tâches telle que nous l'avons mise en œuvre peut s'avérer insuffisante pour des expérimentations réelles qui durent plusieurs heures. Il est important d'offrir à l'ergonome une présentation adaptée pour le grand volume de données qui en découle. Une des solutions envisagée est le Perspective Wall [Robertson 91]. Le principe adopté est illustré par la figure 5.2. Il permet de voir en gros plan une partie des données, l'ensemble des informations étant toujours visible et accessible dans un plan plus lointain.

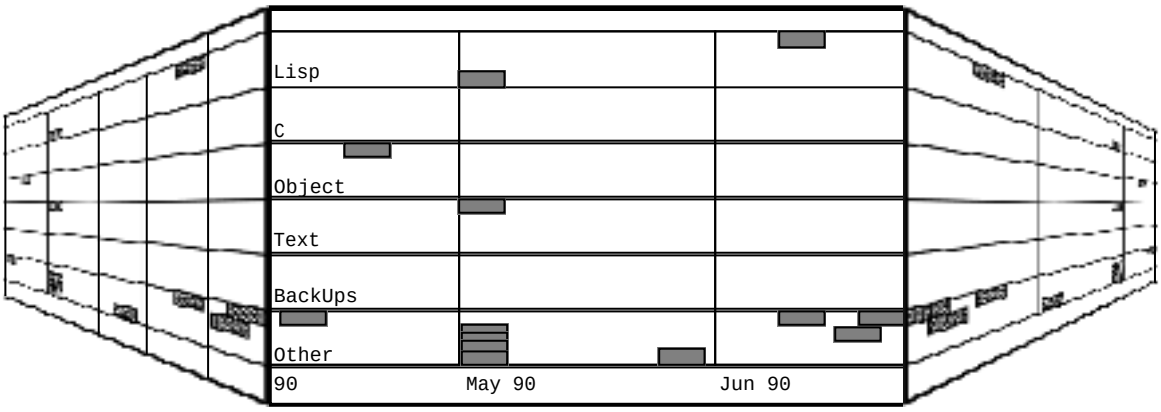


Figure 5.2 : Le Perspective Wall.

Les deux évolutions à court terme pour le projet NEIMO sont l'amélioration de la vitesse transfert des données et le Perspective Wall. Des projets ont démarré en ce sens, dès la rentrée universitaire ; ils sont basés sur les remarques et les réflexions que nous avons menées dans le projet SUPRATEL ★. De plus, un nouveau projet de quatre étudiants en DESS a débuté à la fin de l'année 1994. Il se nomme MEDIATEL ★ et s'appuie fortement sur les travaux réalisés dans SUPRATEL ★. Son objectif est de compléter l'interface homme-machine multimédia à évaluer pour lui donner une dimension multimodale (la multimodalité s'appliquera aux entrées seulement et ne comprendra que la parole et le geste via la souris). Les interfaces

homme-machine des compères seront, elles aussi, reprises et enrichies pour devenir multimédia ; elles permettront la manipulation d'informations de nature textuelle, sonore et visuelle. Ce projet MEDIATEL fournira à SupraAnalyse un second cas d'étude avec une approche de la multimodalité.

Annexes

Annexe 1 : QuickTime

Dans une session d'expérimentation, les différents événements reçus par NEIMO sont synchronisés et datés à l'aide de QuickTime, extension du System 7 Macintosh [Apple 92c] qui offre des services multimédia pour des données relatives au temps. Pour la synchronisation, QuickTime s'appuie sur un format spécifique appelé “movie” qui permet le stockage et la lecture de données de nature différente (son, image, clic souris, frappe au clavier, etc.).

Le format movie est vu selon trois niveaux d'abstraction (figure A.1) : movie, track, média. Le movie est le niveau le plus élevé : il contient différentes tracks et assure leur synchronisation. Chaque track est relative à un média et un seul. Elle indique si le média est actif ou non et précise où se trouve l'information correspondant dans le média. Le média contient les données à l'état brut.

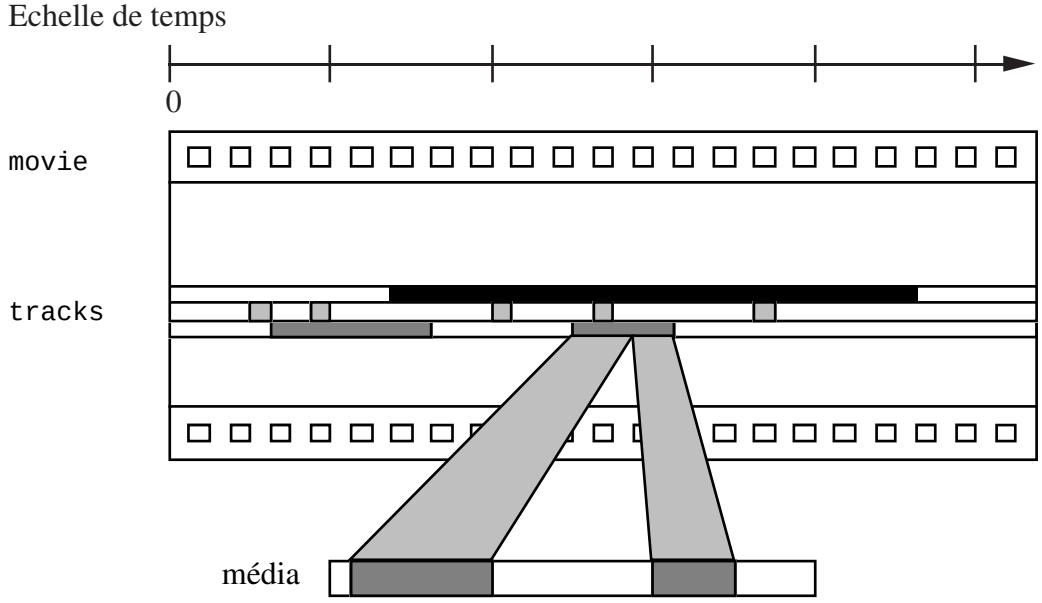


Figure A.1 : Le format movie.

Tous les détails de QuickTime sont pris en charge par la bibliothèque historique.

Annexe 2 : Les Apple Events

NeimoCom par lequel transite tous les messages, y compris les informations sauvegardées dans le fichier historique d'expérimentation, utilise le protocole de communication inter-applications Apple Event Interprocess Messaging Protocol (AEIMP) développé par Apple Computer Inc. [Apple 91a]. Celui-ci est relativement transparent pour l'application analyste. Ce sont les bibliothèques sujet et compère qui se chargent de coder les données sous la forme d'un événement ou Apple Event. La bibliothèque de sauvegarde transcode celles-ci au format QuickTime pour l'enregistrement dans le fichier historique. La bibliothèque historique garantit leur décodage lors de la lecture du fichier. Néanmoins, le cœur des données utiles à l'application analyste est au format des Apple Events.

Comme le montre la figure A.2, la structure d'un Apple Event est composée d'attributs (caractéristiques de l'Apple Event : classe, identificateur, nom de l'application réceptrice, la tâche censée être accomplie) et de paramètres. Les paramètres contiennent un mot-clé pour l'accès aux données et les données proprement dites. Dans NEIMO, les paramètres d'un Apple Event spécifient la date de l'événement, sa durée, l'application émettrice, le type et les données.

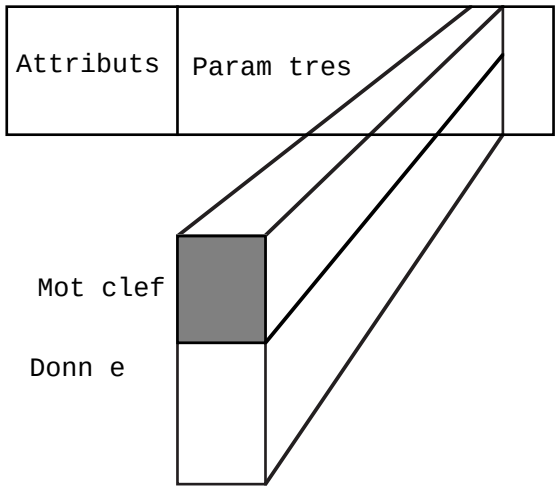


Figure A.2. : La structure interne d'un Apple Event.

Annexe 3 : Le mécanisme des vues

Le mécanisme des vues offert par la bibliothèque historique permet de filtrer les informations du fichier historique suivant l'une quelconque des trois caractéristiques : date, origine et type de l'information. Une vue est un quadruplet défini par un intervalle de temps (date de début, date de fin), par l'origine et le type de l'information cherchée. Lorsque une vue est définie par l'application analyste, la bibliothèque historique ne lui présente que les informations contenue par la vue, les autres informations étant masquées.

On distingue deux niveaux de vues : la vue locale et la vue globale. La vue globale associée au fichier historique est permanente jusqu'à ce que son annulation soit demandée. En complément, une vue locale temporaire peut être ajoutée à la vue globale. Ceci se fait à l'appel d'une fonction de positionnement ou de lecture et n'est valable que pour cet appel. La combinaison de la vue globale et de la vue locale permet de filtrer les données et agit comme un "ET" logique. La figure A.3 illustre ce mécanisme des deux vues superposables en utilisant la métaphore du film cinématographique.

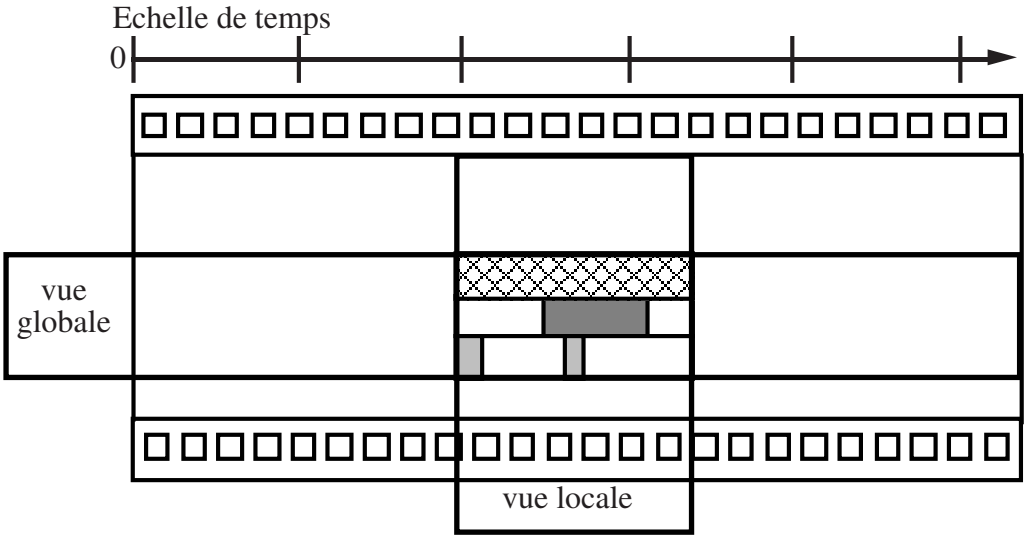


Figure A.3. : Le mécanisme des vues.

Par exemple, dans SupraAnalyse, nous avons déterminé plusieurs vues globales pour trier les informations selon leur origine (sujet et compères). Pour visualiser l'écran du sujet, nous superposons à la vue globale du sujet une vue locale qui définit une tranche de temps d'une seconde et qui se déplace sur toute la durée de l'expérimentation. La combinaison de ces deux vues donne la partie de l'historique qui est rejoué comme un film audiovisuel au rythme de la seconde.

Annexe 4 : L'interface homme-machine de SupraAnalyse

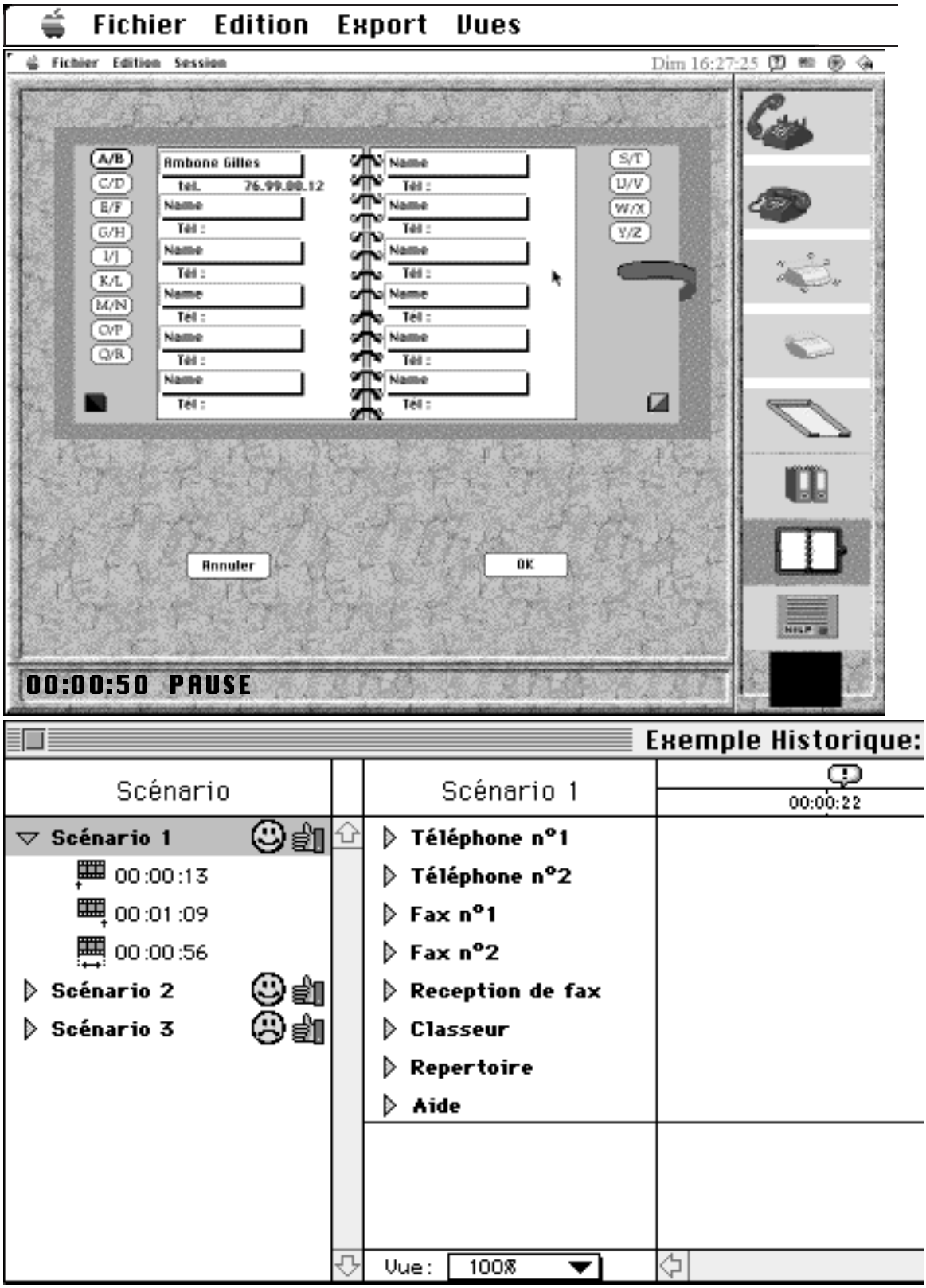
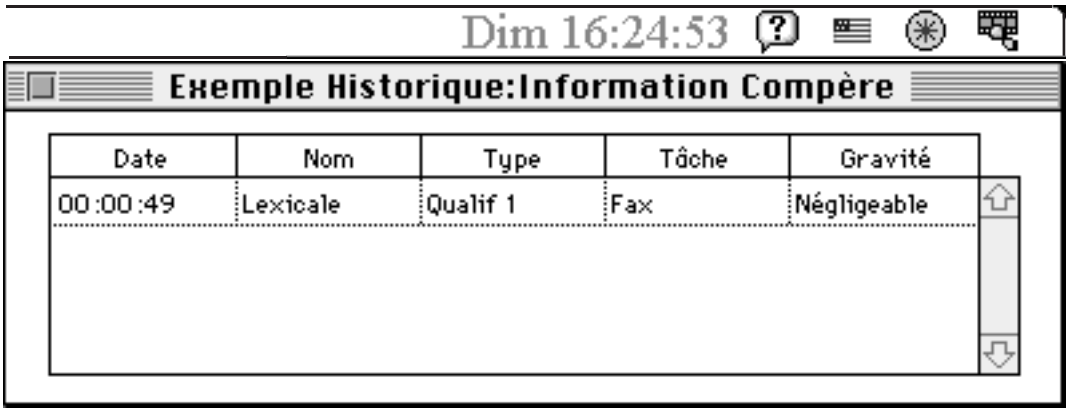
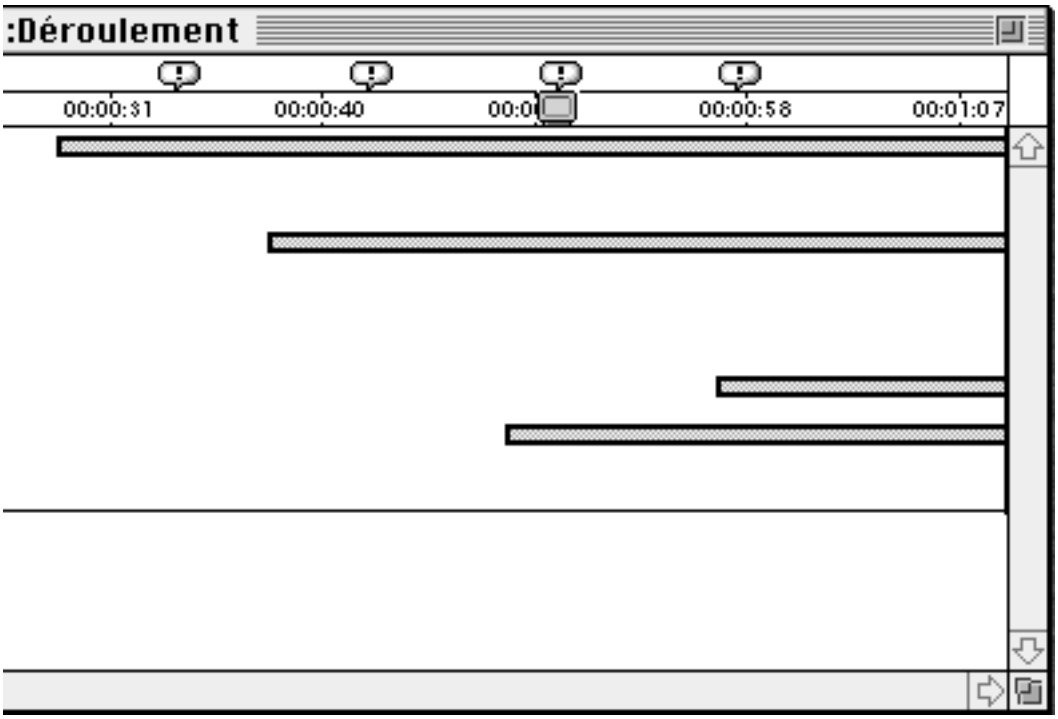


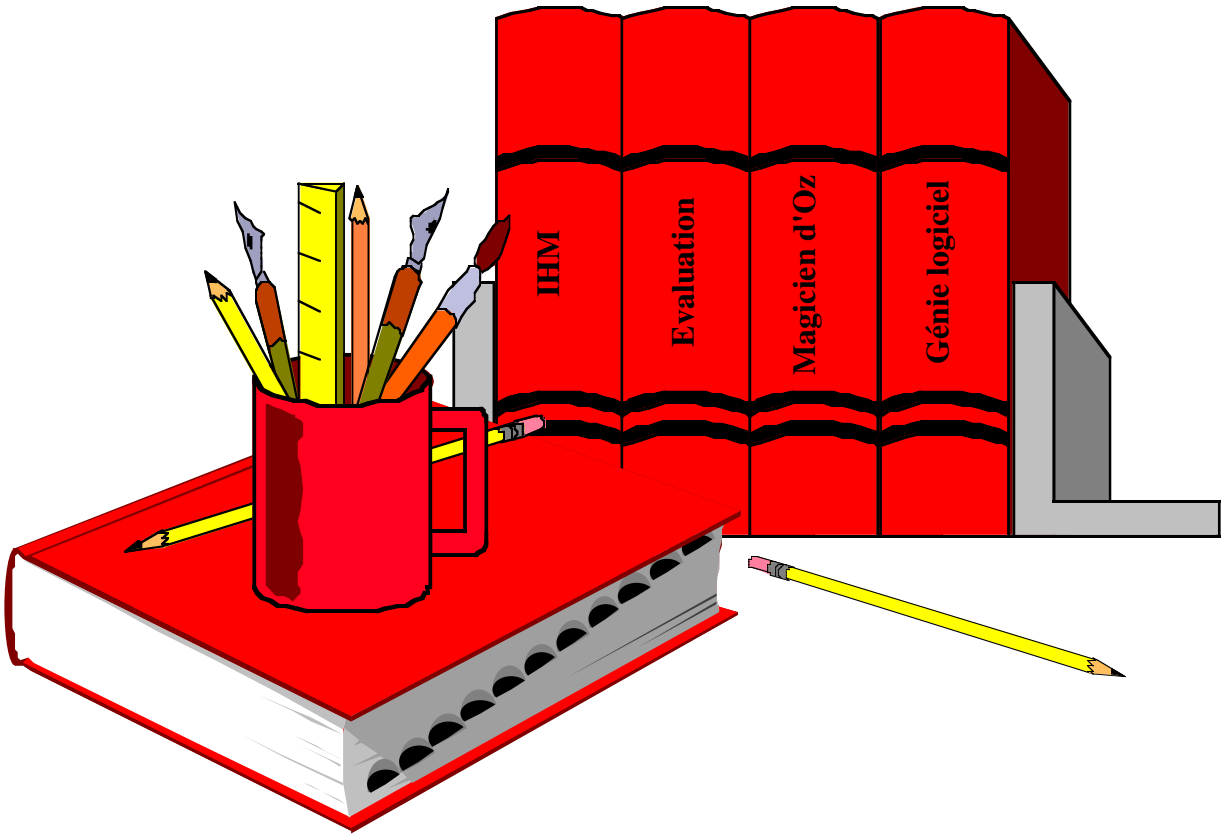
Figure A.4. : L'interface homme-machine de SupraAnalyse.


Dim 16:24:53
Exemple Historique:Information Compère
Date
Nom
Type
Tâche
Gravité
00 :00 :49
Lexicale
Qualif 1
Fax
Négligeable

Pilotage Magnétoscope
Time :
00:00:50


:Déroulement
00:00:31
00:00:40
00:00:58
00:01:07

Bibliographie



Références bibliographiques

Cette bibliographie respecte, dans la mesure du possible, la norme francaise AFNOR Z 44-005 datant de décembre 1987, qui elle-même reproduit intégralement la norme internationale ISO 690-1987.

[Amalberti 91] R. Amalberti, Modèles de l'activité, Introduction, Modèles en analyse du travail, M. de Montmollin, J. Theureau, Liège, Mardaga, 1991.

[Amalberti 93] R. Amalberti, C. Valot, Le Magicien d'Oz, CERMA, journée du PRC Rhône-Alpes, 1993.

[Ambone 92a] G. Ambone, D. Salber, Cahier des charges de NEIMO, Rapport technique interne, équipe IHM, LGI-IMAG, 1992.

[Ambone 92b] G. Ambone, B. Noz, D. Salber, Projet Neimo, Spécifications Externes, Rapport technique interne, équipe IHM, LGI-IMAG, 1992.

[Apple 90a] Apple Computer Inc., Macintosh Programmer's Workshop, Development Environment, vol. 1 et vol. 2, Developer Technical Publication, 1990.

[Apple 90b] Apple Computer Inc., Macintosh Programmer's Workshop, C++, Developer Technical Publication, 1990.

[Apple 91a] Apple Computer Inc., The Apple Event Manager, Developer Technical Publication, 1991.

[Apple 91b] Apple Computer Inc., ResEdit Reference, Apple Technical Library, Addison Wesley Publishing Company, 1991.

[Apple 92a] Apple Computer Inc., Inside Macintosh, Macintosh Toolbox Essentials, Apple Technical Library, Addison Wesley Publishing Company, October 1992.

[Apple 92b] Apple Computer Inc., Inside Mac, Files, Addison Wesley Publishing Company, 1992.

[Apple 92c] Apple Computer Inc., QuickTime 1.5 Developer kit, Developer Technical Publication, Apple Computer Inc., 1992.

[Apple 92d] Apple Computer Inc., Component Manager Developer kit, Developer Technical Publication, 1992

[Apple 92e] Apple Computer Inc., Inside Mac, Memory, Addison Wesley Publishing Company, 1992.

[Apple 92f] Apple Computer Inc., Inside Mac, Processes, Addison Wesley Publishing Company, 1992.

[Apple 92g] Apple Computer Inc., Programmer's Guide to MacApp, Addison Wesley Publishing Company, 1992.

[Apple 92h] Apple Computer Inc., Guide to MacApp Tools, Addison Wesley Publishing Company, 1992.

[Balbo 92a] S. Balbo, J. Coutaz, Une taxonomie pour les méthodes d'évaluation des interfaces homme-machine, IHM'92, Paris, décembre 1992.

[Balbo 92b] S. Balbo, J. Coutaz, Un pas vers l'évaluation automatique des interfaces homme-machine, ERGO-IA'92, Biarritz, 1992.

[Balbo 93a] S. Balbo, J. Coutaz, Automatic Evaluation in Human Computer Interaction, The Ergonomics Society 1993 Annual Conference, Edinburgh, 13-16 April, 1993.

[Balbo 93b] S. Balbo, J. Coutaz, D. Salber, Towards Automatic Evaluation of Multimodal User Interfaces, Workshop on Intelligent User Interfaces'93, 1993.

[Balbo 93c] S. Balbo, J. Coutaz, Modèles de tâches : Analyse comparative, utilité, limitations. IHM'93, Lyon, octobre 1993.

[Balbo 94] S. Balbo, Evaluation ergonomique des interfaces utilisateur : Un pas vers l'automatisation, Thèse de doctorat de l'Université Joseph Fourier, septembre 1994.

[Barnard 87] P.J. Barnard, Cognitive Resources and the Learning of Human-Computer Dialogs, Interfacing Thought, Cognitive Aspects of Human-Computer Interaction, J.M. Carroll Ed., MIT Press Publ., pp.112-158, 1987.

[Barthet 88] M.F. Barthet, Logiciels interactifs et ergonomie, modèles et méthodes de conception, Editions Dunod Informatique, 1988.

[Bass 90] L. Bass, J. Coutaz, Developing Software for the User Interface, SEI Series in Software engineering, Addison Wesley Publishing Company, 1990.

[Bastien 91] C. Bastien, Validation de critères ergonomiques pour l'évaluation d'interfaces utilisateurs, INRIA, Unité de recherche INRIA Rocquencourt, rapport de recherche no 1427, Programme 3, mai 1991.

[Bastien 93] J.M. Bastien, D.L. Scapin, Preliminary findings on the effectiveness of ergonomic criteria for the evaluation of human-computer interfaces, Proceedings of the InterCHI'93 conference, Amsterdam, April 1993.

[Blattner 90] M.M. Blattner, R.B. Dannenberg, CHI'90 Workshop on multimedia, multimodal interface design, SIGCHI Bulletin, Volume 22, Number 2, pp. 54-58, octobre 1990.

[Boehm 81] B.W. Boehm, Software Engineering Economics, Prentice Hall, 1981.

[Boehm 88] B.W. Boehm, A spiral Model of Software Development and Enhancement, IEEE Computer, May 1988.

[Bourguet 92a] M.L. Bourguet, Conception et réalisation d'une interface de dialogue personne-machine multimodale, thèse de l'Institut National Polytechnique de Grenoble, 1992.

[Bourguet 92b] M. L. Bourguet, ICP plan : dialogue multimodal pour la conception de plans architecturaux, 19ème J.E.P., Bruxelles pp 369-374, 1992.

[Carlsen 91] N. V. Carlsen, Modelling User Interface Software, thèse de l'Université Technique du Danemark, département de la communication graphique, vol. 1, 1991.

[Carli 94a] M. Carli, J.P. Charon, S. Liska, L. Tézier, Dossier de Conception de SUPRATEL ★, Rapport technique interne, équipe IHM, LGI-IMAG, 1994.

[Card 83] S. Card, T. Moran, A. Newell, The psychology of Human-Computer Interaction, Hillsdale, New Jersey, Lawrence Erlbaum Associates Publ., 1983.

[Colbert 92] M. Colbert, J. Dowell, J. Long, The application of Dowell, Long's engineering conception of Human-Computer Interaction to Military Command, Control : a view of the domain of C2, its fitness for purpose, Ergonomics Unit, University College London, UK, 13th March, 1992.

[Coutaz 87] J. Coutaz, PAC: an Implementation Model for Dialog Design, Proceedings of Interact'87, H-J. Bullinger, B. Shackel ed., North Holland, Stuttgart, pp. 431-436, September 1987.

[Coutaz 90] J. Coutaz, Interfaces homme-ordinateur : Conception et réalisation, Dunod Informatique, 1990.

[Coutaz 91a] J. Coutaz, A. Gourdol, Communication homme-machine multimodale : perspectives pour la recherche, deuxièmes journées nationales du GRECO-PRC, communication homme-machine, Actes, Toulouse, janvier 1991.

[Coutaz 91b] J. Coutaz, L. Nigay, Seeheim et architecture Multi-agent, Colloque Ingénierie des Interfaces, IHM'91, Dourdan, octobre 1991.

[Coutaz 92b] J. Coutaz, Interfaces Homme-Machine : un regard critique, Journées d'étude Interface homme-machine, Architectures et outils, AFCET Génie logiciel, Paris, octobre 1992.

[Coutaz 93a] J. Coutaz, G. Faconti, F. Paterno, L. Nigay, D. Salber, MATIS : a UAN Description and Lesson Learned, Amodeus Report, SM/WP14, 1993.

[Coutaz 93b] J. Coutaz, L. Nigay, D. Salber, The MSM framework : A Design Space for Multi-Sensori-Motor Systems, Human Computer Interaction, 3rd International Conference EWHCI'93, East/West Human Computer Interaction, Moscow. L. Bass, J. Gornostaaev, C. Unger Eds. Springer Verlag Publ., Lecture notes in Computer Science, Vol. 753, pp. 231-241, 1993.

[Coutaz 93c] 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 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 94] J. Coutaz : Evaluation Techniques : exploring the Intersection between HCI, Software Engineering, Workshop on Software Engineering, Human Computer-Interaction, International Conference on Software Engineering, ICSE'94, Sorrento, May, 1994.

[Dahlbäck 89] N. Dahlbäck, A. Jönsson, Empirical studies of discourse representations for natural language interfaces, Fourth Conference of the European Chapter of the ACL Proceedings, pp. 291-298, 1989.

[de Montmollin 86] M. de Montmollin, L'intelligence de la tâche, Eléments d'ergonomie cognitive, Berne, Peter Lang, 1986.

[Dix 91] A.J. Dix, Formal Methods for Interactive Systems, Computers, Poeple Series, Academic Press, 1991.

[Dowell 89] J. Dowell, J. Long, Toward a conception for an engineering discipline of human factors, Ergonomics, vol. 32, no 11, pp. 1513-1535, November 1989.

[Foley 88] 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., pp. 67-72, 1988.

[Fraser 92] N. Fraser, N. Gilbert, C. McDermid, The Value of Simulation Data, Third Conference on Applied Natural Language Processing, Trento, Italy, 31 March-3 April 1992.

[Gandriau 93] N. Gandriau, J.P. Charon, Cahier des Charges de AppliLevelt, rapport technique interne, équipe IHM, LGI-IMAG, 1993.

[Gourdol 92] A. Gourdol, L. Nigay, D. Salber, Multimodal Systems : Aspects of Event Fusion, a Taxonomy, IFIP Conference'92 Proceedings, 1992.

[Gray 94] P. Gray, D. England, S. McGowan, XUAN : Enhancing the UAN to capture temporal relations among actions, Computing Science Research Report, University of Glasgow, February 1994.

[Guérin 91] F. Guérin, A. Laville, F. Daniellou, J. Duraffourg, A. Kerguelen, Comprendre le travail pour le transformer, la pratique de l'ergonomie, Montrouge, ANACT, 1991.

[Hammontree 92] M.L. Hammontree, J.J. Hendrickson, B.W. Hensley, Integrated data capture, analysis tools for research, testing on graphical user interfaces, Proceedings of the CHI'92 Conference, ACM Press, Monterey, pp. 431-432, 3-7 May 1992.

[Hartson 92] H.R. Hartson, P.D. Gray, Temporal Aspects of Tasks in the User Action Notation, Human Computer Interaction, 7(1), pp. 1-46, 1992.

[Harvey 88] G. Harvey, Understanding HyperCard, for version 1.1, Sybex Inc., 1988

[Hauptmann 89] A.G. Hauptmann, Speech and Gesture for Graphic Image Manipulation, CHI'89 Proceedings, pp. 241-245, 1989.

[Hayes-Roth 79] B. Hayes-Roth, F. Hayes-Roth, A Cognitive Model of Planning, Cognitive Science 3, pp. 275-310, 1979.

[Hix 93] D. Hix, R. Hartson, Developing User Interfaces, Ensuring Usability through Product, Process, Wiley, 1993.

[IEEE 84] IEEE, Guide to Software Requirement Specification, IEEE, Std 830, 1984.

[Jeffries 91] R. Jeffries, J.R. Miller, C. Wharton, K.M. Uyeda, User interface evaluation in the real word : a comparison of four techniques, Proceedings of the CHI'91 Conference on Computer Human Interaction, New Orleans, ACM New York, pp. 119-124, May 1991.

[Kieras 85] D. Kieras, P.G. Polson, An Approach to the Formal Analysis of User Complexity, International Journal of Man-Machine Studies, 22, pp. 365-394, 1985.

[Kolski 89] C. Kolski, Contribution à l'ergonomie de conception des interfaces graphiques homme-machine dans les procédés industriels : application au système expert SYNOP, Thèse présentée à l'Université de Valenciennes et du Hainaut-Cambrésis, janvier 1989.

[Lerch 89] F.J. Lerch, M.M. Mantei, J.R. Olson, Skilled financial planning : The Cost of Translating Ideas into Actions, Proceedings of CHI'89, ACM Press, pp. 121-126, 1989.

[Lewis 90] C. Lewis, P. Polson, C. Wharton, J. Rieman, Testing a Walkthrough Methodology for Theory-Based Design of Walk-Up-and-Use Interfaces, Proceedings of the CHI'90 Conference on Computer Human Interaction, Seattle. ACM New York, pp. 235-242, April 1990.

[Lim 92] K.Y. Lim, J.B. Long, N. Silcock, Integrating human factors with system development methods : an overview of a structured human factors method, Proceedings of the IDASCO'92 Conference, Oxford(UK), IEE, D.K. Hitchins (Eds), April 6-9 1992.

[Lischetti 94a] N. Lischetti, M. Carli, J.P. Charon, S. Liska, L. Tézier, Cahier des charges de SUPRATEL★, Rapport technique interne, équipe IHM, LGI-IMAG, 1994.

[Lischetti 94b] N. Lischetti, M. Carli, J.P. Charon, S. Liska, L. Tézier, Dossier de Spécifications externes de SUPRATEL★, Rapport technique interne, équipe IHM, LGI-IMAG, 1994.

[Lischetti 94c] N. Lischetti, Dossier de Conception de SupraAnalyse, Rapport technique interne, équipe IHM, LGI-IMAG, 1994.

[Lischetti 94d] N. Lischetti, Manuel Utilisateur de SupraAnalyse, Rapport technique interne, équipe IHM, LGI-IMAG, 1994.

[Long 89] J. Long, J. Dowell, Conceptions of the Discipline of HCI : Craft, Applied Science, Engineering, Proceedings of the Fifth Conference of the BCS HCI SIG, A. Sutcliffe, L. Macaulay (Eds), Cambridge University Press, 1989.

[Löwgren 90] J. Löwgren, T. Nordqvist, A Knowledge-Based Tool for User Interface Evaluation, its integration in a UIMS, Human-Computer Interaction-INTERACT'90, pp. 395-400, 1990.

[MacroMind 91] MacroMind Director, Version 3.0, Seconde édition, MacroMind Inc., 1991.

[Macleod 93] M. Macleod, DRUM, Diagnostic Recorder for Usability Measurement, NPL, DITC HCI group, Teddington, Middlesex, TW11 OLW, UK, 1993.

[Mackay 88] W.E. Mackay, Video : Data for Studying Human-Computer Interaction, CHI '88, pp. 133-137, 1988.

[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.

[Meinadier 91] J.P. Meinadier, L'interface utilisateur : pour une informatique plus conviviale, Dunod, Paris, 1991.

[Meyer 88] B. Meyer, Object-oriented Software Construction, Prentice Hall (O-O design, the Eiffel™ language), 1988.

[Monk 87] A.F. Monk, A. Dix, Refining early design decisions with a black box model, People Computer III, D. Diaper et R. Winder (Eds), Cambridge University Press, pp. 147-158, 1987.

[Moran 81] T. Moran, The Command Langage Grammar : a representation for the user interface of interactive computer systems, International Journal of Man-Machine Studies, 15, pp. 3-50, 1981.

[Mulligan 92] R.M. Mulligan, M. Dieli, J. Nielsen, S. Poltrock, D. Rosenberg, S.E. Rudman : Designing usable systems under real world constraints : A practioners forum, Panel, CHI'92, pp. 149-152, 1992.

[Myers 88] B.A. Myers, Tools for creating user interfaces : an introduction, survey, CMU-CS-88-107, pp. 16,17, January 1988.

[Newell 86] A. Newell, S. Card, Straightening out softening up : Response to Carroll, Campell, Human-Computer interaction, 2(3), pp. 251-267, 1986.

[Niel 83] Niel, Simon, Playback : a method for evaluating the usability of software, its documentation, Proceedings of the CHI'83 Conference, 1983.

[Nielsen 89] J. Nielsen, Usability Engineering at a Discount, in Designing, Using Human Computer Interfaces, Knowledge-Based Systems, Salvendy, Smith Eds., Elsevier North Holland, pp. 394-401, 1989.

[Nigay 92] L.Nigay, J. Coutaz - Interfaces multimodales et architecture : fusion et parallélisme, IHM'92, Paris, décembre 1992.

[Nigay 93] L. Nigay, J. Coutaz, A design space for multimodal interfaces : concurrent processing and data fusion, INTERCHI'93 Proceedings, Amsterdam, pp. 172-178, May 1993.

[Nigay 94] L. Nigay, Conception et modélisation logicielles des systèmes interactifs : application aux interfaces multimodales, thèse de l'Université Joseph Fourier, Grenoble, janvier 1994

[Normand 92] V. Normand, Le modèle SIROCO : de la spécification conceptuelle des interfaces utilisateur à leur réalisation, thèse de l'Université Joseph Fourier, Grenoble, 1992.

[Norman 86] D.A. Norman, S.W. Draper, Cognitive Engineering, User Centered System Design, Lawrence Erlbaum Associates, pp. 31-61, 1986.

[Pinsky 89] L. Pinsky, De l'interface à la situation de travail, L'ergonomie de conception, V. de Keyser, A. van Daele, Bruxelles, De boeckWesmael, 1989.

[Pfaff 85] G.E. Pfaff, User Interface Management System, Eurographics Seminars, Springel-Verlag, 1985.

[Polity 90] Y. Polity, J.M. Francony, R. Palermiti, P. Falzon, S. Kazma, Recueil de dialogues homme-machine en langue naturelle écrite, Les Cahiers du Criss, no 17, décembre 1990.

[Polity 91] Y. Polity, J.M. Francony : Sophocle : un atelier pour l'analyse de la production langagière dans les dialogues homme-machine, Criss-notes et rapports de recherche no 5, avril 1991.

[Pollier 91] A. Pollier, Evaluation d'une interface par des ergonomes : diagnostics et stratégies, Rapport de recherche INRIA no 1391, février 1991

[Richards 84] M. Richards, K. Underwood, How Should People, Computers Speak to Each Other, Proceedings of Interact '84, First IFIP Conference on Human-Computer Interaction, pp. 268-273, 1984.

[Robertson 91] G.G. Robertson, J.D. Mackinlay, S.K. Card, The perspective wall : detail and context smoothly integrated, Proceedings of CHI'91, pp. 173-177, 1991.

[Royce 70] W.W. Royce, Managing the development of large software systems, Proc. WESTCON, California, U.S.A., 1970.

[Rumbaugh 91] B. Rumbaugh, E. Premerlani, Lorensen, Object-Oriented Modeling and Design, Prentice-Hall, 1991.

[Scapin 86] D.L. Scapin, Guide ergonomique de conception des interfaces homme-machine, Rapport technique INRIA no 77, octobre 1986

[Scapin 89] C. Pierret-Goldreich, I. Delouis, D.L. Scapin, Un Outil d'Acquisition et de Représentation des Tâches Orientées Objet, Rapport de recherche INRIA no 1063, Programme 8 : Communication homme-machine, août 1989.

[Scapin 90] D.L. Scapin, Des critères ergonomiques pour l'évaluation et la conception d'interfaces utilisateurs, Acte du XXVI Congrès de la SELF, Montréal, Canada, 3-5 octobre, 1990.

[Schmucker 86a] K. Schmucker, MacApp : an application framework, Byte volume 11, pp. 189-193, (8), 1986.

[Schmucker 86b] K. Schmucker, Object-Oriented Languages for the Macintosh, Byte volume 11, pp. 177-188, (8), 1986.

[Séguret-Geye 89] M. Séguret-Geye, C. Argoud, TILT ou les codages universels de l'information sur écran, Les éditions d'organisation, Paris, 1989.

[Senach 90a] B. Senach, Evaluation de l'ergonomie des interfaces homme-machine : du prototype aux systèmes experts, Congrès ERGO-IA'90, Biarritz, 19-21 septembre 1990.

[Senach 90b] B. Senach, Evaluation ergonomique des interfaces homme-machine : une revue de la littérature, Rapport de recherche INRIA no 1180, Programme 8, Communication Homme-Machine, mars 1990.

[Serpent 89] SEI Serpent Application Develloper's Guide, User's Guide CMU/SEI-89-UG-6, ESD-TR-89-12, January 1989.

[Siochi 91] A. Siochi, D. Hix, A study of Computer-Supported User interface Evaluation Using Maximal Repeating Patern Analysis, Proceedings of the CHI'91 Conference, ACM Press, pp. 301-305, 1991.

[Shneiderman 87] B. Shneiderman, Designing the user interface : strategies for effective Human-Computer Interaction, Addison Wesley Publishing Compagny, 1987.

[Smith 86] S.L. Smith, J.N. Mosier, Guidelines for designing user interface software, Report MTR-10090 ESD-TR-86-278, The MITRE Corporation, Bedford, MA, August 1986.

[Sperandio 84] J.C. Sperandio, L'ergonomie du travail mental, Paris, Masson, 1984.

[Sutcliffe 94] A. Sutcliffe, P. Faraday, Designing Presentation in Multimedia Interfaces, CHI'94 proceedings, Boston, pp. 92-98, 1994.

[Symantec 94a] Symantec Corporation, Visual Architect and THINK Class Library Guide, C++ Development, Environment for Macintosh, Version 7, California, 1994.

[Symantec 94b] Symantec Corporation, User's Guide, C++ Development, Environment for Macintosh, Version 7, California, 1994.

[Symantec 94c] Symantec Corporation, Compiler Guide, C++ Development, Environment for Macintosh, Version 7, California, 1994.

[Szekely 93] P. Szekely, P. Luo, R. Moriyon, R. Neches, P. Aberg, D. Benjamin, B. Harp, L. Hu, The integrated user-support environnement, Proceedings of CHI'93,New York, U.S.A., pp. 107-114, 1993.

[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, West Germany, November 1983.

[Tardieu 83] H. Tardieu, A Rochfeld, R. Coletti, La méthode Merise, Edition des Organisations, Paris, 1983

[Theaker 89] C.J. Theaker, R. Phillips, T.M.E. Frost, W.R. Love, HIMS : a tool for HCI evaluations, Proceedings of the Fifth Conference of the BCS HCI SIG, A. Sutcliffe, L. Macaulay (Eds), Cambridge University Press, 1989.

[Valentin 87] A. Valentin, R. Lucongsang, L'ergonomie des logiciels, Montrouge, ANACT, 1987.

[Valentin 93] A. Valentin, G. Vallery, R. Lucongsang, L'évaluation ergonomique des logiciels, une démarche itérative de conception, ANACT, Collection Outils et Méthode, 1993.

[Vanderdonckt 90] J. Vanderdonckt, Modèle de la présentation d'une application : les règles ergonomiques, rapport IHM-Ergo-3, Facultés Universitaires Notre-Dame de la Paix, Institut d'Informatique, 21 rue Grandgagnage, B-5000 Namur, 27 août 1990.

[Webster 89] B.F. Webster, The NeXT Book, Addison-Wesley, Reading, Mass., 1989.

[Wilson 90] D.A. Wilson, L.S. Rosenstein, D. Shafer, C++ Programming with MacApp, Addison Wesley Publishing Company, 1990.

[Whitefield 91] A. Whitefield, F. Wilson, J. Dowell, A framework for human factors evaluation, Behaviour, Information technology, vol. 10, no 1, pp. 65-79, 1991.

[Whittaker 89] S. Whittaker, P. Stenton, User studies and the design of Natural Language Systems, Fourth Conference of the European Chapter of the ACL, Proceedings, pp. 291-298, 1989.

[Whiteside 88] J. Whiteside, J. Bennett, K. Holtzblatt, Usability engineering : our experience, evolution, Handbook of Human-Computer Interaction, M. Helander (Ed.), Elsevier Science Publishers, 1988.

[Young 90] R.M. Young, J. Whittington, A knowledge Analysis of interactivity, Proceedings of INTERACT'90, edited by D. Diaper, G. Cockton, B. Shackel, Elsevier Scientific Publishers B.V., Cambridge, United Kingdom, pp. 207-212, 27-31st August 1990.

Références par thèmes

Ergonomie :

[Amalberti 91], [Balbo 92a], [Balbo 92b], [Balbo 93a], [Balbo 93b], [Balbo 93c], [Balbo 94], [Barthet 88], [Bastien 91], Bastien 93], [Card 83], [Dahlbäck 89], [de Montmollin 86], [Dix 91], [Dowell 89], [Guérin 91], [Kolski 89], [Lerch 89], [Lim 92], [Macleod 93], [McCal 77], [Monk 87], [Niel 83], [Nielsen 89], [Pinsky 89], [Polity 91], [Pollier 91], [Scapin 86], [Scapin 89], [Scapin 90], [Séguret-Geye 89], [Senach 90a], [Senach 90b], [Sperandio 84], [Theaker 89], [Valentin 87], [Valentin 93], [Vanderdonckt 90], [Whitefield 91], [Whittaker 89], [Whiteside 88].

Génie Logiciel :

[Boehm 81], [Boehm 88], [Gray 94], [Hartson 92], [IEEE 84], [McDermid 84], [Robertson 91], [Rumbaugh 91], [Tardieu 83].

Interfaces Homme-Machine et multimodalité :

[Barnard 87], [Bass 90], [Blattner 90], [Bourguet 92a], [Bourguet 92b], [Carlsen 91], [Colbert 92], [Coutaz 87], [Coutaz 90], [Coutaz 91a], [Coutaz 91b], [Coutaz 92b], [Coutaz 93a], [Coutaz 93b], [Coutaz 93c], [Coutaz 93d], [Coutaz 94], [Gourdol 92], [Hammontree 92], [Hauptmann 89], [Hayes-Roth 79], [Hix 93], [Jeffries 91], [Kieras 85], [Lewis 90], [Long 89], [Mackay 88], [Meinadier 91], [Moran 81], [Mulligan 92], [Myers 88], [Newell 86], [Nigay 92], [Nigay 93], [Nigay 94], [Norman 86], [Pfaff 85], [Polity 90], [Polity 91], [Richards 84], [Royce 70], [Siochi 91], [Shneiderman 87], [Smith 86], [Sutcliffe 94], [Tanner 83], [Young 90].

Magicien d'Oz et NEIMO :

[Amalberti 93], [Ambone 92a], [Ambone 92b], [Carli 94a], [Fraser 92], [Gandriau 93], [Lischetti 94a], [Lischetti 94b], [Lischetti 94c], [Lischetti 94d].

Outils et techniques informatiques :

[Apple 90a], [Apple 90b], [Apple 91a], [Apple 91b], [Apple 92a], [Apple 92b], [Apple 92c], [Apple 92d], [Apple 92e], [Apple 92f], [Apple 92g], [Apple 92h], [Foley 88], [Harvey 88], [Löwgren 90], [MacroMind 91], [Meyer 88], [Normand 92], [Robertson 91], [Schmucker 86a], [Schmucker 86b], [Serpent 89], [Symantec 94a], [Symantec 94b], [Symantec 94c], [Szekely 93], [Webster 89], [Wilson 90].

Nom : Nathalie LISCHETTI

Titre du mémoire :

Conception et réalisation d'un outil d'aide à l'analyse de données comportementales.

Date de soutenance :

30 mars 1995.

Résumé :

Ce mémoire présente le travail de conception et de réalisation effectué en vue de l'obtention du diplôme d'ingénieur C.N.A.M. en informatique.

L'ingénierie des Interfaces Homme-Machine (IHM) s'intéresse en particulier aux mécanismes de mise en oeuvre de l'interaction entre l'utilisateur et le système informatique. Elle s'est consacrée jusqu'ici au développement de modèles et d'outils de construction pour satisfaire les exigences grandissantes des utilisateurs et suivre la technologie très innovante et évolutive. Malgré cela, la réalisation des IHM reste techniquement difficile et leur évaluation est souvent négligée par les développeurs. Aujourd'hui, le domaine de l'IHM est pluridisciplinaire (informatique, ergonomie, psychologie, etc.), et l'absence de synergies dans les méthodes et le retard sur les concepts et les techniques poussent à avoir recours à l'évaluation ergonomique expérimentale. Dans ce cadre, notre but est de faire un premier pas vers l'analyse automatique des données recueillies lors d'une expérimentation.

Une revue de l'état de l'art nous permet de présenter la conception des IHM et de classer les méthodes d'évaluation actuelles. Nous présentons l'exemple de la plate-forme d'évaluation expérimentale NEIMO basée sur la technique du Magicien d'Oz. Cette dernière consiste à faire exécuter par un compère humain dissimulé les fonctions manquantes d'un système interactif utilisé par un opérateur représentatif accomplissant une tâche déterminée.

Nous proposons SupraAnalyse, un outil d'aide à l'analyse de données comportementales recueillies avec la plate-forme d'accueil NEIMO. Son application sur le projet SUPRATEL★ a permis d'exploiter des données enregistrées lors de l'évaluation expérimentale d'une interface destinée à un terminal de communication multiservice. La démarche de développement que nous avons suivie sert d'introduction à la présentation des formalismes et des méthodes appliqués à l'IHM. Nous en faisons une analyse critique basée sur notre propre expérience et nous montrons leur usage dans la réalisation.

En conclusion, nous dressons un bilan du travail réalisé et nous envisageons les extensions possibles pour SupraAnalyse.

Mots-clés :

Interaction Homme-Machine, Ergonomie des logiciels, Evaluation de l'utilisabilité d'interfaces utilisateur, Magicien d'Oz.

Keywords:

Human-Computer Interaction, Software ergonomics, Usability evaluation of user interfaces, Wizard of Oz.