THÈSE

présentée par

Daniel SALBER

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

♦♦♦♦♦♦♦

De l'interaction homme-machine individuelle aux systèmes multi-utilisateurs

L'exemple de la communication homme-homme médiatisée

♦♦♦♦♦♦♦

Date de soutenance : 8 septembre 1995

Composition du jury

Président :	**M. Jean CAELEN**
Rapporteurs :	**M. Michel BEAUDOUIN-LAFON** **Mme Marilyn MANTEI**
Examinateurs :	**M. Yves CHIARAMELLA** **Mme Joëlle COUTAZ** **M. Michel RIVEILL**

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

Plan de la thèse

Remerciements

Un travail de recherche n'est jamais l'œuvre d'un seul individu. De nombreuses personnes m'ont permis de mener à bien le travail présenté ici et je tiens à les en remercier.

En premier lieu Joëlle Coutaz m'a accueilli dans son équipe et m'a accordé sa confiance malgré ma situation alors peu ordinaire. Elle m'a fait découvrir un domaine passionnant, m'a fait profiter de sa connaissance approfondie de sa discipline, m'a laissé une liberté exceptionnelle pour orienter et mener à bien mes travaux, et a régulièrement soutenu et encouragé mes efforts. Pour tout ceci allié à ses qualités humaines et scientifiques, je la remercie du fond du cœur. Je lui dois beaucoup.

Les membres du jury m'ont fait l'honneur de bien vouloir juger mon travail et l'ont apprécié. Je tiens à les remercier de leurs encouragements et en particulier les rapporteurs de cette thèse, Marilyn Mantei et Michel Beaudouin-Lafon.

Laurence Nigay a partagé son bureau avec moi ainsi que les moments heureux et difficiles de ces trois et quelque années de thèse. Malgré les hautes doses de Springsteen, qu'elle en soit remerciée. :-)

Je n'aurais peut-être jamais découvert que l'interface homme-machine est une discipline de recherche ni l'existence de l'équipe de Joëlle Coutaz sans Annie Chabert. Mille mercis à elle !

André Coulon m'a régulièrement confié des travaux qui m'ont permis de vivre pendant les premières années de ma thèse. Je le remercie aussi et surtout pour des discussions nombreuses et stimulantes et souhaite que nos aventures se poursuivent dans des conditions plus propices. A lui aussi, je dois beaucoup.

Philip Barnard et Jon May m'ont accueilli dans leur équipe à MRC-APU à Cambridge pour un trop court séjour pendant l'été 1993. Je les remercie de m'avoir permis de découvrir un nouvel environnement et une discipline, la psychologie cognitive, dans laquelle j'étais béotien. Je tiens aussi à remercier David, Sandra, Rob et bien d'autres ainsi que le "pub across the street" qui ont rendu ce séjour aussi agréable qu'instructif.

Le projet européen Amodeus a été une occasion unique de découvrir la diversité de la discipline de l'interface homme-machine. Je remercie ici tous les membres de ce projet pour des discussions fructueuses et des moments amicaux aux quatre coins de l'Europe. Ma gratitude va aussi aux membres du groupe de travail IFIP WG 2.7 qui m'ont invité à quelques-unes de leurs réunions. Je n'oublie pas non plus le climat intense et extrêmement enrichissant des quinze jours de l'Ecole d'Eté INRIA-CEA-EDF 1994 sur les interfaces homme-machines où j'étais assistant. Mes remerciements vont à ses participants et animateurs, tout particulièrement à Gérard Brisson et à Len Bass.

De nombreux étudiants ou membres de l'équipe ont contribué à la réalisation du projet Neimo qui n'aurait pu aboutir sans leur concours. Mes remerciements vont à Gilles, Eric, Sandrine, Nathalie, Rachid, Laurent, Marie-Laure, David, Stéphane, Jean-Pascal, une autre Nathalie, un autre Stéphane, encore un autre Stéphane, un autre Laurent, Martine, Bérangère, Bernie, et sans doute bien d'autres à venir…

Un chercheur n'est rien sans une équipe et je voudrais remercier tout particulièrement tous les membres de l'équipe interface homme-machine qui contribuent ou ont contribué à son atmosphère de confiance et à son dynamisme et sans qui je n'aurais sûrement pas eu autant de plaisir à travailler : François, Sébastien, Francis, Eric, Laurent, Sandrine (merci pour la photo !), Gilles, Arno et d'autres… Les membres d'autres équipes du laboratoire y contribuent parfois tout autant. Merci à eux et special thanks à Philippe "Crazy Gratteux" Mulhem !

Finalement, des remerciements tout particuliers vont à mes parents, ma famille et mes amis qui m'ont soutenu, encouragé ou supporté pendant ces années de thèse.

Introduction

It would appear that we have reached the limits of what it is possible to achieve with computer technology, although one should be careful with such statements, as they tend to sound pretty silly in 5 years.

John von Neumann (vers 1949)

Des interfaces homme-machine aux interfaces hommes-machines

La discipline de l'interaction homme-machine s'est attaquée à un défi formidable : harmoniser la technologie informatique aux nouveaux besoins humains, sociaux et économiques. Si l'on se penche sur ce qui sépare les êtres humains des machines informatiques, le fossé semble incommensurable. D'un côté l'homme, dont l'expérience du monde est analogique[1]. De l'autre côté, l'ordinateur qui ne sait traiter que de l'information numérisée. D'un côté l'homme, qui, si l'on prend le parti de Descartes plutôt que de Spinoza, dispose de son libre-arbitre, de l'autre, l'ordinateur qui est, en général, fondamentalement déterministe. L'interaction homme-machine a à son actif plusieurs succès dans le rapprochement de ces deux mondes si opposés mais ses créations se sont penchées, pour l'essentiel, sur l'activité mono-utilisateur.

En vérité, l'activité humaine n'est qu'exceptionnellement le fait d'un être humain isolé. Comme Rousseau le constatait, l'homme est un animal social et son activité ne peut être considérée isolément. Parallèlement, la multiplication du nombre d'ordinateurs et la possibilité de les faire communiquer conduit à considérer l'outil informatique comme une multitude de machines plutôt qu'une machine prise isolément. Ces deux constatations ouvrent la voie à un nouveau champ d'investigation : l'interaction entres les hommes et les machines plutôt que la simple interaction homme-machine.

Ces deux aspects indissociables du domaine de recherche, l'humain et le technique, nous ont guidés dans le travail que nous présentons ici. D'un côté une technologie qui progresse de façon considérable, transformant progressivement l'ordinateur d'outil de calcul en un outil de communication, de l'autre des enjeux humains et de société qu'il est indispensable de considérer. Certains futurologues, tel Alvin Toffler, n'hésitent pas à parler de la "troisième vague" de la "révolution de l'information", faisant suite aux révolutions agricole et industrielle.

Depuis plusieurs mois, les signes avant-coureurs de la troisième vague annoncée par Toffler semblent se multiplier. Les technologies de l'information sont au centre d'enjeux sociaux, politiques et économiques. Le succès et la vulgarisation croissantes du réseau Internet, les débats sur les "autoroutes de l'information", le développement du "multimédia" en sont les manifestations les plus évidentes. Pour un nombre croissant

[1] Même si les physiciens comme Niels Bohr nous ont révélé la nature quantique de la matière, l'homme n'a pas la faculté de la percevoir.

d'utilisateurs, le concept World-Wide Web permet de toucher du clic de souris une préfiguration du "village planétaire" cher à McLuhan.

Mais au-delà de ce qui peut apparaître comme un simple phénomène de mode, il semble vain de vouloir prédire l'impact de ces nouveaux outils sur les individus et sur les sociétés dans lesquelles ils vivent. Comme pour toute avancée technologique majeure, nous manquons de références pour évaluer les conséquences. Louis Lumière, co-inventeur du cinéma, n'affirmait-il pas que "le cinématographe est une invention sans avenir" ? John von Neumann, dès 1949, nous mettait en garde contre toute velléité de prévision du futur dans le domaine informatique.

Face à ces incertitudes, l'objectif du travail présenté dans ce mémoire est double : *construire* en nous appuyant sur les travaux existants, et *modéliser* afin de guider et comprendre les évolutions futures de notre domaine. Pour construire, nous étudions comment mettre à profit les travaux et résultats issus de l'étude des systèmes mono-utilisateurs. Comment étendre l'acquis des interfaces homme-machine aux interfaces multi-utilisateurs ? Notre deuxième objectif, modéliser, répond à une exigence de rigueur scientifique. La modélisation, sous forme de classification ou d'espace problème par exemple, permet d'explorer systématiquement les composantes d'une question. Les systèmes multi-utilisateurs couvrent un domaine applicatif aux frontières mal cernées. Dans cet espace, nous avons retenu la communication homme-homme médiatisée qui permet à un ensemble d'individus de communiquer via un dispositif informatique.

Plan du mémoire

Ce mémoire reflète l'état de notre réflexion sur les systèmes multi-utilisateurs et la communication homme-homme médiatisée en particulier. Il comprend trois parties : la première est centrée sur l'apport des sciences sociales et humaines. La seconde étudie l'intégration de ces apports dans un processus de développement informatique. La troisième concerne la technologie informatique.

- Ces trois volets de notre réflexion sont introduits dans le chapitre 1 qui cerne le cadre terminologique et conceptuel de notre domaine d'étude. Nous y définissons les systèmes multi-utilisateurs et certaines de leurs qualités. Nous introduisons aussi une grille d'analyse à trois niveaux : principes, propriétés et techniques. Cette décomposition nous permet de mieux situer les apports des différentes disciplines que nous détaillons dans les chapitres suivants.

- Le chapitre 2 présente les apports des sciences sociales à l'étude de notre domaine. Nous évaluons l'intérêt de disciplines comme la sociologie et l'ethnographie. Nous nous penchons aussi sur les problèmes éthiques que peuvent soulever les nouvelles technologies de communication.

- Le chapitre 3 décrit les apports de la psychologie cognitive. Nous décrivons un modèle cognitif et montrons comment il peut aider à modéliser le comportement de l'utilisateur dans une situation de communication via des moyens technologiques avancés. Nous présentons ensuite l'expérimentation "Garden Movie" à laquelle nous avons participé.

La seconde partie du mémoire étudie l'articulation entre les sciences humaines et sociales présentées dans la première partie et les technologies informatiques discutées dans la troisième partie. L'évaluation est un outil privilégié pour lier ces deux points de vue.

- Le chapitre 4 présente le concept de propriétés. Les propriétés nous permettent de définir des caractéristiques objectives et vérifiables d'un système informatique interactif. Informées par les disciplines des sciences de l'homme présentées dans la première partie, les propriétés nous aident à faire le pont entre les requis humains et sociaux et les caractéristiques intrinsèques du système informatique.

- Le chapitre 5 explore les apports de l'évaluation ergonomique, en particulier pour la vérification des propriétés. Nous constatons l'insuffisance des outils ergonomiques classiques pour notre domaine d'étude. Nous y présentons aussi NEIMO, notre plate-forme d'observation du comportement des utilisateurs.

La troisième partie du mémoire se concentre sur les aspects techniques plus directement informatiques : nous y étudions les particularités de la conception et de la réalisation des systèmes multi-utilisateurs dans une perspective génie logiciel.

- Le chapitre 6 discute les modèles d'architecture logicielle. Nous y constatons les insuffisances des modèles actuels.

- Le chapitre 7 présente notre propre modèle d'architecture, CoPAC. Nous présentons aussi plusieurs exemples de mise en œuvre du modèle.

- Le chapitre 8 propose un cadre de réflexion pour les mécanismes de connexion pour la communication homme-homme médiatisée. Nous y présentons notre outil de transmission de médias continus, UserLink.

Chaque chapitre étudie un aspect significatif du domaine de l'interface homme-machine. La plupart des chapitres adopte une structure commune : notre point de départ est l'étude du domaine du point de vue des systèmes mono-utilisateurs. Nous évaluons et critiquons les acquis dans la perspective des systèmes multi-utilisateurs. Puis nous proposons notre contribution en étendant les concepts et techniques présentés aux systèmes multi-utilisateurs et à la communication homme-homme médiatisée en particulier.

Première Partie

Les apports de sciences non-informatiques

Le cadre conceptuel : Terminologie, Principes, Propriétés et Techniques

Everything has now become digitized.
We have created a unimedia, really.

Nicholas Negroponte
in SIGGRAPH'95 advance program

Le cadre conceptuel :
Terminologie, Principes, Propriétés et Techniques

1.1. Introduction

De machine expérimentale qu'il était il y a moins de cinquante ans, l'ordinateur a conquis aujourd'hui un nombre de domaines impressionnant et s'installe dans notre vie quotidienne. Ce mouvement semble s'accélérer constamment et, comme le dit Nicholas Negroponte, tout—au moins toute information—peut être codé sous forme numérique. En particulier, l'image et le son sont maintenant couramment numériques et bénéficient donc des deux possibilités les plus originales de l'information numérisée : la duplication exacte sans dégradation et la transmission par des électrons ou des ondes électromagnétiques. Ces qualités rendent possibles la transmission d'images et de sons par l'intermédiaire de réseaux informatiques, en plus de la transmission de données informatiques pour laquelle ils étaient utilisés jusqu'à présent.

Dans le cadre de notre travail, nous nous sommes intéressés aux systèmes permettant à plusieurs utilisateurs de travailler en commun et de communiquer par des moyens audio et vidéo. Dans ce chapitre, nous commençons par préciser la terminologie et cerner le sujet de notre étude : les systèmes multi-utilisateurs. Puis nous exposons la grille d'analyse qui nous a guidée pour appréhender la conception et la réalisation de ces systèmes, et qui structure aussi ce mémoire. Cette grille d'analyse a trois niveaux d'abstraction : principes, propriétés, techniques.

1.2. Définitions

De la façon la plus élémentaire, on peut définir un système collecticiel comme un système interactif pouvant être utilisé par plusieurs utilisateurs. Mais cette définition manque de précision : la plupart des systèmes informatiques présentent en effet cette caractéristique. Qu'il s'agisse d'un système tel qu'Unix, ou d'un micro-ordinateur relié à un réseau et permettant le partage de ressources (imprimante, par exemple), ces systèmes sont utilisables par plusieurs utilisateurs, simultanément ou successivement, et le système est conçu pour permettre cette utilisation. Pourtant, l'appellation "collecticiel" (ou "groupware") ne recouvre traditionnellement pas ce type de systèmes et suscite plutôt l'évocation de systèmes favorisant l'interaction entre les utilisateurs comme CoLab [Stefik 1988], un des précurseurs. Il nous faut donc définir de façon plus rigoureuse les systèmes qui sont l'objet de notre étude.

Le terme "collecticiel" est traditionnellement utilisé dans la communauté française pour désigner les systèmes interactifs qui permettent le travail coopératif assisté par ordinateur ("computer-supported cooperative work" ou CSCW). Notons que la désignation officielle

est “synergiciel”[1] mais ce terme n’a jamais été utilisé par la communauté française. Pour certains auteurs, les systèmes permettant principalement la communication, tels les mediaspaces dont nous reparlerons au chapitre suivant, ne sont pas couverts par l’appellation collecticiel. Afin d’éviter toute ambiguïté, nous utiliserons le terme plus général de “système multi-utilisateur”.

Pour définir l’espace des systèmes multi-utilisateurs, nous nous appuyons sur le modèle proposé récemment par Ellis et Weiner [Ellis 1994]. Ce modèle définit un système multi-utilisateur comme une combinaison de trois aspects fonctionnels :

- l’aspect *ontologique* : les objets manipulés et les opérations possibles sur ces objets,

- l’aspect *coordination* : l’organisation et les relations entre les activités des participants,

- l’aspect *interface* : la façon dont les participants interagissent avec le système et entre eux.

Avec son modèle ontologique et sa représentation explicite des services de coopération, cette décomposition a le mérite de dégager deux classes de concepts fonctionnels caractérisants des systèmes multi-utilisateurs. Toutefois dans ce modèle, l'interface utilisateur recouvre à la fois l'interaction utilisateur-système et la communication entre utilisateurs. Selon le précepte bien connu de la distinction entre fonctions pures et interface utilisateur, il convient de considérer l'interaction homme-machine comme une conséquence des services de fond dont il est la vitrine perceptible. En conséquence, nous proposons de reprendre les fondements de la décomposition tripartite d’Ellis, mais nous en éliminons l'interface utilisateur. Cette élimination ne signifie pas que l'interface est un composant de seconde classe mais que son rôle doit être distingué de celui du noyau fonctionnel.

Comme le montre la figure 1.1, nous considérons que les services d’un système multi-utilisateur couvrent trois espaces : la production, la coordination et la communication. Ce modèle constitue notre modèle conceptuel de base : le trèfle des systèmes multi-utilisateurs.

[1] Loi du 4 août 1994, Journal Officiel de la République Française.

- L'*espace de production* correspond au modèle ontologique d'Ellis : il désigne les objets qui résultent d'une activité de groupe, par exemple un livre, une œuvre de cinéma, un logiciel, etc. L'espace de production correspond au *modèle conceptuel* tel que nous l'entendons en conception des systèmes mono-utilisateurs. Pour les systèmes multi-utilisateurs, ce modèle décrit les concepts qui motivent l'action de groupe, qui dénotent l'œuvre tangible commune, mais aussi l'espace privé de chaque utilisateur comme dans un système mono-utilisateur.

- L'*espace de coordination* reprend la définition d'Ellis : il s'agit de définir les acteurs (et notamment les individus, les groupes, les rôles, voire des agents logiciels "intelligents"), d'identifier les activités et les tâches (et notamment leurs relations temporelles), de désigner enfin les acteurs responsables des tâches et des activités. Tandis que l'espace de production offre une vue statique du système, l'espace de coordination en définit la dynamique. Sur ce point, nous adhérons au point de vue d'Ellis.

- L'*espace de communication* offre aux acteurs du système multi-utilisateur la possibilité d'échanger de l'information. Le contenu sémantique de cette information concerne les acteurs communicants. Il est étranger au système qui se contente de servir de messager. Cet aspect, qui constitue l'essence de la *communication homme-homme médiatisée*, n'est pas explicite en tant que tel dans le modèle d'Ellis.

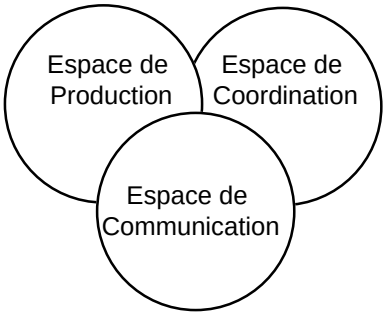


Figure 1.1. Le trèfle des systèmes multi-utilisateurs.

Selon les systèmes multi-utilisateurs, les trois espaces du trèfle fonctionnel n'ont pas la même importance. Par exemple, les systèmes d'édition partagée actuels mettent l'accent sur les services de production, le workflow sur la coordination, et les mediaspaces privilégient la communication. En poussant plus loin l'observation, l'importance relative de chacun de ces trois espaces peut varier au cours de l'interaction. Pour exprimer la variabilité de l'équilibre entre les trois classes de service, nous reprenons le principe du métamodèle Slinky qui, appliqué au modèle d'architecture Arch des systèmes interactifs

[Bass 1992], traduit la variation de l'importance des composants fonctionnels d'un système et leur perméabilité. Cette mouvance peut être statique (l'importance relative des trois composantes est décidée une fois pour toute à la conception) ou bien dynamique (elle varie alors en cours de session). La figure 1.2 montre les possibilités de variation des composants du modèle du trèfle.

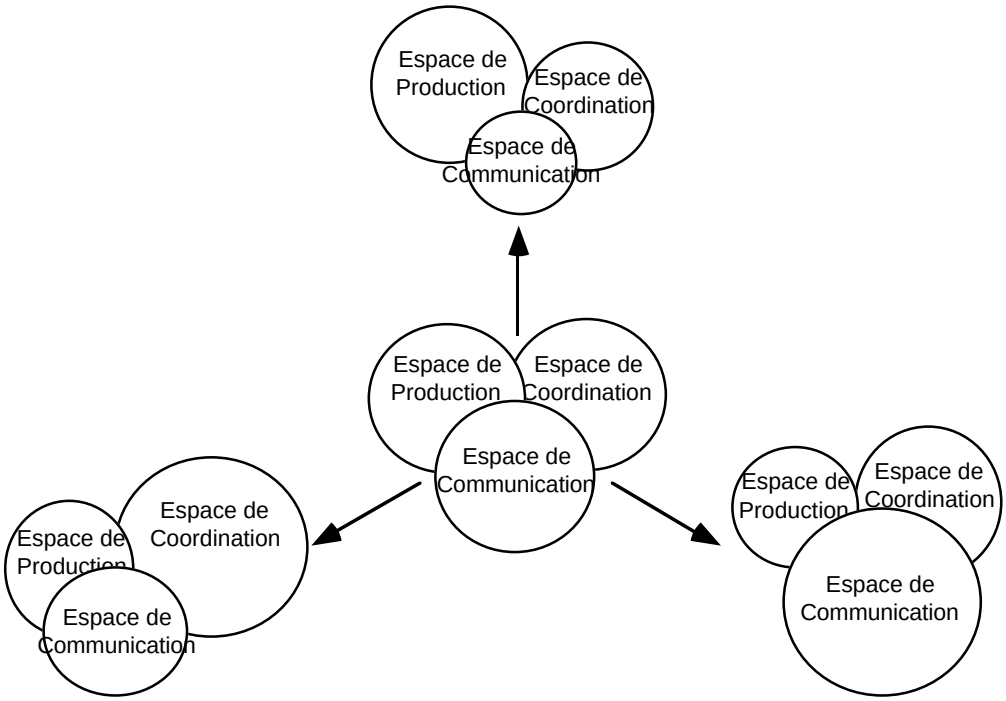


Figure 1.2. Le métamodèle Slinky appliqué au trèfle des systèmes multi-utilisateurs. L'importance de chaque facette peut varier.

L'intérêt du modèle du trèfle est double : il décompose distinctement en trois classes les services que l'on attend d'un système multi-utilisateur et il fournit une définition conceptuelle de ces systèmes. En effet, nous avons trop souvent vu des définitions à partir d'exemples d'applications (par exemple : "les collecticiels sont les systèmes de communication et les éditeurs partagés"). En identifiant clairement les trois aspects information, coordination, communication, ce modèle nous permet d'englober tous les systèmes multi-utilisateurs : à la fois les collecticiels existants et usuellement reconnus comme tels, mais aussi les collecticiels atypiques comme les outils de communication, ainsi que d'éventuelles nouvelles catégories de collecticiels à venir.

Afin d'éprouver la validité du modèle du trèfle, il est intéressant de le rapprocher de modèles organisationnels de l'utilisation de la technologie. Thompson par exemple distingue trois rôles de la technologie [Thompson 1967] :

- un rôle *médiateur* : la technologie est un creuset qui regroupe les résultats de tâches effectuées par différents utilisateurs, séquentiellement ou simultanément (figure 1.3),

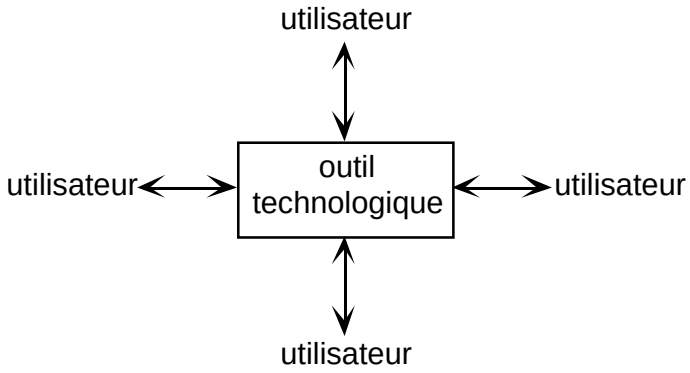


Figure 1.3. L'outil technologique comme médiateur. Différents utilisateurs réalisent des tâches distinctes et l'outil technologique rassemble les différents résultats. Les doubles flèches indiquent que les utilisateurs confient à l'outil le résultat de leurs tâches mais peuvent aussi réutiliser le résultat d'autres tâches pour accomplir leurs tâches.

- un rôle *d'ordonnancement* : les tâches sont interdépendantes séquentiellement dans le temps. Ce modèle est le modèle classique du travail à la chaîne (figure 1.4),

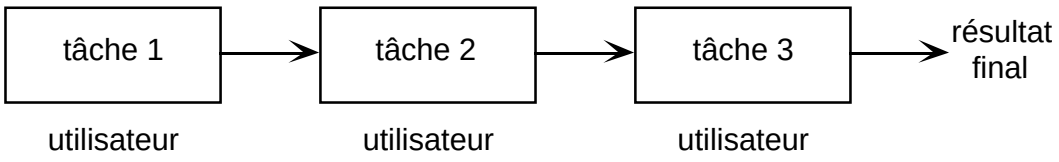


Figure 1.4. Le rôle d'ordonnancement de l'outil technologique. Chaque utilisateur réalise une tâche précise avec le support de l'outil technologique et la tâche (n+1) ne peut être réalisée avant terminaison de la tâche (n).

- enfin la technologie peut être utilisée de façon *intensive* : différentes tâches doivent être réalisées simultanément et peuvent être fortement interdépendantes. Les différents utilisateurs négocient entre eux directement (figure 1.5).

Notons que, même si ce modèle date d'il y a presque trente ans, il nous semble toujours pertinent. La vague récente de "business process reengineering" n'a fait que modifier l'allocation des tâches aux utilisateurs (principalement en leur assignant un plus grand éventail de tâches), mais n'a pas fondamentalement remis en cause les schémas de Thompson.

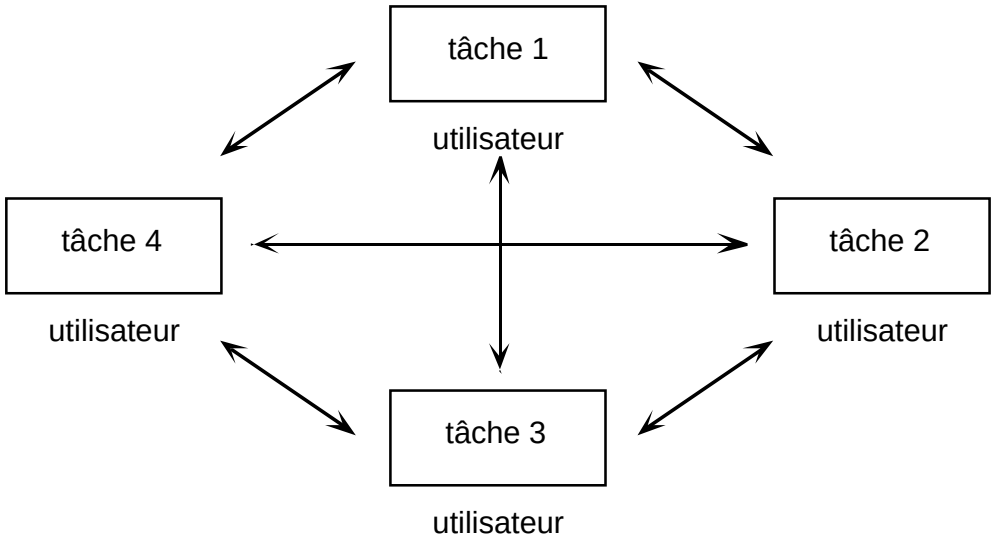


Figure 1.5. Le rôle intensif de la technologie. Chaque utilisateur réalise une tâche qui peut être liée de façon complexe aux tâches des autres utilisateurs. Les flèches indiquent que les utilisateurs doivent négocier entre eux pour clarifier et organiser les dépendances entre les tâches.

Il est instructif de mettre en regard ces trois rôles de la technologie et les trois composantes du trèfle des systèmes multi-utilisateurs. Même si l'on peut en faire une analyse plus subtile, il nous semble que l'on peut rapprocher ces deux modèles de la façon suivante. Le rôle médiateur de la technologie est à rapprocher de l'espace de production du modèle du trèfle. Le but de la technologie est ici un rôle intégrateur, les tâches des utilisateurs concourant à un but commun. Le rôle d'ordonnancement de la technologie privilégie la composante coordination du trèfle. La technologie a alors un rôle d'orchestration des différentes tâches et des différents utilisateurs. Enfin, le rôle intensif de la technologie met en avant la composante communication du trèfle.

Cette analyse rapide d'un modèle organisationnel nous conforte quant à la pertinence du modèle du trèfle pour décrire l'espace fonctionnel des systèmes multi-utilisateurs. Nous terminons ce paragraphe en définissant deux grandes classes de systèmes multi-utilisateurs.

Une distinction usuelle est faite entre systèmes multi-utilisateurs synchrones et asynchrones. La classification espace-temps proposée par Ellis [Ellis 1991] nous permet de clarifier cette distinction. La classification espace-temps repose sur deux axes : la distance temporelle entre les utilisateurs du système, et leur distance spatiale (figure 1.6).

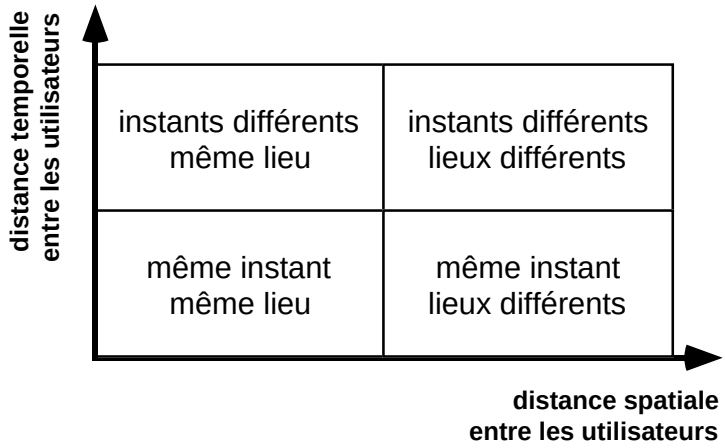


Figure 1.6. La classification espace-temps.

Notons que cette classification peut être affinée. Par exemple, il est plus réaliste de considérer les axes espace et temps comme continus [Salber 1994]. On peut alors considérer un ordre de grandeur pour la distance spatiale, par exemple distance métrique et distance kilométrique. Pour tenir compte des systèmes informatiques mobiles, [Grudin 1994] propose de distinguer “lieux différents mais imprévisibles” et “lieux différents mais prévisibles”. Il convient, à notre sens, d'insister sur le caractère imprévisible de la coopération et, par conséquent de considérer les cas "mêmes lieux mais imprévisibles" et “mêmes lieux mais prévisibles”.

La classification de la figure 1.6 nous permet déjà de distinguer le caractère synchrone et asynchrone des systèmes multi-utilisateurs. Les deux quadrants du haut de la classification, ceux pour lesquels les utilisateurs interagissent à des instants différents représentent les *systèmes asynchrones*. Les deux quadrants du bas, pour lesquels les utilisateurs interagissent au même instant (“même instant” du point de vue des utilisateurs et non du système), représentent les *systèmes synchrones*[1]. Notons qu’il est tentant de rapprocher la classification espace-temps du modèle du trèfle. Potentiellement, chacune des composantes du trèfle peut être considérée dans le cadre de la classification espace-temps. Dans [Salber 1995], nous proposons un modèle intégrant le modèle du trèfle des systèmes multi-utilisateurs et le classique modèle espace-temps.

1.3. La conception des systèmes multi-utilisateurs

La spécification, la réalisation et l’évaluation d’un système interactif sont réputées difficiles. Les systèmes interactifs multi-utilisateurs font apparaître de nouvelles

[1] Le terme “collecticiel temps réel” est également employé pour désigner les systèmes multi-utilisateurs synchrones.

difficultés. Une bonne part des problèmes rencontrés sont d'ordre technique ; mais en amont de la réalisation du système, la définition et la conception du système ne sont pas des tâches simples. La diversité des disciplines qui apportent leurs concours à la conception, chacune ayant une culture, un vocabulaire et un champ d'action différents, est pour beaucoup dans cette complexité. Les approches étudiant la conception des systèmes, telle l'approche Design Rationale peuvent apporter une aide à l'intégration de ces différentes disciplines. Nous avons aussi trouvé utile de structurer notre réflexion suivant une grille d'analyse à trois niveaux que nous présentons ensuite.

1.3.1. Intégrer l'approche multidisciplinaire

Il est maintenant reconnu que le développement des systèmes interactifs requiert la participation de plusieurs disciplines : psychologues, ergonomes, informaticiens apportent tous des compétences et des savoir-faire indispensables. Les systèmes multi-utilisateurs s'adressant à un groupe d'utilisateurs, des disciplines supplémentaires doivent intervenir, telles les sciences sociales. C'est là une des difficultés majeures du développement de ces systèmes. L'expérience montre que la coopération de plusieurs disciplines, chacune ayant des préoccupations, une culture, un vocabulaire différents, est un exercice complexe. Il est nécessaire de disposer d'un canevas intégrateur permettant de prendre en compte les apports de chacune des disciplines intervenant dans le processus de conception. Le Design Rationale se propose justement de jouer ce rôle d'intégration.

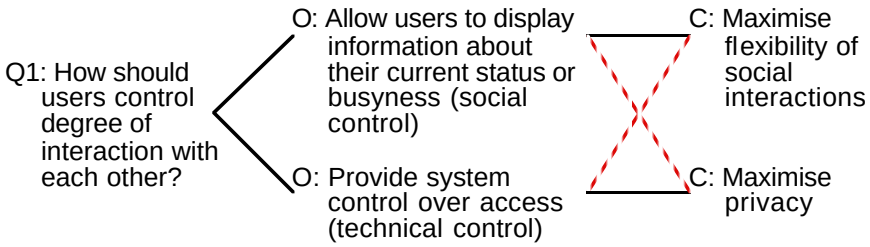


Figure 1.7. Exemple de mise en œuvre de la méthode QOC pour la conception d'un système mediaspace. La question posée Q1 (comment les utilisateurs peuvent-ils contrôler leur degré d'interaction ?) a suscité deux options qui sont examinées en fonction de deux critères. D'après [Duke 1995].

L'approche "Design Rationale" du processus de conception consiste à documenter, à l'aide de diverses techniques, toutes les décisions prises lors de la conception. Cette documentation participe à la traçabilité en génie logiciel. Nous avons choisi de détailler la méthode QOC, représentative de cette approche, mais on peut citer d'autres méthodes comme Design Space Development (DSD) [Bernsen 1994]. La méthode QOC (Questions, Options, Criteria) [MacLean 1991] est une représentante de ces techniques de Design Rationale. Elle consiste à explorer le plus systématiquement possible l'espace solution

correspondant à un problème donné. Pour chaque question qui apparaît lors de la conception d'un système, les options possibles constituant l'espace solution sont explorées et documentées. En regard sont indiqués les critères que satisfont ou non chacune des options. La figure 1.7 montre un exemple de mise en œuvre de la méthode QOC pour la conception du système mediaspace ECOM. Deux options sont envisagées pour la question Q1 et deux critères sont envisagés. Les traits pleins entre options et critères indiquent que le critère est satisfait par l'option considérée. Un trait pointillé indique que le critère s'oppose à l'option choisie.

Trois critiques peuvent être faites à QOC : la première porte sur l'utilisabilité pratique de la méthode, les deux autres portent sur le fait que, si QOC documente bien les choix de conception, la méthode n'offre pas de guide pour déterminer les critères et leur pertinence.

L'utilisabilité pratique de la méthode QOC peut être mise en question. L'espace solution, même pour un choix de conception précis, peut être d'une taille imposante. Il n'est pas rare de trouver cinq à dix options et autant de critères pour une question donnée. Plus il y a d'intervenants, plus il risque d'y avoir d'options et de critères. Dans une approche multidisciplinaire, chaque discipline fournit son propre jeu d'options et de critères correspondants. Plus il y a de disciplines impliquées, plus l'espace de conception est vaste. En l'absence de support informatique, l'espace de conception devient vite ingérable avec la méthode QOC [Bellotti 1994]. Dans [Nigay 1994], nous proposons des mécanismes d'encapsulation permettant d'améliorer la lisibilité de grands espaces de conception décrits avec la méthode QOC.

Une critique plus sérieuse que l'on peut faire à la méthode QOC est qu'elle n'inclut pas de guide permettant de déterminer les critères pertinents. Lorsque nous avons utilisé QOC pour décrire un espace de conception logicielle, nous avons choisi comme critères des propriétés issues du génie logiciel. Nous pensons que cette approche est généralisable pour d'autres espaces de conception. Au paragraphe suivant, nous proposons une grille d'analyse qui situe les propriétés et nous présenterons la nature des propriétés au chapitre 4.

Notre troisième critique met en évidence une contradiction de la méthode QOC. QOC ne permet pas de trancher le choix d'une option dans le cas où deux options satisfont des critères différents et non contradictoires. La figure 1.7 en donne un exemple simple : les deux options considérées satisfont chacune un critère différent. Comment le concepteur peut-il trancher ? Il est possible de donner plus ou moins d'importance aux différents critères, par exemple en les pondérant. Mais dans ce cas les décisions ayant mené à

l'attribution de tel poids à tel critère ne sont pas explicitement documentées dans le diagramme QOC. A la rigueur, ces décisions peuvent être consignées implicitement dans le cahier des charges ou le plan d'assurance qualité. Ce choix des poids, essentiel puisqu'il permet de prendre une décision dans des cas difficiles (puisque "tangents"), n'est pas capturé par le Design Rationale. Ce point est en contradiction avec la motivation même du Design Rationale. Même si les critères sont exprimés sous forme de propriétés, favoriser une propriété plutôt qu'une autre est influencé par des choix de conception d'un plus haut niveau. Ces choix de haut niveau sont les principes de notre grille d'analyse.

1.3.2. Une approche à plusieurs niveaux : principes, propriétés et techniques

Pour essayer de structurer notre vision du processus de développement des systèmes multi-utilisateurs, nous proposons une grille d'analyse à trois niveaux. Cette proposition vise à prendre en compte les apports des différentes disciplines qui participent à la conception des systèmes multi-utilisateurs et à intégrer leurs apports sous forme d'une "lingua franca" de propriétés. Les propriétés ainsi déterminées peuvent ensuite être utilisées avec une technique d'intégration telle que QOC, comme nous l'avons évoqué plus haut. Nous avons identifié trois niveaux échelonnés dans l'ordre chronologique du processus de conception : les principes, les propriétés et les techniques. La figure 1.8 résume notre approche.

Précisons tout de suite pour fixer les idées que l'on peut tracer un parallèle entre nos trois niveaux de principes, propriétés et techniques et un espace référentiel proposé pour le génie logiciel par McCall [McCall 1977]. McCall identifie aussi trois niveaux dans un processus de développement logiciel mettant en jeu un client, un concepteur et un réalisateur : les facteurs traduisent les attentes du client telles qu'exprimées dans le cahier des charges. Les critères s'adressent au concepteur : ils traduisent les facteurs dans les termes du concepteur. Enfin les solutions s'adressent au réalisateur et traduisent les facteurs en termes de solutions techniques. On retrouve dans notre proposition ces niveaux d'affinement successifs, chacun s'adressant à différents acteurs du cycle de développement.

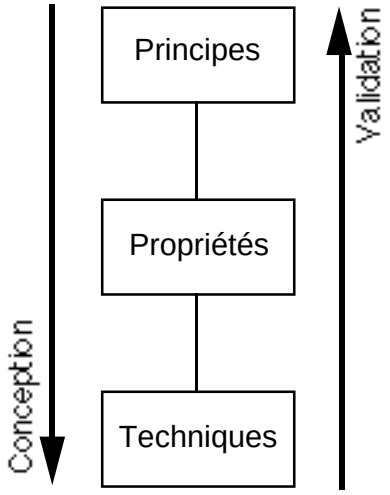


Figure 1.8. Les trois niveaux des principes, propriétés, et techniques. Le processus d'affinement descendant est effectué lors de la conception. Il doit être suivi d'un processus "remontant" de validation.

1.3.2.1. Principes

Nous utilisons ici le mot *principe* dans son sens "cause première"[1]. Un principe est une contrainte permettant de guider les choix de conception et qui est établi très tôt dans le processus de conception. Ce peut aussi être une philosophie qui préside à la conception. Un principe décrit un aspect qualitatif du système à concevoir. Un principe est en général indépendant de l'artefact informatique ; les indications exprimées par un principe sont d'un haut niveau et ne préjugent pas de la réalisation : elles doivent s'appliquer au système informatique mais un principe pourrait être tout autant valide dans le cas d'une réalisation non informatique. Ce dernier point est utile pour intégrer les apports des disciplines non informatiques. En effet, les enseignements que ces disciplines peuvent apporter sont souvent exprimés indépendamment de toute réalisation informatique.

En fait, les principes ne sont pas spécifiques aux systèmes multi-utilisateurs et on peut déjà les identifier dans la conception des systèmes interactifs mono-utilisateurs. Dans une approche génie logiciel, les principes seront exprimés, parfois implicitement, dans le cahier des charges. Ils peuvent aussi être modélisés, par exemple sous forme d'un modèle d'utilisateur établi au début de la conception, ou être intégrés dans le modèle de tâche. Pour les systèmes multi-utilisateurs cependant, de nouveaux types de principes doivent être pris en compte. Ils peuvent être exprimés sous forme d'un modèle du groupe utilisateur du système, et d'un modèle social ou organisationnel. Pour concrétiser cette notion de principe, nous donnons maintenant quelques exemples de principes sociaux ou organisationnels. Notons que ces principes sont présentés hors de leur contexte et qu'ils

[1] Principe : Cause première active, primitive et originelle. (*Le Robert*, dictionnaire de la langue française)

ne sont pas des principes universels. Ils ne sont valides que dans le cadre de la conception d'un système donné.

- "La vie privée des individus doit être protégée."
  Ce principe est issu des réflexions sur les aspects éthiques de l'usage des outils informatiques. Il est particulièrement important pour les systèmes de communication homme-homme médiatisée. Nous justifions ce principe au chapitre 2, en nous appuyant sur les travaux des sciences sociales. Toutefois, la conception d'un système particulier pourrait choisir d'ignorer ce principe.

- "Les utilisateurs sont souvent interrompus dans leur travail."
  Ce principe a été identifié par [Rouncefield 1994], lors d'une étude ethnographique. Les auteurs en déduisent implicitement des propriétés que devrait respecter un système informatique adapté à ces utilisateurs. Par exemple, les écrans ne doivent pas s'éteindre automatiquement après quelques minutes d'inactivité de l'utilisateur, ni surtout perdre des données en cours de saisie en s'éteignant. Ici, une propriété telle que la stabilité de l'écran serait pertinente.

- "L'automatisation totale du travail n'est pas souhaitable."
  Nous avons rencontré ce principe dans la description du travail des contrôleurs aériens. Ce principe a une influence certaine sur les choix de conception du système : on visera à fournir une information adéquate aux contrôleurs aériens, mais jamais le système ne se substituera à eux, en particulier pour prendre des décisions. La propriété de non-préemption présentée au chapitre 4 devrait être appliquée à la conception d'un tel système.

- "Les réunions doivent être courtes et efficaces."
  Ce principe pourrait guider le développement d'un système de réunion informatisée. Ce principe est organisationnel et n'est pas lié à un artefact technologique. On peut citer à titre d'anecdote la façon dont Hubert Beuve-Méry, fondateur et directeur du quotidien *Le Monde*, appliquait ce principe : les réunions de rédaction se déroulaient autour d'une table sans chaises, tous les participants restant debout pendant la réunion. De fait, les réunions de rédaction étaient toujours courtes ! Si l'on voulait appliquer ce même principe dans le cadre d'un outil de réunion informatisé, on pourrait par exemple envisager une limitation du temps de parole de chaque participant. La propriété qui étudie le rythme de l'interaction serait pertinente dans ce cas.

- "Laisser les utilisateurs résoudre leurs différends."
  Ce principe est proposé par Mike Godwin pour les "communautés virtuelles" [Godwin 1994]. Il montre encore une fois qu'un principe a un domaine d'application précis. Ce principe ne serait sans doute pas acceptable dans beaucoup d'entreprises où un conflit est généralement arbitré par un supérieur hiérarchique. Pour le cas des systèmes adaptés aux communautés virtuelles, il plaide contre les "modérateurs", au moins pour l'arbitrage des différends. Il est à rapprocher du principe suivant, plus général.

- "Préférer les solutions sociales, et seulement si tout échoue, recourir aux solutions techniques."
  Ce principe a été proposé par Amy Bruckman, suite à son expérience avec la communauté virtuelle LambdaMOO [Bruckman 1994]. Il recommande de privilégier les solutions négociées entre utilisateurs, et de ne recourir à des solutions techniques contraignantes que lorsque tous les autres recours ont été épuisés. La propriété contrôle technique/contrôle social serait alors applicable.

Bien sûr, nous ne pouvons pas prétendre décrire exhaustivement les principes. Chaque contexte de conception et les différentes disciplines impliquées dans le processus de conception feront apparaître des principes différents. Pour illustrer ce point, nous donnons deux exemples, le premier tiré d'un contexte particulier, le second concernant un point d'interface précis.

Le premier exemple montre comment un contexte de développement particulier peut générer un principe en contradiction absolue avec un principe de la conception centrée utilisateur. Un des principes de la conception centrée utilisateur est de viser à ce qu'une interface soit adéquate à la tâche de l'utilisateur. Or il est un domaine notable où ce principe ne doit *pas* être respecté : les jeux. En effet si l'on admet que, dans un jeu, la tâche de l'utilisateur est de gagner, un bon jeu sera justement celui où gagner est plus difficile. Dans cet objectif, l'interface d'un jeu viole certaines propriétés de façon flagrante. On pense aux jeux d'aventure comme Myst [Myst 1993] dont l'interface contient de nombreux éléments cachés et où la structure même de la tâche n'est ni explicite ni prévisible (les indices sont à rechercher dans un certain ordre, lui-même à découvrir). Les propriétés d'affordance, d'observabilité, d'honnêteté[1], pour ne citer que celles-là, sont délibérément contredites.

[1] Ces propriétés sont définies au chapitre 4.

Un autre exemple nous est fourni par un problème courant et important pour certains systèmes de communication homme-homme médiatisée : la conception de l'interface d'entrée d'un mot de passe par l'utilisateur. Un principe applicable ici est celui du respect de la confidentialité des utilisateurs. Des principes de sécurité peuvent être aussi considérés. Ces principes conduisent en général à ne pas respecter la propriété d'observabilité : le mot de passe ne s'affiche pas. Notons des différences d'approche cependant : sous Unix, taper un mot de passe ne provoque absolument aucun écho. En revanche, taper un mot de passe pour se connecter à un serveur AppleShare renvoie un caractère "•" à la place de chaque caractère tapé. Dans ce deuxième cas, une partie de l'état interne est observable : le nombre de caractères tapés. On peut se demander dans quelle mesure la satisfaction partielle de la propriété d'observabilité contredit des principes de sécurité (par exemple, on peut savoir que mon mot de passe AppleShare a 8 caractères). Un autre conflit entre la sécurité et l'observabilité peut apparaître : lorsque l'utilisateur doit entrer son nom et son mot de passe, en cas d'erreur, un système Unix affichera un message "login error". Pour satisfaire un principe de sécurité, le système n'affichera pas un message d'erreur informatif. En particulier il ne précisera pas si le nom est inconnu ou si le mot de passe est incorrect. Notons qu'AppleShare affiche deux messages d'erreur distincts. Mais en règle générale, ce choix d'interface contredit les propriétés de réparation d'erreur, d'informativité des messages et d'observabilité pour satisfaire des principes de sécurité.

A notre connaissance, il n'y a pas de méthode a priori pour vérifier que le système construit satisfait les principes énoncés au début de la conception. L'évaluation de la conformité du système aux principes souhaités ne peut se faire que par observation de l'utilisation du système et des transformations qu'il peut induire dans l'organisation considérée. Il est probable qu'une évaluation multidisciplinaire sera nécessaire. Force est de constater que ce domaine de recherches est encore peu abordé.

En résumé, les principes sont des contraintes de haut niveau, en général qualitatives et indépendantes de toute réalisation informatique, élaborées très tôt dans le processus de conception. Les principes peuvent trouver leur place dans le cahier des charges du système à réaliser, mais certains principes de conception, en particulier les principes organisationnels ou sociaux pour les systèmes multi-utilisateurs sont souvent implicites dans le cahier des charges. Une fois clairement exprimés, les principes peuvent être affinés en propriétés.

1.3.2.2. Propriétés

Une *propriété* d'un système est une caractéristique du système directement vérifiable. A l'aide de métriques appropriées, une propriété est quantifiable. Au contraire des principes,

les propriétés caractérisent le système informatique. Elles peuvent donc être utilisées comme critères de la notation QOC. Comme les principes, les propriétés sont élaborées à partir de l'apport des différentes disciplines intervenant dans la conception du système. Pour un développement donné, un ensemble de propriétés souhaitables est identifié. Le choix de ces propriétés et leur importance relative sont guidés par les principes énoncés au début de la conception.

Les propriétés pertinentes pour le développement des systèmes interactifs et en particulier des systèmes multi-utilisateurs sont exposées plus en détail au chapitre 4. Comme les propriétés concernent directement le système informatique et sont quantifiables, elles sont vérifiables par analyse du système construit. L'évaluation de la conformité du système aux propriétés énoncées est donc possible. Cette évaluation détermine l'utilisabilité du système. Dans la deuxième partie de ce mémoire, nous définirons des propriétés et montrerons de quelle façon elles participent à l'évaluation du système.

1.3.2.3. Techniques

Les *techniques* sont mises en œuvre pour réaliser le système une fois qu'il est spécifié. Il s'agit principalement de modèles et d'outils informatiques, et leur choix est guidé par les propriétés souhaitées du système. En effet, si l'on utilise une méthode telle que QOC, l'examen des propriétés souhaitées (c'est-à-dire des critères) va permettre de faire un choix dans l'espace des solutions et souvent induire un choix de techniques et d'outils. Par exemple, pour satisfaire la propriété de portabilité, on voudra utiliser une boîte à outils virtuelle ou un environnement de développement multi-plate-forme.

Nous montrons dans les deux chapitres suivants comment les disciplines non informatiques peuvent générer des principes. Nous en avons déduit des propriétés, présentées avec d'autres propriétés usuelles au chapitre 4. Le chapitre 5 étudie comment la satisfaction des propriétés peut être évaluée. Enfin, les techniques, sous forme de techniques d'évaluation, de modèles d'architecture et d'outils, font l'objet des chapitres 5, 6, 7 et 8 de ce mémoire.

1.4. Synthèse

Pour répondre à la complexité du développement des systèmes multi-utilisateurs, due principalement à la multiplicité des participants impliqués dans le processus de conception, nous avons identifié trois niveaux d'affinement dans le processus de développement. Ces trois niveaux mettent en évidence des principes, des propriétés et des techniques que le système construit devra utiliser ou satisfaire. Le passage d'un niveau à un autre correspond à un processus de réification pour finalement aboutir à la

spécification du système et à l'ensemble des outils nécessaires à son développement effectif. En ce qui concerne le processus de validation, qui permet de vérifier la conformité du système construit aux niveaux des propriétés et des principes, nous devons constater les limites actuelles. A chaque niveau correspond un ensemble de concepts et un espace de référence. Notre structuration en trois niveaux n'est pas une méthode de conception, mais une méthode d'analyse. Elle ne contraint pas l'ordonnancement et est donc adaptée au comportement souvent opportuniste des concepteurs : la conception n'est jamais exclusivement "top-down" ou "bottom-up". Le processus descendant de conception fait pour le moment l'objet de l'essentiel des recherches, et il faudra sans doute attendre que celui-ci soit plus élucidé avant de voir apparaître des méthodes de validation. En particulier, l'évaluation multidisciplinaire du système vis-à-vis des principes énoncés lors de la conception du système nous semble une voie de recherche à explorer.

Références

[Bass 1992] L. Bass, R. Little, R. Pellegrino, S. Reed, R. Seacord, S. Sheppard et M. R. Szczur. *The UIMS Tool Developers' Workshop: A Metamodel for the Runtime Architecture of an Interactive System,* in *SIGCHI Bulletin*, 24(1), janvier 1992. pp. 32-37.

[Bellotti 1994] V. Bellotti et A. MacLean. *Integrating and Communicating Design Perspectives with QOC Design Rationale*, Amodeus Project, Working Paper, ID/WP29, 1994.

[Bernsen 1994] N. O. Bernsen et J. Ramsay. *An executive summary of the DSD framework illustrated by two worked exemplars*, Amodeus Project, 1994.

[Bruckman 1994] A. Bruckman. *Approaches to Managing Deviant Behaviour in Virtual Communities (Panel)*, CHI 95, ACM Conference on Human Factors in Computing Systems, Boston, Massachussets, USA, 1994. pp. 183-184.

[Duke 1995] D. J. Duke, (editor), A. Aboulafia, A. E. Blandford, S. J. Buckingham-Shum, J. Darzentas, J. May, L. Nigay, J. Ramsay, D. Salber et S. Verjans. *Integration Techniques for Multi-Disciplinary HCI Modelling: A Survey*, Projet européen ESPRIT BRA 7040 AMODEUS 2, RP3 Working Paper (en cours de soumission), ID/WP, 1995.

[Ellis 1994] C. Ellis et J. Wainer. *A Conceptual Model of Groupware*, CSCW 94, ACM Conference on Computer Supported Cooperative Work, Chapel Hill, North Carolina, USA, 1994. pp. 79-88.

[Ellis 1991] C. A. Ellis, S. J. Gibbs et G. L. Rein. *Groupware: some issues and experiences,* in *Communications of the ACM*, 34(1), janvier 1991. pp. 38-58.

[Godwin 1994] M. Godwin. *How to make virtual communities work,* in *Wired*, 2(6), juin 1994. pp. 92.

[Grudin 1994] J. Grudin. *CSCW: History and Focus,* in *IEEE Computer*, 27(5), mai 1994. pp. 19-26.

[MacLean 1991] A. MacLean, R. M. Young, V. Bellotti et T. Moran. *Questions, Options and Criteria: Elements of Design Space Analysis,* in *Human-Computer Interaction*, 6(3 & 4), 1991. pp. 201-250.

[McCall 1977] J. McCall. *Factors in Software Quality,* General Electric Eds., 1977.

[Myst 1993] *Myst 1.0.* Logiciel pour Macintosh (CD-ROM). 1993.

[Nigay 1994] L. Nigay, J. Coutaz et D. Salber. *Initial draft preliminary on-the-fly PAC analysis for the ECOM Interface*, Projet européen ESPRIT BRA 7040 AMODEUS 2, Integration Report, SM/IR 4, 1994.

[Rouncefield 1994] M. Rouncefield, J. A. Hughes, T. Rodden et S. Viller. *Working with "Constant Interruption": CSCW and the Small Office*, CSCW 94, ACM Conference on Computer-Supported Cooperative Work, Chapel Hill, North Carolina, USA, 1994. pp. 275-286.

[Salber 1994] D. Salber et J. Coutaz. *Taxinomie des mécanismes de connexion pour la communication homme-machine-homme*, IHM'94, Sixièmes Journées sur l'Ingénierie des Interfaces Homme-Machine, Lille, France, 1994. pp. 183-189.

[Salber 1995] D. Salber, J. Coutaz, D. Decouchant et M. Riveill. *De l'observabilité et de l'honnêteté : le cas du contrôle d'accès dans la Communication Homme-Homme Médiatisée*, soumis à IHM'95, Conférence sur l'Ingénierie des Interfaces Homme-Machine, Toulouse, France, 1995.

[Stefik 1988] M. Stefik, G. Foster, D. G. Bobrow, K. Kahn, S. Lanning et L. Suchman. *Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings,* in *Computer-Supported Cooperative Work: A Book of Readings.* I. Greif, (ed.) Morgan Kaufman, San Mateo, California, 1988. pp. 335-366.

[Thompson 1967] J. D. Thompson. *Organizations in Action,* McGraw-Hill, New York, New York, USA, 1967.

Apport des Sciences Sociales

Nos prolongements électriques échappent tout simplement à l'espace et au temps, et suscitent des problèmes de participation et d'organisation humaines auxquels nous ne connaissons aucun précédent.

Marshall McLuhan

Apport des Sciences Sociales

2.1. Introduction

Déjà utiles pour la conception des systèmes mono-utilisateurs, les sciences sociales se révèlent riches d'enseignements pour la compréhension et l'analyse du fonctionnement des groupes et des organisations. Ces enseignements peuvent contribuer de façon déterminante à la conception des systèmes multi-utilisateurs. Dans le cas des systèmes de communication homme-homme médiatisée, la sociologie, les sciences de la communication et l'étude des aspects éthiques apportent des éléments de réflexion importants. Dans ce chapitre, nous présentons un aperçu des contributions effectives et potentielles des sciences sociales à l'étude des systèmes multi-utilisateurs. Nous décrivons d'abord les apports de la sociologie et de l'ethnographie en nous appuyant sur des études de cas de la littérature. Nous nous intéressons ensuite aux aspects éthiques, en particulier pour les systèmes de communication homme-homme médiatisée. Puis nous montrons comment les sciences de la communication et les disciplines juridiques peuvent aussi aider à réfléchir à la conception des systèmes multi-utilisateurs. Enfin, nous présentons la façon dont les principes issus de ces disciplines ont été intégrés dans des systèmes de communication homme-homme médiatisée : les mediaspaces et notamment VideoPort, notre propre mediaspace.

2.2. Sociologie et Ethnographie

La sociologie étudie scientifiquement l'homme vivant en société. C'est une science positive et non normative, c'est-à-dire qu'elle constate ce qui est et ne décrète pas ce qui devrait être. Même si elle a longtemps été considérée comme une science mineure, ses objectifs sont ambitieux et les réalités qu'elle décrit sont complexes. La sociologie est une science à facettes multiples. Nous n'essaierons donc pas d'en donner une vue exhaustive, mais nous montrerons juste sur quelques exemples les objets qu'elle étudie et les bénéfices que nous pouvons en retirer pour les systèmes multi-utilisateurs. Notons que la sociologie entretient des liens étroits avec d'autres disciplines des sciences humaines qui peuvent aussi informer la conception des systèmes multi-utilisateurs comme la psychologie sociale, l'ethnologie, l'anthropologie, etc. L'ethnographie est une forme particulière de recherche sociologique. Elle consiste à faire un inventaire détaillé des modes de vie et de relation d'une population de faible effectif [Coiffier 1990]. L'ethnographie est à la base de l'anthropologie sociale et nous verrons des exemples tirés de la littérature qui en montrent l'utilité appliquée à notre cas d'étude.

2.2.1. Les théories sociologiques fondamentales

La sociologie s'est intéressée très tôt aux organisations, par exemple aux communautés, aux syndicats et aux entreprises. Deux auteurs sont généralement considérés comme les précurseurs de cette réflexion : Taylor et Weber [Ballé 1992]. Taylor, avec sa théorie de l'organisation scientifique du travail, propose une approche normative qui recommande la prise en compte des contraintes technologiques par l'organisation et l'adaptation des comportements humains à l'organisation (figure 2.1).

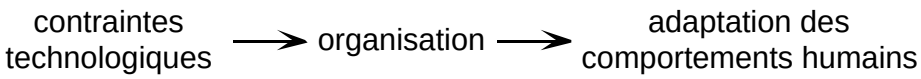


Figure 2.1. La théorie de Taylor. Les contraintes technologiques déterminent l'organisation et l'organisation façonne les comportements humains.

L'application la plus connue de la théorie taylorienne est le travail à la chaîne. Les contraintes de production de masse engendrent une mécanisation du travail. Ces contraintes technologiques mènent à une décomposition du travail en tâches élémentaires et à leur rationalisation. L'organisation conduit à une adaptation des comportements humains aux contraintes technologiques. L'approche de Taylor est contraire aux règles de la sociologie moderne qui, nous l'avons dit, refuse les théories normatives. Elle est aussi en opposition totale avec les principes de la conception des systèmes interactifs. Notre but n'est pas d'adapter le comportement humain aux contraintes technologiques, mais au contraire de concevoir des outils centrés sur les besoins des utilisateurs. De ce point de vue, la théorie taylorienne est intéressante comme "repoussoir" : elle représente exactement la démarche inverse de celle que nous souhaitons mettre en œuvre.

Weber a travaillé sur le rôle de la bureaucratie dans les sociétés modernes [Weber 1947]. Contrairement à Taylor, son approche est descriptive, analytique et non normative. La théorie de la bureaucratie de Weber propose un modèle qui décrit le développement des grandes organisations, comme les administrations. Nous en retenons quelques aspects et des exemples de principes. La théorie de la bureaucratie étudie les types d'autorité ou de domination[1] [Ballé 1992]. L'autorité légale repose sur la légalité des règlements et la légitimité de chefs désignés conformément à la loi. L'autorité traditionnelle a pour fondement la croyance dans les traditions en vigueur et la légitimité de ceux qui sont appelés au pouvoir en vertu de la coutume. Enfin, l'autorité charismatique a pour base l'abandon des membres à la valeur personnelle d'un homme qui se distingue par son exemplarité. Weber inscrit la bureaucratie dans le cadre de l'autorité légale et propose notamment les principes suivants :

---

[1] "Authority" en anglais.

- "L'existence de services définis et donc de compétences rigoureusement déterminées par les lois ou règlements, de sorte que les fonctions sont nettement divisées et distribuées ainsi que les pouvoirs de décision nécessaires à l'accomplissement des tâches correspondantes."

- "La hiérarchie des fonctions, ce qui veut dire que le système administratif est fortement structuré en services subalternes et en postes de direction, avec possibilité de faire appel de l'instance inférieure à l'instance supérieure. (…)"

- "Le droit qu'a l'autorité de contrôler le travail de ses subordonnés (…)"

- "La séparation complète entre la fonction et l'homme qui l'occupe, car aucun fonctionnaire ne saurait être propriétaire de sa charge ou des moyens de l'administration."

Notons tout de suite que ce que Weber appelle ici principe est très proche de la notion de principe que nous avons définie au chapitre 1. Bien entendu, les principes de Weber font partie d'une théorie et sont donc très généraux. Ils mériteraient d'être nuancés pour chaque contexte particulier. Mais pour les systèmes multi-utilisateurs adaptés à ce type d'organisation, ils peuvent déjà nous amener à considérer des propriétés de dialogue et de coordination comme la délégation (définie au chapitre 6), les caractéristiques du contrôle d'accès ou encore l'authentification des utilisateurs.

Au-delà des théories fondatrices de la discipline, la sociologie moderne propose des outils pratiques d'analyse du fonctionnement des groupes humains. Par exemple, [Bernoux 1985] propose une grille d'analyse complète des organisations que nous présentons maintenant.

2.2.2. Une grille d'analyse sociologique

Bernoux prône une approche progressive. Il recommande d'éviter la tentation de commencer par identifier les relations au sein d'un groupe. Il insiste sur la nécessité d'une description globale, c'est-à-dire une description du groupe dans son contexte d'évolution. Cette analyse fait ressortir que le fonctionnement d'un groupe n'est pas inhérent au groupe mais qu'il est le produit d'un ensemble de contraintes externes et internes. La description globale peut prendre en compte des données aussi diverses que l'activité de l'organisation, son histoire, sa localisation géographique, son statut juridique, des données économiques, etc. Après cette première description, la méthode d'analyse

propose de distinguer deux types de fonctionnement de l'organisation : les règles formelles et le fonctionnement informel.

Les règles formelles peuvent être déduites de l'organigramme de l'organisation et de ses règlements. Elles recouvrent :

- le système officiel de division du travail et de répartition des tâches,
- la définition des statuts et des rôles,
- le système hiérarchique,
- les communications,
- le système de contribution-rétribution (par exemple, les sanctions, les promotions, etc.).

L'analyse des règles formelles permet de faire apparaître le système de valeurs et la culture d'une organisation. Mais la non-existence de règles formelles est aussi un élément de la culture et du système de valeurs.

Ensuite, Bernoux propose d'étudier le fonctionnement informel de l'organisation. Il propose pour cela deux approches. La première consiste à évaluer l'écart entre le fonctionnement effectif et les règles formelles. Une technique proposée à cet effet consiste à analyser un imprévu, par exemple une commande inhabituelle ou une panne exceptionnelle. Il est intéressant d'observer dans ce cas comment les circuits formels sont respectés ou non, et comment fonctionnent effectivement les systèmes hiérarchiques et de communication. Bernoux propose aussi d'analyser les conflits : comment naissent-ils, comment sont-ils résolus ? La deuxième approche consiste à étudier les rapports affectifs entre les individus et entre les groupes. L'hypothèse qui motive cette approche est qu'une relation de compétition qui n'est pas reconnue formellement engendre une très forte tension qui se traduit par de l'agressivité. L'étude du fonctionnement informel permet donc d'évaluer la conformité de la réalité objective aux règles formelles.

Le but de cette analyse en trois étapes (description globale, règles formelles, fonctionnement informel) est de décrire l'organisation comme un système (au sens de la systémique) ou un ensemble de systèmes interdépendants. Bernoux identifie trois entités que cette description doit faire apparaître : les acteurs, les systèmes d'action concrets, et les objectifs et les enjeux des acteurs. Les *acteurs* peuvent être des individus ou des groupes d'individus. Les *systèmes d'action concrets* définissent les relations entre les acteurs. Les *objectifs* sont les buts des acteurs, et les *enjeux* représentent ce que les acteurs peuvent s'attendre à gagner ou à perdre dans ce qu'ils entreprennent, et l'importance qu'ils attribuent à ce gain ou à cette perte.

Nous remarquons que cette grille d'analyse fait ressortir des concepts déjà présents dans l'étude des systèmes multi-utilisateurs, quoiqu'exprimés dans un vocabulaire sociologique et structurés de façon différente. Par exemple, la notion d'acteur recouvre à la fois la notion d'utilisateur ou de groupe d'utilisateurs et celle de rôle. Les systèmes d'action concrets définissent un éventail de tâches de coordination et de communication. Les objectifs correspondent principalement aux tâches de production. En revanche, la notion d'enjeu n'apparaît pas dans les analyses classiques des systèmes multi-utilisateurs. Or [Poltrock 1995] constate que l'échec des systèmes multi-utilisateurs tient souvent au fait que les utilisateurs ne voient pas les bénéfices de leur travail. Ces auteurs attribuent ce problème couramment rencontré au fait que nous manquons d'intuition pour concevoir les systèmes multi-utilisateurs. Nous souscrivons entièrement à cette opinion. Cependant nous insistons sur le fait que la sociologie a identifié depuis plusieurs années que la notion d'enjeu est une composante importante d'une tâche d'un individu dans une organisation. L'apport des méthodes d'analyse de la sociologie nous permet donc de comprendre ce qui a été constaté après coup : un utilisateur d'un système multi-utilisateur doit être informé des enjeux de ses tâches. Nous constatons ici que l'analyse sociologique nous permet de compenser le manque d'intuition constaté par Poltrock et Grudin. Nous tirons immédiatement de cette méthode d'analyse sociologique le principe suivant : "l'utilisateur qui réalise une tâche doit être informé des enjeux de la tâche". Nous en déduisons une propriété au chapitre 4.

2.2.3. Ethnographie

L'ethnographie est une démarche sociologique particulièrement adaptée à l'étude d'une population de faible effectif. Elle est donc pertinente pour la conception des systèmes multi-utilisateurs lorsqu'ils concernent un petit groupe d'utilisateurs. Elle consiste à faire un inventaire détaillé des modes de vie et des relations entre les individus. Une des premières applications de la démarche ethnographique décrit exhaustivement un petit village français de 1150 habitants [Wylie 1970]. Cet ouvrage de près de 500 pages présente pour cette communauté son histoire, son économie, sa démographie, son organisation sociale, les attitudes religieuses et politiques, et enfin les relations entre l'individu et la communauté. Ce livre est caractéristique de la démarche ethnographique. Comme l'indique l'origine étymologique du mot ethnographie, il s'agit de consigner par écrit toutes les caractéristiques sociales d'un groupe d'individus. On mesure tout de suite à la fois l'ambition et les problèmes pratiques posés par cette approche. Le temps nécessaire à une étude ethnographique ferait probablement pousser les hauts cris à n'importe quel chef de projet ! En fait, on trouve dans la littérature des exemples d'étude ethnographique réussies, mais portant souvent sur des aspects très spécifiques et limités du système à réaliser. Par exemple, [Mantei 1989] étudie trois aspects de réunions de

cadres dans une même salle autour d'une table : l'arrangement des sièges, la possibilité qu'ont les participants de se voir (malgré la présence d'équipements informatiques sur la table), et les protocoles d'utilisation d'un tableau noir électronique commun. Même si cette étude ne revendique pas être une étude ethnographique, elle est caractéristique de cette démarche. Le contexte, les participants et leurs relations sont soigneusement analysées et mènent à des choix de conception. [Suchman 1987] présente d'autres études pour des systèmes de bureautique. [Rouncefield 1994] rapporte l'étude ethnographique d'un environnement particulier : celui d'une petite entreprise où les utilisateurs sont fréquemment interrompus dans leurs tâches. Nous avons déjà mentionné ce travail au chapitre 1 et avons présenté un principe qui peut en être déduit.

Un autre domaine d'activité constitue un champ d'application intéressant pour la démarche ethnographique : celui du contrôle de procédés. [Heath 1991] par exemple, étudie la coordination dans les salles de contrôle du métro de Londres. [Bentley 1992] présente l'apport d'une démarche ethnographique pour la conception d'un système de contrôle aérien. Ces auteurs notent toutefois la difficulté de passer directement de l'étude ethnographique aux spécifications du système. Notre approche, qui identifie des principes tirés de l'étude ethnographique et des propriétés informées par ces principes, vise à structurer l'analyse des résultats de l'étude ethnographique. Les auteurs remarquent aussi le fossé qui sépare la sociologie de l'informatique, à la fois au niveau du vocabulaire, des points de vue, et des méthodes de travail. Il est certain qu'il y a encore de ce côté d'importants progrès à faire de part et d'autre.

Il nous faut aussi mettre en garde contre la difficulté des analyses sociologiques. [Harrison 1994] relate une collaboration avec des sociologues pour la conception d'un système de communication audio/vidéo. Cet article en tire des leçons utiles sur par exemple l'établissement d'un vocabulaire commun, la validité des observations de groupes d'utilisateurs, et la poursuite de la collaboration avec les sociologues tout au long du projet, même après les résultats de l'analyse sociologique. Ce dernier point nous renforce dans notre conviction que la collaboration pluridisciplinaire est indispensable non seulement pour la conception du système mais aussi pour son évaluation, comme nous l'avons indiqué au chapitre 1. Enfin, notons qu'une erreur bien connue des sociologues guette les éventuels sociologues amateurs. Il s'agit de l'effet Hawthorne que l'on peut résumer en disant que les gens réagissent positivement au fait que l'on s'occupe d'eux pour améliorer leur situation. Ce cas particulier du problème courant de l'influence de l'observateur sur l'objet de l'observation doit nous rappeler que la sociologie requiert des techniques adaptées qui ne peuvent être mises en œuvre que par des sociologues.

Les études sociologiques et ethnographiques peuvent fournir des indications précieuses à la conception des systèmes multi-utilisateurs, par exemple sous forme de principes. Les collaborations entre sociologues et informaticiens restent malgré tout marginales. Un effort de compréhension réciproque reste encore nécessaire. D'autres disciplines des sciences sociales, telles que l'éthique, commencent à se préoccuper d'appliquer leurs méthodes et concepts à l'étude des systèmes informatiques. Nous allons voir que l'éthique peut nous apporter des principes importants pour la conception des systèmes de communication homme-homme médiatisée.

2.3. Éthique informatique

L'éthique informatique[1] est une discipline récente qui puise ses sources dans les textes classiques sur l'éthique d'Aristote et de Kant. Le texte séminal de James Moor ne date que de 1985 [Moor 1985]. Maner justifie la pertinence de la discipline par les caractéristiques uniques de l'outil informatique [Maner 1995]. Comme la sociologie, l'étude de l'éthique rassemble une grande diversité de disciplines : philosophie, droit, morale, management, marketing, informatique, etc. Le problème de l'intégration des apports de l'éthique se pose donc avec une acuité accrue. Toutefois, l'utilisation des principes va nous permettre de surmonter cette difficulté. L'éthique informatique est en effet une discipline qui génère explicitement des principes sous forme de règles. On pourrait abusivement assimiler ces principes à des principes moraux ou juridiques. En réalité, l'éthique est une discipline plus subtile. Elle se pose des questions qui se situent au-delà de la morale ou de la loi, soit parce que ses domaines d'étude sont trop récents pour avoir déjà fait l'objet d'une législation, soit parce qu'elle examine des cas tangents où justement les seules règles d'une morale ne peuvent apporter de réponse. Les principes issus de l'éthique ne sont pas des principes prescriptifs comme ceux de la morale, mais des principes indicatifs, valides dans une situation donnée et à un instant donné. La médecine a eu une démarche pionnière dans le domaine de l'éthique : le serment d'Hippocrate, les comités d'éthique sur les nouvelles possibilités de la science et de la technologie médicales sont peut-être des exemples à suivre pour l'informatique.

En effet, les menaces que certaines utilisations de l'informatique font peser sur les individus et les sociétés sont loin d'être négligeables. Les exemples abondent, qu'il s'agisse de risques liés à des défauts de logiciels [Neumann 1995] ou d'utilisations délibérées d'outils informatiques. Notons que les pouvoirs publics sont souvent conscients de certains de ces risques, mais qu'ils ne se donnent pas toujours les moyens de les prévenir, soit par ignorance, soit par impuissance. En Angleterre par exemple, le

[1] "Computer Ethics" en anglais.

Data Protection Registrar, équivalent de la Commission Nationale Informatique et Libertés en France, estime que seule une infime fraction des fichiers nominatifs lui sont déclarés [France 1995]. Les grandes associations d'informatique comme l'ACM[1] ou l'IFIP[2] se sont dotées de codes éthiques. Mais leurs prescriptions s'adressent plus aux utilisateurs qu'aux concepteurs. En ce qui concerne les risques liés aux défauts des logiciels, le génie logiciel vise à améliorer et vérifier la qualité des logiciels. En ce qui concerne les risques liés à des utilisations illicites ou "déviantes" de l'informatique (on pense par exemple au croisement abusif de bases de données nominatives), nous pensons que la communauté informatique doit prendre conscience des menaces et proposer des stratégies, voire des mécanismes permettant de limiter les abus. Pour notre domaine d'étude, et plus particulièrement pour les systèmes de communication homme-homme médiatisée, nous nous intéresserons principalement à l'étude de la protection de la vie privée.

2.3.1. Protection de la vie privée

Le respect de la vie privée des individus est un prérequis indispensable de la vie en société. Fried par exemple (cité ainsi que les auteurs suivants dans [Johnson 1994]) voit dans le respect de la vie privée une condition nécessaire à l'établissement de relations d'amitié, d'intimité et de confiance. D'autres auteurs affirment que ce respect est une des conditions de viabilité des régimes démocratiques (on pense au secret du vote, par exemple). D'autres encore écrivent que nous ne pouvons exercer notre autonomie sans vie privée (dans la théorie kantienne, l'autonomie est une valeur fondamentale de l'homme). Enfin, dans le cas extrême d'un régime autoritaire ou en temps de guerre, la protection de la vie privée des individus peut même être une question de vie ou de mort. Ce constat nous permet de réfuter immédiatement un argument trop souvent entendu : "seuls se préoccupent du respect de la vie privée ceux qui ont quelque chose à cacher". Dans un contexte plus large, le respect de la confidentialité est une exigence des organisations : qu'il s'agisse des entreprises, des organisations politiques ou gouvernementales, toutes manipulent des données qui leur sont privées et dont elles entendent protéger la confidentialité. Il faut sans doute y voir une des raisons de l'échec récent du système de cryptage Clipper aux États-Unis, dont les clés secrètes de n'importe quel utilisateur auraient été accessibles au gouvernement [Platt 1995]. Cette discussion sur l'importance de la vie privée nous permet de déduire le principe suivant, applicable aux systèmes multi-utilisateurs : "le respect de la vie privée est un droit fondamental de

---

[1] Association for Computing Machinery

[2] International Federation for Information Processing

chaque individu". Nous proposons au chapitre 4 la propriété d'observabilité publiée qui est issue de ce principe.

2.3.2. PAPA : un espace de réflexion

De façon plus générale, [Mason 1986] propose l'espace de réflexion PAPA (Privacy, Accuracy, Property, Access) qui est applicable à la communication homme-homme médiatisée. Cet espace propose quatre dimensions : vie privée, exactitude, propriété, accès. La première dimension concerne le respect de la vie privée que nous avons détaillé au paragraphe précédent. La seconde exprime l'importance de l'exactitude de l'information communiquée. Cette caractéristique peut par exemple s'appliquer à un système comme celui des newsgroups d'Internet. L'ubiquité quasi-instantanée de l'information électronique permet la circulation rapide d'informations, qu'elles soient exactes ou non. La propriété d'intégrité que nous proposons au chapitre 4 permet de vérifier que le système ne déforme pas l'information, mais au niveau social, il n'est pas possible de garantir l'exactitude de toute information. D'autant plus que l'exactitude, surtout lorsqu'il s'agit d'opinions, est un concept à manipuler avec précaution. Cette dimension de l'espace PAPA est à rapprocher de plusieurs autres notions : la censure, que nous discutons plus loin, et la responsabilité des individus, qui est une composante importante de l'éthique. La troisième dimension s'intéresse à la propriété intellectuelle des informations circulant sur un média électronique. Nous présentons plus loin quelques problèmes liés aux droits d'auteur. Enfin, la dernière dimension de l'espace PAPA concerne l'accès à l'information. Cet aspect recouvre par exemple la question de l'accès universel aux nouveaux réseaux de communication. [Schneiderman 1995] analyse les problèmes liés à cette notion d'accès universel et le risque de scinder la société en nantis et démunis de l'information[1]. De façon plus précise, Mason distingue trois composantes dans la notion d'accès : l'accès à l'éducation et aux connaissances permettant d'utiliser les nouvelles technologies, l'accès à la technologie et enfin l'accès à l'information elle-même.

Cette discussion sur l'éthique informatique nous laisse entrevoir que cette nouvelle discipline est à même d'informer utilement la conception des systèmes multi-utilisateurs, en particulier sous forme de principes. Notons que le champ d'application de l'éthique est vaste et concerne aussi d'autres domaines liés à l'étude des systèmes interactifs. [Mackay 1995] présente les problèmes éthiques liés à l'utilisation de la vidéo enregistrée, par exemple pour les tests d'utilisabilité. Comme pour la sociologie, la collaboration avec des spécialistes de l'éthique permet de guider la conception des systèmes multi-utilisateurs et

[1] Les auteurs anglais parlent de "have" et "have-not" de l'information ou encore d' "info-rich" et d' "info-poor". La langue française n'a pas encore de formule équivalente.

de mettre en garde les informaticiens contre des écueils dont ils ne sont pas toujours conscients.

2.4. Sciences de la Communication

Les sciences de la communication font aussi partie des sciences sociales. Elles peuvent apporter un éclairage indispensable à l'étude des systèmes multi-utilisateurs et de communication homme-homme médiatisée.

2.4.1. Les sciences de la communication

Les théories de la communication humaine et du langage, issues des travaux de Jakobson, Austin ou Searle, par exemple, ont déjà apporté des contributions importantes à l'étude des systèmes multi-utilisateurs. La notion de tour de parole, l'importance de la communication non-verbale ou du contact visuel sont des apports directs des sciences de la communication. Plus encore qu'avec la sociologie ou l'éthique, la diversité des approches au sein même des sciences de la communication en fait une discipline d'un abord difficile. Pourtant, c'est cette discipline des sciences sociales qui a été jusqu'à présent la plus utilisée pour l'étude des systèmes multi-utilisateurs. [McCarthy 1994] donne un exposé complet des différentes théories de la communication et de leurs mises en pratique dans les systèmes multi-utilisateurs. On peut citer par exemple l'approche de Flores et Winograd qui repose sur la théorie des actes de langage de Searle [Flores 1988]. Plusieurs travaux ont étudié les modifications qui surviennent dans la communication lorsque celle-ci a lieu à travers un artefact technologique comme un lien audio/vidéo. Dès 1978, un ouvrage de référence étudie l'influence sur la communication humaine de l'utilisation de moyens technologiques de communication [Hiltz 1978]. On peut citer [Egido 1988] pour un récapitulatif des limitations de la vidéoconférence. Egido constate la perte d'une partie de la communication non-verbale (par exemple le contact visuel), ou le fait qu'un utilisateur ne voit qu'une fraction de l'environnement de son interlocuteur et ne sait pas par exemple ce qu'il regarde. Le problème de la perte du contact visuel entre les interlocuteurs a suscité des solutions originales comme le dispositif Hydra [Buxton 1993] ou MAJIC [Okada 1994]. Nous reviendrons sur ces solutions techniques au chapitre 8. [Heath 1991] suggère que l'asymétrie de la communication audio/vidéo facilite certaines interactions sociales. D'autres travaux étudient la modification de la structure de la communication lorsqu'un lien audio/vidéo est utilisé. [Sellen 1992] étudie les temps de parole et les tours de parole et montre qu'un système comme Hydra ou un système d'incrustation d'images ne modifie pas sensiblement ces caractéristiques. Des expériences comme [Daly-Jones 1994] montrent qu'une communication audio/vidéo est plus utile qu'une communication uniquement audio pour le travail coopératif. Au chapitre 3, nous

étudions avec l'expérimentation Garden Movie la compréhension de gestes alliés à la parole dans le cadre de l'utilisation de la communication audio/vidéo.

2.4.2. L'étude des mass-médias

Une branche particulière des sciences de la communication s'intéresse aux mass-médias et peut aussi apporter des éléments de réflexion. Nous avons par exemple trouvé dans [McQuail 1987] un cadre d'analyse du rôle des mass-médias dans une société. Cette classification s'intéresse aux cas particuliers des mass-médias qui constituent un système de diffusion d'informations d'un émetteur vers une population réceptrice. Cependant, nous pensons que l'on peut extrapoler cette classification au cas de systèmes multi-utilisateurs concernant une large population comme la publication d'informations sur Internet, par exemple via World-Wide Web, les newsgroups ou les mailing-lists. McQuail distingue six théories de fonctionnement des mass-médias dans une société : la théorie autoritaire, la théorie de la presse libre, la théorie de la responsabilité sociale, la théorie des médias soviétiques, la théorie des médias pour le développement, la théorie de la participation démocratique. Nous résumons rapidement chacune de ces théories.

- Dans *la théorie autoritaire,* les médias sont sous le contrôle de l'autorité établie et s'interdisent tout ce qui pourrait affaiblir cette autorité. Les professionnels des médias sont subordonnés à l'organisation dans laquelle il travaillent, elle-même subordonnée au pouvoir en place.

- *La théorie de la presse libre* est en vigueur dans la majorité des régimes démocratiques. Chaque individu ou groupe est libre de publier toute information sans autorisation préalable ni censure. Il n'y a pas de restriction à l'importation ou à l'exportation d'information hors des frontières nationales. Ce modèle, conforme au fameux premier amendement de la constitution des États-Unis qui garantit la liberté d'expression[1], est à la base du fonctionnement d'Internet. La déclaration des Droits de l'Homme comporte un principe similaire.

- *La théorie de la responsabilité sociale* impose aux médias certaines obligations envers la société. Par exemple, les médias doivent se conformer à des exigences d'objectivité, de vérité et d'exactitude. Ces exigences sont déterminées par les médias eux-mêmes, considérés comme des organismes socialement responsables. Les journalistes sont personnellement responsables des informations qu'ils

[1] "Congress shall make no law respecting an establishment of religion, or prohibiting the free exercise thereof; or abridging the freedom of speech, or of the press; or the right of the people peaceably to assemble, and to petition the Government for a redress of grievances." (Amendment I to the Constitution of the United States)

diffusent. Certains organismes de presse dans nos sociétés respectent ces principes.

- *La théorie des médias soviétiques* est une variante de la théorie autoritaire qui prend en compte les contraintes de la doctrine marxiste-léniniste. Par exemple, les médias ne sont pas contrôlés par des intérêts privés, sont au service de et contrôlés par la classe ouvrière, etc.

- *La théorie des médias pour le développement* est appliquée dans de nombreux pays en voie de développement : les médias sont libres mais doivent se conformer aux exigences de l'autorité établie pour promouvoir des actions en faveur du développement de l'organisation ou du pays. Ils doivent aussi favoriser la culture et la langue nationales.

- *La théorie de la participation démocratique* peut se résumer en une formule : "la communication est trop importante pour être confiée aux professionnels". Cette théorie prône une décentralisation maximale des médias et la possibilité pour tout individu ou tout groupe de s'exprimer librement sur son propre média, dont l'influence est principalement locale. C'est sur un modèle de ce type que sont fondés les "freenets" et d'autres réseaux communautaires qui font partie d'Internet.

Ces théories du rôle des mass-médias dans une organisation sociale sont constituées d'un ensemble de principes. Même si leur domaine d'étude recouvre plutôt les organes de presse traditionnels, elles peuvent servir de cadre de réflexion pour l'étude de certains systèmes de communication homme-homme médiatisée.

2.5. Le Droit

A titre d'exemple, nous décrivons maintenant les apports potentiels d'une discipline des sciences sociales qui a encore été peu mise à contribution pour l'étude des systèmes multi-utilisateurs : le droit. Pour les systèmes multi-utilisateurs et surtout pour les services de communication largement accessibles comme Internet, les aspects juridiques doivent être considérés. Notons que d'autres sciences sociales et humaines sont potentiellement intéressantes pour notre domaine d'étude : on peut citer le journalisme, dont on pourrait étudier les règles de déontologie, la publicité, le marketing ou le management, qui ont tous étudié les comportements collectifs et organisationnels.

Le droit commence juste à prendre en compte les problèmes juridiques posés par les nouvelles technologies de communication. Pour cette discipline, une double collaboration est nécessaire : les concepteurs de systèmes informatiques doivent réfléchir au respect des lois existantes, mais ils peuvent aussi informer les juristes sur les possibilités des nouveaux systèmes qu'ils conçoivent et leur fournir matière à réfléchir sur les implications juridiques. L'exemple d'Internet et des réseaux internationaux pose plusieurs problèmes juridiques. Nous citerons par exemple la censure, le droit d'auteur, et le cryptage.

Comme nous l'avons vu plus haut, Internet repose sur la théorie de la presse libre. Dans cette théorie, la censure n'est pas acceptable. Cependant, un mécanisme comme la modération des newsgroups permet à un utilisateur privilégié de contrôler les messages avant leur diffusion. En règle générale, seuls sont refusés les messages hors sujet ou non conformes aux règles de bon usage d'Internet[1]. Récemment, un message publicitaire a été sciemment envoyé par deux avocats aux plusieurs milliers de newsgroups accessibles par Internet [Lewis 1994]. Cette première violation des règles d'usage d'Internet a été par la suite imitée et une "police" a été chargée de limiter la diffusion de ces messages abusifs. Le fait que le premier message abusif ait été le fait d'avocats n'est pas innocent. Ils ont exploité délibérément une des faiblesses du système : les règles de bon usage ne sont pas des lois et les enfreindre n'est pas légalement sanctionné. De ce fait, des questions juridiques se posent : un tel comportement est-il effectivement légal ? L'émetteur d'un message peut-il porter plainte (par exemple en vertu du 1er amendement de la constitution des États-Unis) si son message n'est pas publié ? La communauté Internet doit-elle se référer à un appareil légal existant ? Et dans ce cas, comment tenir compte du fait que le réseau est international, quelle juridiction adopter ? Les systèmes juridiques américain et français, par exemple, reposent sur des pratiques et même des postulats de base complètement différents.

Le droit d'auteur (ou "copyright") est aussi un problème sur lequel doit se pencher le droit. Les possibilités du multimédia permettent maintenant de dupliquer et de communiquer sans dégradation de qualité toute information, qu'il s'agisse de textes, de photographies, de musiques, ou même de films. Tous ces documents sont en général protégés par le droit d'auteur, au moins sous leur forme non-électronique. Mais pour leur forme électronique, le droit d'auteur tel que nous le connaissons devient caduc. [Barlow 1994] affirme que nous devons reconsidérer la définition même de la propriété intellectuelle. Barlow envisage un déplacement de la protection des auteurs depuis le

[1] La "netiquette" (contraction de Net etiquette), document largement accessible sur Internet, définit les règles de bon usage du réseau et de ses différents services.

domaine juridique vers le domaine de l'éthique et de la technologie, avec des outils comme le cryptage.

Le cryptage est depuis quelque temps l'objet d'une controverse : des algorithmes de cryptage à clé publique comme RSA permettent d'apporter des solutions technologiques partielles aux problèmes de respect de la vie privée et de protection du droit d'auteur. Mais le cryptage est considérée par beaucoup de gouvernements comme une arme de guerre : son usage est restreint et l'exportation de logiciels de cryptage est interdite. Dans ces conditions, elle n'est pas réellement utilisable, en particulier pour des communications internationales. Ici encore, une collaboration avec des juristes s'avère indispensable. Nous présentons la propriété d'authentification, qui peut reposer sur le cryptage, au chapitre 4. Nous revenons sur l'utilisation du cryptage au chapitre 8.

En résumé, les théories de la communication apportent des éléments utiles à l'analyse de la communication humaine pour la conception des systèmes multi-utilisateurs. D'autres sciences sociales, tel le droit, doivent aussi être considérées. Nous présentons maintenant les systèmes mediaspace et la façon dont ils prennent en compte les principes éthiques que nous avons exposés plus haut.

2.6. Application aux mediaspaces

Un mediaspace (littéralement "espace de médias" ou "espace médiatique") est un dispositif permettant à un groupe géographiquement dispersé de communiquer par l'intermédiaire de moyens audiovisuels et informatiques. Techniquement, un mediaspace repose sur un réseau audio/vidéo piloté par des moyens informatiques. Chaque utilisateur dispose d'une station de travail, ainsi que d'un moniteur vidéo, d'une caméra, de haut-parleurs et d'un microphone. A l'aide de logiciels adaptés, tout utilisateur peut établir une connexion audio/vidéo avec n'importe quel autre membre du groupe. Comme le remarque [Gaver 1995], un mediaspace comporte des différences importantes avec un vidéophone : un mediaspace n'est jamais "éteint" ou "raccroché" ; "on n'utilise pas un mediaspace, on vit dedans". Effectivement, le mediaspace semble créer un espace virtuel qui se superpose à l'espace physique. Ainsi les mediaspaces répondent à l'appel de [Hollan 1992] : les outils de communication homme-homme médiatisée ne doivent pas essayer de se substituer à la communication face à face, mais doivent proposer des possibilités originales qui n'ont pas d'équivalent dans la communication directe.

Depuis quelques années, plusieurs laboratoires ont développé des systèmes mediaspace à titre expérimental. Nous décrivons les services offerts par les principaux systèmes de ce type, en mettant en évidence leurs différences et leurs similitudes. Nous nous intéressons

en particulier à la façon dont ils respectent le principe éthique de protection de la vie privée. Nous décrivons aussi la façon dont nous avons tenu compte de ce principe dans notre propre mediaspace, VideoPort [Salber 1994].

2.6.1. Les services des mediaspaces

Même si l'on peut trouver quelques références antérieures, l'expérimentation des mediaspaces a vraiment commencé dans la deuxième partie des années 80 au Xerox PARC dans le but d'explorer les possibilités des systèmes de communication homme-homme médiatisée, et comme une aide au travail d'un groupe géographiquement dispersé. Le mediaspace du PARC [Stults 1986] visait à l'origine à faciliter la collaboration entre deux équipes de chercheurs du même laboratoire, mais situées dans deux villes différentes. Les parties communes publiques (cafétéria, salle de conférence) sont équipées de moyens de communication audiovisuels. Les utilisateurs peuvent établir une connexion audio/vidéo permanente entre les parties communes des deux sites. Lorsque, après quelques années, les deux sites furent rassemblés dans un même lieu géographique, le mediaspace fut cependant jugé utile et conservé.

Le mediaspace RAVE a été développé au Xerox EuroPARC en Grande-Bretagne, à la fois pour communiquer à l'intérieur du laboratoire et avec le mediaspace du PARC en Californie [Gaver 1992]. Plusieurs aspects de RAVE sont différents du système originel du PARC.

- Tous les membres d'EuroPARC (direction, chercheurs, personnel administratif) ont accès à RAVE et disposent donc de l'équipement nécessaire dans leur bureau. Cet aspect répond au principe éthique d'accès universel que nous avons vu précédemment.

- Le son est utilisé systématiquement pour donner à l'utilisateur des retours d'information sur les connexions qu'il reçoit ou sur certaines activités du groupe. L'utilisation du son est entièrement configurable par l'utilisateur qui peut, s'il préfère un environnement de travail plus silencieux, en désactiver certains voire tous. Nous verrons que ces mécanismes participent à la protection de la vie privée.

- Plusieurs types de connexions audio/vidéo sont disponibles, "glance" (jeter un coup d'œil), vidéophone, "office-share" (bureau partagé), et "background" (connexion en arrière-plan). Le "glance" est une connexion très brève, de quelques secondes, qui n'est pas interruptible. Ce type de connexion permet de "jeter un coup d'œil" dans le bureau d'un autre utilisateur ; celui-ci est prévenu

par des messages sonores non-verbaux du fait que quelqu'un est sur le point d'établir, puis a établi, une connexion avec lui et un message vocal annonce le nom du correspondant qui "jette un coup d'œil". Le vidéophone est une connexion réciproque entre deux utilisateurs ; l'un est appelant, et l'appelé doit donner son accord pour que la communication s'établisse ; la communication est interrompue à l'initiative de l'un quelconque des deux correspondants. L'"office-share" est une connexion réciproque de longue durée entre deux bureaux. Enfin la connexion "background" est une connexion permanente qui ne peut être établie qu'avec une partie commune publique, et uniquement vidéo (sans audio). Le mediaspace RAVE comporte aussi un service original, Portholes [Dourish 1992], qui permet à chaque membre du groupe d'être au courant de l'activité des autres membres du groupe. Ce service répond à la propriété d'"awareness" que nous présentons au chapitre 4. Le système prend des clichés de chaque participant à intervalles réguliers, et visualise sur l'écran de la station de travail de chacun les clichés pris. Là encore, un retour d'information sous forme audio non verbale est fourni à l'utilisateur à chaque fois qu'un cliché est pris (par exemple le son de l'obturateur d'un appareil photographique).

CAVECAT [Mantei 1991] est un mediaspace développé à l'Université de Toronto à partir de RAVE. Il offre des services analogues, mais intègre aussi des outils de travail coopératif (dessin partagé et édition de textes partagée). De plus, des connexions multiples entre utilisateurs sont possibles (analogues à la conversation téléphonique à plusieurs) : l'écran du moniteur de chaque correspondant montre tous les autres correspondants simultanément. Par exemple, pour une connexion à cinq personnes, l'écran de chacun est divisé en quatre images qui montrent les quatre autres correspondants.

Cruiser, développé et utilisé aux laboratoires Bellcore, est un mediaspace dont le but premier est différent [Fish 1992]. A l'origine, Cruiser est un moyen de stimuler les communications informelles au sein d'un groupe. Dans ce dessein, la métaphore sous-jacente est celle du couloir, lieu de rencontres fortuites. Toutes les connexions sont réciproques, et l'utilisateur peut demander au système de choisir un correspondant au hasard ; Cruiser comporte même un service où le système prend l'initiative des connexions entre deux utilisateurs choisis au hasard. Les types de connexion disponibles sont les suivants :

- Le "glance" est une connexion très brève avec un autre utilisateur choisi par l'appelant ou, à défaut, choisi au hasard par le système.

- Les “cruises” sont des connexions du type vidéophone qui doivent être confirmées par l’appelé pour être établies : là encore, l’utilisateur peut laisser au système le choix aléatoire du correspondant.

- Les “autocruises” sont des connexions effectuées à l’initiative du système qui choisit au hasard deux correspondants, mais qui ne sont établies que si les deux correspondants donnent leur accord.

L’originalité de Cruiser par rapport aux systèmes décrits ci-dessus provient de ce que toutes les connexions sont réciproques (même des connexions très brèves comme les “glances”), et du fait que des initiatives peuvent être laissées au système dans le choix des correspondants. Montage [Tang 1994] adopte comme Cruiser la métaphore du couloir mais propose des services légèrement différents. D’autres systèmes comme TeleCollaboration de US West Advanced Technologies ([Bulick 1989], cité dans [Bly 1993]) offrent des services analogues aux systèmes décrits ci-dessus.

2.6.2. La protection de la vie privée

De Charlie Chaplin dans *Les Temps Modernes*, poursuivi jusque dans les toilettes par l’image du directeur de l’usine, aux caméras de surveillance que l’on trouve couramment depuis quelques années en divers endroits publics ou privés, la vidéo dans un espace public ou de travail est étroitement associée à la notion de surveillance de l’individu par un pouvoir central. L’exemple le plus flagrant, et qui ressurgit régulièrement lorsque l’on évoque les mediaspaces, est le “Big Brother” de George Orwell dans son roman *1984*, tyran omniprésent qui peut surveiller en permanence tout individu par l’intermédiaire des “télécrans”. La société totalitaire décrite par Orwell porte la surveillance à son paroxysme puisque les individus sont surveillés partout, aussi bien dans les lieux publics que chez eux, et ne savent pas à quel moment ils sont surveillés. Voici comment Orwell décrit ce dispositif dans les premières pages de *1984* : “Le télécran recevait et transmettait simultanément. Il captait tous les sons émis par Winston [le héros du roman] au-dessus d’un chuchotement très bas. De plus, tant que Winston demeurait dans le champ de vision de la plaque de métal, il pouvait être vu aussi bien qu’entendu. Naturellement, il n’y avait pas moyen de savoir si, à un moment donné, on était surveillé.” [Orwell 1950]. Cette description inquiétante dégage ce qui rend le dispositif insupportable et dangereux. On ne sait pas si l’on est vu, ni par qui, ni quand, et il n’y a pas de réciprocité possible : on est vu mais l’on ne peut pas voir. A la lumière de notre présentation des principes éthiques, nous pouvons dériver du principe de protection de la vie privée pour les mediaspaces les trois propriétés suivantes :

- *Accès et réciprocité* : tout utilisateur qui peut voir doit pouvoir être vu et tout utilisateur qui peut être vu doit pouvoir voir ; on ne peut accepter par exemple qu'un utilisateur ait un moniteur mais pas de caméra. Cette propriété est aussi inspirée par le principe d'accès universel.

- *Retour d'information* à l'utilisateur qui est vu : par qui est-il vu et quand ? Cette propriété est une spécialisation de la propriété usuelle de retour d'information dans la conception des systèmes mono-utilisateurs. Mais au lieu de concerner uniquement l'état du système, elle s'applique ici à l'état du système en relation avec un autre utilisateur.

- *Contrôle* : chaque utilisateur doit pouvoir accepter ou refuser une connexion. Cette propriété est un cas particulier de la propriété d'initiative des systèmes mono-utilisateurs (l'utilisateur a l'initiative de l'action à effectuer, et non le système), mais elle est ici appliquée dans un contexte multi-utilisateur.

Nous allons maintenant voir comment ces trois propriétés sont vérifiées dans les exemples de mediaspace décrits plus haut.

2.6.2.1. Contrôle

Le mediaspace du PARC n'intègre pas de protection explicite de l'espace privé de l'utilisateur. Toute demande de connexion aboutit et résulte en l'établissement de la connexion demandée. La propriété de contrôle n'est donc pas vérifiée. Toutefois les utilisateurs peuvent recourir à une protection "artisanale" consistant à masquer l'objectif de leur caméra et à débrancher leur microphone ; cette pratique n'interdit pas les connexions, mais les rend ineffectives, sans toutefois indiquer à l'appelant la raison de l'absence d'image sur son écran. Dans RAVE et CAVECAT, l'utilisateur gère librement une liste de droits d'accès et peut autoriser ou interdire à chacun des autres utilisateurs du système tel ou tel type de connexion. La propriété de contrôle est ici vérifiée. Mais CAVECAT comporte une caractéristique originale : il y a en fait dans ce système deux niveaux de contrôle. D'une part, l'utilisateur peut fixer une liste de droits d'accès comme dans RAVE. Cette liste est ensuite utilisée par le système pour accepter ou refuser les appels. Mais CAVECAT permet aussi à chaque utilisateur de fixer son degré d'accessibilité. La métaphore des portes est utilisée à cet effet : la porte de chaque utilisateur peut être ouverte, entrouverte, fermée ou condamnée. Mais cet aspect du contrôle est uniquement social. Le système n'a pas connaissance de ce degré d'accessibilité et ne peut s'en servir pour accepter ou refuser les appels. Le système

n'interdit pas de se connecter à un utilisateur dont la porte est fermée[1]. Dans RAVE, l'appelant n'a pas connaissance de ce degré d'accessibilité : si le correspondant n'accepte pas la connexion, la station du correspondant renvoie un message qui a été défini par le correspondant indisponible. Cette différence dans l'approche d'un même problème est révélatrice de la différence de philosophie entre ces deux mediaspaces. CAVECAT privilégie la coordination : l'utilisateur est conduit à ne même pas essayer d'établir une connexion qui n'aboutira pas. En revanche, RAVE privilégie la communication : l'utilisateur peut établir une connexion infructueuse mais elle lui apportera une information explicative. Dans le système Cruiser, un utilisateur peut se rendre temporairement indisponible ; dans ce cas l'appelant verra l'image du correspondant partiellement cachée par des barres noires horizontales superposées à l'image. Ces barres noires, qui rappellent un store vénitien, présentent ainsi une analogie forte avec un objet et une pratique du monde réel. Cependant, cette solution ne garantit pas une protection absolue de la vie privée ; en effet, l'image de l'appelé est quand même visible à travers les barres qui ne le masquent que partiellement. Le principe de protection de la vie privée n'est pas entièrement respecté. Mais cette approche est conforme à l'objectif de Cruiser qui est de stimuler la communication informelle dans un groupe de travail. On distingue ainsi à travers les trois exemples de CAVECAT, RAVE et Cruiser trois niveaux différents de protection de l'espace privé et les solutions proposées par ces trois systèmes sont révélatrices des objectifs qu'ils privilégient. L'approche que nous avons choisie pour notre mediaspace VideoPort[2] est différente : toute connexion est négociée "au vol". Lorsqu'il reçoit un appel, un utilisateur se voit systématiquement présenter une boîte de dialogue identifiant le correspondant et lui demandant s'il accepte ou non la connexion. Cette solution a toutefois un inconvénient : le système est intrusif, et la propriété de non-préemption du chapitre 4 n'est pas respectée. Nous examinons maintenant comment est vérifiée la propriété de retour d'information dans les systèmes mediaspace.

2.6.2.2. Retour d'information

En ce qui concerne le retour d'information, Cruiser apporte certainement la solution la plus radicale : dans ce système, toute connexion est réciproque, c'est-à-dire systématiquement bidirectionnelle. De ce fait l'utilisateur appelé dispose d'un retour d'information immédiat dès qu'une connexion est établie, même s'il s'agit d'une connexion de courte durée de type "glance". Dans RAVE ou CAVECAT, les connexions de type "glance" ne sont pas réciproques mais unidirectionnelles. Cependant, l'utilisateur

---

[1] Notre description de CAVECAT s'appuie sur des informations recueillies lors d'une discussion avec un utilisateur de CAVECAT en avril 1994 et a été contredite depuis par un autre utilisateur du même système. Il semble que CAVECAT ait connu plusieurs évolutions et que les caractéristiques décrites ici aient été modifiées dans le système actuel.

[2] VideoPort est présenté plus en détail au chapitre 7.

appelé dispose d'un retour d'information sous forme audio qu'il peut configurer. En règle générale, le retour d'information consiste en un bruit de porte qui s'ouvre ou du "toc-toc" de quelqu'un frappant à une porte, suivi du nom de l'appelant puis d'un bruit de porte qui se ferme. Ce type de retour d'information renforce la métaphore du monde réel sur laquelle repose une connexion de type "glance" : frapper à la porte d'un bureau et entrouvrir la porte pour voir si la personne est disponible. Dans notre mediaspace VideoPort, nous avons choisi de donner à l'utilisateur le choix de configurer le son qui accompagne une demande de connexion : porte en bois, porte métallique, sonnerie de téléphone, obturateur d'appareil photographique, etc. Cette configurabilité (propriété définie au chapitre 4) vise à laisser à l'utilisateur le choix de sa propre métaphore d'utilisation du mediaspace. Les connexions "glance" sont unidirectionnelles, mais le retour d'information fourni à l'appelé permet de décourager toute utilisation mal intentionnée du système : personne ne se risquera à établir des "glances" à répétition dans un but de surveillance sachant que l'utilisateur appelé saura non seulement qu'une connexion est établie, mais aussi par qui elle est établie. La propriété de retour d'information à l'utilisateur appelé participe donc au respect des règles de bon usage du système ; elle prend toute son importance lorsque la connexion est unidirectionnelle.

2.6.2.3. Accès

La propriété d'accès au système mediaspace est étroitement liée au souci de garantir la protection de l'espace privé. En effet, à l'exception de RAVE, tous les mediaspaces ont pour utilisateurs des groupes de chercheurs qui se connaissent bien et collaborent étroitement. Dans RAVE en revanche, tout le personnel d'EuroPARC a accès au système, aussi bien les chercheurs que le personnel administratif ou les dirigeants du laboratoire. Comme le remarque [Gaver 1992], le succès d'un mediaspace tel que RAVE repose sur l'existence d'une relation de confiance mutuelle entre les utilisateurs. [Harper 1992] dans une étude sociologique explique que les laboratoires de recherche sur les technologies de l'information, où ont lieu l'essentiel des expérimentations mediaspace, sont des endroits où les relations entre collaborateurs sont particulières. Même si des travaux récents ont montré l'intérêt des mediaspaces dans "le monde réel" [Pagani 1993], on peut se demander si les mediaspaces sont transposables par exemple dans une entreprise, où les relations hiérarchiques sont souvent plus marquées. Des études comme [Heath 1992] sont utiles pour comprendre les spécificités de ces environnements de communication homme-homme médiatisée et les nouveaux comportements et les nouveaux types de relations qui s'y développent.

En résumé, nous avons vu qu'un principe issu de l'éthique informatique, affiné sous forme de propriétés dont nous avons examinée la vérification, nous a permis d'analyser dans le détail les mediaspaces existants, leurs similarités et leurs différences.

2.7. Conclusion

Dans ce chapitre, nous avons vu comment les diverses disciplines des sciences humaines peuvent participer à l'étude de la conception des systèmes multi-utilisateurs. Les principes qu'elles génèrent ou que l'on peut déduire de leurs études sont des guides utiles pour comprendre le fonctionnement d'un groupe, et permettent aussi d'éviter des erreurs communes dans la conception. Nous verrons au chapitre suivant comment une discipline des sciences humaines comme la psychologie cognitive peut également générer des principes. Nous avons aussi présenté dans ce chapitre les systèmes mediaspaces et leur intérêt pour la communication homme-homme médiatisée. En appliquant un principe de l'éthique informatique, nous avons pu analyser en détail les systèmes mediaspace et les comparer à VideoPort, notre propre contribution.

Références

[Ballé 1992] C. Ballé. *Sociologie des organisations,* Presses Universitaires de France, Paris, France, 1992.

[Barlow 1994] J. P. Barlow. *The Economy of Ideas,* in *Wired*, 2(03), mars 1994.

[Bentley 1992] R. Bentley, J. A. Hughes, D. Randall, T. Rodden, P. Sawyer, D. Shapiro et I. Sommerville. *Ethnographically-informed systems design for air traffic control*, CSCW'92, ACM Conference on Computer-Supported Cooperative Work, Toronto, Canada, 1992. pp. 123-129.

[Bernoux 1985] P. Bernoux. *La sociologie des organisations,* Editions du Seuil, Paris, France, 1985.

[Bly 1993] S. A. Bly, S. R. Harrison et S. Irwin. *Media Spaces: Bringing People Together in a Video, Audio, and Computing Environment,* in *Communications of the ACM,* 36(1), janvier 1993.

[Bulick 1989] S. Bulick, M. Abel, D. Corey, J. Schmidt et S. Coffin. *The US WEST Advanced Technologies Prototype Multi-media Communications System*, GLOBECOM'89, IEEE Global Telecommunications Conference, Dallas, Texas, USA, 1989.

[Buxton 1993] W. A. S. Buxton. *Telepresence: Integrated Shared Task and Person Spaces,* in *Readings in Groupware and Computer-Supported Work.* R. M. Baecker, (ed.) Morgan Kaufman, San Mateo, California, USA, 1993. pp. 816-822.

[Coiffier 1990] E. Coiffier, Y. Crozet, D. Dehoux-Grafmeyer, F. Faure et J.-F. Renaud. *Sociologie basique,* Nathan, Paris, France, 1990.

[Daly-Jones 1994] O. Daly-Jones. *Characterising the Social Salience of Electronically Mediated Communication,* CHI'94, ACM Conference on Human Factors in Computing Systems, Boston, Massachussets, USA, 1994. pp. 93-94.

[Dourish 1992] P. Dourish et S. A. Bly. *Portholes: Supporting Awareness in a Distributed Work Group,* CHI'92, ACM Conference on Human Factors in Computing Systems, Monterey, California, USA, 1992. pp. 541-547.

[Egido 1988] C. Egido. *Videoconferencing as a Technology to Support Group Work: A Review of its Failures,* CSCW'88, ACM Conference on Computer-Supported Cooperative Work, 1988. pp. 13-24.

[Fish 1992] R. S. Fish, R. E. Kraut, R. W. Root et R. E. Rice. *Evaluating Video as a Technology for Informal Communications*, CHI'92, ACM Conference on Human Factors in Computing Systems, Monterey, California, USA, 1992. pp. 32-48.

[Flores 1988] F. Flores, M. Graves, B. Hartfield et T. Winograd. *Computer Systems and the Design of Organizational Interaction,* in *ACM Transactions on Office Information Systems*, 6(2), avril 1988. pp. 153-172.

[France 1995] E. France. *UK Data Protection Registrar, Keynote speech*, Ethicomp'95, An International Conference on the Ethical Issues of Using Information Technology, Leicester, UK, 1995.

[Gaver 1992] W. W. Gaver, T. Moran, A. MacLean, L. Lövstrand, P. Dourish, K. Carter et W. Buxton. *Realizing a Video Environment: EuroPARC's RAVE System*, CHI'92, ACM Conference on Human Factors in Computing Systems, Monterey, California, USA, 1992. pp. 27-36.

[Gaver 1995] W. W. Gaver, G. Smets et K. Overbeeke. *A Virtual Window on Media Space*, CHI'95, ACM Conference on Human Factors in Computing Systems, Denver, Colorado, USA, 1995. pp. 257-264.

[Harper 1992] R. Harper. *Looking at Ourselves: An Examination of the Social Organization of Two Research Laboratories*, CSCW'92, ACM Conference on Computer-Supported Cooperative Work, Toronto, Canada, 1992. pp. 330-337.

[Harrison 1994] B. Harrison, M. Mantei, G. Beirne et T. Narine. *Communicating About Communicating: Cross-Disciplinary Design of a Media Space Interface*, CHI'94, ACM Conference on Human Factors in Computing Systems, Boston, Massachussets, USA, 1994. pp. 124-130.

[Heath 1991] C. Heath et P. Luff. *Collaborative Activity and Technological Design: Task Coordination in the London Underground Control Rooms*, ECSCW'91, European Conference on Computer-Supported Cooperative Work, Amsterdam, Pays-Bas, 1991.

[Heath 1992] C. Heath et P. Luff. *Media Space and Communicative Asymmetries: Preliminary Observations of Video-Mediated Interaction*, in *Human-Computer Interaction*, 7(3), 1992. pp. 315-246.

[Hiltz 1978] S. R. Hiltz et M. Turoff. *The Network Nation*, MIT Press, Cambridge, Massachussets, USA, 1978.

[Hollan 1992] J. Hollan et S. Stornetta. *Beyond Being There*, CHI'92, ACM Conference on Human Factors in Computing Systems, Monterey, California, USA, 1992. pp. 119-125.

[Johnson 1994] D. G. Johnson. *Computer Ethics*, Prentice Hall, 1994.

[Lewis 1994] P. H. Lewis. *An Ad (Gasp!) in Cyberspace*, in *The New York Times*, 1994(19/4), avril 1994. pp. D1:2.

[Mackay 1995] W. E. Mackay. *Ethics, Lies and Videotape…*, CHI'95, ACM Conference on Human Factors in Computing Systems, Denver, Colorado, USA, 1995. pp. 138-145.

[Maner 1995] W. Maner. *Unique Ethical Problems in Information Technology*, Ethicomp'95, An International Conference on the Ethical Issues of Using Information Technology, Leicester, UK, 1995.

[Mantei 1989] M. Mantei. *Observation of Executives Using a Computer Supported Meeting Environment*, in *Readings in Groupware and Computer-Supported Cooperative Work*. R. M. Baecker, (ed.) Morgan Kaufman, San Mateo, California, US, 1989. pp. 695-708.

[Mantei 1991] M. Mantei, R. M. Baecker, A. Sellen, W. Buxton, T. Milligan et B. Wellman. *Experiences in the use of a Media Space*, CHI'91, ACM Conference on Human Factors in Computing Systems, New Orleans, Louisiana, USA, 1991. pp. 203-208.

[Mason 1986] R. O. Mason. *Four Ethical Issues of the Information Age*, in *MIS Quarterly*, 10(1), janvier 1986. pp. 486-498.

[McCarthy 1994] J. C. McCarthy et A. F. Monk. *Channels, conversation, cooperation and relevance: all you wanted to know about communication but were afraid to ask,* in *Collaborative Computing*, 1(1), mars 1994. pp. 35-60.

[McQuail 1987] D. McQuail. *Mass Communication Theory, An Introduction,* SAGE Publications, London, UK, 1987.

[Moor 1985] J. H. Moor. *What is Computer Ethics ?,* in *Metaphilosophy*, 16(4), octobre 1985. pp. 226-275.

[Neumann 1995] P. G. Neumann. *Computer-related Risks,* Addison-Wesley, 1995.

[Okada 1994] K.-i. Okada, F. Maeda, Y. Ichikawaa et Y. Matsushita. *Multiparty Videoconferencing at Virtual Social Distance: MAJIC Design*, CSCW'94, ACM Conference on Computer-Supported Cooperative Work, Chapel Hill, North Carolina, USA, 1994. pp. 385-393.

[Orwell 1950] G. Orwell. *1984,* Gallimard, Paris, France, 1950.

[Pagani 1993] D. Pagani et W. E. Mackay. *Bringing Media Spaces into the Real World*, ECSCW'93, European Conference on Computer-Supported Cooperative Work, Milan, Italie, 1993.

[Platt 1995] R. G. Platt et B. Morrison. *Ethical and Social Implications of the Internet,* Ethicomp'95, An International Conference on the Ethical Issues of Using Information Technology, Leicester, UK, 1995.

[Poltrock 1995] S. Poltrock et J. Grudin. *Groupware and Workflow: A Survey of Systems and Behavioral Issues*, Tutorial at CHI'95, ACM Conference on Human Factors in Computing Systems, Denver, Colorado, USA, 1995.

[Rouncefield 1994] M. Rouncefield, J. A. Hughes, T. Rodden et S. Viller. *Working with "Constant Interruption": CSCW and the Small Office*, CSCW 94, ACM Conference on Computer-Supported Cooperative Work, Chapel Hill, North Carolina, USA, 1994. pp. 275-286.

[Salber 1994] D. Salber et J. Coutaz. *Fenêtres sur groupes: des MediaSpaces pour collaborer et communiquer*, 3èmes Journées Internationales sur L'Interface des Mondes Réels et Virtuels '94, Montpellier, France, 1994. pp. 309-318.

[Schneiderman 1995] B. Schneiderman. *The Info Superhighway: For The People,* in *Communications of the ACM*, 38(1), janvier 1995. pp. 162.

[Sellen 1992] A. J. Sellen. *Speech Patterns in Video-Mediated Conversations*, CHI'92, ACM Conference on Human Factors in Computing Systems, Monterey, California, USA, 1992. pp. 49-59.

[Stults 1986] R. Stults. *MediaSpace*, Xerox PARC, Rapport technique, 1986.

[Suchman 1987] L. Suchman. *Plans and Situated Actions,* Cambridge University Press, Cambridge, 1987.

[Tang 1994] J. C. Tang et M. Rua. *Montage: Providing Teleproximity for Distributed Groups*, CHI'94, ACM Conference on Human Factors in Computing Systems, Boston, Massachussets, USA, 1994. pp. 37-43.

[Weber 1947] M. Weber. *The Theory of Social and Economic Organization,* Oxford University Press, New York, New York, USA, 1947.

[Wylie 1970] L. Wylie. *Chanzeaux, village d'Anjou*, Gallimard, Paris, France, 1970.

Apport de la Psychologie Cognitive

Minds are simply computers made of meat

Marvin Minsky

Apport de la Psychologie Cognitive

3.1. Introduction

L'intérêt de la psychologie cognitive, à la fois théorique et expérimentale, pour la conception des interfaces homme-machine est maintenant reconnu. Le travail séminal de Card, Moran et Newell [Card 1983] a ouvert la voie à de nombreuses collaborations fructueuses entre psychologues et informaticiens. La psychologie cognitive vise à comprendre et modéliser les mécanismes cognitifs mis en jeu dans les activités humaines. Les activités étudiées par la psychologie cognitive comprennent par exemple la planification et l'exécution d'un plan, l'apprentissage, la mémorisation, la perception de notre environnement.

Nous présentons dans ce chapitre le modèle cognitif Interacting Cognitive Subsystems (ICS) [Barnard 1985], qui offre l'intérêt de prendre en compte les aspects cognitifs de l'interaction d'un utilisateur avec des systèmes informatiques avancés comme les systèmes multimédias ou multimodaux[1]. ICS apporte des éléments précieux pour la conception d'interfaces de communication homme-homme médiatisée en offrant un support théorique pour l'analyse de la perception et la compréhension humaines de différentes sources d'information simultanées, comme par exemple son et image ou bien son et texte. Nous présentons ensuite l'expérimentation "Garden Movie" que nous avons menée en collaboration avec Jon May et Phil Barnard au MRC-Applied Psychology Unit à Cambridge (Grande-Bretagne). Cette expérience nous a permis d'étudier la collaboration entre deux utilisateurs disposant de moyens de communication audio/vidéo et s'est révélée riche d'enseignements pour la conception des systèmes de communication homme-homme médiatisée.

3.2. Sciences Cognitives et Psychologie Cognitive

Il nous semble opportun de rappeler les hypothèses traditionnelles sur lesquelles repose l'ensemble des sciences cognitives. Selon [Andler 1992], le paradigme classique ou "cognitivisme" propose les trois hypothèses fondamentales suivantes :

1. *Le complexe esprit/cerveau est susceptible d'une double description, matérielle ou physique au sens large (la physique intervenant en réalité par le biais des neurosciences), et informationnelle ou fonctionnelle ; ces deux niveaux sont largement indépendants, et le rapport qui s'établit entre eux est à l'image de celui*

[1] Pour une discussion détaillée des mots multimodal et multimédia, voir [Nigay 1994].

qui lie un ordinateur en tant que système physique à la description du même appareil en tant que système de traitement de l'information.

2. *Au niveau informationnel, le système cognitif de l'homme (…) est caractérisé par ses* états *internes ou mentaux et par les* processus *qui conduisent d'un état au suivant. Ces états sont* représentationnels *: ils sont dotés d'un contenu renvoyant à des entités externes (on dit aussi qu'ils sont* sémantiquement évaluables*).*

3. *Les états ou* représentations *internes sont des formules d'un langage interne (ou "mentalais") proche des langages formels de la logique. (…)*

Notons tout d'abord que ces trois postulats, fondamentaux pour les sciences cognitives, se veulent universels : ils sont applicables à tout être humain quelle que soit l'activité humaine considérée. Ils sont donc pertinents pour l'étude de l'interaction homme-machine.

On pourra regretter l'analogie faite dans le premier postulat avec l'ordinateur : en effet, l'ordinateur est un outil créé par l'homme et même s'il s'agit probablement de l'outil le plus complexe imaginé à ce jour, sa complexité et ses capacités sont dans l'ensemble bien inférieures à celle de notre cerveau. Cette analogie a cependant le mérite de clarifier les positions de l'approche "matérialiste" qui étudie les mécanismes biologiques et, à l'opposé, de l'approche qui considère le cerveau comme une "boîte noire" et l'étudie de façon externe.

L'approche cognitiviste est contestée en particulier par le mouvement connexionniste issu de l'Intelligence Artificielle. [Smolensky 1992] distingue le connexionnisme de l'approche reposant sur le "paradigme symbolique". Le paradigme symbolique affirme pouvoir formaliser les structures mentales sans connaître la relation que les structures neuronales sous-jacentes entretiennent avec les structures mentales. Selon Smolensky, les modèles reposant sur le paradigme symbolique ne permettent qu'une approximation grossière des comportements cognitifs. En revanche, l'approche connexionniste vise à décrire exactement le comportement cognitif directement à partir des structures neuronales, sans recourir aux structures mentales (qui incluent les notions de buts, connaissances, perceptions, actions, etc., sur lesquelles reposent largement les théories psychologiques appliquées jusqu'à présent à l'interaction homme-machine). Une approche intermédiaire que propose Smolensky repose sur le "paradigme subsymbolique" qui tente d'établir un pont entre les deux approches.

Pour résumer l'opposition entre les écoles cognitiviste et connexionniste et pour clarifier notre position, nous serions tentés de reprendre la formule provocatrice de Marvin Minsky citée en exergue de ce chapitre en la modifiant : "L'esprit humain *peut être modélisé* comme un ordinateur".

Le modèle psychologique ICS que nous présentons ci-dessous repose, comme toutes les modélisations utilisées classiquement pour l'étude de l'interaction homme-machine, sur le paradigme symbolique et cognitiviste classique. Cependant, nous sommes conscients des possibles approximations que contient un tel modèle, approximations sur lesquelles insistent d'ailleurs ses auteurs. Malgré cette limitation, le modèle ICS permet de prédire et d'expliquer des comportements cognitifs et apporte une contribution déterminante au domaine de l'interaction homme-machine.

3.3. ICS, un modèle cognitif

Un des premiers modèles issu de la psychologie cognitive et qui a été appliqué à l'étude de l'interaction homme-machine est le célèbre modèle du processeur humain [Card 1983]. Ce modèle décompose le système cognitif humain en un ensemble de trois processeurs spécialisés (perceptif, moteur et cognitif) et de mémoires (de travail et à long terme). Comme le reconnaissent eux-mêmes ses auteurs, ce modèle est très simplifié, en particulier en ce qui concerne le système perceptif. Cette simplicité, suffisante pour des interfaces graphiques traditionnelles, est une limitation gênante pour l'étude des interfaces multimodales ou multimédia. En effet, en interagissant avec un système multimodal ou multimédia, l'utilisateur doit percevoir et interpréter des informations non seulement visuelles, mais aussi auditives, tactiles, etc. Le système perceptif monolithique tel qu'il est décrit dans le modèle du processeur humain ne permet pas d'étudier la façon dont, par exemple, nous intégrons des informations visuelles et auditives et parvenons à fusionner les deux sources d'information et à les interpréter comme un ensemble cohérent. Ce cas de figure correspond par exemple, outre l'utilisation de nos sens dans la vie quotidienne, au spectateur de cinéma, mais aussi à l'utilisateur d'un système de communication multimédia ou d'une interface multimodale.

Le modèle Interacting Cognitive Subsystems (ICS) [Barnard 1985] peut dans un premier temps être vu comme un affinement du modèle du processeur humain. Mais ces deux modèles ont des différences fondamentales : en particulier, ICS repose sur une architecture parallèle multi-processus, et non sur un processeur cognitif central ; il n'y a pas dans ICS de mémoire de travail centralisée et banalisée, mais un ensemble de mémoires locales ; d'autre part, ICS ne vise pas à expliquer précisément la nature de l'information traitée par le système cognitif humain, ni les mécanismes précis du

traitement de l'information, mais il tente de construire des modèles approchés des opérations cognitives en s'intéressant particulièrement à l'utilisation des ressources cognitives. Enfin, l'intérêt majeur d'ICS pour notre domaine d'étude est dû au fait qu'il s'intéresse plus aux phénomènes sensoriels qu'au raisonnement, à la différence de beaucoup de modèles cognitifs. Cet aspect d'ICS le rend plus adapté que le modèle du processeur humain à l'étude des interfaces multimodales et multimédia.

3.3.1. Présentation générale du modèle

Le modèle Interacting Cognitive Subsystems (ICS) structure le système de traitement de l'information humain en un ensemble de neuf sous-systèmes (figure 3.1).

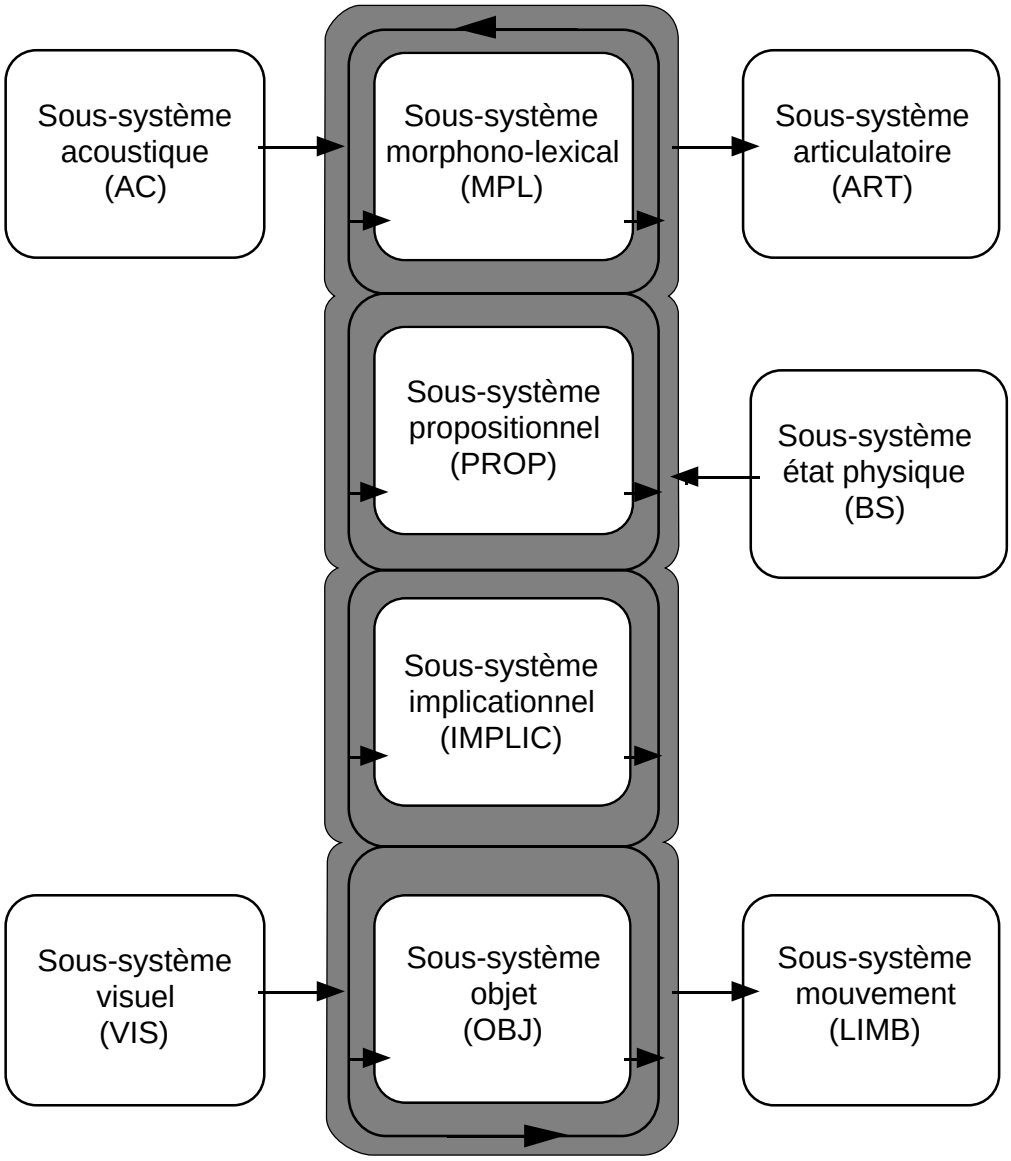


Figure 3.1. Architecture générale du modèle ICS. La partie grisée représente le réseau permettant aux sous-systèmes de communiquer.

Les sous-systèmes perceptifs (visuel, acoustique, état physique) acquièrent l'information en provenance de l'environnement ou du corps. Les sous-systèmes centraux (objet, morphono-lexical, propositionnel, implicationnel) réalisent des transformations de l'information acquise. Ces sous-systèmes sont au cœur des processus d'interprétation et de compréhension, ainsi que des mécanismes du raisonnement. Les sous-systèmes moteurs (articulatoire, mouvement) contrôlent les actions physiques. Tous ces sous-systèmes fonctionnent en parallèle.

Les sous-systèmes centraux sont reliés entre eux par un réseau grâce auquel ils peuvent communiquer. Les sous-systèmes perceptifs peuvent uniquement transférer des informations vers ce réseau et les sous-systèmes moteurs ne peuvent que recevoir des informations en provenance de ce réseau.

Les neuf sous-systèmes qui constituent ICS ont tous la même structure. Nous décrivons d'abord la structure d'un sous-système, puis nous détaillons le rôle de chacun des sous-systèmes et ses interactions possibles avec les autres sous-systèmes.

3.3.2. Description d'un sous-système d'ICS

Un sous-système du modèle ICS est constitué d'un ensemble d'entrées et de sorties, d'une capacité de traitement de l'information, et d'une mémoire locale (figure 3.2).

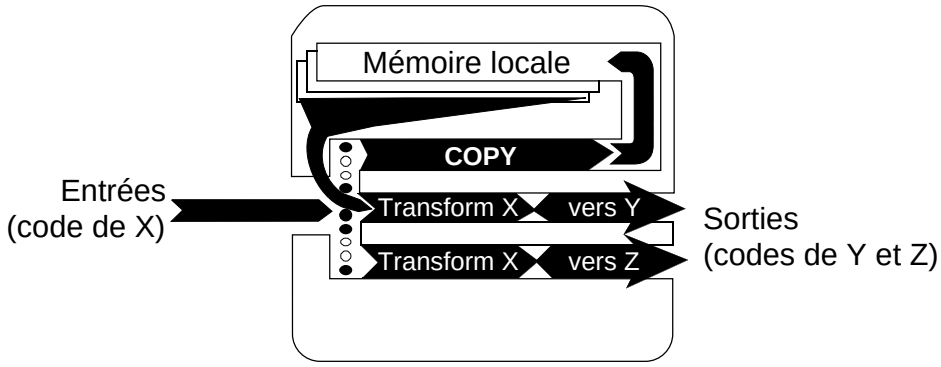


Figure 3.2. Un sous-système X. Le sous-système reçoit en entrée des représentations dans son code et fournit en sortie des représentations dans le code d'autres sous-systèmes (ici Y et Z).

Chaque sous-système dispose d'un code de représentation de l'information qui lui est propre : il ne peut traiter que des informations arrivant à ses entrées qui sont représentées dans son propre code. Il peut en revanche générer des représentations dans le code d'autres sous-systèmes. Par exemple, le sous-système objet acquiert les représentations en provenance du sous-système visuel (ces représentations sont donc exprimées dans le

code du sous-système objet) et les transforme par exemple en représentations exprimées dans le code du sous-système propositionnel.

La capacité de traitement de l'information d'un sous-système est représentée par un ensemble de processus qui opèrent sur les représentations manipulées par le sous-système. Ces processus sont de deux sortes : le processus de recopie locale (COPY) copie systématiquement toute représentation arrivant en entrée du sous-système vers la mémoire locale. Les processus de transformation (TRANSFORM) reçoivent en entrée soit les représentations présentes aux entrées du sous-système, soit des représentations provenant de la mémoire locale. Ces processus transforment les représentations et fournissent en sortie des représentations exprimées dans le code d'autres sous-systèmes. A un instant donné, un seul processus de transformation peut-être actif. Deux processus TRANSFORM d'un même sous-système ne peuvent pas fonctionner en parallèle. En revanche, COPY peut fonctionner en même temps qu'un processus TRANSFORM. Ce mode de fonctionnement montre que les capacités de traitement de l'information d'un sous-système sont limitées et peuvent donc être sujettes à surcharge.

La mémoire locale peut contenir quatre types de représentations :

- *Active Task Record* (ATR) : une représentation que le sous-système vient de traiter et qui est disponible pour une réutilisation immédiate,

- *Experiential Task Record* (ETR) : une représentation qui a été rencontrée dans le passé et qui peut être rappelée si le sous-système reçoit une représentation analogue,

- *Common Task Record* (CTR) : abstraction regroupant plusieurs représentations déjà rencontrées (ETR) et constituée de leur rappel simultané ; les similarités entre les représentations sont combinées et les différences gommées,

- *Entity Property Record* (EPR) : partie d'une représentation déjà rencontrée, élaborée à partir des deux catégories précédentes (CTR et ETR), et qui peut servir à compléter une représentation partielle qui se présente à l'entrée du sous-système.

La mémoire locale d'un sous-système est à rapprocher de la mémoire de travail décrite dans le modèle du processeur humain : le nombre de représentations qu'elle peut contenir est réduit, et la mémoire locale est alimentée par les entrées du sous-système (sous forme d'ATRs) et par la mémoire à long terme (ETRs, CTRs, et EPRs). Le modèle du processeur humain fait aussi intervenir des mémoires très fugitives liées au système

perceptif (*auditory image store* et *visual image store*) qui transfèrent immédiatement leur contenu à la mémoire de travail. Dans ICS, ces mémoires fugitives sont considérées comme faisant partie intégrante de la mémoire de travail (ce sont des ATRs des sous-systèmes acoustique et visuel) et la mémoire de travail est ici distribuée dans l'ensemble des sous-systèmes. La distinction entre les différents types de représentation manipulées par la mémoire locale d'un sous-système d'ICS, et en particulier les CTRs et EPRs permet de mettre en évidence la nature associative de la mémoire humaine. ICS montre aussi que les informations stockées en mémoire à long terme sont représentées dans le code du sous-système concerné et ne peuvent donc être rappelées que par ce sous-système. Enfin les EPRs montrent qu'une représentation partielle présentée à un sous-système peut être complétée grâce à l'expérience accumulée par ce sous-système.

Les neuf sous-systèmes du modèle ICS fonctionnent tous comme décrit ci-dessus. Mais chaque sous-système est spécialisé dans le traitement d'une partie de l'activité cognitive. Nous décrivons ci-dessous le rôle joué par chacun d'entre eux.

3.3.3. Les sous-systèmes perceptifs

Les sous-systèmes perceptifs sont les sous-systèmes visuel (VIS), acoustique (AC) et état physique (BS pour *body state*). Chacun de ces sous-systèmes reçoit de l'information du monde physique par nos récepteurs sensoriels et transforme ces représentations de l'information à l'intention des sous-systèmes centraux.

Plus précisément, le sous-système visuel transforme les informations vues (par exemple, couleur, contour, luminosité) en représentations à l'intention du sous-système objet (OBJ). Ce processus de transformation, que l'on note VIS $\Rightarrow$ OBJ, fournit en sortie du sous-système VIS des informations exprimées dans le code de OBJ (par exemple, formes des objets vus).

Le sous-système acoustique opère sur les informations auditives (par exemple, hauteur des sons, rythme, timbre). Si les sons entendus sont par exemple des paroles, le sous-système acoustique va fournir en sortie des représentations pour le sous-système morphono-lexical (MPL) par la transformation AC $\Rightarrow$ MPL.

De façon similaire, le sous-système BS va acquérir des informations en provenance du corps (par exemple, retour d'information tactile, température, texture) et transformer ces informations à l'intention des sous-systèmes centraux.

En règle générale, chacun des sous-systèmes perceptifs a un partenaire privilégié parmi les sous-systèmes centraux, comme OBJ pour VIS ou MPL pour AC. Les transformations correspondantes abstraient l'information du niveau perceptif vers un niveau intermédiaire qui permettra ensuite l'interprétation des informations perçues.

3.3.4. Les sous-systèmes centraux

Quatre sous-systèmes forment les sous-systèmes centraux : les sous-systèmes propositionnel (PROP) et implicationnel (IMPLIC) qui sont parfois appelés *moteur central de la cognition* et les sous-systèmes objet (OBJ) et morphono-lexical (MPL) dont nous avons vu au paragraphe précédent qu'un de leurs rôles est de fournir une représentation abstraite des perceptions.

Le sous-système objet (OBJ) peut être vu comme ayant la charge de "l'imagerie mentale". C'est là par exemple que se forment les images mentales qui ne sont pas directement perçues mais qui sont générées par le système cognitif. Ce sous-système va en général, à partir des représentations fournies par VIS à ses entrées, élaborer des propositions, c'est-à-dire des relations sémantiques entre entités, à l'intention du sous-système PROP. Cette transformation correspond à l'identification d'un objet et à l'établissement de relations avec les objets qui l'entourent (par exemple, l'objet vu est un stylo et il est posé sur une table). Un cas particulier notable est celui de la perception du texte écrit : dans ce cas, OBJ va élaborer une représentation à l'intention du sous-système MPL qui est en charge du langage.

Le sous-système morphono-lexical (MPL) est lié principalement au langage. On peut décrire les représentations qu'il manipule comme "ce que nous entendons dans notre tête". Ce sous-système a la connaissance des mots et des formes lexicales. Les représentations fournies à ses entrées par AC pour le langage parlé ou par OBJ (après transformation VIS $\Rightarrow$ OBJ) pour le langage écrit sont transformées en propositions pour le sous-système PROP. Cette transformation extrait le sens des formes langagières entendues ou lues. C'est aussi MPL qui permet la génération de formes langagières. Pour cette génération, le sous-système PROP va transformer une proposition en une représentation pour MPL. Cette représentation va ensuite être transformée par MPL à destination du sous-système articulatoire qui va l'exprimer sous forme de langage parlé ou écrit.

Le sous-système propositionnel (PROP) manipule des représentations correspondant à des propositions. Ces propositions peuvent être représentées sous forme de prédicats logiques. Les entrées de PROP sont principalement alimentées par les sous-systèmes

OBJ, MPL et IMPLIC. Les représentations circulant dans PROP correspondent au contenu sémantique des informations perçues par les sous-systèmes perceptifs mais aussi des propositions engendrées par IMPLIC.

Le sous-système PROP a pour partenaire privilégié le sous-système IMPLIC. IMPLIC reçoit en entrée des représentations qui proviennent principalement de PROP et génère de nouvelles représentations à l'intention de PROP. Pour simplifier et très schématiquement, on peut assimiler le couple PROP/IMPLIC à un système expert où PROP serait la base de connaissances et où IMPLIC jouerait le rôle du moteur d'inférence. Le cycle PROP $\Rightarrow$ IMPLIC et IMPLIC $\Rightarrow$ PROP modélise le raisonnement tel que l'entend le sens courant.

Le sous-système PROP génère aussi des représentations dans le code des sous-systèmes OBJ et MPL qui vont nous permettre d'agir sur l'environnement par l'intermédiaire des sous-systèmes moteurs.

3.3.5. Les sous-systèmes moteurs

Les deux sous-systèmes moteurs contrôlent nos actions sur le monde physique. Le sous-système articulatoire (ART) est spécialisé dans l'émission du langage qu'il soit articulé, mais aussi écrit ou tapé au clavier. Le sous-système mouvement (LIMB) se charge de tous les autres mouvements du corps. Notons qu'en règle générale, le sous-système LIMB reçoit des représentations du sous-système objet, traduisant le fait que nous préparons visuellement un geste avant de l'effectuer. Toutefois dans le cas particulier des actes réflexes (par exemple, on retire sa main rapidement lorsque l'on touche un objet brûlant), le sous-système état physique communique directement avec le sous-système mouvement. C'est l'un des cas où deux sous-systèmes peuvent communiquer sans mettre en jeu un sous-système central (un autre cas survient dans la même circonstance : on criera "Aïe!", ce qui fera aussi intervenir une communication directe BS $\Rightarrow$ ART).

3.4. Application à la communication homme-homme médiatisée

Notre présentation d'ICS met en évidence deux intérêts majeurs du modèle :

- ICS est un modèle qualitatif ; contrairement au modèle du processeur humain qui s'intéresse à la performance de l'utilisateur et aux aspects quantitatifs de la cognition (temps de réaction, limitations de capacité de la mémoire à court terme, etc.), ICS met en avant les aspects qualitatifs de la cognition en termes d'utilisation de ressources. Les différents sous-systèmes peuvent être considérés

comme un ensemble de ressources cognitives et ICS peut indiquer et justifier qu'une ressource risque d'être surchargée dans une circonstance donnée.

- ICS affine le système perceptif en trois sous-systèmes : VIS, AC et BS. De plus il met en évidence différents niveaux d'abstraction et modélise l'interprétation et la compréhension des informations perçues comme une suite de transformations, chaque transformation élevant le niveau d'abstraction. Pour la compréhension d'une information visuelle non textuelle par exemple, un parcours typique des représentations dans l'architecture ICS sera VIS $\Rightarrow$ OBJ puis OBJ $\Rightarrow$ PROP, suivi éventuellement d'un ou plusieurs cycles PROP $\Rightarrow$ IMPLIC / IMPLIC $\Rightarrow$ PROP.

Ces deux aspects d'ICS en font un bon candidat en tant que théorie psychologique adaptée aux nouveaux médias et combinaisons de médias utilisés aujourd'hui dans les interfaces homme-machine, et en particulier dans les interfaces des systèmes de communication homme-homme médiatisée. Nous nous sommes particulièrement intéressés à l'aspect perceptif de l'interaction dans le cadre de ces systèmes. Nous verrons qu'ICS apporte des éléments utiles pour l'étude de l'intégration et de la combinaison des nouveaux médias. ICS permet en particulier de raisonner sur la surcharge cognitive qui peut provenir de la présentation simultanée à l'utilisateur de plusieurs sources d'information. Le modèle permet aussi de déterminer les conditions que les différentes sources d'information doivent respecter afin d'être interprétées comme un tout cohérent. Ces points sont maintenant illustrés au paragraphe suivant.

3.4.1. Informations combinant plusieurs médias : l'évidence expérimentale

Dans de multiples situations de la vie courante, nous sommes amenés à combiner des informations provenant de différents sens. Parmi les possibilités, la combinaison d'informations visuelles et auditives est privilégiée. Lorsque nous discutons, lorsque nous assistons à la projection d'un film, nous construisons la signification de notre conversation ou du film à partir d'informations perçues à la fois par la vue et l'ouïe. Lorsque des informations visuelles et auditives sont présentées simultanément, elles doivent cependant respecter certains critères pour faire sens.

Un exemple familier est celui d'un film dont la bande-son est désynchronisée : si l'écart entre l'image et la bande-son est trop important, le film devient incompréhensible. Dans certains cas, une incohérence entre les informations auditives et visuelles peut même conduire à une interprétation erronée. L'effet McGurk [McGurk 1976] est une bonne

illustration de ce phénomène : on présente à un sujet une bande vidéo montrant un orateur. L'orateur prononce la syllabe "ga" mais la bande-son a été réenregistrée avec la syllabe "ba". Lors des tests, 98% des sujets adultes rapportent entendre la syllabe "da".

Ces exemples montrent que lorsque des informations visuelles et auditives sont présentées à l'aide de moyens techniques, elles doivent satisfaire certaines propriétés afin d'être correctement perçues et interprétées. Ce point est particulièrement crucial pour la conception d'outils de communication audio/vidéo. En effet, les techniques disponibles à l'heure actuelle ne permettent pas en général de transmettre une communication audio/vidéo avec une qualité équivalente à celle du cinéma ou de la télévision. Il est alors nécessaire de rechercher des compromis. En nous donnant un fondement théorique pour analyser les mécanismes perceptifs, ICS nous permet de déduire des principes pour guider ces compromis. Nous avons appliqué les propriétés issues de ces principes dans la conception du système de communication UserLink décrit au chapitre 8 dans la partie plus technique de ce mémoire.

3.4.2. Critères de combinaison d'informations dans ICS

ICS distingue plusieurs types de combinaisons d'informations perçues et donne des critères pour que ces combinaisons soient possibles. La combinaison d'informations distinctes est principalement effectuée par les sous-systèmes centraux (MPL, IMPLIC, PROP ou OBJ). Les informations susceptibles d'être combinées peuvent provenir soit des sous-systèmes perceptifs, soit des sous-systèmes centraux. Différentes sources d'informations perçues simultanément peuvent être combinées à différents niveaux cognitifs : au niveau capteur, dans les sous-systèmes perceptifs, dans les sous-systèmes centraux.

3.4.2.1. Combinaison au niveau capteur

Le tout premier niveau de combinaison est celui des capteurs physiques : les limitations intrinsèques de nos sens ou certaines de leurs particularités nous permettent une perception globale d'informations discrètes. Par exemple, l'image sur un écran est perçue comme un tout alors qu'elle est composée d'un ensemble de points. La persistance rétinienne nous fait voir l'image cinématographique comme un mouvement continu alors qu'elle est constituée de 24 images par seconde. Notons que ces limitations de nos capteurs sensoriels sont déjà exploitées par certains systèmes de compression. La norme JPEG de compression des images photographiques dégrade volontairement l'image originale en éliminant des détails que notre sens de la vue n'est pas capable de discerner. De façon similaire, la norme MPEG audio pour la compression du son tient compte de particularités physiologiques de l'audition pour éliminer certains détails. En d'autres

termes, le principe psychologique de limitation de nos capacités sensorielles permet de s'affranchir de la propriété d'intégrité (définie au chapitre 4) pour certains types d'informations.

3.4.2.2. Combinaison dans les sous-systèmes perceptifs

La combinaison d'informations perçues sur un même canal sensoriel se produit également au niveau des sous-systèmes perceptifs. Sur un même canal, nous pouvons combiner différentes informations à condition qu'elles soient liées par une structure commune. Barnard [Barnard 1992] cite l'exemple d'un vol d'oiseaux : on voit l'ensemble des oiseaux comme un tout, et non chaque oiseau individuellement. D'autres théories psychologiques, en particulier la Gestalt théorie peuvent ici apporter une aide. Cette théorie définit en effet un certain nombre de principes (telles la similarité, la symétrie, la proximité, etc.) qui ont une influence sur la façon dont nous percevons des informations visuelles (figure 3.3).

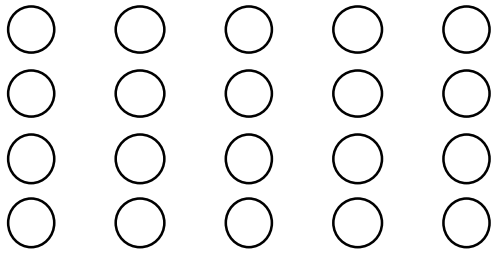


Figure 3.3. Application de la Gestalt théorie. La disposition de ces cercles nous conduit à les voir comme "cinq colonnes de cercles" plutôt que comme "quatre lignes de cercles". La proximité des cercles fait apparaître une propriété émergente de la figure : une disposition en colonnes.

La combinaison au niveau perceptif (c'est-à-dire dans les sous-systèmes perceptifs et sur un seul canal sensoriel) peut déjà avoir des conséquences importantes sur les niveaux supérieurs. En effet, le modèle ICS prédit que la combinaison d'informations provenant de différents canaux sensoriels dans les niveaux supérieurs ne peut avoir lieu que si les flots d'information issus des sous-systèmes perceptifs sont cohérents. Un contre-exemple est donné par les chaînes de télévision dites "mosaïque" qui affichent simultanément plusieurs chaînes sur un même écran (par exemple 16 pour la mosaïque des chaînes du câble à Grenoble). Si le son correspondant à la mosaïque est celui d'une des chaînes affichées, il est très difficile, voire impossible d'identifier à quelle image correspond le son entendu. L'attention du spectateur passant d'une image à l'autre, le flot fourni par le sous-système VIS n'est pas cohérent avec AC et par la suite, la combinaison entre les informations visuelles et auditives ne peut s'effectuer dans les sous-systèmes centraux.

3.4.2.3. Combinaison dans les sous-systèmes centraux

A un niveau cognitif supérieur, dans les sous-systèmes centraux de l'architecture ICS, la combinaison d'informations par un sous-système peut avoir lieu si les informations se présentant aux entrées du sous-système sont :

- exprimées dans le code du sous-système : cette contrainte est nécessaire pour que le sous-système puisse traiter les informations,

- cohérentes à l'intérieur d'un même flot d'informations, comme expliqué ci-dessus,

- cohérentes entre deux flots d'informations : si par exemple, la bande-son d'un film ne correspond pas à l'image, la combinaison est impossible ou donne lieu à une interprétation erronée. C'est le cas de l'effet McGurk exposé plus haut,

- simultanées : les flots d'informations doivent être présents simultanément aux entrées du sous-système, avec une certaine marge de tolérance qui peut être calculée expérimentalement et qui dépend des informations considérées. L'exemple évoqué plus haut de la désynchronisation de la bande-son d'un film est un cas typique où les informations ne sont pas simultanées et gênent, voire empêchent la combinaison.

Ces conditions de combinaison peuvent être vues comme des principes issus de la psychologie. Nous en déduisons des propriétés des médias au chapitre 4, comme la synchronisation et la régularité des flots. La combinaison d'informations peut se produire dans chacun des sous-systèmes centraux de l'architecture ICS sous réserve de respecter les critères énoncés ci-dessus. La combinaison d'informations visuelles et auditives par exemple peut avoir lieu dans le sous-système MPL (via OBJ pour les informations visuelles). Dans les autres sous-systèmes centraux ont lieu des combinaisons d'information à plus haut niveau, après abstraction des informations sensorielles par les sous-systèmes PROP et IMPLIC.

PROP est le lieu de la combinaison d'informations sensorielles (via OBJ et MPL) et sémantiques (via IMPLIC). Nous décrivons plus loin l'expérimentation Garden Movie qui fait intervenir une combinaison au niveau du sous-système PROP. Dans le sous-système IMPLIC a lieu la combinaison d'informations perçues et sémantiques. La particularité de la combinaison à ce niveau est que les informations perçues parvenant à IMPLIC arrivent directement des sous-systèmes perceptifs sans passer ni par OBJ, ni par

MPL, ni par PROP. C'est le cas des informations qui sont ressenties mais pas interprétées : un bruit violent, par exemple, mais aussi toutes les informations que nous percevons "inconsciemment". C'est le cas en particulier des attitudes physiques de nos interlocuteurs, des gestes qui accompagnent les discours, bref d'une grande partie de la communication non-verbale.

Enfin un cas particulier de combinaison peut également se produire dans le sous-système OBJ : des informations sémantiques issues de PROP vont être combinées avec les informations visuelles afin de guider l'interprétation. C'est le cas par exemple lorsque nous sommes dans un environnement familier plongé dans la pénombre et que nous devinons un objet parce que nous savons qu'il est là.

En appliquant ICS à la perception d'informations complexes, il est donc possible de déterminer des critères afin que des informations combinées puissent être perçues et interprétées correctement. ICS montre aussi que la combinaison d'informations est un phénomène complexe qui peut intervenir à différents niveaux du système cognitif. A travers quelques exemples liés à la perception d'informations visuelles et auditives, nous avons montré qu'ICS peut aider à réfléchir sur la présentation de ces informations en générant des principes et ainsi guider des choix de conception.

Au paragraphe suivant, nous présentons l'expérimentation Garden Movie qui a permis d'étudier un cas de combinaison d'informations au niveau du sous-système PROP.

3.5. Expérimentation avec ICS : Garden Movie

J'ai participé à l'expérimentation Garden Movie lors d'un séjour de trois mois au MRC-Applied Psychology Unit à Cambridge (Grande-Bretagne), en collaboration avec Jon May et Phil Barnard. J'ai pris part à la conception et au dépouillement des résultats de l'expérimentation. J'ai aussi réalisé l'application support (avec MacroMedia Director [MacroMedia 1993]) ; j'ai également testé la plupart des sujets.

Pour ma part, la motivation de cette collaboration était double :

- je souhaitais étudier les processus psychologiques mis en œuvre dans la coopération entre utilisateurs d'un système permettant la communication audio/vidéo,

- je souhaitais aussi me familiariser avec la démarche expérimentale en psychologie, dans le but de l'appliquer aux tests d'utilisabilité avec la plate-forme NEIMO (voir chapitre 5).

La découverte d'un nouvel environnement et l'appréhension "de l'intérieur" d'une discipline dans laquelle j'étais naïf a été une expérience extrêmement enrichissante. Ce paragraphe décrit les motivations de l'expérimentation, le dispositif expérimental mis en place. Il présente brièvement les résultats obtenus, les discute et les explique, et enfin tire les leçons de cette collaboration.

3.5.1. Motivations de l'expérimentation

Dans la communication humaine face à face, une grande partie des informations échangées l'est sous forme non-verbale : gestes, attitudes, contact visuel, etc. Mais lorsque la communication se fait via un lien audio/vidéo, des difficultés apparaissent [Heath 1992]. Les gestes par exemple sont moins bien compris que dans la communication face à face. Une raison en est l'absence d'un cadre de référence commun. L'usage traditionnel de la vidéo, qui simule la communication face à face, semble laisser penser aux utilisateurs que leur cadre de référence est parallèle à celui des autres participants alors qu'il est en fait en vis-à-vis. Les systèmes de vidéoconférence commerciaux comme ceux décrits dans [Reinhardt 1993] offrent aux utilisateurs une disposition face à face et la possibilité de partager des vues sur des documents. Dans ce type de systèmes, un télépointeur ou des annotations partagées sont souvent utilisés pour remplacer la communication gestuelle.

Les systèmes de dessin partagé sont un champ d'investigation intéressant pour l'étude de la communication gestuelle via vidéo. Commune [Minneman 1991] permet aux utilisateurs de dessiner sur une surface commune horizontale et de se voir de face sur des écrans vidéo. Ce système utilise des télépointeurs mais les auteurs ont constaté que des gestes étaient aussi échangés par les utilisateurs. Dans ClearBoard [Ishii 1992], deux utilisateurs peuvent dessiner simultanément sur un espace partagé en voyant l'image de l'autre participant superposée sur le dessin. Une caractéristique originale de ClearBoard est qu'il renverse l'image du participant. Les deux utilisateurs ont ainsi une référence commune de la droite et de la gauche du dessin et peuvent utiliser des gestes pour montrer une partie du dessin. VideoWhiteboard [Tang 1991], précurseur de ClearBoard, adoptait une approche similaire mais projetait l'ombre des participants au lieu de leur image. De même que dans ClearBoard, l'ombre projetée était inversée. Cependant, pour ces deux systèmes, l'efficacité de cette technique comparée à la disposition classique face à face n'a pas été étudiée.

Enfin, [Gaver 1993] propose une approche originale pour une tâche où deux utilisateurs coopèrent pour arranger des objets d'une maison de poupées miniature. La disposition choisie utilise plusieurs caméras qui donnent aux utilisateurs le choix de plusieurs vues de la maison de poupées et de leur partenaire. L'expérimentation montre que les possibilités de choix des caméras rend difficile l'établissement d'un cadre de référence commun entre les utilisateurs. Mais un résultat intéressant de cette étude indique que les utilisateurs préfèrent une vue qui montre l'espace de travail commun plutôt que la vue face à face. Ils peuvent ainsi créer un espace de référence partagé centré sur la tâche.

3.5.2. Dispositif expérimental

Le but de notre expérimentation "Garden Movie" est d'étudier l'influence de plusieurs paramètres sur la réalisation coopérative d'une tâche de dessin par deux utilisateurs dans un environnement permettant des communications audio et vidéo. Pour des raisons pratiques et de rigueur expérimentale, l'un des utilisateurs est simulé, l'autre est le sujet testé. Le sujet voit sur son écran un espace de travail (similaire à MacDraw) de trois cases sur trois qu'il est censé partager avec un autre utilisateur. A côté de cet espace de travail, une fenêtre vidéo montre l'utilisateur avec lequel il coopère (figures 3.4 et 3.5).

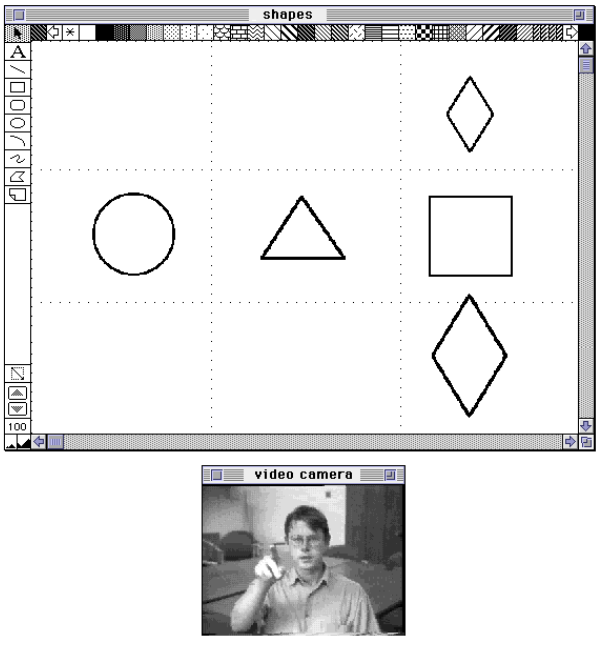


Figure 3.4. Une copie d'écran de l'expérimentation Garden Movie. On présente ici la vue de face et l'espace de travail est en configuration verticale (c'est-à-dire sans perspective).

Le collègue présenté dans la fenêtre vidéo (en fait un enregistrement) demande au sujet de déplacer un objet de l'espace de travail vers une nouvelle position. Il accompagne sa

demande de gestes désignant l'objet à déplacer et sa nouvelle position. Le collègue est censé voir sur son écran exactement le même espace de travail. Un *essai* se déroule de la façon suivante :

- au début de l'essai, le sujet a sur son écran une configuration analogue à celle de la figure 3.4,
- la séquence vidéo est présentée au sujet ; le collègue demande au sujet de déplacer deux objets et accompagne la désignation des objets et de leurs nouvelles positions de gestes de la main droite,
- lorsque la séquence est terminée, elle est remplacée par un bouton “Play again”. Ce bouton permet au sujet de revoir la séquence vidéo,
- le sujet exécute la demande du collègue en déplaçant les objets avec la souris et signale qu’il est prêt à passer à l’essai suivant en appuyant sur un bouton “OK”. Le bouton “OK” n’apparaît que lorsque la séquence vidéo est entièrement jouée.

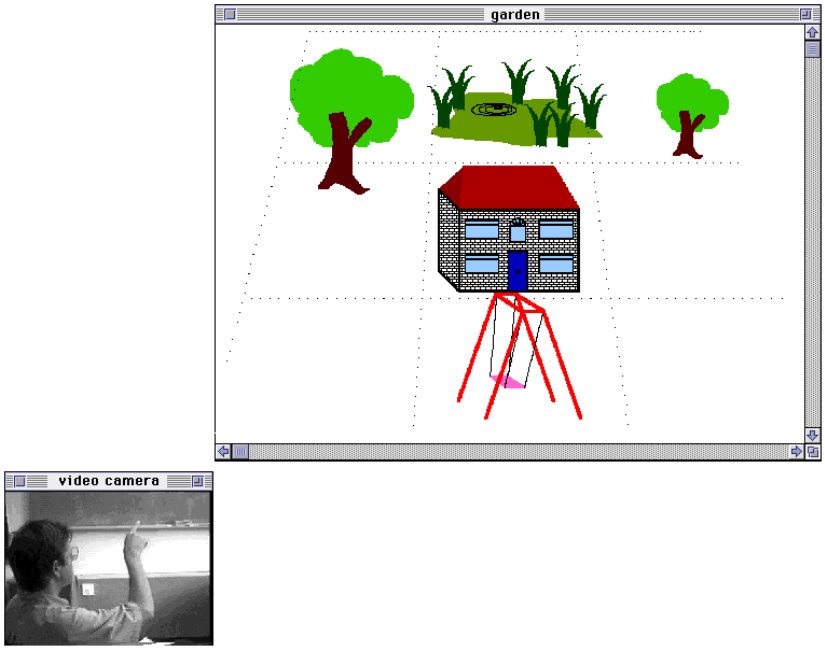


Figure 3.5. Une copie d'écran de l'expérimentation Garden Movie. On présente cette fois la vue “par-dessus l’épaule” et l’espace de travail est en configuration horizontale (c’est-à-dire avec perspective).

Dans l’expérimentation, les effets de trois variables ont été étudiés : la position de la caméra, la disposition de l’espace de travail, et l’usage des déictiques. Trois positions de

caméra ont été étudiées : une vue de face, une vue de face “miroir” et une vue “par-dessus l’épaule”. La vue de face telle que celle présentée sur la figure 3.4 reproduit la disposition de deux personnes face à face, par exemple de part et d’autre d’une table. Elle reproduit donc une situation réelle ; cependant, les deux utilisateurs ayant la même vue du document, un objet situé à gauche dans l’espace de travail sera désigné par le collègue sur sa propre gauche, donc à droite sur l’image vidéo observée par le sujet. La vue miroir est une première tentative pour résoudre ce problème de latéralisation : la vue miroir présente la vue de face transformée par une simple symétrie horizontale. Un objet situé à gauche de l’espace de travail sera désigné par le collègue par un geste sur la gauche de l’image vidéo. Enfin la troisième vue étudiée est donnée par une caméra située derrière l’épaule de l’utilisateur. Cette vue est une autre tentative de solution au problème de latéralisation : les deux utilisateurs regardent la figure selon un point de vue commun ; on souhaite ainsi créer un espace de travail partagé. Afin de renforcer cette illusion d’espace partagé, la fenêtre vidéo est placée dans ce cas en bas à gauche de l’espace de travail en sorte que les gestes du collègue semblent désigner l’espace de travail du sujet.

L’influence de la disposition de l’espace de travail a également été étudiée : deux configurations ont été utilisées. La première (cf. figure 3.4) présente des figures géométriques sur le plan vertical de l’écran. La seconde (cf. figure 3.5) présente des objets que l’on peut trouver dans un jardin (d’où le nom de l’expérimentation !) et grâce à une illusion de profondeur, donne l’impression que les objets sont disposés sur un plan horizontal.

Enfin, l’utilisation de références déictiques a été étudiée : les instructions données par le collègue peuvent soit être explicites, soit comporter des déictiques. Une instruction explicite comporte le nom des objets à déplacer et indique l’endroit où les déplacer relativement à un autre objet. Par exemple, pour la figure 3.4, “déplace le petit losange sous le cercle” est une instruction explicite. Les déictiques ont été utilisés pour référencer les objets à déplacer (comme dans “*cet* objet”, “*ceci*”) et les relations spatiales dans l’espace de travail (comme “de *ce* côté du cercle”). La figure 3.6 récapitule les variables de l’expérimentation.

Vue	**Espace de travail**	**Déictique**
de face	horizontal (jardin)	non (explicite)
par-dessus l’épaule	vertical (MacDraw)	oui (objet ou spatial)
de face miroir		

Figure 3.6. Les variables de l’expérimentation Garden Movie. Les noms des variables sont indiqués en haut de chaque colonne et les valeurs possibles en-dessous.

Une expérimentation avec un sujet est divisée en trois phases qui correspondent aux trois positions de caméra possibles (face, face miroir, derrière l'épaule) (figure 3.7).

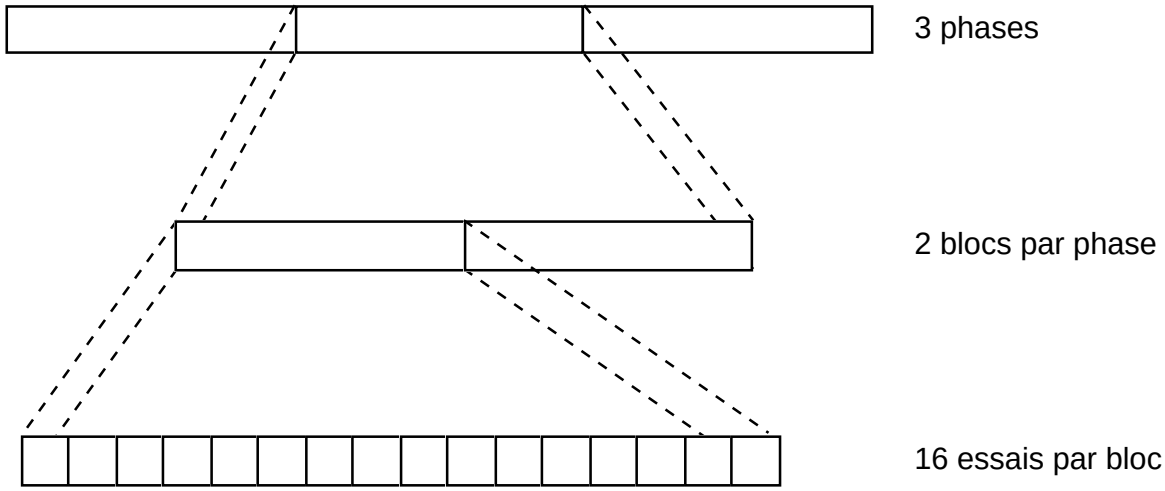


Figure 3.7. La structure d'une session d'expérimentation Garden Movie. Un essai correspond à une configuration comme en montrent les figures 3.4 et 3.5.

Lors d'une phase, la position de la caméra est maintenue constante. Chaque phase est découpée en deux blocs de seize essais chacun. Chaque bloc correspond à une configuration différente de l'espace de travail (verticale, horizontale). Enfin, dans chaque bloc la moitié des essais fait intervenir des instructions explicites, l'autre moitié fait intervenir des instructions comportant des déictiques (dans chaque bloc, quatre déictiques portant sur les objets et quatre déictiques portant sur les relations spatiales).

Lors d'une expérimentation, trois types d'informations ont été collectées : l'utilisation du bouton "Play again", les erreurs dans l'exécution des instructions, et le temps d'exécution de chaque essai. L'information concernant l'utilisation du bouton "Play again" permet de déterminer combien de fois chaque séquence vidéo a été vue par le sujet. (Nous indiquions aux sujets qu'ils pouvaient utiliser le bouton "Play again" autant de fois qu'ils le voulaient tant que les instructions du collègue "n'étaient pas claires".) Les erreurs dans l'exécution des instructions sont des erreurs de placement des objets (par exemple, objet placé à droite alors que l'instruction donnée par le collègue demandait de le placer à gauche). Le temps d'exécution de l'essai est calculé à partir du moment où la dernière visualisation de la séquence s'arrête et jusqu'au moment où le sujet indique qu'il a terminé l'essai en appuyant sur le bouton "OK".

Lors de l'expérimentation, vingt-quatre sujets choisis au hasard dans le "panel" de sujets de MRC-APU ont été testés. L'expérimentation comportant trois variables pouvant prendre douze valeurs différentes (3x2x2) et les différents blocs de l'expérimentation

étant présentés dans un ordre différent pour chaque sujet, douze sujets suffisent pour couvrir tous les cas possibles. Nous avons testé deux séries de douze sujets pour augmenter la fiabilité de nos résultats, soit treize femmes et onze hommes d'âge moyen 31,2 ans. Les instructions lues aux sujets sont présentées en annexe B.

3.5.3. Résultats

On trouvera dans [Barnard 1994] la description détaillée des résultats de l'expérimentation. Nous nous contenterons ici d'en donner la synthèse.

Le taux d'erreur est défini comme le pourcentage d'essais dans lesquels les sujets ont fait une erreur de placement d'objet. Le taux de "replay" est le pourcentage d'essais dans lesquels les sujets ont redemandé à voir la séquence au moins une fois. Ces deux taux ont été étudiés en fonction des variables de l'expérimentation. Les résultats font apparaître que la disposition de l'espace de travail a une influence limitée sur ces taux ou sur le temps nécessaire à la réalisation de l'essai. En revanche, la position de la caméra et l'utilisation des déictiques ont une influence significative. Nous donnons d'abord les résultats globaux, puis nous approfondissons les résultats concernant l'intelligibilité des déictiques. Lorsque les instructions données sont explicites (figure 3.8), le taux d'erreur et le taux de replay ont des valeurs négligeables (proches de 5%), quelle que soit la vue présentée. Pour les instructions contenant des déictiques, le taux d'erreur moyen est proche de 30% et le taux de replay proche de 25%.

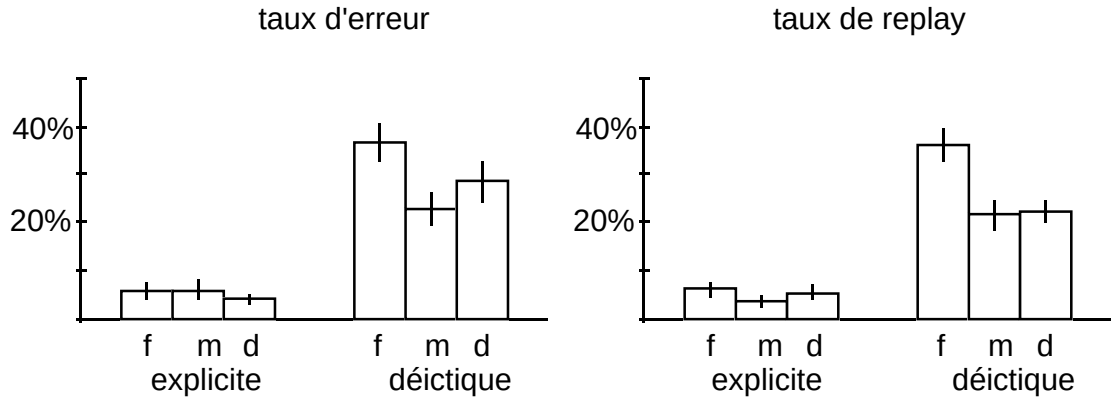


Figure 3.8. Taux d'erreur et de replay suivant les instructions (explicites ou déictiques) et les positions de caméra (f pour face, m pour miroir, d pour derrière l'épaule).

L'analyse des temps d'exécution d'un essai montre également une nette différence suivant que les instructions comportent des déictiques ou non : le temps moyen d'exécution d'un essai est de 8,27 secondes lorsque les instructions sont explicites, et de 9,20 secondes avec des déictiques.

Nous nous intéressons maintenant uniquement aux instructions comportant des déictiques (figure 3.9).

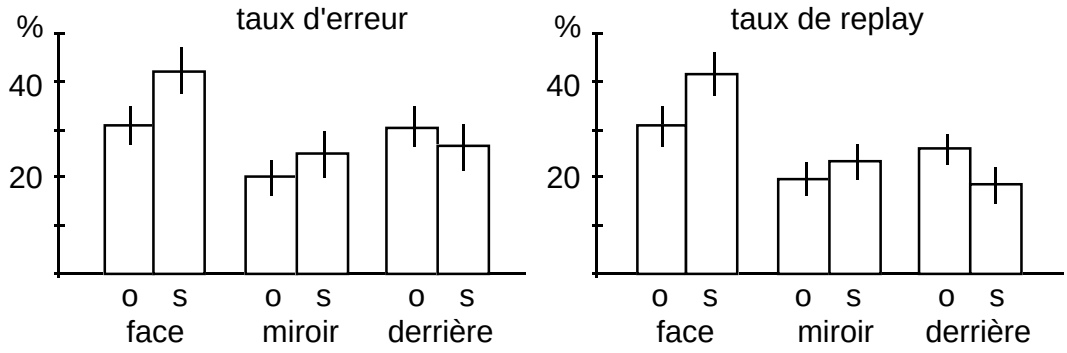


Figure 3.9. Taux d'erreur et de replay suivant les positions de la caméra et les types de déictiques (o pour objet, s pour spatial).

En fonction de la vue présentée, le taux d'erreur et de replay est plus important pour la vue de face (proche de 40%) que pour les deux autres (20% à 30%). Dans les cas de la vue de face et de la vue de face miroir, les taux d'erreur et de replay sont moins importants lorsque le déictique désigne un objet que lorsqu'il désigne une relation spatiale. Dans le cas de la vue derrière l'épaule, les déictiques spatiaux donnent lieu à de plus faibles taux que les déictiques d'objets.

En ce qui concerne les temps d'exécution d'un essai, un essai comportant un déictique spatial est effectué en sensiblement plus de temps (9,49 secondes) qu'un essai avec un déictique d'objet (8,89 secondes).

Dans les cas des instructions contenant des déictiques désignant des relations spatiales, l'analyse fait apparaître une différence significative suivant le type de relation spatiale. Suivant la configuration de l'espace de travail (vertical plan ou horizontal avec effet de perspective), les relations spatiales sont exprimées différemment. Si les relations horizontales "à gauche de" et "à droite de" sont les mêmes dans les deux cas, les relations spatiales verticales "au-dessus de" et "au-dessous de" sont utilisées dans l'espace plan alors que ces relations deviennent "derrière" et "devant" pour l'espace avec effet de perspective. Les résultats montrent que les taux d'erreur et de replay sont nettement plus importants pour les relations spatiales verticales que pour les relations spatiales horizontales (figure 3.10).

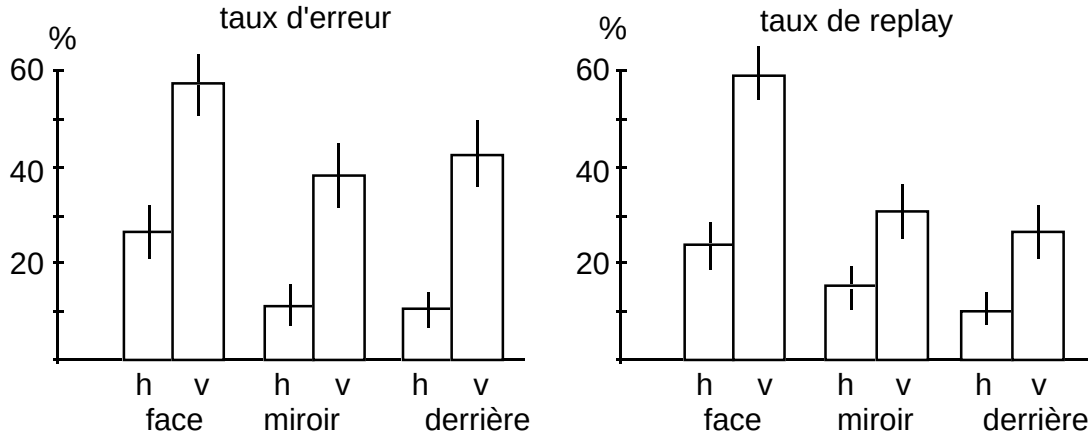


Figure 3.10. Taux d'erreur et de replay suivant les positions de caméra et les types de relations spatiales (h pour horizontale, v pour verticale).

L'analyse des temps d'exécution d'un essai montre que le temps d'exécution est plus important pour les relations spatiales verticales (9,87 secondes) que pour les relations spatiales horizontales (9,06 secondes).

3.5.4. Discussion

Les résultats de l'expérience montrent que les différentes variables de l'expérimentation ont une influence sur les erreurs, les replays et le temps d'exécution. Nous analysons ci-après les résultats obtenus et donnons une interprétation avec le modèle ICS des mécanismes cognitifs mis en œuvre.

Les résultats font clairement apparaître que les instructions explicites donnent lieu à moins d'erreur et sont exécutées plus rapidement que les instructions comportant des références déictiques. Cependant, les taux d'erreur et de replay, bien que faibles, ne sont pas nuls dans le cas des instructions explicites. Ce fait met en évidence que la tâche demandée aux sujets est complexe même sans déictique. Le sujet doit en effet partager son attention visuelle entre la séquence vidéo et l'espace de travail. On peut se demander si des instructions uniquement orales auraient facilité la tâche du sujet et donné de meilleurs résultats pour les essais sans déictique.

Sur l'ensemble de l'expérimentation, le positionnement de la caméra "de face" donne lieu au plus grand nombre d'erreurs, de replays et au plus long temps d'exécution. Ce résultat était attendu : avec cette position de caméra, il y a systématiquement conflit entre les gestes du collègue et la disposition des objets dans l'espace de travail. Ce conflit peut être résolu par le sujet par un renversement mental des gestes du collègue. Dans notre contexte expérimental, l'utilisation de la vue de face pour la réalisation d'une tâche déjà complexe semble déborder les capacités cognitives. Les sujets ne semblent pas toujours parvenir à

effectuer la transformation nécessaire pour interpréter correctement les gestes du collègue. Cette supposition est confortée par l'analyse des processus mis en jeu dans le cadre de l'architecture ICS (figure 3.11).

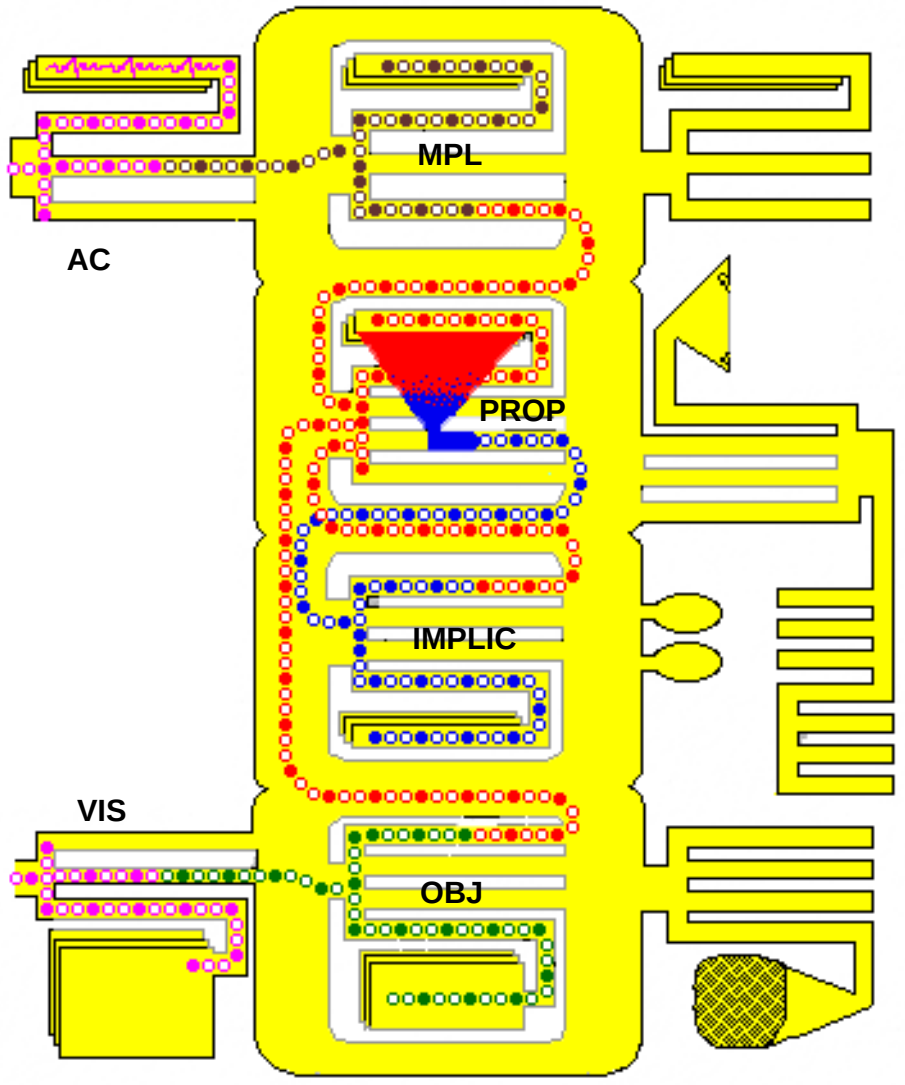


Figure 3.11. Fusion des informations au niveau du sous-système PROP dans l'architecture ICS.

La résolution des déictiques nécessite plusieurs combinaisons d'informations : tout d'abord, la séquence vidéo requiert la combinaison d'informations auditives (acquises via AC et MPL) et visuelles (via VIS et OBJ). Ces informations sont combinées au niveau du sous-système PROP et la séquence vidéo est comprise après passage de l'information dans le sous-système IMPLIC. La compréhension d'un déictique demande ensuite la combinaison de l'information élaborée par IMPLIC avec celle correspondant à l'interprétation de la disposition des objets dans l'espace de travail. Cette interprétation est fournie par OBJ via VIS. A ce moment du traitement des informations, deux flots

d'information sont disponibles à l'entrée de PROP : celui correspondant à la séquence vidéo et celui correspondant à l'espace de travail. Or la séquence vidéo étant renversée par rapport à l'espace de travail, les deux flots d'information ne sont pas cohérents et ne remplissent donc pas le critère de cohérence exposé en 3.4.2.3. La combinaison peut cependant s'effectuer, mais a de fortes chances de donner une information incohérente et donc de conduire le sujet à mal interpréter le déictique. Une étape supplémentaire de transformation (le renversement mental) est nécessaire pour que la compréhension du déictique soit correcte. Il y a alors cohérence entre les deux informations et la combinaison peut s'effectuer correctement.

ICS permet de détailler la complexité des traitements cognitifs nécessaires à l'interprétation de la vue de face et d'expliquer à quel niveau cognitif et pourquoi une surcharge cognitive peut intervenir. En revanche, ICS fournit peu d'indications pour expliquer les autres résultats. En particulier, la meilleure adéquation de la vue miroir par rapport à la vue par derrière ne peut pas être justifiée simplement. Les traitements cognitifs mis en œuvre sont similaires dans les deux cas et rien dans ICS ne tranche de façon claire en faveur de l'une ou l'autre configuration. De même les différences observées dans les résultats entre les déictiques spatiaux et d'objets et entre les déictiques spatiaux horizontaux et verticaux ne sont pas éclaircies par ICS.

La vue de derrière permettant de créer un point de vue commun entre le sujet et le collègue de la séquence vidéo, on aurait pu s'attendre à ce que la vue de derrière donne lieu à moins d'erreurs que les autres vues. En fait la vue miroir se révèle être souvent plus adéquate que la vue de derrière. Une exception notable est la résolution des déictiques spatiaux et d'objets (cf. figure 3.8) : la vue miroir et la vue de derrière donnent des résultats équivalents pour les déictiques spatiaux, mais la vue de derrière est moins performante pour les déictiques d'objets. La vue de derrière ne permet donc pas une meilleure résolution des déictiques.

Les résultats montrent aussi que, quelle que soit la position de la caméra, les sujets ont plus de difficultés à résoudre les déictiques spatiaux verticaux que horizontaux. Une explication possible est que, comme mentionné plus haut, les déictiques spatiaux font intervenir des formulations différentes suivant la disposition de l'espace de travail (horizontal/vertical) et ces différences ont pu perturber l'interprétation de ces déictiques.

Comme avec tous les résultats d'expérimentation, il convient d'être extrêmement prudent dans l'interprétation des résultats. Les résultats obtenus montrent la meilleure adéquation de la vue miroir *pour le dispositif expérimental considéré.* Il serait abusif d'en conclure que la vue miroir est mieux adaptée à des tâches coopératives dans un environnement

audio/vidéo. Cependant, les résultats montrent que, pour certaines tâches coopératives et pour la résolution de déictiques, la possibilité d'avoir une vue miroir peut améliorer l'intelligibilité des déictiques. Nous examinons maintenant cette constatation d'un point de vue technique.

La position des caméras dans un environnement de communication audio/vidéo peut donc contribuer à l'utilisabilité du système. Cependant, la position des caméras est également soumise à des contraintes qui rendent certaines configurations difficiles à mettre en œuvre. La plupart des systèmes de vidéoconférence, par exemple, placent la caméra face à l'utilisateur et à proximité de l'écran. Nous avons vu que cette solution est utilisable dans un contexte proche de celui de l'expérimentation Garden Movie à condition de fournir une vue miroir. La vue miroir peut être fournie par manipulation informatique de l'image captée par la caméra : une transformation simple (symétrie verticale) appliquée systématiquement aux images captées est suffisante. Nous remarquons avec intérêt que CU-SeeMe [CU-SeeMe 1995], un outil de vidéoconférence largement diffusé dans la communauté Internet, dispose précisément d'une fonction miroir.

La fonction miroir peut cependant avoir des effets indésirables et il est souhaitable qu'elle puisse être désactivée par l'utilisateur. En effet, puisque l'image est renversée, l'environnement de l'utilisateur observé est également renversé ; par exemple un environnement familier, tel un bureau, peut être surprenant vu "à l'envers". Dans certains cas, voir une image à l'envers peut être plus déplaisant : si l'utilisateur observé sur l'image vidéo porte un tee-shirt avec une inscription, cette inscription apparaîtra en miroir à l'image.

La vue derrière l'épaule est plus difficile à réaliser en pratique que les deux autres : placer la caméra derrière soi est contraignant, surtout si cette position inhabituelle n'est pas utilisée constamment. Bien sûr, on peut utiliser un système à plusieurs caméras et activer la caméra située derrière soi en cas de besoin ; d'autres problèmes subsistent néanmoins. La caméra doit être placée de façon précise afin que l'utilisateur et ses gestes soient bien visibles : pour un utilisateur de station informatique, placer une caméra derrière soi pour que les gestes soient visibles, et en évitant de préférence de filmer l'écran de sa station impose des contraintes contradictoires.

L'expérimentation Garden Movie nous a fourni un nouveau principe lié à l'interprétation cognitive des expressions déictiques. Nous déduisons de ce principe la propriété de réversibilité du média vidéo présentée au chapitre 4.

3.5.5. Leçons de l'expérimentation Garden Movie

Un des objectifs de la réalisation et de la conduite de cette expérimentation était d'étudier la démarche expérimentale et de l'appliquer aux tests d'utilisabilité que nous serions amenés à réaliser avec la plate-forme NEIMO. L'expérimentation Garden Movie nous a effectivement permis de découvrir et surtout d'appliquer la démarche expérimentale. Et nous nous sommes rendus compte que la psychologie expérimentale et les tests d'utilisabilité, même s'ils ont en apparence des points communs, sont deux mondes bien différents. La mise en œuvre de tests d'utilisabilité peut cependant bénéficier des méthodes de la psychologie expérimentale.

La différence majeure entre les deux démarches réside dans les postulats de départ. Une expérimentation psychologique est conçue pour étudier le comportement humain. Partant d'une hypothèse de départ sur un comportement humain, éventuellement étayée par une théorie psychologique, l'expérimentation permet de confirmer ou d'infirmer cette hypothèse, et éventuellement fait apparaître de nouvelles questions. Un test d'utilisabilité vise à étudier une interface homme-machine. Partant d'un aspect de l'interface que l'on souhaite étudier, le test d'utilisabilité permet de mettre en évidence les défauts éventuels liés à cet aspect de l'interface. On voit ici se dégager deux différences importantes : la psychologie expérimentale s'intéresse au comportement humain, les tests d'utilisabilité étudient une interface homme-machine. D'autre part, en psychologie expérimentale, l'expérimentateur a en général une hypothèse *a priori* qu'il cherche à valider ; au contraire, les ergonomes qui réalisent un test d'utilisabilité s'efforcent de ne pas avoir d'*a priori* sur le système qu'ils testent.

Dans la préparation du test, les approches sont aussi différentes. Pour préparer une expérimentation psychologique, l'expérimentateur identifie un ensemble de variables indépendantes ; l'expérimentation a pour but de présenter au sujet des stimuli qui combinent les valeurs possibles de cet ensemble de variables et de mesurer les réactions du sujet en réponse à ces stimuli. Afin de limiter son champ d'investigation et pour ne pas perturber les résultats, l'expérimentateur cherche à limiter le nombre de variables ; une expérimentation psychologique fait rarement intervenir plus de quatre ou cinq variables. Pour préparer un test d'utilisabilité, l'ergonome prépare un scénario représentatif des tâches pour lesquelles l'interface a été conçue. Le test consiste en la réalisation de ce scénario par le sujet. L'ergonome observe le comportement du sujet et en déduit d'éventuels défauts de l'interface, en s'appuyant sur son expérience et sur des règles heuristiques.

La conduite d'une expérimentation psychologique est une école de rigueur. Comme les variables de l'expérimentation sont clairement recensées, il est essentiel que d'autres variables parasites ne viennent pas perturber l'expérimentation. Par exemple, les instructions que l'expérimentateur donne au sujet doivent être strictement les mêmes pour tous les sujets, et le comportement de l'expérimentateur doit être identique avec tous les sujets pendant l'expérimentation. Un test d'utilisabilité est beaucoup moins strict ; l'ergonome peut par exemple se permettre d'intervenir et d'aider le sujet si le besoin s'en fait sentir.

Dans l'analyse et l'utilisation des résultats enfin, les approches varient. Les données à recueillir lors d'une expérimentation psychologique sont déterminées à l'avance et sont ensuite analysées avec des outils statistiques afin de déterminer des constantes de réactions indépendamment des individus. Dans un test d'utilisabilité classique, les données recueillies le sont de façon très informelle : il s'agit des notes des expérimentateurs, de bandes vidéo, etc. D'autres données peuvent également être recueillies, tel le temps nécessaire à la réalisation d'une tâche donnée. Contrairement aux expérimentations psychologiques, il n'y a pas d'analyse statistique poussée de façon à gommer les comportements individuels. Un défaut de l'interface mis en évidence par un seul sujet sera pris en compte. En psychologie expérimentale, un sujet fournissant des résultats très en-dehors de la moyenne des sujets sera minimisé par l'analyse statistique, voire écarté de l'analyse. Un test d'utilisabilité numérique comme ceux que nous pouvons réaliser avec la plate-forme NEIMO se rapprocherait plus de la méthode expérimentale en psychologie : les données recueillies peuvent être traitées avec des outils statistiques, mais le but reste différent. Il ne s'agit pas d'identifier des constantes du comportement humain, mais de détecter les difficultés que peut poser une interface aux utilisateurs.

En conclusion, la démarche de la psychologie expérimentale est plus rigoureuse parce qu'elle sert un but différent. Au même titre qu'une expérience de physique, une expérimentation psychologique relève d'une démarche scientifique classique : on veut vérifier ou infirmer une hypothèse de départ ; l'objet de l'expérimentation, c'est-à-dire ses variables sont clairement identifiées, et l'expérimentation est reproductible. Un test d'utilisabilité en revanche vise à mettre en évidence les défauts d'une interface. Les ergonomes, sans idée préconçue sur l'interface, observent le comportement du sujet et, à partir de leur expérience et de règles heuristiques, détectent les faiblesses de l'interface.

3.6. Synthèse

La psychologie cognitive apporte une contribution majeure à l'étude des systèmes de communication homme-homme médiatisée dans le cadre des systèmes multi-utilisateurs.

Contrairement aux approches traditionnelles, représentées par le modèle du processeur humain, nous avons préféré privilégier les aspects qualitatifs plutôt que quantitatifs de l'interaction de l'utilisateur avec ces systèmes. Il y a à cela plusieurs raisons : tout d'abord, les données fournies par les modèles quantitatifs ne sont que des valeurs moyennes ; certains de ces résultats sont régulièrement contestés et dépendent fortement de l'expérience de l'utilisateur. Dans le cas des systèmes multi-utilisateurs, où l'adaptation de l'interface à chacun des utilisateurs du système est une nécessité, l'utilisation de valeurs moyennes comme base de travail n'est pas pertinente. D'autre part, l'apparition de nouveaux médias pour la communication humaine informatisée, tels le son et la vidéo, pose des questions nouvelles en lien direct avec la perception. Dans la communication humaine, informatisée ou non, il peut être difficile de déterminer la tâche et encore plus d'évaluer quantitativement un échange entre participants. L'utilisation d'un modèle qualitatif et prédictif tel ICS, appliqué à la perception d'informations complexes, permet d'évaluer l'utilisation des ressources cognitives de l'utilisateur et ainsi de déterminer la validité des choix de conception. Comme pour toute théorie psychologique, un tel modèle trouve idéalement sa place en début de conception, et permet de critiquer des choix et d'en suggérer de nouveaux. Dans le cas d'un système existant, il peut aussi servir à expliquer des difficultés d'utilisation. Il faut cependant noter que, même si le modèle ICS peut être présenté assez simplement, son utilisation effective dans un processus de conception requiert une certaine expertise et la participation de psychologues.

Références

[Andler 1992] D. Andler. *Introduction,* in *Introduction aux sciences cognitives.* D. Andler, (ed.) Gallimard, Paris, France, 1992.

[Barnard 1985] P. J. Barnard. *Cognitive Resources and the Learning of Computer Dialogs,* in *Interfacing Thought, Cognitive aspects of Human Computer Interaction.* J. M. Carroll, (ed.) MIT Press, 1985. pp. 112-158.

[Barnard 1992] P. J. Barnard et J. May. *Real time blending of data streams: a key problem for the cognitive modelling of user behaviour with multimodal systems,* Amodeus Project, Working Paper, UM/WP 26, 1992.

[Barnard 1994] P. J. Barnard, J. May et D. Salber. *Deixis and Points of View in Media Spaces,* Amodeus Project, Working Paper, UM/WP 19, 1994.

[Card 1983] S. K. Card, T. P. Moran et A. Newell. *The Psychology of Human-Computer Interaction,* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1983.

[CU-SeeMe 1995] *CU-SeeMe* . Logiciel pour Apple Macintosh. Cornell University, 1995.

[Gaver 1993] W. W. Gaver, A. Sellen, C. Heath et P. Luff. *One is not Enough: Multiple Views in a Media Space,* InterCHI'93, ACM/IFIP Conference on Human Factors in Computing Systems, Amsterdam, Pays-Bas, 1993. pp. 335-341.

[Heath 1992] C. Heath et P. Luff. *Media Space and Communicative Asymmetries: Preliminary Observations of Video-Mediated Interaction,* in *Human-Computer Interaction,* 7(3), 1992. pp. 315-246.

[Ishii 1992] H. Ishii et M. Kobayashi. *ClearBoard: A Seamless Medium for Shared Drawing and Conversation with Eye Contact,* CHI'92, ACM Conference on Human Factors in Computing Systems, Monterey, California, USA, 1992. pp. 525-532.

[MacroMedia 1993] *Director 3.1.1.* Logiciel pour Apple Macintosh. MacroMedia Inc., 1993.

[McGurk 1976] H. McGurk et J. MacDonald. *Hearing lips and seeing voices,* in *Nature,* numéro(264), 1976. pp. 746-748.

[Minneman 1991] S. L. Minneman et S. A. Bly. *Managing a trois: A Study of a Multi-User Drawing Tool in Distributed Design Work,* CHI'91, ACM Conference on Human Factors in Computing Systems, New Orleans, Louisiana, USA, 1991. pp. 217-224.

[Nigay 1994] L. Nigay. *Conception et réalisation des systèmes interactifs: Application aux Interfaces Multimodales.* Thèse de doctorat, Université Joseph Fourier Grenoble I, 1994.

[Reinhardt 1993] A. Reinhardt. *Video Conquers the Desktop,* in *BYTE,* 18(9), septembre 1993. pp. 64-80.

[Smolensky 1992] P. Smolensky. *IA connexionniste, IA symbolique et cerveau,* in *Introduction aux sciences cognitives.* D. Andler, (ed.) Gallimard, Paris, 1992. pp. 77-106.

[Tang 1991] J. C. Tang et S. L. Minneman. *VideoWhiteboard: Video Shadows to Support Remote Collaboration*, CHI'91, ACM Conference on Human Factors in Computing Systems, New Orleans, Louisiana, USA, 1991. pp. 315-322.

Deuxième Partie

L'articulation entre
les sciences non-informatiques
et la mise en œuvre :

Propriétés et Évaluation

Propriétés

As far as the laws of mathematics refer to reality, they are not certain; and as far as they are certain, they do not refer to reality.

Albert Einstein

Propriétés

4.1. Introduction

Notre grille d'analyse à trois niveaux présentée au chapitre 1 comprend des principes, propriétés et techniques. Nous avons vu aux deux chapitres précédents comment des disciplines comme les sciences sociales et la psychologie cognitive peuvent générer des principes. Nous nous intéressons dans ce chapitre aux propriétés. Nous définissons d'abord ce que sont les propriétés et la façon de les utiliser. Puis nous présentons des propriétés d'utilisabilité proposées pour les systèmes mono-utilisateurs. Nous étendons les propriétés d'utilisabilité aux systèmes multi-utilisateurs. Nous présentons ensuite les propriétés de qualité du logiciel. Nous introduisons aussi les propriétés CARE pour les systèmes multimodaux. Nous montrons que ces propriétés sont génériques et peuvent être appliquées à d'autres cas : les systèmes multi-utilisateurs et notamment les systèmes de communication homme-homme médiatisée. Enfin, nous présentons d'autres propriétés spécifiques aux systèmes multi-utilisateurs et de communication homme-homme médiatisée.

4.2. Définition et utilisation des propriétés

Les propriétés sont des caractéristiques objectivement vérifiables d'un système interactif. Une propriété est mesurable et peut être exprimée sous la forme d'un prédicat dans un système formel approprié [Salber 1994]. On pourra consulter le chapitre 9 de [Dix 1993] pour des exemples de formalisation mathématique des propriétés. Notons que cet ouvrage appelle "principes"[1] les propriétés. Les propriétés présentent les qualités d'un outil scientifique. Elles sont formalisables et procurent une caractérisation d'un système qui est vérifiable, soit expérimentalement directement sur le système, soit mathématiquement sur une description formelle du système. D'autre part, leur caractère général en fait un outil réutilisable pour différents systèmes. Les propriétés ne sont pas liées à un système particulier mais à une classe de systèmes.

Comme nous l'avons précisé au chapitre 1, toutes les propriétés ne sont pas souhaitables pour tous les systèmes. C'est là que se situe une des difficultés de l'usage des propriétés. Le choix d'un jeu de propriétés pertinent pour l'analyse d'un système donné requiert une expertise. Notre grille d'analyse qui tente de lier les propriétés à des principes de plus haut niveau laisse entrevoir la possibilité d'une modélisation systématique. Mais une telle approche se heurterait néanmoins à un obstacle : nous avons vu que l'ensemble des principes est étroitement lié à un contexte de développement, et qu'il n'est pas possible de

[1] "Principles" en anglais.

présenter un ensemble exhaustif de principes. Il en est de même pour les propriétés : nous présentons dans la suite de ce chapitre des propriétés pertinentes pour l'étude des systèmes multi-utilisateurs, mais nous ne pouvons garantir que nos propriétés sont exhaustives. Des préoccupations spécifiques à un système donné pourraient faire apparaître de nouvelles propriétés. En fait, on retrouve ici un débat plus général lié aux modélisations : dans une activité de modélisation, l'expertise de l'utilisateur du modèle est un paramètre dont il faut tenir compte. Il n'est pas toujours possible de distinguer dans les résultats d'une activité de modélisation ce qui est inspiré par le modèle lui-même et ce qui découle de l'expertise de l'utilisateur du modèle.

En résumé, les propriétés présentent l'intérêt d'être un outil scientifique et sont applicables aussi bien pour l'analyse ou la critique d'un système existant que pour la conception d'un nouveau système. Mais nous pensons qu'une certaine expertise est nécessaire pour en tirer pleinement parti.

4.3. Propriétés d'utilisabilité des systèmes mono-utilisateurs

Nous donnons quelques exemples de propriétés d'utilisabilité des systèmes mono-utilisateurs qui sont particulièrement intéressantes pour les systèmes multi-utilisateurs. On trouvera dans [Coutaz 1992], [Salber 1994] et [Dix 1993] ainsi que dans l'annexe A un ensemble de propriétés plus complet pour les systèmes mono-utilisateurs. Nous distinguons deux classes de propriétés d'utilisabilité : des propriétés de présentation et des propriétés du dialogue. La première classe de propriétés caractérise l'aspect présentation du système interactif. La seconde classe caractérise le dialogue que permet le système.

Les propriétés de la classe présentation s'appliquent aux éléments d'interaction proposés par le système. Nous présentons d'abord une propriété générale d'un artefact, l'"affordance". Nous définissons ensuite la propriété d'observabilité et plusieurs de ses cas particuliers, propriétés moins strictes adaptées à des conditions particulières.

- Affordance
  La propriété d'*affordance* est issue de l'approche écologique de la perception [Gibson 1979]. Nous gardons le terme anglais qui n'a pas d'équivalent exact en français. L'affordance indique qu'un artefact, par son aspect perceptible, incite l'utilisateur à effectuer les actions que permet cet artefact. Par exemple, un bouton suggère que l'on peut appuyer dessus. Norman propose l'exemple célèbre des poignées de portes dont la disposition et la forme inciteront plutôt à tirer, pousser ou tourner [Norman 1988]. L'affordance peut aussi exprimer une propriété plus

générale d'un système ou d'une classe de systèmes. Gaver par exemple décrit les affordances des mediaspaces pour la collaboration, telles que la possibilité de collaboration entre utilisateurs distants, ou l'anisotropie des communications audio/vidéo qui permet de regarder sans être vu ou de partager un bureau à travers le mediaspace [Gaver 1992].

- Observabilité

  L'*observabilité* (*observability*) est la capacité du système à rendre perceptibles à l'utilisateur les variables d'état internes pertinentes pour la tâche en cours. Les variables pertinentes sont déterminées lors de l'analyse de tâche. La figure 4.1 présente une boîte de dialogue du Finder du Macintosh qui indique la progression d'une opération de copie.

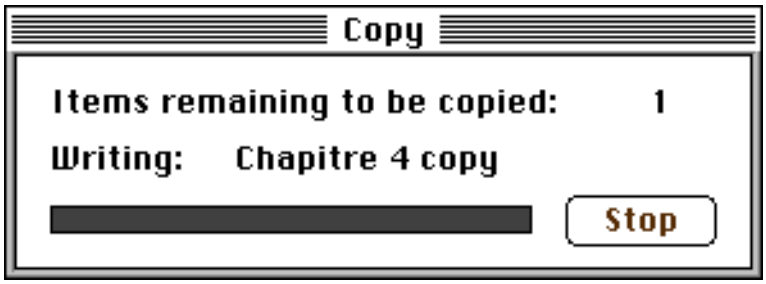


Figure 4.1. Boîte de dialogue affichant la progression d'une opération de copie. En haut, le système rend observable le nombre d'éléments restant à copier. Cependant la barre de progression est ambiguë car l'utilisateur n'a pas de moyen de savoir quelle variable interne elle représente.

Le système rend observable le nombre d'éléments restant à copier. Le système affiche aussi une barre de progression tout au long de la copie. Mais cette barre de progression est ambiguë : l'utilisateur n'a pas de moyen de savoir quelle variable interne elle représente. Sa longueur est-elle proportionnelle au nombre de fichiers, à la taille des fichiers, ou au temps écoulé ? Cet exemple montre qu'il ne suffit pas de présenter l'information pertinente. Il faut aussi donner à l'utilisateur les moyens de l'interpréter correctement. La propriété suivante, l'honnêteté, affine l'observablité pour en tenir compte. Nous donnons au chapitre suivant un moyen de vérifier la propriété d'observabilité à partir d'une description semi-formelle de l'interface. Au paragraphe 4.6, nous affinons la propriété d'observabilité pour les systèmes multi-utilisateurs avec la propriété d'observabilité publiée.

- Honnêteté

  L'*honnêteté* (*honesty*)[1] est un affinement de l'observabilité. Le système est dit honnête si 1) le rendu de l'état observable est conforme à l'état interne et si 2) la forme de ce rendu conduit l'utilisateur à interpréter correctement cet état : le résultat de cette interprétation est conforme à la valeur de l'état interne. L'honnêteté implique l'observabilité, l'état interne pertinent doit être perceptible, mais aussi la conformité entre l'état interne et son rendu, et aussi la conformité entre l'état interne et la représentation mentale de cet état élaborée par l'interprétation du rendu. Dans [Salber 1995], nous étudions en détail un exemple où l'honnêteté du système ne peut être garantie. Cet exemple étudie le contrôle d'accès aux individus dans un mediaspace. Chaque utilisateur dispose de l'interface présentée figure 4.2 pour établir des connexions audio/vidéo avec les autres utilisateurs.

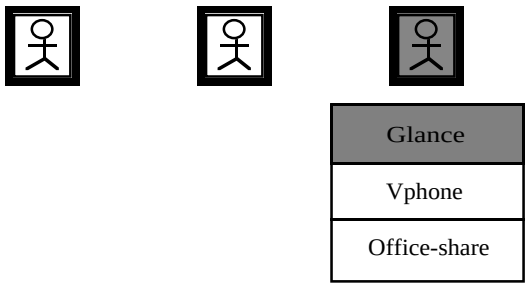


Figure 4.2. Interface permettant d'établir des connexions audio/vidéo dans un mediaspace. Lorsque l'utilisateur clique sur l'icône représentant un autre utilisateur, un menu lui indique les types de connexions que l'utilisateur distant a autorisé, ici le vphone et l'office-share.

  Dans le cas où les droits d'accès sont gérés localement par chaque site et répliqués sur tous les sites, l'honnêteté peut être compromise. Lorsque l'utilisateur distant modifie ses droits d'accès, la modification est ensuite répercutée sur tous les autres sites. Dans l'intervalle de temps correspondant à cette mise à jour, il est possible que le système présente à l'utilisateur une information inexacte car périmée. A cause des délais de transmission dûs au réseau, l'honnêteté ne peut être garantie "qu'à un epsilon près". Il s'agit alors d'une "honnêteté relâchée".

- Parcourabilité

  La *parcourabilité* (*browsability*) est une forme relâchée de l'observabilité. Elle n'impose plus que l'état du système soit directement perceptible à l'utilisateur, mais permet que des commandes syntaxiques[2] soient exécutées pour rendre cet

[1] Le terme français *honnêteté* est plus connoté que le terme anglais *honesty*. On pourra lui préférer "véracité".
[2] On trouve aussi les termes "commande articulatoire" ou "commande passive".

état perceptible. Une *commande syntaxique* est une commande qui s'adresse à l'interface et ne modifie pas l'état du noyau fonctionnel. L'utilisateur peut ainsi modifier la vue du noyau fonctionnel que lui présente l'interface. Un exemple typique de parcourabilité est donné par la fonction "scroll". Un grand document peut être présenté dans une fenêtre agrémentée de scrollbars. Toute l'information présentée dans le document n'est pas observable simultanément. Mais l'utilisateur peut effectuer une commande syntaxique (scroller) pour parcourir l'ensemble du document.

Les propriétés que nous venons d'exposer caractérisent la partie présentation d'un système interactif. La seconde classe de propriétés d'utilisabilité que nous présentons maintenant concerne le dialogue qu'offre un système interactif à l'utilisateur.

- Dialogue à fils multiples

  La propriété de *dialogue à fils multiples* (*multi-threading*) définit la capacité du système à permettre la réalisation de plusieurs tâches. On distingue deux types de dialogues à fils multiples : parallèle et entrelacé. Dans le cas entrelacé, les tâches peuvent être simultanées au sens de l'utilisateur, mais à un instant donné, le dialogue est restreint à une seule tâche. Dans le cas parallèle, le dialogue n'est pas restreint et peut concerner plusieurs tâches. Par exemple, avec le système du Macintosh [Apple 1994] lorsque l'on réalise une copie avec le Finder, il est possible d'effectuer une autre tâche dans une autre application. Il y a alors entrelacement : à un instant donné, une seule application est active (c'est-à-dire capable de traiter les actions de l'utilisateur). Un système autorisant l'usage de la parole et de la manipulation directe, comme l'éditeur de dessin ICP-Draw [Bourguet 1992], permet un parallélisme vrai et un dialogue à fils multiples parallèle. La notion de dialogue à fils multiples est à rapprocher de la propriété suivante, la non-préemption. Dans le cas d'un système multi-utilisateurs, il convient de s'interroger sur les possibilités de parallélisme et d'entrelacement entre les tâches de différents utilisateurs. Cet aspect correspond à la dimension coordination de notre modèle du trèfle du chapitre 1.

- Non-préemption

  La *non-préemption* est liée à la notion d'initiative. La non-préemption est assurée si l'utilisateur a toujours l'initiative du dialogue. Si le système contraint l'utilisateur à effectuer une action (par exemple cliquer le bouton "OK" d'un message d'erreur) avant de pouvoir poursuivre sa tâche courante, il y a préemption. Notons que l'on peut distinguer deux types de préemption : la préemption globale interdit à l'utilisateur d'effectuer toute autre action que celle

requise par le système. C'est souvent le cas des messages d'erreur. La préemption locale ne bloque qu'un fil de dialogue et laisse l'utilisateur continuer les autres fils de dialogue. Toutefois, le fil de dialogue "préempté" ne pourra être repris qu'après que l'utilisateur ait satisfait la requête du système. Il y a aussi préemption lorsque le système est ininterruptible. Dans le cas des systèmes multi-utilisateurs, il faut aussi considérer un autre cas de préemption : la préemption d'une ressource partagée par un des utilisateurs. Par exemple dans les systèmes à "tour de parole" (*turn-taking*), chaque utilisateur peut accaparer à tour de rôle un curseur partagé.

- Accessibilité

  L'*accessibilité* (*reachability*) définit la possibilité de navigation dans l'ensemble des états observables d'un système. Un état q est accessible à partir d'un état p s'il existe une suite de commandes $\{ c_i \}$ qui font passer de l'état p à l'état q. On voudra vérifier que l'ensemble des états est accessible par l'utilisateur. Nous donnons un moyen de vérifier cette propriété au chapitre suivant.

- Annulabilité

  L'*annulabilité* définit la capacité qu'a l'utilisateur de revenir de l'état courant dans un état antérieur. Même si cette propriété semble simple (nous avons tous déjà utilisé le service "annuler" ou "undo"), elle recouvre en fait plusieurs aspects. Par exemple, dans beaucoup de systèmes, on ne peut annuler que les commandes non syntaxiques de l'application, c'est-à-dire celles qui modifient le noyau fonctionnel. Nous n'avons jamais vu par exemple, un système dans lequel on puisse annuler le déplacement d'une fenêtre, ou une opération de scroll. On peut aussi considérer des niveaux d'annulation : en s'appuyant sur un historique des actions de l'utilisateur, il est possible d'annuler la dernière commande, mais également toute commande précédente. Dans ce cas, on distingue en général un service "annuler" et un service "refaire" qui permet d'annuler l'effet de la dernière commande annuler. [Kosbie 1994] propose une gestion de l'historique des actions de l'utilisateur adaptée à diverses formes d'annulabilité. Dans le cas multi-utilisateurs, l'annulabilité peut encore prendre d'autres formes [Young 1994]. Les propriétés de récupérabilité avant et arrière (*forward and backward recovery*), de réparation (*repair*), que nous ne présentons pas ici, sont liées à l'annulabilité. On pourra consulter [Salber 1994] pour une définition de ces propriétés (voir aussi l'annexe A).

- Temps de réponse

  Nous distinguons deux propriétés liées au temps de réponse du système : la conformité et la stabilité. Ces propriétés prennent une dimension nouvelle avec les

systèmes multi-utilisateurs. En effet, l'utilisation d'un réseau de communication introduit souvent un élément non-déterministe qui a une influence sur le temps de réponse du système.
La *conformité du temps de réponse* mesure la capacité du système à réagir dans un laps de temps en accord avec l'attente de l'utilisateur.
La *stabilité du temps de réponse* dénote la capacité du système à entretenir un "même" temps de réponse pour un traitement donné.

Avec les systèmes multi-utilisateurs, la possibilité pour chaque utilisateur d'adapter le système à ses besoins et ses habitudes est particulièrement importante. La propriété de configurabilité prennent en compte cette nécessité. [Mackay 1991] présente une étude sur la façon dont les utilisateurs adaptent leur environnement de travail. Cette étude, réalisée avec des utilisateurs d'un environnement destiné à des experts (Unix), montre que la configuration d'un environnement est une tâche difficile. Ce travail plaide pour une configurabilité plus accessible à des utilisateurs non-experts.

- Configurabilité
  La configurabilité (*customizability*) définit la modifiabilité de l'interface par l'utilisateur du système. Comme pour l'annulabilité, cette propriété recouvre plusieurs aspects. Notons que le vocabulaire n'est pas exactement défini pour les divers affinements de la configurabilité. On pourra trouver dans la littérature les termes *tailorability*, *adaptivity*, *adaptability*, *parameterazibility* avec diverses définitions [Coutaz 1992], [Dix 1993]. La configurabilité peut se rencontrer à différents niveaux et sous différentes formes. L'utilisateur peut modifier la présentation, le dialogue, voire le comportement de certains aspects du système. Un système comme OpenDoc [Apple 1993] autorise une grande flexibilité et permet à l'utilisateur de construire son environnement en assemblant des composants logiciels de haut niveau interopérables. Le niveau le plus élémentaire de la configurabilité est la *paramétrisation*. L'utilisateur peut modifier des paramètres ou préférences du système. Ces paramètres concernent la présentation (par exemple les couleurs des fenêtres) ou le dialogue (par exemple, dans Word, l'utilisateur peut choisir de se voir suggérer de remplir un formulaire "résumé" à chaque création de document). L'identification des paramètres pertinents sont fournis par l'analyse de tâche et la définition d'un modèle d'utilisateur. Remarquons que beaucoup de dialogues de réglages des préférences comportent un choix "valeurs par défaut" qui fixe les réglages à des valeurs déterminées par le concepteur. Cette caractéristique tient compte des utilisateurs non-experts. Un cas

extrême de configurabilité est fourni par la possibilité d'"attachabilité"[1] du système OpenDoc. Ce système repose sur le langage de script AppleScript. L'attachabilité permet de redéfinir le comportement de n'importe quel élément d'interaction (par exemple un bouton) en glissant-déposant un script sur cet élément d'interaction. Notons que cette utilisation des langages de script (interprétés donc prêts à l'emploi sans compilation) est particulièrement intéressante. Elle permet de contraindre la distinction entre mécanisme et politique d'utilisation en exprimant la politique sous forme d'un script. Cette approche a des applications à certains services des systèmes multi-utilisateurs, comme la mise en œuvre du contrôle d'accès. Nous avons expérimenté cette approche dans [Berne 1995] pour le contrôle d'accès à une caméra mobile. Le modèle réflectif proposé par [Dourish 1995] est une généralisation particulièrement intéressante de cette technique. Nous y reviendrons au chapitre 8.

Les propriétés que nous venons de voir concernent l'utilisabilité d'un système interactif. Nous présentons maintenant des propriétés centrées sur les préoccupations du génie logiciel.

4.4. Propriétés de qualité du logiciel

Les propriétés de qualité du logiciel que nous présentons maintenant correspondent aux facteurs proposés par McCall [McCall 1977]. Ici encore, nous ne visons pas l'exhaustivité et nous ne présentons que trois propriétés représentatives et pertinentes pour les systèmes multi-utilisateurs : la réutilisabilité, la portabilité, et l'interopérabilité. Nous utiliserons ces propriétés pour vérifier la validité du modèle d'architecture au chapitre 7. Nous les utiliserons aussi pour évaluer les outils au chapitre 8.

- Réutilisabilité
  La *réutilisabilité* définit dans quelle mesure un composant logiciel écrit dans le cadre de la réalisation d'un système donné peut être réutilisé dans la conception d'un autre système. Cette propriété est étroitement liée à des propriétés d'organisation des composants logiciels d'un système, comme la modularité, l'indépendance fonctionnelle, ou le couplage (on parle aussi d'orthogonalité, nous y reviendrons au chapitre 8). L'approche de conception objet a permis de progresser dans l'étude et la satisfaction effective de la réutilisabilité [Meyer 1990].

[1] "Attachability" en anglais.

- Portabilité
  La *portabilité* exprime l'indépendance du logiciel vis-à-vis de la plate-forme matérielle et logicielle d'accueil. Il nous faut distinguer deux types de portabilité, la portabilité avec et sans recompilation.

  La *portabilité avec recompilation* est la plus courante et a été popularisée par le système Unix. En théorie, les sources d'un logiciel écrit sous Unix peuvent être recompilés sur n'importe quelle variante d'Unix. En pratique, la satisfaction de la propriété de portabilité impose des contraintes au réalisateur. Les services d'un système en effet sont composés de services communs à l'ensemble des systèmes Unix et de services plus spécifiques, propres à chaque plate-forme d'accueil. Le réalisateur doit donc se limiter à n'utiliser que les services communs à tous les systèmes Unix pour que son logiciel soit effectivement portable. La portabilité peut aussi s'envisager pour l'interface graphique. Des systèmes de boîte à outils virtuelles ou des outils de développement d'interface multi-plate-formes permettent de satisfaire dans une certaine mesure la portabilité. Le modèle d'architecture Arch prévoit explicitement un composant logiciel pour assurer la portabilité. Nous présentons au chapitre 8 des outils de construction d'interfaces qui visent à satisfaire la portabilité.

  La *portabilité sans recompilation* connaît une vogue récente. Cette possibilité repose sur l'utilisation d'interpréteurs ou d'émulateurs. Les émulateurs sont déjà couramment utilisés, par exemple pour les logiciels de communication. Il s'agit de permettre l'exécution d'un code objet spécifique d'une plate-forme X sur une plate-forme Y. L'émulateur traduit le code X en code Y à l'exécution. Cette approche a été appliquée avec succès par Apple pour la transition de ses matériels vers le microprocesseur PowerPC. L'émulateur SoftWindows sur Macintosh [Insignia 1994] permet d'exécuter sans recompilation des exécutables Windows sur Macintosh. Bien sûr, la traduction du code objet par l'émulateur ralentit l'exécution, mais la puissance des processeurs actuels et de nouvelles techniques d'optimisation permettent d'obtenir des performances acceptables. Notons que la technique de *translation*, qui consiste à effectuer une traduction directe du code objet avant exécution, permet de s'affranchir du délai de traduction lors de l'exécution.

Notons que la portabilité peut être examinée de façon plus précise et pour un même type de plate-forme. Par exemple, la taille de l'écran ou la vitesse du processeur doivent être prises en compte pour le rendu correct des animations sur une même famille de machines. La propriété de portabilité est particulièrement importante pour les systèmes multi-

utilisateurs. Elle permet en effet de prendre en compte les environnements hétérogènes. Cette propriété est liée à la propriété d'interopérabilité.

- Interopérabilité
  L'*interopérabilité* définit la capacité qu'ont plusieurs systèmes de s'interfacer. Cette propriété est communément satisfaite par l'utilisation d'un protocole de communication commun aux deux systèmes, comme TCP/IP. Elle suppose aussi la définition des informations pouvant être échangées entre les deux systèmes, ainsi que le format que doivent respecter ces informations. Notons que l'interopérabilité peut être envisagée à un haut niveau d'abstraction. C'est par exemple le cas du système OpenDoc. Le système OpenDoc promet d'être interopérable sur plusieurs plates-formes matérielles. Quelle que soit la configuration (réseau ou local), les composants communiquent en échangeant des scripts AppleScript. Ces scripts utilisent un vocabulaire normalisé d'objets, adapté à différentes classes d'outils et de tâches. Par exemple, la classe "traitement de texte" définit des objets *chapitre*, *section* ou *paragraphe* et les opérations possibles sur ces objets (suppression, déplacement, changement de style, etc.). Cette approche, qui vise à normaliser les échanges entre des composants logiciels de haut niveau, nous semble particulièrement intéressante pour les systèmes multi-utilisateurs. Il est ainsi envisageable de faire interopérer deux traitements de texte quelconques. Cette interopérabilité à haut niveau permet d'obtenir une grande configurabilité au niveau système : chaque utilisateur peut utiliser son traitement de texte habituel, et collaborer avec d'autres utilisateurs utilisant un traitement de texte différent.

Les trois propriétés que nous venons de présenter définissent des propriétés du logiciel. Pour le cas particulier des systèmes multimodaux, les propriétés CARE permettent de réfléchir aux combinaisons possibles des dispositifs physiques et des langages d'interaction. Nous présentons ces propriétés et nous verrons comment elles peuvent être réutilisées dans d'autres cas que les systèmes multimodaux.

4.5. Les propriétés CARE pour les systèmes multimodaux

Les propriétés CARE définissent quatre propriétés (Complémentarité, Assignation, Redondance, Equivalence) qui peuvent être appliquées à deux niveaux : les dispositifs physiques et les langages d'interaction des systèmes multimodaux [Nigay 1994]. Le *dispositif physique* définit la réalité physique observable : un clavier, une souris, un écran, ou n'importe quel transducteur de propriétés physiques en informations numériques et réciproquement. Le *langage d'interaction* est un système conventionnel de

signes qui assure la communication d'informations entre l'utilisateur et le système, par exemple un langage de commande. En utilisant un langage d'interaction à travers un dispositif physique, l'utilisateur et le système échangent des *expressions*.

Les propriétés CARE caractérisent les relations possibles entre dispositifs physiques et langages d'interaction pour la réalisation d'une tâche. Quatre cas sont envisagés :

- Complémentarité
  La *complémentarité entre dispositifs physiques* est obtenue si l'expression peut être décomposée en deux sous-expressions, chacune étant formée en utilisant un dispositif différent. Par exemple, pour ouvrir un document depuis le Finder du Macintosh, on peut utiliser la souris pour sélectionner le document puis taper l'équivalent clavier ⌘O.
  La *complémentarité entre langages d'interaction* correspond à l'utilisation de plusieurs langages pour former une expression. Par exemple, avec le système de reconnaissance de la parole PlainTalk sur Macintosh, il est possible d'utiliser deux langages pour ouvrir un fichier : la manipulation directe pour sélectionner un document du Finder, puis le langage de commande du reconnaisseur de parole en prononçant "open this".

- Assignation
  L'*assignation des dispositifs physiques* traduit le fait que l'utilisateur (ou le système) est contraint d'utiliser un dispositif physique donné pour former une expression. Par exemple, un menu pour lequel il n'existe pas d'équivalent clavier doit être manipulé exclusivement à la souris.
  L'*assignation des langages d'interaction* traduit qu'un langage donné doit être utilisé pour former une expression. Par exemple, avec un éditeur de dessin classique, la création d'un rectangle ne peut être effectué que par manipulation directe.

- Equivalence
  L'*équivalence entre dispositifs physiques* traduit la possibilité de choix entre dispositifs pour former une expression. Les équivalents-clavier des menus en sont un exemple courant : l'utilisateur a le choix entre la souris et le clavier pour sélectionner l'item d'un menu.
  L'*équivalence entre langages d'interaction* correspond à la possibilité de choix entre langages pour former une expression. Par exemple, dans le système MATIS d'information sur les transports aériens, l'utilisateur a le choix de remplir un

formulaire ou de prononcer une phrase dans un langage de commande pour exprimer une requête [Nigay 1994].

- Redondance
  La *redondance des dispositifs physiques* exprime que plusieurs dispositifs peuvent être utilisés simultanément et qu'ils peuvent être utilisés pour transmettre la même information (ils sont donc aussi équivalents mais de plus, ils sont utilisés simultanément). Ce cas correspond par exemple à l'utilisation d'un reconnaisseur de parole utilisé conjointement avec une caméra observant le mouvement des lèvres afin de rendre plus robuste la reconnaissance de la parole.
  La *redondance des langages d'interaction* correspond à l'utilisation simultané de langages équivalents. Les systèmes de télé-exploration qui exigent une grande fiabilité en offrent un exemple : l'utilisateur donne des instructions à un robot distant par manipulation directe et confirme simultanément ces instructions en utilisant un langage de commande parlé.

Ces propriétés sont présentées plus en détail et sont formalisées dans [Coutaz 1995]. On trouvera aussi dans cet article une extension des propriétés CARE du point de vue des capacités cognitives de l'utilisateur. Notons que dans notre présentation nous avons considéré uniquement les dispositifs et langages pour l'entrée d'informations. Les propriétés CARE sont aussi applicables à l'analyse des expressions en sortie. Nous allons maintenant voir que les propriétés CARE peuvent être appliquées à l'analyse des systèmes multi-utilisateurs.

4.6. Propriétés des systèmes multi-utilisateurs et des systèmes de communication homme-homme médiatisée

Pour les systèmes multi-utilisateurs, nous examinons d'abord l'application des propriétés CARE aux systèmes multi-utilisateurs en général puis aux systèmes de communication homme-homme médiatisée. Nous présentons ensuite d'autres propriétés spécifiques à ces deux types de systèmes.

4.6.1. Extension des propriétés CARE

Les propriétés CARE peuvent être appliquées au cas général des systèmes multi-utilisateurs. Les propriétés CARE distinguent des dispositifs physiques et des langages d'interaction pour la réalisation d'une tâche donnée. Il est possible de transposer cette approche aux cas des utilisateurs et des rôles pour une tâche dans un système multi-utilisateur. Par exemple, la *complémentarité* entre rôles se rencontre dans les jeux : deux joueurs ne peuvent pas jouer l'un sans l'autre. Nous verrons aussi un exemple de

complémentarité entre les compères du système NEIMO au chapitre suivant. L'*assignation* d'une tâche à un rôle est un exemple commun : par exemple, seul l'administrateur système pourra effectuer la configuration du système. La *redondance* entre les rôles pour une même tâche est aussi possible : par exemple dans un éditeur partagé, un rédacteur et un lecteur peuvent annoter le document en même temps. Nous verrons un autre exemple de redondance avec les observateurs du système NEIMO au chapitre suivant. Enfin, l'*équivalence* correspond au cas où deux rôles peuvent indifféremment exécuter la même tâche. Par exemple dans un système médical, le médecin ou l'infirmière peuvent modifier une même partie du dossier d'un patient. Les propriétés CARE peuvent aussi être utilisées pour réfléchir aux utilisateurs qui doivent réaliser une tâche. Dans ce cas des phénomènes de délégation sociale peuvent par exemple avoir lieu. On peut de la même façon étudier la complémentarité, l'assignation, la redondance et l'équivalence des utilisateurs pour une tâche en fonction de leur statut ou de leur compétence par exemple. Dans ce cas, des phénomènes de délégation sociale peuvent aussi intervenir (voir au chapitre 6). Cependant, l'aspect le plus intéressant et qu'il faut prendre en compte dans la conception logicielle des systèmes multi-utilisateurs est l'application des propriétés CARE aux rôles. CARE pour les utilisateurs relève plus d'un protocole social qui ne sera en général pas implémenté dans le système.

Les propriétés CARE peuvent aussi être appliquées à l'usage des moyens technologiques de communication. La *complémentarité* peut être obtenue par combinaison de différents outils de communication. Par exemple, le service TPC.INT [Savetz 1994] permet en émettant un courrier électronique d'envoyer en fait un fax à son correspondant. De même les télégrammes acheminés par la poste sont en général transmis par téléphone ou par télex jusqu'au bureau distributeur où ils sont retranscrits sur papier et distribués. L'*équivalence* est un cas de plus en plus courant : pour transmettre une information à un correspondant, nous avons souvent le choix entre le fax, le téléphone ou le courrier électronique. Bien entendu, le choix répondra à des contraintes (par exemple, l'urgence du message), et n'aura pas nécessairement le même contenu suivant l'outil de communication utilisé. Cependant, au niveau de la tâche globale (transmettre une information), les différents outils sont équivalents et permettent tous de réaliser la tâche. L'*assignation* est un cas que l'on rencontre aussi dans la vie courante : par exemple, un correspondant en déplacement ne sera joignable que par son téléphone de voiture. Enfin la *redondance* peut être parfois observée, en particulier lorsque la fiabilité d'un moyen de communication n'est pas assurée. Par exemple, si l'on utilise le courrier électronique qui ne permet pas toujours d'avoir un accusé de réception, on pourra vouloir confirmer en même temps le message par téléphone ou par fax.

Ces deux exemples montrent l'intérêt des propriétés CARE et la possibilité de les étendre à d'autres cas que les systèmes multimodaux. Nous présentons maintenant des propriétés originales des systèmes multi-utilisateurs.

4.6.2. Propriétés des systèmes multi-utilisateurs

Une caractéristique importante d'un système multi-utilisateur est que les utilisateurs n'utilisent pas le système isolément les uns des autres. Notre modèle du trèfle du chapitre 1 nous indique en effet que les tâches de coordination et de communication sont des composantes importantes. Toutes deux nécessitent que chaque utilisateur puisse avoir connaissance des activités des autres utilisateurs. Mais ce lien entre les utilisateurs peut être plus ou moins fort. Nous utilisons ici une distinction établie par [Buxton 1994] entre tâches de premier plan et tâches d'arrière-plan. Selon cette classification, une tâche de premier plan est une tâche consciente et intentionnelle de l'utilisateur. Cette définition correspond à la définition courante de la tâche en interface homme-machine. Une tâche d'arrière-plan est une tâche qui n'est ni nécessairement consciente ni même nécessairement intentionnelle. Il s'agit d'une tâche "périphérique", qui n'est pas le centre d'intérêt principal de l'utilisateur. Cette distinction nous permet de présenter deux propriétés qui caractérisent le lien par lequel chaque utilisateur est informé de l'activité des autres utilisateurs : la rétroaction de groupe lorsque cette information participe à une tâche de premier plan et l'"awareness" lorsque cette information est une tâche périphérique.

- Rétroaction de groupe
  La *rétroaction de groupe* (parfois appelée *feedthrough*) est l'extension de la classique propriété de retour d'information (ou *feedback*) au cas des systèmes multi-utilisateurs. Elle caractérise le fait que chaque utilisateur dispose d'une information sur les actions des autres utilisateurs et que cette information est pertinente pour la réalisation de la tâche commune. Les scrollbars multi-utilisateurs de l'éditeur SASSE [Baecker 1992] (cité dans [Karsenty 1994]) permettent à chaque utilisateur de connaître la position des autres dans le document commun. Dans l'éditeur de dessin partagé GroupDesign [Karsenty 1994], plusieurs stratégies de rétroaction de groupe ont été étudiées, par exemple en utilisant des retours d'information sous forme graphique ou sonore, ou en combinant les deux.

La rétroaction de groupe est lié à un concept bien connu des systèmes multi-utilisateurs : le WYSIWIS[1]. Il stipule que chaque utilisateur est informé des actions des autres utilisateurs au moment où elles ont lieu. Le WYSIWIS relaxé autorise un certain délai dans la répercussion des actions. Plusieurs stratégies sont envisageables pour mettre en

[1] What You See Is What I See

œuvre le WYSIWIS relaxé. L'éditeur de textes partagé Alliance par exemple [Decouchant 1994] permet le WYSIWIS relaxé et les utilisateurs peuvent négocier le délai des répercussion des actions. De notre point de vue, le WYSIWIS strict et l'awareness que nous présentons ci-dessous sont les deux extrêmes d'un continuum qui définit différentes possibilités de WYSIWIS plus ou moins relaxé.

- Awareness
  L'*awareness*[1] traduit la possibilité pour chaque utilisateur d'être informé de l'état ou des actions des autres utilisateurs de façon périphérique (au sens de Buxton). Ce retour d'information n'est pas nécessairement pertinent pour la tâche de l'utilisateur ou du groupe mais contribue à rendre chaque utilisateur conscient de l'activité du groupe. L'exemple le plus représentatif en est le système Portholes [Dourish 1992]. Portholes, qui repose sur le mediaspace RAVE, présente à chaque utilisateur un ensemble de photos des autres membres du groupe, prises à intervalles de quelques minutes. Il donne ainsi une vue synthétique de l'activité du groupe. Notons que ces informations périphériques d'arrière-plan peuvent passer en premier plan si la tâche de l'utilisateur change. Par exemple, s'il a tout à coup besoin d'une information qu'un autre utilisateur peut lui fournir, il peut utiliser Portholes pour déterminer si cet autre utilisateur est disponible et ouvrir une connexion vidéo avec lui. Le système Khronika, qui est lié à Portholes, généralise cette notion d'awareness [Lövstrand 1991]. Le système centralise des événements génériques fixés par chaque utilisateur, par exemple une visite, un rendez-vous, voire un événement automatique généré par un capteur, comme le fait qu'il commence à pleuvoir. Pour avertir les utilisateurs concernés, le système envoie des notifications sous forme de sons, de messages ou de courriers électroniques.

Nous avons vu au chapitre 2 que l'enjeu des tâches est un élément important dans les activités d'un membre d'un groupe. Les propriétés de rétroaction de groupe et d'awareness sont adaptées pour prendre en compte cette notion. Lorsqu'une tâche peut avoir une influence sur les tâches des autres membres du groupe, la propriété de viscosité, que nous définissons maintenant, doit aussi être considérée.

- Viscosité
  La *viscosité* caractérise un phénomène social. Mais dans certains cas, le système doit pouvoir la prendre en compte. Elle indique que l'action d'un utilisateur a un effet secondaire sur les tâches des autres utilisateurs. Cet effet secondaire peut

[1] Nous conservons ici le terme anglais qui n'a pas d'équivalent exact en français. La traduction la plus proche que nous pouvons en proposer est "conscience de groupe". Notons que ce terme est parfois traduit par "rétroaction de groupe". Comme nous le montrons, il s'agit de deux concepts différents.

consister en une charge de travail supplémentaire. L'harmonisation des termes dans l'écriture d'un article avec un éditeur de texte partagé en est un exemple. Si l'un des auteurs décide de changer un terme par un autre dans sa partie, chacun des autres auteurs devra faire de même pour assurer la cohérence de l'article. Ces tâches supplémentaires pourraient être prises en compte par le système, mais nous ne connaissons pas d'exemple de système de ce type.

Lorsque plusieurs utilisateurs travaillent ensemble à la réalisation d'une tâche, il est indispensable de prendre en compte leurs différences de compétences, d'habitudes, etc. Nous avons vu précédemment que la propriété de configurabilité tenait compte de cet aspect. La possibilité d'adapter le rythme de l'interaction est une spécialisation de la configurabilité et est aussi liée aux différentes variantes du WYSIWIS dont nous avons parlé plus haut. Elle est aussi inspirée par la propriété correspondante pour les systèmes mono-utilisateurs.

- Contrôle du rythme de l'interaction
  Le *contrôle du rythme de l'interaction* indique la possibilité pour chaque utilisateur de fixer le rythme de son interaction à la fois avec le système et avec les autres utilisateurs. Pour un éditeur partagé par exemple, elle recouvre la configurabilité du délai du WYSIWIS relaxé. On en trouve aussi un exemple dans les mailing-lists : il est souvent possible de choisir si l'on veut recevoir les messages des autres adhérents dès qu'ils sont postés, ou bien sous forme d'un "digest" quotidien qui regroupe tous les messages du jour.

Dans un système multi-utilisateur, l'identification des utilisateurs est une caractéristique sur laquelle il faut réfléchir lors de la conception. Par exemple, à un utilisateur peut être associé systématiquement un rôle. Ou bien au contraire on souhaitera que les utilisateurs puissent être anonymes, par exemple pour satisfaire à des principes éthiques. Nous définissons maintenant la propriété d'identification.

- Identification
  L'*identification* définit sous quelle forme les utilisateurs sont connus par le système. Ils peuvent être connus sous leur nom réel, sous un pseudonyme ou encore être anonymes. Sur Internet par exemple, la plupart des utilisateurs sont identifiés par leur nom réel. Cependant, certains peuvent utiliser un nom d'emprunt (c'est le cas par exemple des utilisateurs du service America Online). Notons que dans le cas d'un nom d'emprunt, le nom réel peut quand même être connu du système mais ne pas être diffusé. Il s'agit là d'un cas particulier de la propriété d'observabilité publiée que nous définissons ci-dessous. Enfin, en

utilisant un “remailer” qui va supprimer toute identification, les utilisateurs peuvent par exemple envoyer des messages anonymes à certains newsgroups.

L’observabilité publiée est un cas particulier de la propriété d’observabilité [Salber 1995]. Dans une activité de groupe, l'observabilité se heurte au principe de protection de la vie privée. L'espace privé peut se définir comme l'ensemble des variables d'état "personnelles" (par exemple, le fait que X est dans son bureau en train de lire son courrier électronique). Le caractère personnel d'une variable se définit dès l'analyse du problème. Si l'observabilité des variables personnelles peut être pertinente pour le groupe, elle est potentiellement contraire au principe du respect de la vie privée.

- Observabilité publiée
  L’*observabilité publiée* caractérise le fait que les variables d'état personnelles sont rendues observables seulement si le propriétaire en autorise la publication. L'autorisation de publication peut être statique ou dynamique. En fait cette première approche peut être affinée : dans [Salber 1995], nous proposons d’associer à chaque variable publiée un *filtre de publication*. On peut alors parler d’*observabilité filtrée*. Par exemple, le fait que X est en train de lire son courrier peut être simplement publié sous la forme “X est indisponible” sans plus de précisions. On peut aussi envisager de définir différents filtres à l’intention de différents utilisateurs. Par exemple X sera indisponible pour tout autre utilisateur à l’exception de ses collaborateurs directs pour lesquels il publiera son activité réelle. Le filtre utilisé sera alors le filtre *identité*. Dans [Salber 1995], nous définissons aussi la propriété de *réflexivité* qui permet à l’utilisateur de vérifier sous quelle forme sont publiées ses variables personnelles. Notons que la propriété d’observabilité publiée ou filtrée déporte la responsabilité depuis le domaine technique vers le domaine social. Rien, sauf peut-être les conventions sociales, n’empêche un utilisateur de définir un filtre qui ne correspond pas à la réalité.

Nous venons de voir que de nouvelles propriétés doivent être définies pour les systèmes multi-utilisateurs. Dans le cas particulier des systèmes de communication homme-homme médiatisée, on doit aussi envisager des propriétés spécifiques.

4.6.3. Propriétés des systèmes de communication homme-homme médiatisée

Pour les systèmes de communication homme-homme médiatisée, nous définissons d’abord des propriétés caractérisant les médias utilisés pour communiquer, puis des propriétés concernant les possibilités d’utilisation de ces médias.

[Clark 1991] définit des propriétés caractérisant les médias utilisés comme canaux de communication dans la communication homme-homme médiatisée. Nous en retenons deux : la rejouabilité et la révisabilité.

- Rejouabilité
  La *rejouabilité* (*reviewability*) définit la possibilité de rejouer un message exprimé sur un média donné. Au téléphone par exemple, il n'est pas possible de réécouter la dernière phrase de son interlocuteur à moins de l'avoir enregistrée. De même dans un mediaspace, une communication audio/vidéo est fugitive et ne laisse pas de trace qui puissent être rejouée. En revanche, le courrier électronique permet de relire un message, ou de réécouter un courrier audio. Cette propriété est à rapprocher de la notion d'historique, courante dans les systèmes multi-utilisateurs. Elle s'applique ici uniquement aux médias de communication.

- Révisabilité
  La *révisabilité* (*revisability*) définit la possibilité de réviser un message avant de l'émettre. Le courrier électronique permet cette opération, mais la communication audio synchrone par exemple, comme le téléphone, l'interdit. Notons que la révisabilité implique la rejouabilité au moins pour l'émetteur, mais que l'inverse n'est pas vrai.

Pour les flots de médias continus, comme le son ou la vidéo, les principes de psychologie cognitive nous indiquent qu'il faut se préoccuper de leur régularité, en particulier pour le son. Autant il est possible, même si cela est désagréable, de suivre une image vidéo dont le débit est irrégulier, autant un débit haché rendra inintelligible un message audio.

- Régularité des flots continus
  La *régularité des flots continus* indique que le débit d'un média continu est constant, dans certaines limites de tolérance. Des données psychologiques expérimentales permettent de déterminer le débit minimal acceptable. Par exemple, si l'on souhaite une communication vidéo de haute qualité, 25 images/seconde est un seuil minimum. Bien sûr, le débit exigé sera imposé par la tâche. S'il s'agit de maintenir l'awareness comme dans Portholes, un débit très faible est tout à fait acceptable.

Lorsque l'on utilise plusieurs médias simultanément pour communiquer, par exemple le son et la vidéo, les principes psychologiques nous demandent de considérer la synchronisation entre les différents médias mis en jeu. Nous avons vu qu'ICS fournit des

principes pour que la combinaison d'informations exprimées dans différents médias puisse avoir lieu.

- Synchronisation
  La *synchronisation* entre médias définit l'écart temporel moyen entre des informations simultanées exprimées dans des médias différents. En effet, le caractère fondamentalement séquentiel des traitements informatiques, des transmissions sur réseau, et du stockage sur mémoire permanente nécessite des traitements adaptés pour garantir la synchronisation. Notons que de nouvelles approches, en particulier pour le stockage sur disque magnétique permettent de stocker en parallèle des informations synchronisées. Là encore, les seuils minimaux acceptables pour l'écart temporel entre les médias est déterminé par la psychologie expérimentale (à peu près 150 ms pour la synchronisation audio/vidéo par exemple).

Lorsque l'on considère la communication d'informations, il faut aussi s'intéresser à la propriété d'intégrité.

- Intégrité
  L'*intégrité* d'un canal de communication est vérifiée si les messages transmis sur ce canal ne sont pas modifiés entre leur émission et leur réception. Cette propriété semble élémentaire mais pourtant de nombreux systèmes de communication ne la respectent pas. Nous avons tous par exemple reçu un jour un courrier électronique dont les lignes commençant par "From" étaient remplacées par ">From". Dans d'autres cas un courrier électronique peut voir son formatage modifié, ou même de longs messages peuvent être tronqués. Ces transformations pittoresques, dues à des passerelles de courrier électroniques capricieuses, peuvent poser un réel problème lorsque les messages sont authentifiés par une signature numérique (comme le permet par exemple PGP [Zimmerman 1993]). Si le message a été modifié, son authentification par le destinataire est alors impossible. Un autre cas de non-respect de la propriété d'intégrité est fourni par certains systèmes de vidéoconférence. CU-SeeMe [CU-SeeMe 1995] par exemple, utilise le protocole de communication UDP qui ne garantit pas un acheminement des informations de bout en bout. La transmission de la vidéo par exemple est avec perte (*lossy*). Suivant la bande passante disponible, des trames peuvent être perdues.

Enfin, nous proposons deux types de propriétés spécifiquement adaptées à la communication vidéo. La première, la réversibilité vidéo, est issue d'un principe établi

par l'expérimentation Garden Movie au chapitre 3. La seconde, vidéo miroir et vidéo reflex, est un affinement de l'observabilité publiée.

- Réversibilité vidéo

  La *réversibilité vidéo* caractérise la possibilité qu'a l'utilisateur de voir sous forme inversée (ou miroir) l'image d'un correspondant. Nous avons vu au chapitre 3 que cette propriété est utile pour la compréhension des gestes déictiques. Notons que l'on peut envisager deux formes de réversibilité : la réversibilité à la réception est possible à l'initiative du destinataire, la réversibilité à l'émission est le fait de l'émetteur. On peut considérer ce dernier cas comme une forme particulière de l'observabilité publiée : la variable personnelle est l'image de l'utilisateur, le filtre de publication est une transformation miroir de l'image.

- Vidéo miroir et vidéo reflex

  Ces deux propriétés caractérisent la possibilité pour l'utilisateur d'une communication vidéo de disposer d'une fonction miroir. Il ne s'agit pas comme dans la propriété précédente d'une transformation miroir (qui renverse l'image par une symétrie verticale) mais d'un service qui permet à l'utilisateur de se voir lui-même lorsqu'il est en communication vidéo. Nos premières expériences avec notre mediaspace VideoPort nous ont montré la nécessité d'un tel service. Non seulement il concourt à une meilleure acceptation de la présence de la caméra par les utilisateurs, mais il permet aussi par exemple à l'utilisateur de vérifier qu'il est bien dans le champ de la caméra. On peut distinguer deux cas, d'où deux propriétés.

  La propriété *vidéo miroir* caractérise la possibilité de voir l'image telle qu'elle est capturée par la caméra. On trouve habituellement ce cas de fonction miroir dans les systèmes de communication vidéo.

  La propriété *vidéo reflex* indique la possibilité de voir l'image telle qu'elle sera reçue par le correspondant. Cette propriété tire son nom du système reflex des appareils photo. Cette image peut être différente de celle fournie par la vidéo miroir. En effet, l'image peut subir un traitement avant d'être envoyée au correspondant, par exemple un filtrage ou une conversion en niveaux de gris. Le débit choisi par le correspondant peut aussi être différent du débit maximal (puisque local) qui sera fourni par la fonction miroir.

  Ces deux propriétés peuvent être rapprochées de la propriété de réflexivité que nous avons brièvement évoquée plus haut : la vidéo miroir montre la variable personnelle (la propre image de l'utilisateur) sans fonction de filtrage. La vidéo reflex montre la même variable personnelle après application de la fonction de filtrage.

En résumé, nous avons présenté pour le cas particulier des systèmes de communication homme-homme médiatisée des propriétés issues de principes de la psychologie cognitive, et des cas particuliers d'application à la vidéo des propriété liées à l'observabilité filtrée.

4.7. Conclusion

Dans ce chapitre, nous avons défini le concept de propriété et nous avons présenté des propriétés issues de l'étude des systèmes mono-utilisateurs, des propriétés inspirées par le génie logiciel, ainsi que des propriétés découlant de principes sociaux que nous avons exposés au chapitre 2 et des principes psychologiques du chapitre 3. Nous allons maintenant voir au chapitre suivant comment les propriétés peuvent être vérifiées à l'aide de techniques d'évaluation ergonomique. Nous verrons dans les chapitres 6 et 7 comment les propriétés issues du génie logiciel permettent de réfléchir aux qualités d'une architecture logicielle. Au chapitre 8, nous verrons comment certains outils peuvent aider à vérifier les propriétés.

Références

[Apple 1993] Apple. *OpenDoc Human Interface Guidelines (preliminary)*, Apple Computer Inc., 1993.

[Apple 1994] *Finder 7.5*. Logiciel pour Macintosh. Apple Computer Inc., Cupertino, CA, USA, 1994.

[Baecker 1992] R. M. Baecker, D. Nastos, L. R. Posner et K. L. Mawby. *The user-centred iterative design of collaborative writing software*, Workshop on Real Time Group Drawing and Writing Tools (CSCW'92, ACM Conference on Computer-Supported Cooperative Work), Toronto, Canada, 1992.

[Berne 1995] Y.-H. Berne et B. Hocq. *SIDECOM (Suivi d'individus destiné à l'extension de la communication médiatisée)*, Equipe IHM, Laboratoire de Génie Informatique, Institut IMAG, Rapport de stage, 1995.

[Bourguet 1992] M.-L. Bourguet. *Conception et réalisation d'une interface de dialogue personne-machine multimodale*. Thèse de doctorat, Institut National Polytechnique de Grenoble, 1992.

[Buxton 1994] W. Buxton. *Integrating the Periphery and Context: A New Taxonomy of Telematics*, ERGO-IA'94, Biarritz, France, 1994.

[Clark 1991] H. H. Clark et S. E. Brennan. *Grounding in Communication*, in *Perspectives on Socially Shared Cognition.* L. B. Resnick, J. M. Levine et S. D. Teasley, (eds.). American Psychological Association, Washington, DC, USA, 1991. pp. 127-149.

[Coutaz 1992] J. Coutaz, G. Abowd et L. Nigay. *Refining Software Engineering Quality Factors with HCI Factors*, HCI'92, UK, 1992.

[Coutaz 1995] J. Coutaz, L. Nigay, D. Salber, A. E. Blandford, J. May et R. M. Y. Young. *Four Easy Pieces for Assessing the Usability of Multimodal Interaction*, INTERACT'95, IFIP Fifth International Conference on Human Computer Interaction, 25-29 juin 1995, Lillehammer, Norvège (à paraître), 1995.

[CU-SeeMe 1995] *CU-SeeMe* . Logiciel pour Apple Macintosh. Cornell University, 1995.

[Decouchant 1994] D. Decouchant. *Rétroaction de groupe et édition coopérative de documents structurés*, IHM'94, Sixièmes Journées sur l'Ingénierie des Interfaces Homme-Machine, Lille, France, 1994. pp. 145-150.

[Dix 1993] A. Dix, J. Finlay, G. Abowd et R. Beale. *Human-Computer Interaction*, Prentice Hall, New York, 1993.

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

[Dourish 1992] P. Dourish et S. A. Bly. *Portholes: Supporting Awareness in a Distributed Work Group*, CHI'92, ACM Conference on Human Factors in Computing Systems, Monterey, California, USA, 1992. pp. 541-547.

[Gaver 1992] W. W. Gaver. *The Affordances of Media Spaces for Collaboration*, Proceedings of ACM CSCW'92 Conference on Computer-Supported Cooperative Work, Toronto, Canada, 1992. pp. 17-24.

[Gibson 1979] J. J. Gibson. *The ecological approach to visual perception*, Houghton Mifflin, New York, New York, USA, 1979.

[Insignia 1994] *SoftWindows 1.0*. Logiciel pour Macintosh. Insignia Solutions, 1994.

[Karsenty 1994] A. Karsenty. *GroupDesign : un collecticiel synchrone pour l'édition partagée de documents.* Thèse de doctorat, Université d'Orsay Paris-Sud, 1994.

[Kosbie 1994] D. S. Kosbie. *Hierarchical Events in Graphical User Interfaces*, CHI'94, ACM Conference on Human Factors in Computing Systems, Boston, Massachussets, USA, 1994. pp. 131-132.

[Lövstrand 1991] L. Lövstrand. *Being Selectively Aware with the Khronika System*, ECSCW'91, European Conference on Computer Supported Cooperative Work, Amsterdam, Pays-Bas, 1991.

[Mackay 1991] W. E. Mackay. *Triggers and Barriers to Customizing Software*, Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems, New Orleans, Louisiana, USA, 1991. pp. 153-160.

[McCall 1977] J. McCall. *Factors in Software Quality*, General Electric Eds., 1977.

[Meyer 1990] B. Meyer. *Conception et Programmation par Objets*, InterEditions, Paris, France, 1990.

[Nigay 1994] L. Nigay. *Conception et réalisation des systèmes interactifs: Application aux Interfaces Multimodales.* Thèse de doctorat, Université Joseph Fourier Grenoble I, 1994.

[Norman 1988] D. Norman. *The Psychology of Everyday Things*, Basic Books Publishing, 1988.

[Salber 1995] D. Salber, J. Coutaz, D. Decouchant et M. Riveill. *De l'observabilité et de l'honnêteté : le cas du contrôle d'accès dans la Communication Homme-Homme Médiatisée*, soumis à IHM'95, Conférence sur l'Ingénierie des Interfaces Homme-Machine, Toulouse, France, 1995.

[Salber 1994] D. Salber, J. Coutaz, L. Nigay, (editors), G. Faconti, F. Paterno', D. Duke et M. Harrison. *The System Modelling Glossary*, Amodeus Project, System Modelling Working Paper, SM/WP 26, 1994.

[Savetz 1994] K. Savetz. *Internet Fax FAQ*, disponible sur Internet, FAQ, 1994.

[Young 1994] R. M. Young et G. D. Abowd. *Multi-perspective Modelling of Interface Design Issues: Undo in a Collaborative Editor*, in *People and Computers IX: Proceedings of HCI'94.* G. Cockton, S. W. Draper et G. R. S. Weir, (eds.). Cambridge University Press, Cambridge, UK, 1994. pp. 249-260.

[Zimmerman 1993] *PGP (Pretty Good Privacy) 2.3ui.* Logiciel pour Macintosh. Phil Zimmerman, 1993.

Techniques d'Évaluation Ergonomique

- Who are you ?
- I am Oz, the Great and Terrible,
said the little man, in a trembling voice.

Lyman Frank Baum
"The Wonderful Wizard of Oz"

Techniques d'Évaluation Ergonomique

5.1. Introduction

Dans le cadre des systèmes interactifs mono-utilisateurs, l'intérêt de l'évaluation ergonomique des interfaces homme-machine s'affirme progressivement. En ce qui concerne les systèmes multi-utilisateurs et bien que [Grudin 1989] ait montré l'importance cruciale de l'évaluation dans la conception des systèmes multi-utilisateurs, peu de méthodes et d'outils sont disponibles. En revanche, on trouve dans la littérature de nombreuses études de cas de conception de systèmes multi-utilisateurs qui relatent souvent une phase d'évaluation. La diversité des types de systèmes multi-utilisateurs, la complexité des interactions qu'ils permettent et les nouvelles approches de conception qu'ils exigent sont à la source de ce manque de méthodes d'évaluation généralisables.

Vis-à-vis de notre structuration en principes, propriétés et techniques présentée au chapitre 1, l'évaluation intervient sur l'axe remontant. Elle permet de s'assurer que le système spécifié ou construit vérifie les propriétés souhaitées. Nous examinons dans ce chapitre ce que les méthodes d'évaluation des systèmes mono-utilisateurs peuvent apporter à l'évaluation des systèmes multi-utilisateurs. Nous présentons d'abord l'ensemble de ces méthodes de façon structurée afin d'analyser leur pertinence pour les systèmes multi-utilisateurs. A partir de cette analyse, nous illustrons l'utilisation d'une méthode prédictive fondée sur User Action Notation (UAN), puis nous nous concentrons sur l'évaluation expérimentale. En nous appuyant sur l'observation du travail des ergonomes lors de tests d'utilisabilité, nous avons établi un cahier des charges pour un laboratoire d'utilisabilité informatisé. NEIMO (Nouvelle Evaluation des Interfaces par le Magicien d'Oz) est une plate-forme informatique d'observation des utilisateurs et de simulation par Magicien d'Oz qui répond à ce cahier des charges.

5.2. Définitions

Avant de caractériser les méthodes d'évaluation des systèmes mono-utilisateurs, nous rappelons les définitions des termes "évaluation ergonomique" et "utilisabilité". Nous discutons la validité de ces définitions pour les systèmes multi-utilisateurs.

5.2.1. Définitions usuelles

L'*évaluation ergonomique* consiste à estimer la conformité des performances effectives avec les performances désirées du système [Dowell 1989]. Le "système" recouvre ici le couple utilisateur-système informatique. Pour les systèmes multi-utilisateurs,

l'imprécision de cette définition est immédiate : faut-il considérer un utilisateur donné et le système informatique, ou bien l'ensemble des utilisateurs et le système informatique ? Cette question n'a pas de réponse claire dans la littérature et nous verrons que les deux approches sont justifiables. Pour contourner l'ambiguïté de cette définition, nous préférons considérer que l'évaluation ergonomique est concernée par l'étude de l'utilisabilité d'un système informatique.

Nous avons défini au chapitre 4 des propriétés d'utilisabilité d'un système qui affinent la notion d'utilisabilité. Cependant, ce point de vue inspiré de l'étude de la qualité du logiciel doit être confronté à l'approche ergonomique de l'utilisabilité.

L'*utilisabilité* telle que la définit la norme ISO 9241 est "l'efficacité et la satisfaction avec laquelle des utilisateurs définis peuvent réaliser des buts définis dans un environnement particulier"[1]. [Shackel 1991] est plus précis : il définit l'utilisabilité d'un système comme "la capacité, en termes de fonctionnement humain, d'être utilisé facilement et efficacement par une classe d'utilisateurs définie, recevant une formation et une aide définies, pour réaliser une classe de tâches définies, dans le cadre d'une classe définie de scénarios prenant en compte l'environnement"[2]. A partir de cette définition générale, les mêmes auteurs donnent une série de critères opérationnels quantifiables : efficacité, facilité d'apprentissage, flexibilité et attitude. Ce dernier critère est constitué de deux aspects : l'acceptabilité en termes de "coût humain" (fatigue, inconfort, frustration, et effort individuel) et la satisfaction de l'utilisateur. Notons que, contrairement aux trois autres, le critère "attitude" n'est que très partiellement couvert par nos propriétés du chapitre 4.

Ces notions sont replacées dans un cadre plus vaste par [Nielsen 1994] qui considère l'utilisabilité comme un sous-ensemble d'une notion plus large : l'acceptabilité globale du système. L'acceptabilité est définie comme "la question de savoir si le système satisfait les besoins et les demandes des utilisateurs et des autres personnes potentiellement concernées, telles que les clients des utilisateurs et les dirigeants. L'acceptabilité globale d'un système informatique est une combinaison de son acceptabilité sociale et de son acceptabilité pratique". Nielsen propose les critères suivants pour quantifier l'utilisabilité d'un système : facilité d'apprentissage, efficacité, facilité de mémorisation, faible taux d'erreur et possibilités de réparation des erreurs, satisfaction des utilisateurs ("le système

1 "The effectiveness, efficiency and satisfaction with which specified users can achieve specified goals in particular environments." (Norme ISO 9241, Part 11 : Ergonomics requirements for office work with VDTs - Guidance on usability specification and measures).

2 "[the usability of a system or equipment is] the capability in human functional terms to be used easily and effectively by the specified range of users, given specified training and user support, to fulfil the specified range of tasks, within the specified range of environmental scenarios."

doit être "agréable" à utiliser, les utilisateurs l'apprécient"). Scapin retient des critères similaires [Scapin 1990].

Comme on le voit à travers ces définitions, l'utilisabilité est encore un concept mal cerné, même si ces définitions présentent des points communs. En fait, ces différences apparentes révèlent des différences de point de vue : du point de vue technique de la norme ISO jusqu'à une perspective sociale plus large proposée par Nielsen. De notre analyse, nous retenons les points suivants :

- l'utilisabilité se traduit par des métriques qui permettent de quantifier et évaluer l'utilisabilité d'un système. Schackel est le plus précis et identifie des paramètres mesurables, par exemple en termes de pourcentage de variation pour la flexibilité. Toutefois, du point de vue d'un concepteur de système, ces critères sont trop généraux. Les propriétés du chapitre 4, qui caractérisent directement les aspects du système pertinents pour l'étude de l'utilisabilité, sont applicables dès les étapes amont de la conception. Les critères exposés ci-dessus sont plutôt orientés vers l'évaluation expérimentale. L'évaluation prédictive ne peut pas être guidée efficacement par ces critères centrés sur l'expérimentation. Nous précisons les différences entre ces pratiques ergonomiques dans le paragraphe suivant.

- Les trois définitions de l'utilisabilité exposées ci-dessus prennent en compte la satisfaction de l'utilisateur. Ici encore, il s'agit d'un critère qui ne peut être évalué qu'expérimentalement. L'évaluation prédictive ne peut fournir d'indications sur la façon de satisfaire ce critère.

- Seul Nielsen donne de l'utilisabilité une vision large incluant les aspects sociaux. Même si elle semble ouvrir une boîte de Pandore en prenant en compte dans l'utilisabilité l'avis de toutes les personnes ayant rapport de près ou de loin avec le système informatique, cette définition est mieux adaptée aux systèmes multi-utilisateurs. En particulier, elle permet d'intégrer la dimension sociale qui est essentielle pour ce type de système.

- Les définitions ci-dessus mettent en évidence la nécessité de spécifier le plus précisément possible les utilisateurs du système et les tâches à réaliser. Cette précision semblera évidente pour les lecteurs familiers avec l'ergonomie. Rappelons toutefois qu'il s'agit d'un prérequis indispensable à toute évaluation ergonomique. Moins connu mais cité dans les trois définitions, l'environnement doit également être spécifié avec précision. Le contexte de travail impose en effet des contraintes qui auront une influence sur l'utilisabilité. Il est intéressant de

remarquer qu'avec l'informatique mobile, cette caractéristique devient plus difficile à déterminer avec précision et un logiciel prévu pour être utilisé dans un contexte de bureau peut aussi être utilisé dans un environnement beaucoup plus contraignant mais plus difficile à cerner. Un exemple révélateur en est donné par [Tognazzini 1991] qui étudie l'utilisation des ordinateurs portables dans les avions de ligne. Il préconise un ensemble de règles, en particulier sur la prévention des erreurs dans l'utilisation des menus et du trackball, en argumentant que l'espace réduit et les vibrations rendent les manipulations moins précises.

Comme nous l'avons vu au chapitre 4, l'utilisabilité peut aussi être affinée en un ensemble de propriétés. Ces propriétés peuvent servir de critères pour l'étude de l'utilisabilité. Nous remarquons que nos propriétés recouvrent les critères de Schackel ou de Nielsen, exception faite du critère de satisfaction de l'utilisateur. Comme nous allons le voir, le choix des critères est guidé par la démarche ergonomique, qui elle-même est influencée par un ensemble de facteurs caractérisant le système et le contexte de développement.

5.2.2. L'évaluation d'un système multi-utilisateur

Comme nous l'avons remarqué plus haut, un système multi-utilisateur peut être vu au moins de deux façons. Soit le système est vu comme l'ensemble de ses parties et l'on peut mener une étude de l'utilisabilité du système pour un utilisateur donné et les tâches que réalise cet utilisateur. Soit le système est considéré dans son ensemble et l'on mène une évaluation globale du système en considérant l'ensemble des utilisateurs et les tâches globales qu'ils réalisent. Ces deux approches ne sont sûrement pas exclusives et sont en fait deux points de vue complémentaires sur un même problème. Pascal écrivait justement : "Je tiens pour impossible de connaître les parties sans connaître le tout, non plus que de connaître le tout sans connaître particulièrement les parties."

Pour réconcilier ces deux points de vue dans le cadre de l'évaluation ergonomique, nous introduisons la distinction faite par [Senach 1990] entre utilité et utilisabilité. L'utilité caractérise l'adéquation fonctionnelle du système : permet-il à l'utilisateur d'atteindre ses objectifs de travail ? L'utilisabilité concerne l'adéquation de l'interface homme-machine : le logiciel est-il facile à apprendre et à opérer ? Ramenée aux systèmes multi-utilisateurs, cette distinction nous aide à fixer des objectifs pour l'évaluation. L'étude de l'utilité permet d'évaluer le système multi-utilisateur dans son ensemble ; l'étude de l'utilisabilité permet d'évaluer le système du point de vue d'un utilisateur donné. Nous définissons maintenant ce qu'est l'utilité d'un système multi-utilisateur.

Pour un système multi-utilisateur, l'utilité a deux facettes : d'abord, elle caractérise l'adéquation fonctionnelle du système pour chaque utilisateur. En ce sens, cette approche de l'utilité est proche de celle pratiquée avec les systèmes mono-utilisateurs. La différence réside dans les tâches à étudier : en effet, comme l'explicite le modèle du trèfle (présenté au chapitre 1), les systèmes multi-utilisateurs introduisent de nouvelles tâches ayant trait à la coordination et à la communication. On peut citer MERMAID, un système de conférence audio/vidéo [Watabe 1990] comme exemple d'étude de l'utilité d'un système multi-utilisateur. Les auteurs notent par exemple que l'utilisation de la voix seule rend la communication plus difficile lorsque plus de quatre utilisateurs qui se connaissent peu communiquent à l'aide du système. Les auteurs remarquent aussi que le télépointage est en général exclusivement utilisé par la personne qui a la parole. Cette étude exploite des données obtenues par observation de l'utilisation du système et présente des résultats qualitatifs. En l'absence de modèles théoriques de la communication et de la coordination et de règles permettant de les appliquer à l'étude des systèmes multi-utilisateurs, l'approche expérimentale est privilégiée.

La deuxième facette de l'utilité d'un système multi-utilisateur est son adéquation fonctionnelle à la tâche globale réalisée par l'ensemble des utilisateurs. Cette utilité "globale" ne peut être déduite de l'étude de l'utilité du système pour chacun des utilisateurs mais requiert de considérer le système et le groupe d'utilisateurs dans leur ensemble (le tout n'est pas simplement la somme des parties). [Brothers 1990] présente une méthode expérimentale pour évaluer l'utilité globale. Le système décrit, ICICLE, est un environnement multi-utilisateur d'inspection de code. Les auteurs suggèrent d'étudier l'utilité du système en comparant différentes inspections du même code, avec ou sans ICICLE. Ils proposent des métriques permettant une évaluation quantitative de l'utilité, telles le nombre de *bugs* découverts ou des métriques mesurant la qualité du logiciel produit, ainsi que des mesures du temps requis par l'inspection et de la productivité.

[Boersma 1994] relate une évaluation d'un système de workflow pour l'accord de prêts bancaires. Ici encore, l'approche est expérimentale et repose sur des métriques comme le temps de traitement des dossiers de prêts, le nombre d'erreurs dans les décisions d'accord des prêts, et le gain de productivité.

Ces deux exemples choisis pour leur représentativité mettent en évidence l'utilisation de deux classes de métriques. Les métriques de la première classe expriment la qualité du travail produit : nombre d'erreurs, qualité du résultat, par exemple. La deuxième classe recouvre des mesures de temps d'exécution des tâches et de productivité. On retrouve ici la traditionnelle distinction entre bénéfice (représenté par la première classe) et coût (deuxième classe). Le système doit permettre d'augmenter la qualité du résultat de la tâche

tout en réduisant le coût nécessaire à son accomplissement. Cette notion est bien sûr également présente dans les systèmes mono-utilisateurs.

La distinction utilité/utilisabilité nous a permis de mieux cerner ce que représente l'évaluation pour les systèmes multi-utilisateurs. Pour examiner comment les techniques d'évaluation existantes peuvent être appliquées à ces systèmes, il nous faut caractériser ces techniques et réfléchir à leurs intégration dans le processus de développement d'un système. Pour tenir compte des pratiques actuelles de conception des systèmes multi-utilisateurs et des pratiques d'évaluation ergonomique, et pour replacer notre travail dans le cadre de ces pratiques, nous pensons utile de présenter les conceptions de l'interaction homme-machine de Long et Dowell.

5.3. Les démarches en IHM et en ergonomie

Long et Dowell distinguent trois types de démarches en interface homme-machine (IHM) : artisanale, sciences appliquées et ingénierie [Long 1989]. Nous présentons pour chacune de ces trois démarches la façon dont elles s'appliquent à l'ergonomie.

La démarche artisanale repose sur un savoir empirique, acquis par l'expérience. Ce savoir est informel et souvent ambigu, donc difficile à transférer parce que sujet à interprétation. Les règles ergonomiques élaborées expérimentalement, telles les recommandations de Smith et Mozier [Smith 1986], sont un exemple typique de connaissance utilisée dans la démarche artisanale. La pratique de la démarche artisanale rappelle l'approche exploratoire qui repose sur le prototypage rapide : une maquette est développée sans spécifications préalables et est évaluée immédiatement. Ce cycle prototypage-évaluation est ensuite réitéré à partir des informations fournies par l'évaluation.

La démarche sciences appliquées s'appuie sur un savoir élaboré à partir d'une théorie scientifique. Mais comme pour la démarche artisanale, le savoir doit être interprété pour être appliqué. La technique du "cognitive walkthrough" est un représentant de cette démarche [Lewis 1991]. Dans la pratique, le cycle de base est constitué des trois phases : spécification partielle, mise en œuvre, et évaluation. Des retours arrière après évaluation permettent de compenser les incertitudes liées à l'ambiguïté du savoir et d'ajuster les spécifications.

La démarche ingénierie est une approche scientifique rigoureuse. Le savoir est formalisé donc non ambigu et transférable. Les propriétés et leur formalisation par [Dix 1993] relèvent de cette approche. En pratique, le cycle de développement comporte :

spécification complète, réalisation, évaluation. L'évaluation mène éventuellement à une modification des spécifications et à un nouveau cycle.

Ces différentes démarches sont appliquées par Long et Dowell aussi bien à la composante informatique qu'à la composante ergonomie de la discipline IHM. Nous pensons que cette vue de l'IHM doit être étendue car l'on retrouve aussi ces trois démarches dans la composante psychologie et dans les composantes sciences humaines. Nous pensons aussi que la vue monolithique de l'IHM que présentent Long et Dowell se trouve aujourd'hui contredite par la pratique. Long et Dowell voient l'ergonomie et l'informatique évoluant de concert, adoptant en même temps l'un des trois types de démarche. En fait, et c'est d'autant plus vrai lorsque l'IHM doit intégrer de nouvelles disciplines comme les sciences humaines, chacune des disciplines partie prenante de l'IHM évolue indépendamment et suit l'une des démarches identifiées par Long et Dowell.

Comme le remarque [Balbo 1994], l'approche ingénierie est une vue idéale vers laquelle devrait tendre l'ergonomie. La rigueur scientifique qui la caractérise garantit que la méthode est généralisable, réutilisable, transférable et testable. Mais les démarches utilisées en réalité sont plus proches de la démarche artisanale ou de la démarche sciences appliquées. Dans les domaines où la théorie scientifique n'est pas encore suffisamment mature, comme pour les systèmes multimodaux ou multi-utilisateurs, la démarche artisanale est largement privilégiée. Notre approche fondée sur l'utilisation des propriétés vise à développer les premières bases d'une ingénierie de la conception ergonomique des systèmes multi-utilisateurs. Mais étant donné l'état actuel encore limité des connaissances théoriques applicables aux systèmes multi-utilisateurs, une approche plus concrète est aussi nécessaire. Nous présentons plus loin dans ce chapitre l'utilisation des propriétés (approche ingénierie) et l'évaluation expérimentale (approche artisanale). Mais il nous faut d'abord situer les techniques d'évaluation existantes et les lier aux processus de développement couramment utilisés pour les systèmes multi-utilisateurs.

5.4. Taxinomies

Nous exposons dans ce paragraphe une taxinomie des techniques d'évaluation afin de situer les techniques que nous allons présenter par la suite. Nous proposons aussi une classification des processus de développement.

5.4.1. Taxinomie des techniques d'évaluation

Une taxinomie classique des méthodes d'évaluation distingue les méthodes prédictives des méthodes expérimentales. Une méthode prédictive ne nécessite pas la présence de

l'utilisateur alors qu'une approche expérimentale repose sur l'observation d'utilisateurs. Cette classification peut être précisée comme le montre la figure 5.1.

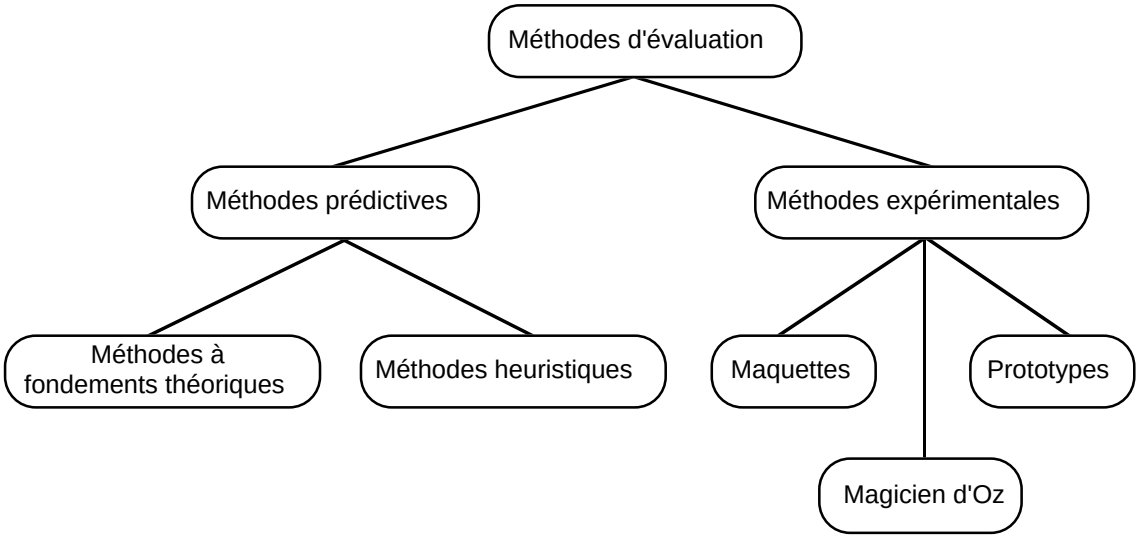


Figure 5.1. Classification des techniques d'évaluation.

L'évaluation expérimentale repose sur l'expérimentation avec un utilisateur final. Les tests de maquettes, prototypes ou de produits finaux, les tests en laboratoire d'utilisabilité, et les expérimentations Magicien d'Oz sont des exemples de techniques d'évaluation expérimentale. L'évaluation expérimentale ne nécessite pas un produit logiciel opérationnel. Les expérimentations à base de maquettes en papier et carton réalisées par IBM pour la messagerie des jeux olympiques de Los Angeles [Gould 1987] sont un exemple célèbre du fait que l'évaluation expérimentale de l'utilisabilité peut être réalisée à coût réduit et tôt dans le développement du système. L'évaluation expérimentale relève de la démarche ergonomique artisanale. Nous présentons plus loin dans ce chapitre l'outil NEIMO, plate-forme Magicien d'Oz et d'observation pour l'évaluation expérimentale.

L'évaluation prédictive est caractérisée par le fait qu'elle ne nécessite pas la présence de l'utilisateur final. Elle ne nécessite pas non plus une réalisation ni même un prototype du système. Elle peut donc intervenir très tôt dans le cycle de développement, dès la phase de spécification. L'évaluation prédictive relève de la démarche sciences appliquées (comme le "cognitive walkthrough" par exemple), de la démarche artisanale (l'utilisation de règles ergonomiques) ou de la démarche ingénierie. Nous illustrons cette dernière approche avec la notation UAN et nos propriétés d'utilisabilité (présentées au chapitre précédent). Un inconvénient des techniques prédictives est lié à la complétude de la théorie sous-jacente ou de l'ensemble des règles heuristiques. La théorie ou les règles sont souvent limitées aux interfaces graphiques traditionnelles. En l'absence d'une extension des supports théoriques, la technique d'évaluation n'est pas applicable à des systèmes plus récents tels

les interfaces multimodales ou les systèmes multi-utilisateurs. Il faut alors recourir à une technique artisanale. Gageons que ces techniques seront étendues tôt ou tard, mais il faut avoir conscience de leurs limitations.

Les techniques d'évaluation ne peuvent pas être envisagées indépendamment du processus de développement. Outre le fait qu'une technique d'évaluation requiert un certain avancement du développement pour être appliquée, et donc présuppose la disponibilité de documents ou d'un artefact pour servir de base à l'évaluation, toutes les techniques d'évaluation ne sont pas adaptées à tous les processus de développement. Nous caractérisons maintenant les processus de développement communément utilisés pour la conception et la réalisation des systèmes multi-utilisateurs.

5.4.2. Classification des processus de développement

Nous avons identifié trois grandes catégories de processus de développement utilisés dans la conception et la réalisation des systèmes multi-utilisateurs. Ces trois catégories sont le cycle génie logiciel, l'approche exploratoire et la conception participative. Nous définissons chacune de ces catégories et proposons une classification de leur usage.

5.4.2.1. Approche génie logiciel

L'approche génie logiciel est maintenant couramment utilisée en informatique. Les modèles sur lesquels elle repose sont typiquement le cycle de vie en cascade, ou les cycles en V ou en spirale. Une telle approche vise à organiser les activités de conception, de réalisation et de test du système de façon structurée. Chacune des étapes est identifiée et donne lieu à la production de documents. Les retours arrière sont possibles tout au long du cycle de vie. Aujourd'hui, les cycles de vie couramment utilisés ne mentionnent pas explicitement l'intégration des tests ergonomiques. Dans la pratique actuelle, ceux-ci ont souvent tendance à être repoussés en fin de cycle, regroupés avec les tests globaux du système. Or, les problèmes mis en évidence par l'évaluation ergonomique requièrent souvent une remise en cause des spécifications du système. Un retour arrière important (des tests du système jusqu'aux spécifications) est extrêmement coûteux. Une étude de Hewlett-Packard évalue qu'un défaut détecté lors des tests du système est cent fois plus coûteux à corriger qu'un défaut détecté lors de la conception (document cité dans [Balbo 1994]). Il est donc indispensable d'intégrer l'évaluation ergonomique le plus tôt possible dans le cycle de développement. N'oublions pas que si beaucoup de grandes entreprises de logiciel ont intégré les pratiques ergonomiques à leur cycle de vie, l'évaluation ergonomique reste encore "la cerise sur le gâteau" dans beaucoup de contextes industriels. Une récente étude révèle que seulement 52% des industriels européens mesurent

l'importance[1] de l'évaluation ergonomique ! Dans la classification de Long et Dowell, le cycle de vie génie logiciel relève de l'approche ingénierie, au moins en ce qui concerne la partie informatique. La structuration rigoureuse du cycle de vie permet aussi d'intégrer les pratiques ergonomiques comme le montre [Balbo 1994].

5.4.2.2. Approche exploratoire

L'approche exploratoire relève suivant les cas de la démarche artisanale ou de la démarche science appliquée. Elle repose sur l'utilisation intensive du prototypage et se caractérise par une suite d'itérations (spécification-)prototypage-évaluation. La granularité des itérations peut varier : le cycle de vie en spirale identifie plusieurs phases de prototypage, mais on observe souvent que le prototypage est utilisé pour tester un ou plusieurs aspects précis du système. On peut ainsi distinguer un "prototypage global" et un "prototypage local", plus concentré sur une caractéristique précise du système (par exemple un détail de l'interface utilisateur). La caractéristique distinctive de l'approche exploratoire est une plus grande implication des utilisateurs : le prototype est destiné à être évalué lors de tests d'utilisabilité et les résultats des tests informent l'itération suivante.

Même si un modèle comme le cycle de vie en spirale propose une structure pour l'approche exploratoire, celle-ci est moins fortement structurée qu'un processus de développement où l'objectif final est bien identifié. Les itérations ne sont pas connues à l'avance et même si l'on peut vouloir limiter leur nombre, leur résultat peut mener à repousser cette limite. En fait, et comme le fait explicitement ressortir le modèle en spirale, chaque itération comporte une phase d'analyse des risques. L'importance de cette phase ne doit pas être négligée : elle nécessite une analyse exhaustive du problème qui fait l'objet du prototypage et constitue à notre sens le "défaut dans la cuirasse" de l'approche exploratoire. Une analyse des risques mal conduite ou incomplète peut entraîner une itération supplémentaire. Notons au bénéfice du modèle en spirale qu'il impose une phase de spécification explicite et qu'il exige de documenter les alternatives lors de la spécification. Cette approche augmente les chances de réaliser un prototype pertinent pour l'évaluation du problème considéré.

5.4.2.3. Conception participative

La conception participative est une approche assez récente qui se distingue par une implication des utilisateurs tout au long du processus de développement. On trouve des exemples d'étude de cas dans la littérature décrivant les mediaspaces, ainsi que dans les présentations de techniques de conception pour les systèmes multi-utilisateurs, notamment celles de l'École Scandinave. Ce processus de développement est

[1] "Mesurer l'importance" ne veut pas dire "appliquer" !

particulièrement adapté aux systèmes multi-utilisateurs. Il prend en compte des aspects sociaux et organisationnels qui ne sont pas abordés par les autres méthodes. Cependant, il relève d'une démarche artisanale. Les études de cas font apparaître un éventail de démarches ad hoc et, si elles permettent d'identifier les grands principes qui sous-tendent cette approche, la méthode n'est pas encore généralisable. Ses résultats sont toutefois très prometteurs et elle a l'avantage de promouvoir la collaboration de disciplines diverses indispensables à la conception des systèmes multi-utilisateurs. Cette multi-disciplinarité de la méthode est sans doute à l'origine de son manque de structuration. Chacune de ces disciplines a en effet ses propres technique, et l'ensemble souffre de l'absence d'un creuset intégrateur permettant de conjuguer leur apports de façon systématique. Des techniques de Design Rationale comme la notation QOC présentée au chapitre 1 peuvent ici apporter une aide.

Suite à cette analyse des processus de développement, nous proposons la classification de la figure 5.2. Les deux axes de cette classification indiquent la structuration du processus de développement et l'implication des utilisateurs dans la conception du système.

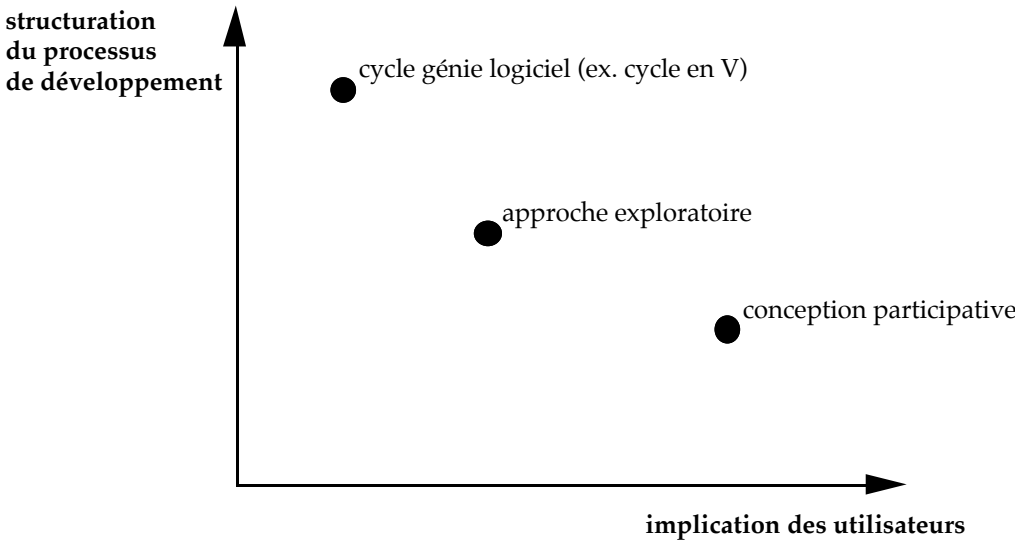


Figure 5.2. Classification des processus de développement reflétant les pratiques actuelles de conception et de réalisation des systèmes multi-utilisateurs.

Il est important de préciser que cette classification reflète les usages des méthodes présentées, et non leurs caractéristiques intrinsèques. Un cycle génie logiciel devrait impliquer davantage les utilisateurs dans la conception. Rien dans un cycle en V par exemple n'empêche de rajouter des phases d'évaluation ergonomique après chaque étape, dans l'esprit du modèle en étoile[1] de [Hix 1993]. De même, si la conception participative

[1] star life model.

apparaît aujourd'hui comme un processus de développement peu structuré, nous espérons qu'elle évoluera vers une plus grande structuration et permettra d'intégrer les apports des différentes disciplines qu'elle rassemble dans un cycle de vie inspiré de ceux du génie logiciel.

Notre classification des processus de développement fait ressortir que les catégories que nous avons identifiées diffèrent dans la façon dont les utilisateurs finaux sont présents lors du développement. La présence ou non de l'utilisateur final à un certain stade du développement conduira à opter pour une technique d'évaluation prédictive ou expérimentale.

De l'analyse des pratiques ergonomiques et des classifications que nous venons de présenter, nous tirons deux conclusions. Tout d'abord, il nous faut tendre vers la démarche ingénierie et les méthodes d'évaluation prédictive sont pour cela les plus adaptées. Cependant, les approches de conception des systèmes multi-utilisateurs relèvent plus de l'approche exploratoire ou participative que de l'approche génie logiciel. Cette contradiction nous a incité à explorer, en nous appuyant sur nos propriétés, une méthode prédictive flexible et peu coûteuse reposant sur la notation UAN.

5.5. Évaluation prédictive : l'exemple d'UAN

Nous avons dit que les propriétés que nous avons présentées au chapitre 4 peuvent être formalisées dans un but de vérification formelle. Compte tenu des pratiques actuelles de développement des systèmes multi-utilisateurs, la lourdeur d'une telle technique est irréaliste. Nous avons donc exploré le potentiel d'une notation semi-formelle plus flexible et plus simple d'emploi, User Action Notation (UAN). Nous constatons que si cette notation peut bien servir à la vérification de propriétés, son extension à la description des systèmes multi-utilisateurs pose des difficultés.

User Action Notation (UAN) [Hix 1993] est un système de notation semi-formel pour les systèmes interactifs développé originellement par Hartson, Siochi et Hix. UAN est orienté tâche et utilisateur : la notation permet de décrire les tâches, les actions de l'utilisateur et le comportement de l'interface homme-machine. A ce titre, c'est un outil précieux pour la représentation des tâches et pour les spécifications externes. UAN remplace avantageusement les descriptions informelles sous forme de texte et copies d'écran qui constituent aujourd'hui la plupart des spécifications externes des interfaces. Les descriptions textuelles sont en effet souvent ambiguës ou incomplètes, et par là même sujettes à interprétations multiples. Il faut toutefois noter que UAN n'est pas une description formelle comme peut l'être une description en langage Z. Cependant, dans les

cas où un vrai formalisme n'est pas nécessaire (donc hors systèmes critiques), UAN offre une notation plus lisible que la moyenne des formalismes. L'utilisation privilégiée d'UAN est la communication de spécifications, par exemple entre le concepteur et le réalisateur, utilisation qui a d'ailleurs motivé son développement.

Nous présentons d'abord dans ses grandes lignes la notation UAN, puis montrons de quelle façon elle peut être utilisée pour vérifier des propriétés d'utilisabilité. Nous envisageons son extension aux systèmes multi-utilisateurs et terminons par une évaluation de la notation.

5.5.1. Présentation de la notation UAN

UAN structure la description du comportement d'une interface en un ensemble de tableaux. Chaque tableau représente une tâche qu'il est possible d'effectuer avec le système. Ces tâches peuvent aussi bien être des tâches de haut niveau que des sous-tâches ou des tâches feuilles de l'arbre de tâches. Chaque tableau se compose de trois colonnes : les actions de l'utilisateur, la réponse de l'interface et l'état de l'interface. La notation propose un ensemble de symboles figurant des actions utilisateur ou des réponses de l'interface. Par exemple, enfoncer le bouton de la souris est représenté par le symbole **v**[1]. Le changement d'état d'une icône, par exemple son passage en vidéo inverse est représenté par **!**. La notation autorise le libre ajout de variables et de prédicats représentant soit des objets d'interaction soit des états internes de l'interface. La figure 5.3 montre une représentation typique en UAN.

Task: delete file
Precondition: visible(file_icon)

User Action	Interface Feedback	Interface State
~[file_icon]v	other_icon-! file_icon!	selected = file_icon
~[trash_icon]	trash_icon!	
^	trash_icon-! trash_icon!!	
	erase(file_icon)	selected = nil

Figure 5.3. Représentation UAN (simplifiée) de la destruction d'un fichier.

La figure 5.3 est une représentation UAN de la tâche de destruction d'un fichier avec le Finder du Macintosh. La représentation est légèrement simplifiée : le fait que le

[1] Nous prenons ici une légère liberté avec la notation. L'action d'enfoncer le bouton de la souris devrait en fait être notée **Mv**. Nous justifions cet ajustement de la notation au paragraphe 5.5.4.

"fantôme" de l'icône suit le mouvement lorsque l'on déplace l'icône a été omis. En haut à gauche du tableau, on indique la tâche décrite et d'éventuelles préconditions. Ici, la précondition pour que l'on puisse détruire un fichier est que l'icône de ce fichier doit être visible. Les lignes suivantes du tableau représentent la succession dans le temps des actions de l'utilisateur et les réponses de l'interface correspondantes. La première ligne indique que l'utilisateur déplace la souris près de l'icône de façon à pouvoir la saisir. [icon] représente la partie de l'élément d'interaction icon qui permet de le manipuler. Dans le cas d'une icône, il s'agit en général de toute sa surface ; pour une fenêtre, il s'agirait de sa barre de titre. L'utilisateur enfonce ensuite le bouton de la souris. La réponse de l'interface est constituée de deux opérations : toute autre icône sélectionnée auparavant (other_icon) repasse en affichage normal (**-!**) et l'icône du fichier est inversée. La variable d'état de l'interface selected prend la valeur de la nouvelle sélection. Les lignes suivantes décrivent successivement le déplacement de l'icône vers la corbeille et, après relâchement de la souris, le changement d'apparence de l'icône de la corbeille qui se gonfle (**!!**) et la disparition de l'icône du fichier déplacé.

UAN offre également un ensemble complet d'opérateurs permettant d'exprimer les relations entre tâches : séquencement, parallélisme, interruptions, entrelacement, tâches optionnelles, … ainsi qu'un ensemble exhaustif de symboles permettant de prendre en compte le temps et les relations temporelles entre tâches ou actions.

On constate que la notation UAN permet d'exprimer de façon très compacte un comportement d'interface qui est lourd et difficile à décrire en langue naturelle : il suffit de compter le nombre de lignes de texte qui sont nécessaires ci-dessus pour décrire la première ligne d'UAN ! Ce bénéfice jouerait à lui seul en faveur de l'usage d'UAN pour la spécification d'interfaces. Comme nous allons le voir, la notation permet aussi de détecter des problèmes d'utilisabilité.

5.5.2. Utilisation d'UAN pour la vérification de propriétés d'utilisabilité

UAN présente en regard les unes des autres les actions utilisateur et les réponses de l'interface. Cette caractéristique permet de vérifier la propriété d'observabilité présentée au chapitre 4. Nous allons montrer sur un exemple (MATIS) comment vérifier la propriété d'observabilité puis nous donnerons quelques règles qui permettent de vérifier d'autres propriétés à partir de l'examen d'une spécification UAN.

MATIS, qui fonctionne sur NeXT, est un système multimodal de recherche d'information sur les transports aériens [Nigay 1994]. L'application MATIS utilise un système de reconnaissance de la parole et pour pouvoir utiliser ce service doit être lancée depuis

l'environnement Office Manager (OM). Chaque application utilisant les services de reconnaissance de la parole d'OM fournit son propre dictionnaire permettant de guider la reconnaissance. La description UAN en figure 5.4 décrit comment l'utilisateur active une application dans l'environnement OM. Notons que l'analyse UAN de MATIS a été faite après développement. Il s'agit donc d'une évaluation "summative" au sens de [Hix 1993], mais la règle que nous donnons est générale et est aussi valide dans le cas d'une évaluation avant implémentation.

Task: SelOMAppli
Precondition: visible(Appli_Icon)

User Action	Interface Feedback	Interface State
~[Appli_Icon]v ^	Other_Icon! : Other_Icon-! Appli_Icon! *Feedback absent* ✗	SelDict = AppliDict

Figure 5.4. Détection d'un problème d'observabilité à partir de la spécification UAN.

On note dans la colonne Interface State qu'une variable SelDict est mise à jour après que l'utilisateur a cliqué sur une icône pour changer d'application. Ceci indique que le dictionnaire actif est maintenant celui correspondant à la nouvelle application sélectionnée. Or aucun feedback n'est fourni correspondant à la modification de cette variable. Cependant, cette variable est utilisée comme précondition dans la description d'une autre tâche sous la forme SelDict = MATISDict. Comme la modification de cette variable n'a jamais été répercutée dans la colonne Interface Feedback, l'utilisateur n'a aucun moyen de savoir si la précondition qui conditionne l'exécution d'une tâche est vraie ou fausse. La propriété d'observabilité du dictionnaire sélectionné n'est donc pas vérifiée.

A partir de cet exemple, nous pouvons généraliser notre constatation par la règle suivante :

❶ Si une variable d'état de l'interface est utilisée dans la précondition d'une tâche, toute modification de cette variable dans la colonne Interface State doit être répercutée dans la colonne Interface Feedback.

Notons que l'on peut être tenté d'aller plus loin que ce que propose ❶ et dire simplement que toute modification d'une variable d'état de l'interface doit être répercutée dans la colonne Interface Feedback. Mais ce serait en fait outrepasser la propriété d'observabilité. En effet, pour garantir que la variable est pertinente pour la tâche de l'utilisateur, comme

défini par la propriété d'observabilité, la variable doit apparaître comme précondition d'une tâche.

D'autres règles peuvent être établies pour vérifier des propriétés liées à l'observabilité. Nous donnons ici les règles permettant de vérifier les propriétés de non-préemption, dialogue à fils multiples, accessibilité, ainsi que les propriétés CARE.

Non-préemption :

❷ La non-préemption est assurée si toute opération apparaissant dans la colonne Interface Feedback est une réponse à une opération figurant dans la colonne User Action. Dans le cas contraire, l'opération considérée constitue une préemption de la part du système.

Dialogue à fils multiples :

❸ Des opérateurs permettant d'exprimer le parallélisme (**||**) et l'entrelacement de tâches (**<->**) sont inclus dans UAN. Le dialogue à fils multiples est donc particulièrement facile à caractériser : il suffit que les opérateurs de parallélisme ou d'entrelacement soient présents dans la spécification. Remarquons que, conformément à l'énoncé de la propriété, le dialogue à fils multiples peut être exprimé dans UAN à différents niveaux d'abstraction avec les mêmes opérateurs : niveaux tâche, sous-tâche, et action physique.

Accessibilité :

❹ Pour vérifier que tous les états de l'interface sont accessibles, il faut d'abord s'assurer qu'il n'y a pas de tâches "orphelines" c'est-à-dire que l'arbre de tâches est un arbre correct. Puis il suffit de vérifier pour chaque tableau UAN que la précondition peut être vraie. Il faut donc vérifier que tous les constituants de toutes les préconditions peuvent être modifiés par l'utilisateur (ou par le système le cas échéant) de sorte que les préconditions puissent être vraies.

Propriétés CARE :

Dans le cadre particulier des interfaces multimodales, la notation UAN doit être étendue pour prendre en compte de nouvelles techniques d'interaction. [Nigay 1994] donne un exemple d'une telle extension pour la reconnaissance de la parole.

❺ La complémentarité, l'assignation, la redondance et l'équivalence des modalités peuvent être déterminées à partir de l'examen de la spécification UAN. L'équivalence est caractérisée par un opérateur de choix (**|**) entre deux suites d'actions physiques chacune exprimée dans une modalité différente.

L'assignation, en revanche, est détectée par l'absence d'un opérateur de choix pour réaliser une tâche.

5.5.3. Extension d'UAN aux systèmes multi-utilisateurs

Nous l'avons noté dans ce chapitre, l'évaluation des systèmes multi-utilisateurs requiert de considérer deux niveaux et donc deux groupes de tâches. Le concepteur peut utiliser UAN pour spécifier les tâches "globales" du système, ou considérer l'interface d'un utilisateur donné et spécifier le comportement de cette interface isolément. Dans un cas comme dans l'autre, UAN révèle des limitations.

Si l'on considère le système de façon globale, la structuration de l'espace des tâches imposée par UAN nécessite de décomposer les tâches globales. Nous avons souligné que cette approche est réductrice, à moins de prendre en compte les tâches de coordination et de communication. [Jambon 1994] va dans ce sens et propose une extension de la notation UAN pour décrire une tâche collaborative faisant intervenir deux participants. Ce travail propose de mettre en regard les actions de chaque participant et les réponses du système ; il introduit également une notation pour indiquer les informations partagées et aborde également les problèmes de coordination (principalement l'attente par un participant d'une réponse de son interlocuteur) et de communication (transmission d'une information d'un utilisateur à un autre). A partir de cette représentation, on peut par exemple garantir qu'une information a été correctement échangée entre les utilisateurs. Mais, comme le remarquent les auteurs, la présentation choisie ne permet de prendre en compte que deux participants ; un plus grand nombre de participants rendrait le tableau UAN illisible. Il est probable qu'un support informatique de la notation permettrait d'envisager d'autres alternatives de représentation. En ce qui concerne l'étude de l'utilité, une spécification UAN prenant en compte la tâche globale et les tâches de coordination et de communication permettrait par exemple d'étudier de façon systématique la coordination, en particulier pour les systèmes de workflow. La complexité de la coordination imposée par le système pourrait être évaluée et une telle étude pourrait aider à détecter des redondances ou des incohérences dans les tâches de coordination. Toutefois, en l'absence d'une représentation de la notation UAN adaptée à plusieurs utilisateurs, et en l'absence d'une notation permettant d'exprimer la coordination et la communication, cette étude ne peut être faite que dans des cas élémentaires.

Si l'on considère la tâche d'un utilisateur donné, les problèmes de représentation sont simplifiés : on se ramène au cas mono-utilisateur. Toutefois, il faut pouvoir exprimer que des événements extérieurs dûs aux autres utilisateurs surviennent. Par sa structure qui fait suivre les actions de l'utilisateur par les réponses du système, il est peu naturel

d'exprimer avec UAN des événements qui ne sont pas une réponse du système à une action de l'utilisateur. Mais la difficulté la plus sérieuse tient au fait que l'on perd la nature multi-utilisateur du système. La spécification du partage d'informations, de l'accès concurrent à des objets d'interaction, des tâches de communication ne sont pas prises en compte par cette approche. Or ce sont précisément ces aspects du comportement du système qui mériteraient de bénéficier d'une description précise comme celle permise par UAN. Pour un éditeur de dessin partagé par exemple, il est intéressant de spécifier précisément le comportement de l'interface lorsque plusieurs utilisateurs sélectionnent le même objet graphique. L'approche considérant un seul utilisateur ne le permet pas.

Dans le cas des systèmes multi-utilisateurs, et quelle que soit l'approche considérée, UAN dans sa forme actuelle se révèle inadapté. Pourtant, les résultats intéressants obtenus avec les systèmes mono-utilisateurs et le potentiel de la notation pour l'étude de l'utilisabilité sont prometteurs. L'adaptation d'UAN à plusieurs utilisateurs reste un problème ouvert, mais certainement une voie de recherche à poursuivre.

5.5.4. Évaluation d'UAN

UAN est d'abord un outil de notation intéressant grâce à sa lisibilité et sa facilité d'apprentissage (au moins de notre point de vue, mais sa diffusion relativement large nous laisse supposer que ce point de vue est partagé). Nous avons cependant jugé utile d'apporter deux modifications à la notation. La première est une simplification : pour exprimer l'appui sur le bouton de la souris, la notation emploie le double symbole **Mv**. Nous l'avons remplacé par **v**, pour alléger la notation puisqu'il n'y a pas ambiguïté dans notre cas (**M** et **v** ne sont jamais utilisés avec une autre signification). Nous avons introduit une deuxième modification moins mineure qui concerne les préconditions : dans UAN, une précondition peut être utilisée à n'importe quel moment dans la colonne User Action. Nous avons souvent trouvé utile de regrouper les préconditions intervenant dans une tâche donnée et de les placer au début du tableau. Cette façon de faire permet aussi de mettre en évidence les variables intervenant dans la précondition. Ainsi elles peuvent être repérées plus facilement pour vérifier la propriété d'observabilité. L'heuristique que nous proposons est donc d'essayer d'isoler les préconditions en début de tableau. D'après notre expérience, si ce n'est pas possible c'est souvent que la tâche en question n'est pas suffisamment décomposée et que l'on peut identifier des sous-tâches dont les préconditions pourront, elles, être isolées en début de leurs tableaux UAN.

D'autres critiques peuvent être faites à UAN. Son caractère semi-formel, qui lui donne l'avantage de la lisibilité, conduit dans certains cas à une sémantique imprécise et à des possibilités d'interprétation multiples. Par exemple la disposition en colonnes amène

parfois à s'interroger sur les relations temporelles entre actions et réponses situées sur une même ligne et dans plusieurs colonnes. UAN souffre aussi de l'absence de certaines facilités d'écriture : il n'y a pas par exemple de mécanismes d'encapsulation comme les macros ou les procédures.

Outre la difficulté d'adaptation d'UAN aux systèmes multi-utilisateurs, nous regrettons aussi que la notation ne permette pas d'exprimer le lien entre l'interface et le noyau fonctionnel. Cette lacune nous a par exemple empêchés de proposer une règle permettant de vérifier la propriété d'annulabilité. En effet, dans ce cas, nous devons pouvoir vérifier qu'après annulation, l'interface et le noyau fonctionnel sont revenus dans leur état antérieur à l'exécution de la commande annulée. Avec UAN, on ne peut vérifier ce fait que pour l'interface ce qui est d'un intérêt limité et ne permet certainement pas de vérifier l'annulabilité. L'ajout d'une quatrième colonne représentant l'état du noyau fonctionnel résoudrait ce problème, mais la lourdeur d'écriture qui en résulterait incite à se demander si les inconvénients ne primeraient pas sur les avantages.

Nous avons montré que l'utilisation d'une technique d'évaluation prédictive reposant sur les propriétés permet de mettre en évidence des défauts d'utilisabilité en raisonnant sur les spécifications. Toutefois, nous avons rencontré des difficultés à étendre cette technique aux systèmes multi-utilisateurs. Nous apportons ainsi confirmation du manque de maturité des techniques prédictives pour l'étude des systèmes multi-utilisateurs. Face à cet état de fait, il est raisonnable de se tourner vers les techniques expérimentales.

5.6. Évaluation expérimentale

L'évaluation expérimentale est le terrain privilégié des ergonomes. Le manque de maturité de beaucoup de techniques d'évaluation prédictive, le rôle d'entraînement de quelques grandes sociétés de logiciel qui se sont équipées de laboratoires d'utilisabilité et l'aspect pratique et tangible des tests d'utilisabilité sont probablement la cause de cet état de fait. On peut se féliciter de cette vogue de l'utilisabilité. Il faut cependant garder à l'esprit les points suivants :

- l'évaluation expérimentale nécessite un prototype ou une maquette du logiciel. Même dans le cas d'une évaluation d'une maquette carton ou papier, des spécifications précises sont nécessaires. Cela signifie que l'évaluation expérimentale intervient plus tard que l'évaluation prédictive dans le cycle de vie. Les problèmes détectés seront donc plus coûteux à corriger.

- L'évaluation expérimentale ne s'improvise pas. La compétence de plusieurs ergonomes est indispensable, à la fois pour la préparation, la conduite et l'exploitation des résultats des tests d'utilisabilité. [Valentin 1993] affirme que la participation de trois ergonomes est nécessaire pour détecter 90% des problèmes d'utilisabilité. Rappelons que l'évaluation expérimentale est une approche artisanale dans la classification de Long et Dowell. La compétence des ergonomes est largement empirique et est donc difficile à transférer à des non-experts.

- Le travail des ergonomes est difficile. Les sessions d'utilisabilité sont longues et nécessitent une attention soutenue. Le dépouillement des sessions d'expérimentation nécessite d'analyser une masse importante de données, sous forme de vidéo, questionnaires, notes d'expérimentation.

L'avantage certain de l'évaluation expérimentale est de fournir des données réelles, obtenues par observation de sujets représentatifs des utilisateurs effectifs. On peut toutefois s'interroger sur la validité de données recueillies dans le contexte d'un laboratoire par opposition à l'observation des utilisateurs dans leur contexte de travail habituel. D'autre part, l'évaluation expérimentale permet d'évaluer, grâce en particulier aux questionnaires, la satisfaction de l'utilisateur. Et nous avons vu que la satisfaction de l'utilisateur contribue à l'utilisabilité d'un système.

Pour les systèmes multi-utilisateurs et les outils de communication homme-homme médiatisée, l'absence de techniques prédictives opérationnelles et prenant en compte explicitement ces systèmes plaide comme nous l'avons vu en faveur de l'approche expérimentale. De plus, la composante humaine et sociale, particulièrement dans les outils de communication, ne peut se satisfaire d'une évaluation uniquement théorique. Cependant, l'impact que ces systèmes peuvent avoir sur l'organisation et le fonctionnement d'un groupe social, ou d'une organisation complexe comme une entreprise, ne peut être raisonnablement estimé lors de sessions d'expérimentation en laboratoire. Des évaluations "sur le terrain" sont alors nécessaires.

Dans cette section, nous présentons d'abord nos observations, "in vivo" ou à partir de discussions ou des comptes rendus, du travail des ergonomes dans quelques laboratoires d'utilisabilité de grandes entreprises informatiques (Hewlett-Packard, CCETT, Lotus, Claris, Apple). A partir de ces observations, nous proposons les éléments d'un cahier des charges pour un ensemble d'outils permettant d'aider les ergonomes dans leurs tâches. Ces éléments ont mené à la réalisation de l'environnement NEIMO présenté dans la section suivante.

5.6.1. L'activité des ergonomes dans l'évaluation expérimentale

Dans l'évaluation expérimentale d'un système interactif, l'activité des ergonomes peut être découpée en trois phases principales : la préparation de l'expérimentation, la conduite de l'expérimentation et l'analyse des résultats obtenus.

5.6.1.1. La préparation de l'expérimentation

Pour mener correctement une expérimentation, les ergonomes doivent connaître parfaitement le système à évaluer. Leur première tâche est donc de se familiariser avec le système dans son ensemble (y compris la documentation) et le domaine applicatif. Ils identifient les tâches caractéristiques qui peuvent être réalisées avec le système et découvrent des problèmes d'utilisabilité potentiels. Notons qu'en règle générale, les tâches caractéristiques ont été déterminées lors de l'analyse de tâche. Sauf dans des cas flagrants, ces problèmes potentiels devront être infirmés ou confirmés par l'expérimentation.

En règle générale, même les problèmes d'utilisabilité flagrants sont confirmés par l'expérimentation. En effet, voir un utilisateur buter sur une difficulté ou plusieurs utilisateurs répéter systématiquement la même erreur est souvent le meilleur moyen pour un ergonome de convaincre un concepteur ou un développeur du bien-fondé de son analyse. Nous avons remarqué à plusieurs reprises que les ergonomes jouent un rôle pédagogique certain auprès des concepteurs, mais qu'ils doivent être prêts à argumenter toutes leurs assertions. Face à un concepteur convaincu de ses choix de conception, le savoir empirique d'un ergonome est moins démonstratif que l'observation des utilisateurs !

Les ergonomes identifient aussi les utilisateurs représentatifs et déterminent des scénarios caractéristiques. Les utilisateurs sont souvent choisis pour leur connaissance du domaine applicatif et leur niveau d'expertise, suivant la population à laquelle le système est destiné. La mise au point des scénarios repose sur le savoir-faire des ergonomes. Un scénario doit être représentatif des tâches et doit être aussi neutre que possible vis-à-vis du problème d'utilisabilité que l'on souhaite étudier. Le scénario ne doit pas laisser deviner aux sujets le point qui intéresse les ergonomes. Le sujet, se sachant dans une situation de test—même si ce n'est pas ses capacités personnelles mais le système qui est testé—aura parfois tendance à vouloir deviner le but du test. Un ergonome de Lotus insiste ainsi sur la nécessaire cohérence des dessins dans la conception d'une maquette papier : "les dessins doivent être cohérents ; si vous avez par exemple un beau dessin en couleur et que le reste est brouillon, l'utilisateur pensera—ah c'est donc ça qu'ils veulent tester !".

Nous retrouverons cette importance de la cohérence dans la conception et la conduite des expérimentations Magicien d'Oz.

Enfin l'environnement représentatif est déterminé. Si la plupart des laboratoires d'utilisabilité mettent les sujets dans une classique situation d'environnement de bureau, certaines applications grand public requièrent un environnement plus personnel. Les tests des décodeurs de télévision VisioPass par le CCETT par exemple se déroulaient dans une salle recréant un salon d'appartement.

5.6.1.2. La conduite de l'expérimentation

Suivant la complexité du système, le nombre de scénarios élaborés, et aussi la facilité à trouver des sujets adéquats, entre trois et une dizaine de sessions d'expérimentation auront lieu. Souvent, la grande majorité des problèmes d'utilisabilité est détectée dès les deux ou trois premières sessions. La durée d'une session d'expérimentation peut varier, suivant le système et les scénarios, d'une demi-heure à une demi-journée. La technique de verbalisation (*thinking aloud*) est quasiment systématiquement utilisée. Les ergonomes préfèrent faire travailler les sujets par couple : ils sont ainsi conduits à parler entre eux, ce qui rend leurs actions plus compréhensibles aux ergonomes. Sans cet artifice, les ergonomes ont souvent du mal, même en observant avec attention l'écran et le sujet, à deviner les actions et les intentions des sujets. Notons que dans le cas multi-utilisateur et en particulier pour la communication homme-homme médiatisée, la technique du "thinking aloud" est compromise. La communication entre utilisateurs interfère avec le "thinking aloud". Dans le meilleur des cas, la surcharge cognitive du sujet est certaine. Ces constatations ont renforcé notre conviction de la nécessité d'une capture informatique des actions du sujet.

L'équipement et la disposition des laboratoires d'utilisabilité varient, mais on repère certaines constantes. Les ergonomes et les observateurs sont dissimulés derrière une glace sans tain et observent les sujets à travers la glace. L'utilisation de caméras est systématique, avec en moyenne trois caméras : une vue d'ensemble de la pièce, une vue du ou des sujets de face, et une vue de l'écran. Parfois, une caméra supplémentaire fixée au plafond donne une vue d'ensemble du bureau afin de surveiller l'usage de la documentation. Une régie vidéo permet de choisir la vue affichée dans la cabine des expérimentateurs et permet aussi de mélanger les vues des différentes caméras. La vue d'ensemble semble en général assez peu utilisée. La vue affichée dans la cabine est aussi enregistrée sur magnétoscope. La bande vidéo est ensuite archivée pour utilisation lors de l'analyse des résultats. Notons que des expérimentateurs entraînés (chez Hewlett-Packard) ont très peu recours à la bande vidéo. La majeure partie de leur analyse se fait "au vol" et à partir de leurs notes prises en cours de session. La bande vidéo est revue

lorsque les notes sont ambiguës ou imprécises. L'utilité de l'enregistrement dans ce cas semble plus être une sécurité, ou une preuve pour justifier un problème d'utilisabilité auprès d'un concepteur. Cet usage réduit de la vidéo (à la fois lors de la session où l'observation directe est privilégiée, et pour l'analyse) a motivé notre choix de ne pas incorporer immédiatement l'enregistrement vidéo dans NEIMO.

Suivant les laboratoires, le nombre de personnes assistant à une session d'expérimentation est très variable. Chez Hewlett-Packard par exemple, une expérimentation a typiquement lieu en présence de deux ergonomes et du rédacteur de la documentation du système. Ce dernier vérifie l'utilisabilité de la documentation papier et en examine la pertinence. Chez Lotus en revanche, la cabine d'observation est spécialement conçue pour accueillir une quinzaine d'invités : concepteurs, développeurs, autres observateurs, … Il leur est demandé de noter toutes les remarques qui leur viennent à l'esprit lors de la session. Les instructions pour les observateurs précisent explicitement que toutes ces notes sont anonymes. Cette importance accordée aux notes prises au vol dans toute session d'expérimentation nous a conduit a introduire des possibilités d'annotation dans l'environnement d'observation NEIMO.

Lors d'une session, les notes prises par les ergonomes ne sont pas uniquement des annotations. Les expérimentateurs remplissent également un formulaire au fur et à mesure de l'avancement de la session. Les données recueillies sont quantitatives (temps d'exécution de chaque tâche du scénario, nombre d'erreurs) et qualitatives (stratégie utilisée en cas de choix, hésitations, difficultés). Les ergonomes observent et notent également les écarts entre activité (tâche effective) et tâche (prévue). Pour les expérimentations concernant des systèmes multi-utilisateurs, les expérimentateurs rapportent compter le nombre de fois où les sujets disent : "qui a fait ça ?", "qui a dit ça ?", "où es-tu ?". Ces observations permettent de vérifier les propriétés de rétroaction de groupe et d'awareness que nous avons définies au chapitre précédent.

Un soin particulier est pris pour l'accueil et le confort des sujets. Les sujets sont explicitement prévenus de leur participation à un test et du but du test (évaluer la facilité d'utilisation d'un logiciel). Ils savent également qu'ils seront observés et filmés. Chez Lotus, on notifie aux sujets que les enregistrements pourront être utilisés, mais de façon anonyme. Pendant le déroulement des scénarios, les expérimentateurs s'interdisent toute communication avec le sujet. En particulier, ils doivent résister à la tentation naturelle de guider un sujet qui s'enferre dans une suite d'erreurs. Ce n'est qu'exceptionnellement et après hésitation que l'expérimentateur se résout à aider un sujet. Typiquement, le problème rencontré par le sujet a déjà été détecté lors d'une expérimentation précédente et laisser le sujet répéter les mêmes erreurs n'apporterait pas d'informations

supplémentaires. Si le sujet demande de l'aide à l'expérimentateur, celui-ci est tenu de fournir une réponse évasive qui ne puisse pas réellement renseigner le sujet. Une fois les scénarios prévus exécutés, les sujets sont interrogés par les expérimentateurs. Cette interview a plusieurs buts : dissiper d'éventuelles incertitudes des ergonomes sur l'interprétation de ce qu'ils ont observé, obtenir des informations qualitatives et des appréciations sur le système de la part des sujets, évaluer leur satisfaction.

5.6.1.3. L'analyse des résultats de l'expérimentation

Sauf pour les ergonomes experts de Hewlett-Packard auxquels nous avons fait allusion plus haut, la phase d'analyse des résultats de l'expérimentation a posteriori est celle qui demande le plus de travail. Les notes prises pendant les sessions, les vidéos et les questionnaires sont dépouillés. Ils servent à analyser les stratégies (en comparant l'activité des sujets à la tâche prévue) et à étudier les erreurs et problèmes rencontrés par les sujets. Les données quantitatives recueillies pendant les différentes sessions sont pondérées, les données qualitatives et les questionnaires sont rapprochés et résumés.

Ces tâches nécessitent la manipulation et l'analyse d'importants volumes de données : les vidéos en particulier nécessitent une visualisation longue et souvent fastidieuse pour traquer les moments "intéressants". Il faut localiser les événements significatifs, les caractériser et éventuellement les quantifier, puis en abstraire un défaut d'utilisabilité. Cette recherche doit être faite sur tous les enregistrements, et mène à établir des recoupements entre les différentes sessions. On devine ici, étant donné l'absence d'outils de recherche élaborés de séquences vidéo ou de systèmes d'indexation performants, les contraintes de manipulation et de recherche manuelle auxquelles sont soumis les ergonomes. Il existe toutefois des outils permettant l'indexation et la recherche d'index sur des enregistrements vidéo. Le système MUSiC [MacLeod 1993] en est un exemple caractéristique ; mais la nécessité d'indexer manuellement les séquences vidéo limite l'intérêt de tels outils.

Après analyse et synthèse des données recueillies, les ergonomes sont en mesure d'établir un rapport détaillé sur les problèmes d'utilisabilité rencontrés, et de suggérer des améliorations. Ces suggestions peuvent porter 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.

5.6.2. Les outils d'aide à l'expérimentation

L'exposé de nos observations des activités des ergonomes montre leur diversité et leur complexité, comme le dépouillement des données recueillies. Bien que les ergonomes

côtoient des outils informatiques, leur travail n'a pas de support informatisé. On peut s'étonner par exemple que chez Lotus les notes sur Post-It soient l'outil principal utilisé lors d'un test d'utilisabilité. Même si les ergonomes utilisent des outils informatiques classiques (traitement de texte, outils d'analyse statistique), il n'existe pas d'outil dédié pour assister toutes les tâches des ergonomes. Cependant, on trouve des outils permettant d'aider certaines tâches, comme la capture des actions de l'utilisateur, ou l'annotation des vidéos. Nous dressons d'abord un bref état de l'art de ces outils, puis, constatant leurs insuffisances et leur manque d'intégration, nous proposons une approche intégrée du support informatisé pour l'évaluation expérimentale. Nous présentons aussi la technique expérimentale dite du Magicien d'Oz et ses liens avec les autres techniques expérimentales.

Des outils peuvent apporter une aide aux ergonomes pour faciliter les tâches d'observation, d'analyse en fournissant des données précises comme les outils de capture ou en facilitant la manipulation des enregistrements vidéo. Notons que la plupart de ces outils sont encore utilisés dans un contexte de recherche et que très peu sont disponibles hors de ce cadre.

Les outils de capture des actions de l'utilisateur sont potentiellement extrêmement utiles pour les ergonomes. Les systèmes existant capturent les événements système dans un fichier à des fins d'analyse ou pour rejouer la session. Le système de [Hammontree 1992] effectue une capture au niveau des éléments d'interaction (widgets) : il enregistre les événements clavier ou souris et les date. Mais cette limitation au niveau widget ne permet qu'une couverture partielle des domaines applicatifs : la manipulation directe du type MacDraw par exemple n'est pas prise en compte. Il est possible de filtrer et de n'enregistrer que certains types d'événements. Ce système est couplé à un enregistreur vidéo et les événements capturés sont synchronisés avec l'enregistrement vidéo. Le système permet aussi d'enregistrer des annotations verbales qui sont elles aussi synchronisées. Il existe d'autres systèmes effectuant des captures d'événements système mais celui de Hammontree est l'un des plus complets. Mais cette approche souffre de limitations.

La capture d'événements d'un bas niveau d'abstraction comme les clics souris est de peu d'utilité directe aux ergonomes. Il faut ensuite à partir de ces événements de bas niveau retrouver les commandes exécutées. Le lien avec l'enregistrement vidéo facilite cette opération. La capture informatique permet d'envisager un traitement automatique comme l'approche EMA proposée par [Balbo 1994]. Rejouer une session peut aussi soulever des difficultés : il faut envoyer les événements au système en respectant exactement les dates des événements. D'autre part, le contexte de départ doit être fixé précisément (position

des fenêtres à l'écran par exemple). Si l'on envoie l'événement "relâcher la souris" un tout petit peu trop tard, la position de la fenêtre que l'on était en train de déplacer n'est plus la même que dans la session originale, le clic suivant n'aura pas un contexte correct et risque de perdre sa signification. Ces problèmes de précision et de contexte typiques de la capture à bas niveau d'abstraction, sont suffisamment sérieux pour être un réel obstacle à l'utilisation de cette technique. Le filtrage proposé dans l'outil de Hammontree est intéressant : il permet de limiter la quantité d'informations capturées. Mais il a l'inconvénient d'être fixé de façon statique et ne peut être changé en cours de session. Or au cours d'une session, au fur et à mesure de l'exécution des différentes tâches par le sujet, les événements intéressants varient.

Les outils d'indexation vidéo comme l'outil d'Hammontree ou MUSiC facilitent la manipulation des enregistrements vidéo. Mais comme nous l'avons vu auparavant, l'indexation doit être faite manuellement, ce qui limite leur intérêt. Dans l'outil d'Hammontree toutefois, la synchronisation avec le fichier de capture et les annotations permet de retrouver rapidement les passages utiles au travail d'analyse.

Le manque d'outils réellement adaptés à l'expérimentation nous a conduit à développer le projet NEIMO, une plate-forme d'observation et de capture du comportement de l'utilisateur. NEIMO étant aussi une plate-forme pour des expérimentations Magicien d'Oz, nous présentons cette technique d'évaluation.

5.6.3. La technique du Magicien d'Oz

Une expérimentation Magicien d'Oz consiste à faire simuler par un "compère" humain les services non implémentés d'un logiciel, utilisé par un sujet qui ignore la présence du compère. En général le compère observe le sujet grâce à des moyens informatiques. Lorsque le sujet déclenche l'exécution d'une commande non implémentée, le résultat de la commande est généré par le compère et transmis au sujet. Lors de l'expérimentation, les échanges informatiques entre le sujet et le compère sont enregistrés pour analyse ultérieure. Jusqu'à présent, les systèmes Magicien d'Oz ont généralement été utilisés pour simuler la reconnaissance du langage naturel, parlé ou écrit, pour des services d'informations téléphonées [Richards 1984], l'interrogation de bases de données [Dahlbäck 1988] ou de systèmes experts [Diaper 1989]. Pour l'interrogation de services téléphoniques, le sujet appelle un service supposé automatique. Pour parfaire l'illusion, la voix du compère qui lui répond est déformée par un filtre électronique qui donne l'impression d'une voix artificielle. Hors des systèmes de reconnaissance de langage naturel, nous n'avons trouvé que deux systèmes visant l'étude d'autres types d'interfaces qui prennent en compte la manipulation directe. [Dahlbäck 1989] décrit un système

Magicien d'Oz développé pour l'étude d'applications graphiques. Mais les auteurs reconnaissent avoir rencontré des difficultés dans la mise en œuvre et l'utilisation du système. Le dispositif duplique la fenêtre de travail du sujet sur l'écran du compère et toutes les événements de bas niveau générés par les actions du sujet sont répercutés vers la machine du compère. Les problèmes de synchronisation évoqués plus haut pour rejouer des événements de bas niveau semblent ici avoir été la cause des problèmes rencontrés.

Le système Turvy [Maulsby 1993] est un agent "intelligent", en fait une simulation d'agent que l'on peut programmer par démonstration. Le système permet d'utiliser la parole et la manipulation directe pour programmer l'agent simulé par le compère. Ce système s'est révélé d'une grande aide pour le prototypage rapide de l'agent logiciel Turvy. Enfin [Mignot 1993] décrit l'utilisation d'un dispositif Magicien d'Oz pour l'étude de l'interaction multimodale. Le système est utilisé pour simuler entièrement un système de dessin utilisant la voix et la manipulation directe.

Cet état de l'art des systèmes Magicien d'Oz met en évidence l'utilisation privilégiée de ce type de systèmes : la simulation pour l'étude de nouvelles techniques d'interaction. Deux contextes sont cependant à distinguer : dans de nombreux cas on étudie une technique d'interaction en la simulant dans le but d'étudier la technique elle-même et son usage, et non une interface déterminée. En reconnaissance de la parole par exemple, cette technique permet de constituer des corpus et de rendre plus robuste un système de reconnaissance existant. Dans d'autres cas, c'est une interface précise qui est étudiée et, en l'absence de la technologie adéquate ou du composant logiciel nécessaire, certains services sont simulés. Cette dernière approche montre que la technique du Magicien d'Oz peut être un auxiliaire précieux pour le prototypage. Cependant les expériences Magicien d'Oz présentent des difficultés de mise en œuvre.

La simulation du langage naturel, donc d'une seule technique d'interaction se heurte à des difficultés lorsqu'un seul compère assure la simulation [Polity 1990]. En particulier, les temps de réponse deviennent vite rédhibitoires pour le sujet qui utilise les services simulés. En effet, les tâches demandées au compère, même si elles semblent simples, exigent un travail cognitif important. Les demandes de l'utilisateur sont difficilement prévisibles et surtout le compère doit se conformer à certaines règles pour parvenir à simuler de façon réaliste les réponses d'une machine : les réponses doivent être cohérentes au regard de leur contenu, leur style et leur régularité. Deux demandes identiques du sujet doivent obtenir la même réponse. Le temps de réponse doit être conforme aux attentes du sujet : si le compère est trop lent, le sujet risque d'éviter l'usage du service simulé. Pour garantir le respect de ces contraintes de cohérence, plusieurs

approches ont été essayées. Tout d'abord, les tâches du compère sont précisément définies ainsi que leurs limites d'action. Par exemple une requête non prévue doit provoquer un message "Je ne comprends pas". Les compères sont préparés à leurs tâches et suivent un entraînement intensif : on ne s'improvise pas compère ! Le compère peut aussi disposer d'un support informatique : dans [Mignot 1993], le compère peut déclencher l'émission de messages parlés préenregistrés ; dans [Dahlbäck 1989], le compère combine des éléments de réponse à l'aide de menus. Enfin, une seule expérimentation à notre connaissance utilise plus d'un compère afin de tenter d'alléger la tâche de simulation. Dans [Mignot 1993], deux compères coopèrent pour la simulation : l'un des compères a la charge de la communication avec le sujet, et l'autre compère génère effectivement le résultat des requêtes sous le contrôle du premier. Dans cette expérimentation, un troisième compère est parfois intervenu, chargé de vérifier la cohérence des réponses fournies au sujet et de coordonner l'action des deux autres. Nous avons poussé plus loin ce découpage des tâches dans NEIMO pour tenir compte des exigences de l'interaction multimodale.

5.7. Réalisation : NEIMO

NEIMO (Nouvelle Evaluation des Interfaces par Magicien d'Oz) est un environnement pour le test d'utilisabilité des logiciels interactifs. Il représente une première étape vers un laboratoire numérique d'utilisabilité. Il fournit des outils pour l'expérimentation et pour l'analyse des données recueillies. Nous exposons ces deux facettes de l'environnement NEIMO, puis nous présentons brièvement une des expérimentations que nous avons réalisées dans cet environnement. Cette expérimentation concerne un système de communication homme-homme médiatisée. Enfin, nous tirons des leçons de cette réalisation et proposons un agenda de recherches pour faire évoluer cet outil.

Le cœur de l'environnement NEIMO est une plate-forme Magicien d'Oz pour l'étude de l'interaction multimodale. Comme nous l'avons indiqué au paragraphe 5.6.3 un dispositif Magicien d'Oz demande aux compères un lourd travail de simulation. C'est pourquoi NEIMO a été conçu dès l'origine pour permettre à plusieurs compères de collaborer pour simuler les services manquants du logiciel testé. De plus, NEIMO est aussi une plate-forme d'observation. Des observateurs prennent part à la session et disposent d'outils d'annotation. La figure 5.5 montre une configuration typique de NEIMO.

Le sujet observé a à sa disposition un ensemble de modalités d'interaction. Ici, par exemple, la parole est utilisée et la reconnaissance de la parole est simulée par un compère. Deux observateurs participent aussi à la session. Le travail de simulation est coopératif : les compères se répartissent la tâche de simulation et peuvent, par exemple,

simuler chacun un service différent. Les observateurs et le sujet sont aussi des participants actifs avec des rôles particuliers. NEIMO est donc un système multi-utilisateur. Nous décrivons l'environnement dans cette perspective au paragraphe 5.7.4. Nous détaillons en particulier les services que nous avons développés pour tenir compte de l'interaction multi-utilisateur. Une description de l'architecture logicielle du système NEIMO est présentée au chapitre 7.

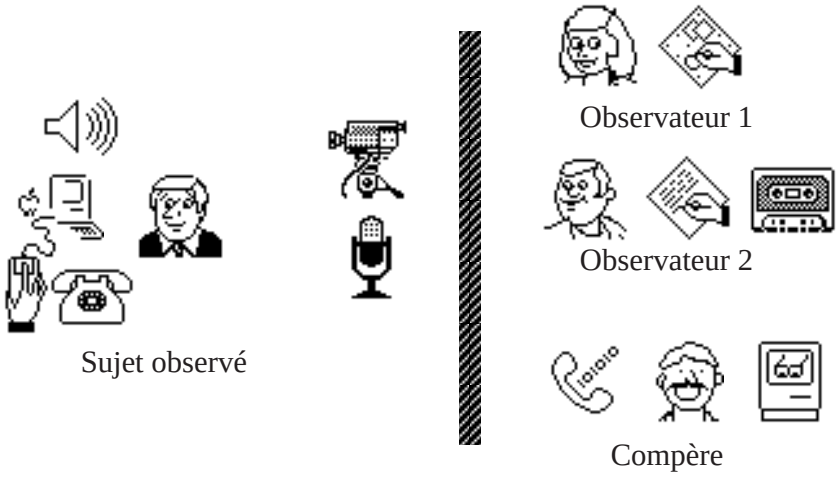


Figure 5.5. Exemple de configuration de NEIMO

5.7.1. NEIMO : outils pour l'expérimentation

Pour assister l'expérimentation, NEIMO offre une plate-forme Magicien d'Oz qui permet la simulation de services manquants dans le logiciel testé, l'observation et la capture informatique du comportement de l'utilisateur. NEIMO fournit aussi des outils aux ergonomes pour la conduite d'une session d'expérimentation.

5.7.1.1. Adaptation d'un logiciel à l'environnement de test

Pour pouvoir utiliser l'environnement de test d'utilisabilité, une application doit être adaptée à NEIMO. Plus précisément, elle doit être interfacée avec NEIMO via une interface de programmation définie. Cette interface permet au développeur de spécifier les actions de l'utilisateur qui doivent être transmises aux compères et/ou capturées pour analyse ultérieure. Par exemple, si le dispositif Magicien d'Oz est utilisé pour simuler la reconnaissance de la parole, le logiciel testé envoie à l'environnement NEIMO les phrases prononcées par le sujet sous forme de sons. Le compère reçoit ces sons et interprète les phrases du sujet et agit sur l'interface utilisée par le sujet pour réaliser les commandes demandées.

L'inconvénient de notre approche est qu'elle nécessite un développement spécifique. Nous avons toutefois essayé de minimiser le coût de ce développement. Une fois identifiées les informations pertinentes à transmettre aux compères ou à capturer, il faut faire un appel à la bibliothèque NEIMO pour chaque information différente. La bibliothèque est réduite pour minimiser son apprentissage (cinq à dix fonctions sont utiles au développeur). Comme le logiciel doit être adapté, il faut disposer de son code source pour pouvoir l'utiliser avec NEIMO. Cette limitation n'est pas gênante pour l'évaluation "formative" utilisée en cours de développement. Elle interdit cependant l'évaluation "summative", pour évaluer par exemple des logiciels commerciaux déjà développés. Nous expliquons au paragraphe 5.7.5 comment nous envisageons de nous affranchir de cette limitation.

Cette nécessaire adaptation a heureusement plus d'avantages que d'inconvénients : en intégrant explicitement dans le code source le lien avec l'environnement de test d'utilisabilité, elle donne plus de souplesse aux concepteurs. Par exemple, ils sont à même de choisir le niveau d'abstraction des actions de l'utilisateur qu'ils souhaitent capturer : du niveau événement jusqu'au au niveau commande. Un clic sur un bouton peut par exemple être enregistré comme { clic en (140, 203) }, comme { appui sur le bouton Raccrocher } ou comme { commande Raccrocher }. Sans un modèle de l'application, il n'est pas possible d'abstraire les commandes à partir des événements de bas niveau.

L'adaptation à NEIMO du logiciel à tester a deux autres avantages qui relèvent aussi d'une plus grande souplesse : la sélectivité et l'extensibilité. Contrairement à une solution automatique capturant les événements de bas niveau ou à une solution fondée sur un modèle de l'application, notre approche permet aux concepteurs de choisir les éléments pertinents à capturer. Ainsi, NEIMO permet une capture sélective et les expérimentateurs peuvent ainsi se concentrer sur un ensemble de problèmes d'utilisabilité précis. Nous avons vu que l'importance en volume des données recueillies lors d'une expérimentation est un des problèmes auxquels sont confrontés les ergonomes. Limiter le volume de données à la source pour éliminer le plus tôt possible le "bruit" de l'information intéressante nous semble une contribution importante de notre solution. Enfin, en laissant au développeur le choix des informations à capturer, cette solution garantit l'extensibilité du système. Ce point est particulièrement important pour le test des interfaces multimodales dans lesquelles les modalités utilisées sont susceptibles d'évoluer.

5.7.1.2. Conduite d'une expérimentation avec NEIMO

Lors d'une expérimentation avec l'environnement NEIMO, quatre types de participants interviennent : sujet, observateur, compère et super-compère.

- Un ou plusieurs *sujets* utilisent le logiciel à tester suivant un scénario mis au point par des ergonomes. Les actions du sujet sont capturées comme indiqué au paragraphe précédent. Même si jusqu'à présent nos expérimentations n'ont mis en jeu qu'un seul sujet, NEIMO permet la participation de plusieurs sujets.

- Des *observateurs* suivent la session. Ils disposent d'une station et d'une interface adaptée grâce à laquelle ils voient sur leur écran une copie de l'écran du sujet. Ils peuvent créer des annotations écrites ou verbales qui seront capturées. Ils surveillent aussi l'exécution du scénario, et pour chacune des tâches identifiées au préalable, ils enregistrent deux indications : leur opinion sur la bonne fin de la tâche, et la satisfaction de l'utilisateur quant à la réalisation de la tâche. Ces informations correspondent à celles inscrites par les ergonomes sur des formulaires dans une session d'utilisabilité. Les temps d'exécution des tâches et des scénarios sont calculés par le système d'après les indications de début et de fin données par les observateurs.

- Les *compères* ont la charge de simuler des services manquants du système, le cas échéant. Chaque compère a une tâche bien identifiée. Par exemple un compère dédié à cette tâche peut simuler un reconnaisseur de parole en écoutant les commandes vocales du sujet et en simulant, via des commandes adaptées, l'effet de ces commandes. Chaque compère dispose d'une station et d'une interface prévue pour sa tâche de simulation. Nous avons été particulièrement attentifs à cette interface. Comme nous l'avons dit plus haut, la tâche d'un compère est cognitivement exigeante. De plus, il doit satisfaire des contraintes de rapidité et de cohérence de ses réactions aux commandes émises par le sujet. L'interface inclut donc un ensemble d'éléments prêts à l'emploi que le compère peut réutiliser. L'objectif ici est double : minimiser les actions du compère dans un but de rapidité (optimisation au niveau "keystroke") et préparer à l'avance tout ce qui peut l'être en fonction du scénario dans un souci de cohérence. Dans le cas multi-utilisateur, un compère peut simuler un autre utilisateur ou un interlocuteur dans le cas de la communication homme-homme médiatisée. Nous en verrons un exemple avec l'expérimentation Supratel au paragraphe 5.7.4.

- Le *super-compère* est un compère qui n'intervient que lorsque plusieurs compères participent à la session et dont le rôle est particulier. Il a la tâche de superviser et d'orchestrer les actions des compères. Il s'occupe aussi de l'administration et de la configuration de la session. Un aspect important de sa tâche est de surveiller la cohérence des réponses des compères. Dans ce but, il n'a pas de support

informatique proprement dit et l'interface qui lui est dédiée est proche de celle d'un observateur.

Lors de la session, des données comportementales correspondant aux actions de l'utilisateur sont capturées à différents niveaux d'abstraction comme nous l'avons vu précédemment. Les annotations des observateurs, et éventuellement les actions des compères sont également enregistrées. Ce fichier historique sert ensuite de support à l'analyse de la session. L'environnement NEIMO comporte un outil d'analyse de la session a posteriori.

5.7.2. NEIMO : outils pour l'analyse

NEIMO offre des outils à l'ergonome pour l'analyse d'une session a posteriori. Ces outils permettent de rejouer la session en visualisant les actions du sujet et les simulations réalisées par les compères, de retrouver les annotations des observateurs et de les analyser en liaison avec un diagramme représentant les tâches. La figure 5.6 montre des copies d'écran de ces différents outils dans leur application à l'analyse d'une session Supratel [Lischetti 1994]. L'expérimentation Supratel est présentée au paragraphe suivant.

Les outils permettent de visualiser les différentes informations capturées durant la session :

- les actions du sujet et les interventions des compères sont visualisées sur une réplique de l'interface. L'interface utilise la métaphore du magnétoscope et permet de se déplacer dans l'enregistrement.

- Les scénarios sont présentés sous une forme synthétique. Pour chaque scénario, les annotations des observateurs sont visibles. Par exemple, la tête souriante et le pouce levé visibles en bas de la figure 5.6 indiquent que le scénario est considéré comme réussi à la fois du point de vue de l'observateur et du point de vue du sujet. Les annotations indiquent aussi les difficultés rencontrées par les utilisateurs et indiquent pour chacune son nom, le type d'erreur, la tâche concernée et la gravité de l'incident. Pour chaque scénario, les tâches réalisées sont présentées sous forme d'un diagramme de Gantt. Il permet de voir les instants de début et de fin de chaque tâche. Les annotations sont matérialisées sur l'échelle de temps par les icônes en forme de point d'exclamation.

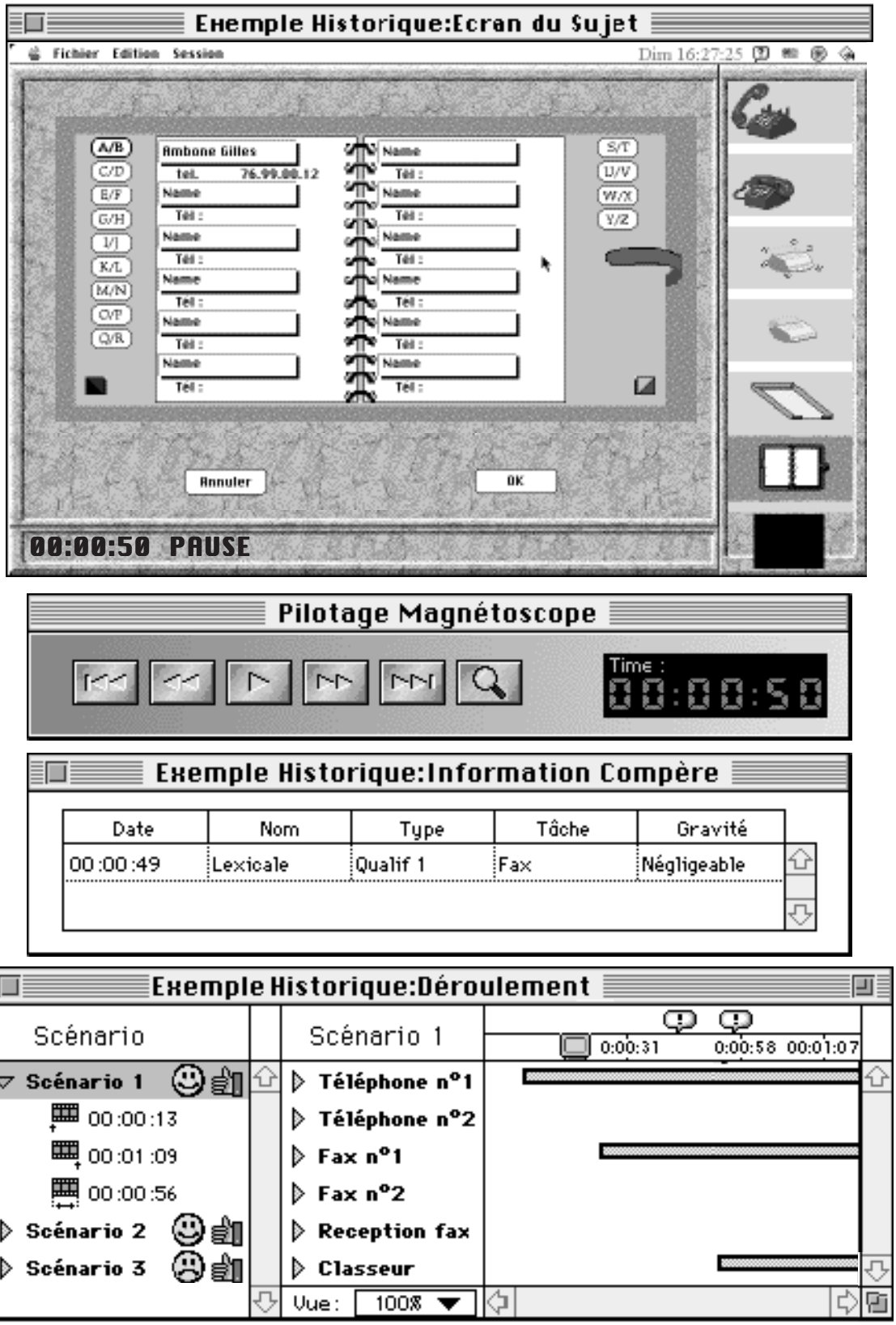


Figure 5.6. Copies d'écran des outils d'analyse de NEIMO. En haut, la visualisation des actions du sujet sur une réplique de l'interface étudiée. En dessous, l'interface de contrôle de la visualisation de la session et la liste des annotations des observateurs. En bas, un récapitulatif des scénarios et des tâches sous forme d'un diagramme de Gantt.

Avec ces outils, les ergonomes peuvent continuer leur analyse commencée pendant la session. Ils peuvent notamment compléter leurs annotations et revoir les passages qui ont posé des difficultés à l'utilisateur. Plusieurs efforts ont été faits pour permettre une analyse sélective des résultats. En effet, même si les mécanismes de NEIMO permettent déjà une capture sélective, la quantité de données capturées au cours d'une session peut être importante. Deux approches complémentaires sont utilisées :

- un mécanisme de vues qui permet de filtrer les informations de l'historique. Les vues agissent comme des filtres suivants des critères définis par l'ergonome : type des actions de l'utilisateur (ces types sont déterminés lors de la conception de l'application étudiée), fenêtre temporelle, utilisateur concerné dans le cas d'une application multi-utilisateur, tâches qui ont donné lieu à des annotations, etc. Différents filtres peuvent être combinés.

- Une présentation hiérarchique des scénarios et des tâches et une présentation synthétique des relations entre les tâches de l'utilisateur sous forme de diagramme de Gantt. Cette présentation permet de voir rapidement les relations temporelles entre tâches pour le cas de dialogue à fils multiples (par exemple entrelacement ou parallélisme).

Enfin, les outils d'analyse de NEIMO peuvent exporter des données numériques comme les dates de début et de fin de chaque tâche vers un tableur comme Excel. Il est ensuite possible d'effectuer des comparaisons entre plusieurs expérimentations avec la même interface.

A la différence des outils existants qui permettent de rejouer une bande vidéo et de l'annoter, cet outil d'analyse présente l'intérêt de visualiser une session d'expérimentation à plus haut niveau d'abstraction : en identifiant les scénarios et les tâches et en les liant aux annotations prises pendant la session, il fournit une aide supplémentaire au travail d'analyse. Nous verrons cependant au paragraphe 5.7.5 que ces outils ne représentent qu'un premier pas vers l'analyse automatique.

5.7.3. L'expérimentation Supratel

Supratel est un prototype de terminal de télécommunication multiservices développé par le CCETT [Charon 1993]. Son interface a été adaptée à l'environnement NEIMO pour l'étude de son utilisabilité. La figure 5.7 montre une copie d'écran de Supratel.

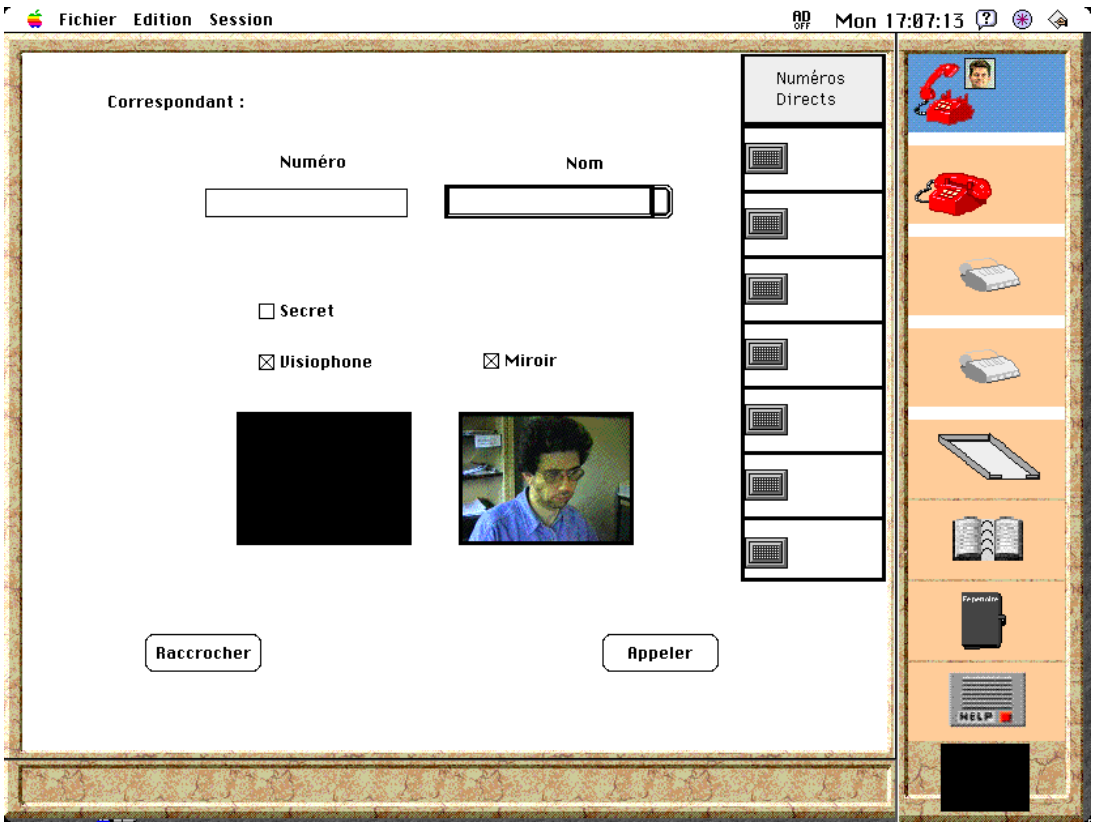


Figure 5.7. Une copie d'écran de l'interface du système Supratel. Le visiophone est en service.

Supratel offre un ensemble de services de communication homme-homme médiatisée et des services plus classiques mono-utilisateur. Ces différents services sont accessibles depuis la rangée d'icônes située à droite de l'écran sur la figure 5.7. On note par exemple des possibilités de communication téléphonique et visiophonique ainsi que fax (le système est destiné à être utilisé avec le réseau RNIS[1]), et des outils classiques comme l'agenda ou le répertoire.

Une des particularités de cette application est le grand éventail de tâches qui s'offrent à l'utilisateur et qui peuvent être utilisées simultanément. L'utilisateur est quasiment systématiquement dans un cas de dialogue à fils multiples. Pour l'expérimentation NEIMO, un ou deux observateurs suivent le comportement du sujet, et un compère simule les correspondants du sujet. Il peut par exemple entrer en communication visiophonique avec le sujet ou simuler la réception d'un fax sur la station du sujet. Les scénarios sont constitués d'un ensemble de tâches simples mais qui exploitent toutes les possibilités de l'application. Par exemple, le sujet doit vérifier sa disponibilité dans

[1] Réseau Numérique à Intégration de Services, traduction de *ISDN*. Réseau téléphonique numérique à débit moyen (64 Kb/s).

l'agenda, appeler un hôtel pour effectuer une réservation et envoyer simultanément un fax de confirmation. Pendant ces opérations, un appel extérieur survient et l'utilisateur doit vérifier un numéro dans son répertoire.

Lors d'une session d'expérimentation, le comportement de l'utilisateur et des compères est capturé à haut niveau d'abstraction, en général au niveau des commandes effectuées. L'analyse de la session permet d'étudier la satisfaction de certaines propriétés et la pertinence vis-à-vis des tâches des propriétés que vérifie l'interface. L'analyse nous permet d'étudier par exemple la vérification des propriétés CARE appliquées aux systèmes de communication, ou l'utilisation de la propriété vidéo miroir, vérifiée par Supratel. Notons que Supratel ne satisfait pas la propriété de réversibilité vidéo. Ce choix est délibéré : les tâches typiques réalisées avec Supratel dans les scénarios sont des tâches de communication "pures" ; le système ne permet pas de tâches de production multi-utilisateurs. L'observabilité publiée peut aussi être étudiée : Supratel comporte une fonction "secret" qui permet de suspendre momentanément la transmission de l'image de l'utilisateur. Enfin, la propriété de contrôle du rythme de l'interaction peut aussi être examinée : il est par exemple possible de mettre les appels en attente.

L'expérimentation Supratel a permis d'appliquer NEIMO à l'étude de la communication homme-homme médiatisée. Cette expérience a permis de vérifier des propriétés sur l'interface de Supratel et aussi la façon dont les sujets tirent parti de ces propriétés. Notons que certaines propriétés n'ont pu être satisfaites, principalement à cause de limitations techniques. A cause de l'environnement réseau utilisé pour les expériences, la régularité des flots par exemple n'est pas satisfaite.

5.7.4. NEIMO comme système multi-utilisateur

NEIMO constitue un système multi-utilisateur original : il comporte une variété de rôles et est à la fois synchrone et asynchrone. Nous avons vu qu'une session d'expérimentation implique plusieurs rôles : sujets, observateurs, compères et super-compère. Il faut y rajouter le rôle d'analyste rempli par l'ergonome qui effectue l'analyse a posteriori. Pendant une session, NEIMO est un système multi-utilisateur synchrone qui permet diverses tâches : la tâche globale est une tâche de production. Il s'agit de réaliser une session d'expérimentation dont le résultat est un fichier de capture. Mais NEIMO fait aussi intervenir des tâches de communication et de coordination entre le sujet et les compères et les observateurs. Si l'on considère l'aspect analyse a posteriori, NEIMO est aussi un système multi-utilisateur asynchrone : à partir du fichier élaboré pendant la session, l'ergonome analyste complète les observations et en tire une synthèse.

5.7.5. Leçons et perspectives

NEIMO nous a permis de réfléchir à la construction des systèmes multi-utilisateurs. Nous présentons au chapitre 7 l'architecture logicielle du système. Par sa complexité et la diversité des rôles qu'il prend en compte, il nous a aussi montré la difficulté de l'analyse des tâches multi-utilisateurs. La notation UAN présentée dans ce chapitre a été utilisée, par exemple pour la réalisation des interfaces du sujet et des observateurs et compère de Supratel. Toutefois, nous avons considérée dans cette analyse les rôles indépendamment les uns des autres. En effet, il n'y a collaboration qu'avec le sujet. Le système ne permet pas de collaboration entre les observateurs par exemple. Et même la collaboration avec le sujet est en général limitée à l'observation, sauf dans le cas du compère. L'absence d'outils d'analyse de tâches permettant de mieux prendre en compte l'interaction multi-utilisateurs a certainement été un frein au développement de possibilités de collaboration plus complètes dans NEIMO.

Deux aspects de NEIMO ouvrent des perspectives intéressantes : l'adaptation de l'interface à étudier à la plate-forme d'expérimentation, et l'analyse a posteriori des résultats d'une session.

Nous avons vu au paragraphe 5.7.1.1 que l'adaptation de l'interface à étudier à l'environnement NEIMO nécessitait un développement spécifique. Cette contrainte empêche notamment l'étude de logiciels dont les sources ne sont pas accessibles. Pour surmonter cette limitation, une voie de recherche à poursuivre est la définition d'une interface standard permettant la connexion d'un logiciel à un environnement d'étude de l'utilisabilité. Comme les circuits numériques qui disposent de points de test normalisés, on peut envisager qu'un logiciel comporte des points de test de l'utilisabilité. Un logiciel pourrait être ainsi connecté à n'importe quelle plate-forme d'étude de l'utilisabilité conforme à ce standard.

En ce qui concerne l'analyse, deux aspects méritent une étude approfondie : l'automatisation partielle ou totale et la visualisation. [Balbo 1994] distingue trois niveaux d'automatisation de l'analyse de l'activité d'évaluation : l'automatisation de la capture des actions de l'utilisateur, de la détection des problèmes d'utilisabilité, et de la correction ou de l'explication de ces problèmes. Dans ce cadre d'analyse, NEIMO offre la capture automatique des actions de l'utilisateur lors de la session d'expérimentation. Nous nous dirigeons vers l'analyse automatique avec l'utilisation de techniques comme la détection MRP (*Multiple Repeating Patterns*) de répétition de schémas d'actions dans le comportement de l'utilisateur. Mais comme le note Balbo, il n'existe pas à l'heure actuelle de solution satisfaisante pour la correction ou l'explication automatique. La technique

EMA proposée par Balbo est un premier pas dans cette direction. Guidée par l'utilisation des propriétés, elle pourrait ouvrir des perspectives prometteuses. Pour la visualisation synthétique de l'ensemble des tâches d'un scénario réalisé pendant une session, nous avons utilisés les diagrammes de Gantt. Cependant nous nous heurtons ici à un problème classique de la visualisation d'un grand espace d'informations : le compromis entre la vue globale contextuelle et la vue détaillée. Même si notre interface autorise la parcourabilité de l'ensemble du diagramme (avec un facteur de zoom, visible en bas de la figure 5.6), cette solution impose à l'analyste de naviguer entre vue globale et vue détaillée. L'intérêt de techniques de visualisation plus synthétiques, comme le "perspective wall" [Mackinlay 1991], bien adapté à la visualisation d'informations linéaires, reste à explorer dans le cadre de NEIMO.

5.8. Synthèse

En conclusion, force nous est de constater l'insuffisance des techniques d'évaluation ergonomique pour les systèmes multi-utilisateurs. Nous confirmons ainsi les conclusions du workshop "Design and Evaluation of Groupware" qui s'est tenu lors de la conférence CHI'95. En ce qui concerne les techniques d'évaluation prédictive, nous avons vu qu'en l'absence de théories prenant en compte les systèmes multi-utilisateurs, elles ne sont pas directement utilisables. Nous avons proposé d'utiliser la notation UAN, alliée à nos propriétés pour la détection de problèmes d'utilisabilité. Mais là encore, l'inadaptation de la notation au cas multi-utilisateur, en particulier pour la prise en compte des tâches de communication et de coordination, est un obstacle. Face à ce constat, nous nous sommes tournées vers des techniques relevant de l'approche artisanale, en particulier l'évaluation expérimentale. Notre observation de l'activité des ergonomes nous a conduit à la réalisation de l'outil NEIMO, un premier pas vers un laboratoire numérique d'utilisabilité. NEIMO est une plate-forme d'observation du comportement de l'utilisateur et de simulation par la technique du Magicien d'Oz. Cet environnement intègre aussi un outil d'analyse des résultats d'une session d'observation. Cette approche expérimentale, guidée par les propriétés du chapitre précédent, a été appliquée à l'étude d'un outil de communication homme-homme médiatisée. Toutefois, l'étude de l'utilisabilité d'un système multi-utilisateur plus complexe se heurte à des difficultés encore non résolues.

Références

[Balbo 1994] S. Balbo. *Evaluation ergonomique des interfaces utilisateur: un pas vers l'automatisation.* Thèse de doctorat, Université Joseph Fourier, Grenoble I, 1994.

[Boersma 1994] P. Boersma. *Experimental Research into Usability and Organisational Impact of Workflow Software.* Master's Thesis, Université de Twente, Pays-Bas, 1994.

[Brothers 1990] L. Brothers, V. Sembugamoorthy et M. Muller. *ICICLE: Groupware for Code Inspection*, CSCW'90, ACM Conference on Computer Supported Cooperative Work, Los Angeles, California, USA, 1990. pp. 169-181.

[Charon 1993] J.-P. Charon, L. Tézier, M. Carli et S. Liska. *Le projet Supratel : étude de l'utilisabilité d'un terminal de communication multiservices*, DESS Génie Informatique, Laboratoire de Génie Informatique, Université Joseph Fourier Grenoble 1, Rapport de stage, 1993.

[Dahlbäck 1988] N. Dahlbäck et A. Jönsson. *Talking to a computer is not like talking to your best friend*, SCAI-88, Scandinavian Conference on Artificial Intelligence, 1988. pp. 53-68.

[Dahlbäck 1989] N. Dahlbäck et A. Jönsson. *Empirical studies of discourse representations for natural language interfaces*, Fourth conference of the european chapter of the ACL, 1989. pp. 291-298.

[Diaper 1989] D. Diaper. *The Wizard's Apprentice: A Program to Help Analyse Natural Language Dialogues*, 5th Conference of the British Computer Society HCI SIG, 1989.

[Dix 1993] A. Dix, J. Finlay, G. Abowd et R. Beale. *Human-Computer Interaction*, Prentice Hall, New York, New York, 1993.

[Dowell 1989] J. Dowell et J. Long. *Towards a conception for an engineering discipline of human factors*, in *Ergonomics*, 32(11), pp. 1513-1535.

[Gould 1987] J. D. Gould, S. J. Boies, S. Levy, J. T. Richards et J. Schoonard. *The 1984 Olympic message system: a test of behavioral principles of system design*, in *Communications of the ACM*, 30(9),

[Grudin 1989] J. Grudin. *Why groupware applications fail: problems in design and evaluation*, in *Office: Technology and People*. Elsevier, 1989. pp. 245.

[Hammontree 1992] M. L. Hammontree, J. J. Hendrickson et B. W. Hensley. *Integrated data capture and analysis tools for research and testing on graphical user interfaces*, CHI'92, ACM Conference on Human Factors in Computing Systems, Monterey, California, USA, 1992. pp. 431-432.

[Hix 1993] D. Hix et H. R. Hartson. *Developing User Interfaces, Ensuring Usability Through Product & Process*, John Wiley & Sons, New York, New York, 1993.

[Jambon 1994] F. Jambon et L. Karsenty. *Formalisation des interfaces et travail coopératif : quelles conséquences ?*, IHM 94, Sixièmes Journées sur l'Ingénierie des Interfaces Homme-Machine, Lille, France, 1994. pp. 163-168.

[Lewis 1991] C. Lewis, P. Polson, C. Wharton et J. Rieman. *Testing a Walkthrough Methodology for Theory-Based Design of Walk-Up-and-Use Interfaces*, CHI'91, ACM Conference on Human Factors in Computing Systems, New Orleans, Louisiana, USA, 1991. pp. 235-242.

[Lischetti 1994] N. Lischetti. *SupraAnalyse : un outil d'aide à l'analyse de l'utilisabilité. Application à un terminal de télécommunication multiservices*. Mémoire CNAM, Laboratoire de Génie Informatique, Université Joseph Fourier Grenoble 1, 1994.

[Long 1989] J. Long et J. Dowell. *Conceptions of the Discipline of HCI: Craft, Applied Science, and Engineering*, Fifth Conference of the British Computer Society HCI SIG, 1989.

[Mackinlay 1991] J. D. Mackinlay, G. G. Robertson et S. K. Card. *The Perspective Wall: Detail and Context Smoothly Integrated*, CHI'91, ACM Conference on Human Factors in Computing Systems, New Orleans, Louisiana, USA, 1991. pp. 173-179.

[MacLeod 1993] M. MacLeod. *DRUM, Diagnostic Recorder for Usability Measurement*, NPL, DITC HCI Group, Teddington, UK, 1993.

[Maulsby 1993] D. Maulsby, S. Greenberg et R. Mander. *Prototyping an Intelligent Agent through Wizard of Oz*, InterCHI'93, ACM/IFIP Conference on Human Factors in Computing Systems, Amsterdam, Pays-Bas, 1993. pp. 277-284.

[Mignot 1993] C. Mignot, C. Valot et N. Carbonell. *An Experimental Study of Future 'Natural' Multimodal Human-Computer Interaction*, InterCHI'93, ACM/IFIP Conference on Human Factors in Computing Systems, Amsterdam, Pays-Bas, 1993. pp. 67-68.

[Nielsen 1994] J. Nielsen. *Usability Engineering*, Academic Press, Boston, Massachussets, USA, 1994.

[Nigay 1994] L. Nigay. *Conception et réalisation des systèmes interactifs: Application aux Interfaces Multimodales.* Thèse de doctorat, Université Joseph Fourier, Grenoble I, 1994.

[Polity 1990] Y. Polity, J.-M. Francony, R. Palermiti, P. Falzon et S. Kazma. *Recueil de dialogues homme-machine en langue naturelle écrite*, Criss, Les cahiers du Criss n°17, 1990.

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

[Scapin 1990] D. L. Scapin. *Des critères ergonomiques pour l'évaluation et la conception d'interfaces utilisateur*, XXVIè Congrès de la SELF, Montréal, Canada, 1990.

[Senach 1990] B. Senach. *Evaluation ergonomique des interfaces homme-machine : une revue de la littérature*, INRIA, Programme 8, Communication Homme-Machine, n° 1180, 1990.

[Shackel 1991] B. Shackel et S. Richardson. *Human Factors for Informatics Usability*, Cambridge University Press, 1991.

[Smith 1986] S. L. Smith et J. N. Mosier. *A design evaluation checklist for user-system interface software*, The MITRE Corporation, Bedford, Massachussets, USA, #MTR-9480 EDS_TR_84-358, 1986.

[Tognazzini 1991] B. Tognazzini. *On High-Altitude Computing*, in *Apple Directions*, 1991.

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

[Watabe 1990] K. Watabe, S. Sakata, K. Maeno, H. Fukuoka et T. Ohmori. *Distributed Multiparty Desktop Conferencing System: MERMAID*, CSCW'90, ACM Conference on Computer Supported Cooperative Work, Los Angeles, California, USA, 1990. pp. 27-38.

Troisième Partie

Mise en œuvre : science et technologie informatiques

Modèles d'architecture

Si j'avais appris la technique, je serais technicien. Je fabriquerais des objets compliqués. Des objets très compliqués, de plus en plus compliqués, cela simplifierait l'existence.

Eugène Ionesco

Modèles d'architecture

6.1. Introduction

Les disciplines dont nous avons décrits les apports dans les chapitres de la première partie contribuent à l'analyse des besoins et à l'établissement des spécifications du système à réaliser. Lorsque vient le moment de construire effectivement le système à partir des spécifications, d'autres guides centrés sur le génie logiciel deviennent nécessaires. La complexité croissante des systèmes informatiques et la diversité des techniques logicielles ont mis en évidence depuis quelques années l'utilité de l'architecture logicielle pour la conception et la réalisation des systèmes. Pour répondre à l'aphorisme de Ionesco, il est vrai que nous créons aujourd'hui des objets informatiques de plus en plus compliqués afin de simplifier l'existence des utilisateurs. Mais nous visons, principalement grâce aux modèles d'architecture logicielle et aux outils, à simplifier aussi l'existence des concepteurs et réalisateurs de ces objets informatiques complexes.

Dans ce chapitre, nous nous intéressons d'abord aux modèles d'architecture logicielle en général et identifions les rôles qu'ils doivent remplir. Nous précisons ensuite les dimensions de l'espace problème des systèmes multi-utilisateurs en termes d'architecture logicielle. Quelles particularités et quels concepts de ces systèmes doivent être pris en compte par un modèle d'architecture logicielle ? A la lumière de cette analyse, nous présentons les modèles d'architecture les plus significatifs pour la conception logicielle des systèmes multi-utilisateurs. Au chapitre suivant, nous présentons notre propre contribution.

6.2. Architecture logicielle : définition

L'architecture logicielle est un domaine récent et il n'existe pas de vrai consensus sur une définition. A défaut, plusieurs auteurs sont cependant d'accord sur le rôle attendu d'une architecture logicielle (voir par exemple [Abowd 1994], [Bass 1994], ou [Garlan 1993]). Une architecture logicielle doit communiquer quatre perspectives sur le système et son organisation : fonctionnelle, structurelle, allocation et coordination.

- La *perspective fonctionnelle* décrit les services du système comme une collection de services élémentaires. Le terme "service élémentaire" est à prendre avec précaution car la granularité d'un service élémentaire dépend du domaine d'application, de sa maturité, mais aussi du contexte de développement. Un domaine mature admet une partition canonique en services. Abowd cite comme domaine d'application mature l'exemple des compilateurs [Abowd 1994]. L'ensemble des services d'un compilateur classique est maintenant bien connu :

analyseur lexical, analyseur syntaxique, analyseur sémantique, générateur de code et service d'optimisation. Cependant, pour un domaine d'application moins bien cerné comme les compilateurs pour les systèmes massivement parallèles, on pourra préciser cette décomposition fonctionnelle et affiner par exemple la phase de génération de code en une phase d'analyse des dépendances et d'allocation des processeurs suivie de la génération de code pour chacun des processeurs.

- La *perspective structurelle* décrit de façon statique l'organisation des services. Traditionnellement représentée sous forme de diagrammes, elle fait intervenir des composants, des connecteurs et des configurations. Un *composant* est un élément qui réalise une activité de calcul. Un objet, un agent ou un processus sont des exemples de composants. Un composant a la faculté de communiquer avec l'extérieur c'est-à-dire avec d'autres composants ou avec des ressources matérielles. Cette communication s'effectue par l'intermédiaire de *connecteurs* qui relient plusieurs composants et définissent un protocole entre ces composants. Des connecteurs classiques sont par exemple l'appel de procédure, l'envoi de message ou un protocole réseau comme TCP/IP. Notons que l'ensemble des connecteurs possibles est bien identifié, au contraire de l'ensemble des composants qui dépend fortement du domaine. Enfin une *configuration*, comme celle de la figure 6.1, rassemble les instances de composants et de connecteurs constituant un système particulier à un instant donné. En général, un système est décrit à l'aide de plusieurs configurations complémentaires. Ces différentes configurations décrivent soit différents ensembles indépendants de composants et de connecteurs, soit l'évolution d'un ensemble de composants et de connecteurs au cours du temps.

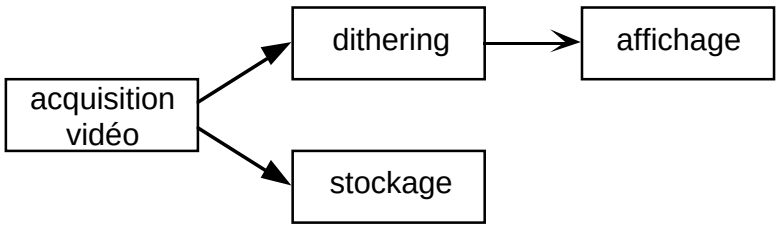


Figure 6.1. Un exemple de configuration pour un module de capture et d'affichage d'images vidéo. Les rectangles représentent des composants identifiés par le nom du service qu'ils réalisent. Les flèches représentent des connecteurs. Le style d'architecture (voir plus loin) est du type flot de données et les connecteurs assurent une simple fonction "copie".

- La *perspective allocation* définit l'allocation des services aux constituants de la structure. Autrement dit, la perspective allocation décrit le lien entre les deux perspectives précédentes, c'est-à-dire de quelle façon la structure véhicule les

fonctions identifiées. C'est principalement à cette étape que sont faits des choix dont l'influence sur la qualité du logiciel sera déterminante. Une stratégie d'allocation adaptée permet en particulier de satisfaire les propriétés de modifiabilité ou de réutilisabilité.

- La *perspective coordination* traduit les aspects dynamiques du système. Deux caractéristiques importantes du fonctionnement du système sont décrites par cette perspective : les protocoles utilisés par les connecteurs et la création dynamique de composants et de connecteurs. Il est important de constater que la dimension coordination est orthogonale à la dimension fonctionnelle. La dimension fonctionnelle prend en compte les services rendus par les composants alors que la dimension coordination s'intéresse au comportement dynamique de l'ensemble des composants et connecteurs.

Notons que cette dernière perspective n'est pas toujours mentionnée (voir par exemple [Kazman 1994]). Pourtant, dans le domaine des interfaces homme-machine il est indispensable de considérer la dimension coordination. La création dynamique de nouveaux éléments d'interaction est souvent occultée dans les architectures pour les interfaces : la responsabilité de la création de nouveaux éléments ou le moment auquel ils doivent être créés ne sont pas toujours clairs. Pour les systèmes multi-utilisateurs, l'analyse fonctionnelle fait apparaître une dimension purement dynamique (la dimension coordination du trèfle du chapitre 1). D'autre part, la communication et donc les connecteurs sont un fondement indispensable d'un système multi-utilisateur. Une architecture logicielle adaptée doit tenir compte de ces deux aspects et pour nous cette perspective coordination est donc particulièrement importante.

Pour être complète, une architecture logicielle doit être accompagnée d'une légende : le *style d'architecture*. Le style est constitué d'un vocabulaire et de règles d'organisation. Le style définit comment doivent être lus les diagrammes et conduit à les interpréter avec un modèle usuel d'organisation de composants : flux de données (*data flow*), client-serveur, tableau noir, etc. Mentionnons que sans le style correspondant, une architecture est ambiguë : suivant le style choisi, elle pourra être interprétée différemment et elle ouvre la voie à des implémentations aux propriétés bien différentes [Shaw 1995]. Pourtant, le style est parfois implicite dans une architecture. En effet, la nature des composants et des connecteurs permet souvent de lever les ambiguïtés. Cependant, sans le style qui sert de guide de lecture, une architecture est incomplète.

Une architecture logicielle répond à un double objectif : organiser le code et les composants du système à construire et documenter cette organisation. Un *modèle*

d'architecture logicielle vise à généraliser les architectures logicielles pour un domaine donné et propose une architecture canonique adaptée au domaine considéré. Au contraire d'une architecture pour un système donné, un modèle d'architecture présente l'avantage de codifier le savoir et donc d'être réutilisable. C'est aussi un objet scientifique : il est possible de raisonner sur ses propriétés et ses qualités. Pour concevoir l'architecture d'un système donné, le modèle d'architecture est instancié avec les composants et connecteurs particuliers au système considéré. Le modèle Arch [Bass 1992] ou le modèle PAC [Coutaz 1987] sont des exemples de modèles d'architecture pour les systèmes interactifs mono-utilisateurs.

6.3. Caractéristiques requises d'une architecture

Une architecture est profondément liée au contexte de développement. Les rôles attendus d'une architecture que nous venons de présenter doivent être modulés par le contexte particulier du système à développer ainsi que par l'expertise des acteurs du développement. Comme le soulignent [Bass 1994] et [Kazman 1994], il n'y a pas de "bonne" ou de "mauvaise" architecture dans l'absolu. Une architecture répond à des critères et à des besoins dans un contexte donné. Toutefois, dans le cadre des systèmes interactifs et des systèmes multi-utilisateurs, on peut identifier des caractéristiques générales qu'une architecture et donc un modèle d'architecture doivent satisfaire.

[Nigay 1994] identifie quatre règles qui traduisent les bénéfices attendus d'un modèle d'architecture logicielle pour les interfaces utilisateur. Ces règles sont aussi applicables au cas des systèmes multi-utilisateurs :

① Un modèle d'architecture logicielle implique une organisation spécifique du code de l'interface utilisateur. Cette organisation doit faciliter les modifications dues, pour l'essentiel, à la mise au point itérative des interfaces.

② Un modèle d'architecture logicielle doit réduire la complexité de réalisation d'une interface utilisateur en proposant des éléments de structuration.

③ Un modèle d'architecture logicielle doit faciliter la décomposition du travail.

④ Un modèle d'architecture doit tenir compte de l'existence des outils logiciels de développement d'interfaces utilisateur et doit englober la plate-forme physique d'accueil. En d'autres termes, un modèle d'architecture doit prendre en compte les contraintes logicielles et matérielles stipulées dans le cahier des charges.

La règle ① impose une contrainte importante due au domaine particulier des systèmes interactifs : elle met en exergue la modifiabilité du code développé en suivant le modèle d'architecture. Nous lui ajoutons volontiers une propriété liée : la réutilisabilité. La satisfaction de ces propriétés est garantie par une allocation adéquate des fonctions à la structure comme dit au paragraphe précédent. La règle ② est contenue en partie dans la perspective structurelle d'une architecture. Mais la réduction souhaitée de la complexité dépend de la pertinence de l'analyse fonctionnelle du domaine. La règle ③ découle des deux précédentes mais souligne un aspect important d'une architecture : en structurant l'ensemble du code à développer, elle permet d'allouer le travail de réalisation des composants à plusieurs individus ou à plusieurs équipes. Cette règle plaide pour une spécification précise des connecteurs afin de préparer l'intégration qui sera nécessaire à l'issue de la phase de réalisation. Nous avons progressé dans cette direction avec le modèle CoPAC présenté au chapitre suivant et avec la classification IMPACT du chapitre 8. Enfin la règle ④ dénote la validité dans le monde réel du modèle. Cette règle est particulièrement importante pour qu'un modèle soit opérationnel. C'est elle qui garantira qu'un modèle est véritablement utilisable, *in fine*, par les praticiens.

Un problème important est la vérification des caractéristiques annoncées par un modèle d'architecture. La méthode SAAM [Kazman 1994] propose une méthodologie permettant ces vérifications. Nous l'avons appliquée à notre modèle CoPAC, en particulier pour vérifier les règles ① et ④. Ces résultats sont présentés avec le modèle CoPAC au chapitre 7.

Pour les systèmes multi-utilisateurs, des facteurs supplémentaires sont à considérer. Premièrement, comme ces systèmes présentent une interface utilisateur à chacun des participants, il faut envisager la duplication de certains composants. Identifier clairement dans l'architecture les composants dupliqués permet de factoriser leur réalisation et réduit ainsi le coût de développement. Notons que cette éventuelle duplication de composants est indépendante du type d'architecture et n'est pas propre à l'architecture répliquée présentée plus loin. Nous proposons donc la règle suivante :

❶ Un modèle d'architecture pour les systèmes multi-utilisateurs doit permettre de factoriser la réalisation en identifiant les composants identiques.

Deuxièmement, étant donné la grande variété des systèmes multi-utilisateurs et la diversité des groupes auxquels ils s'adressent, un modèle d'architecture doit indiquer pour quel facteur d'échelle il est valide. Un modèle peut être adapté à un groupe restreint et se révéler inopérant à plus grande échelle. D'où la règle :

❷ Un modèle d'architecture doit mentionner ses limitations en termes de taille du groupe et indiquer à quelle échelle il est applicable.

Mais ces règles générales sur les modèles d'architecture ne sont pas d'une aide suffisante. Il nous faut maintenant expliciter les composantes de l'espace problème des architectures pour les systèmes multi-utilisateurs afin d'être à même d'évaluer les différents modèles proposés dans la littérature.

6.4. Espace problème

L'objectif de notre espace problème est d'identifier un ensemble de concepts pertinents pour les modèles d'architecture des systèmes multi-utilisateurs et de les organiser. Cet espace problème pourra ensuite servir de base structurante à notre réflexion sur les modèles d'architecture existants.

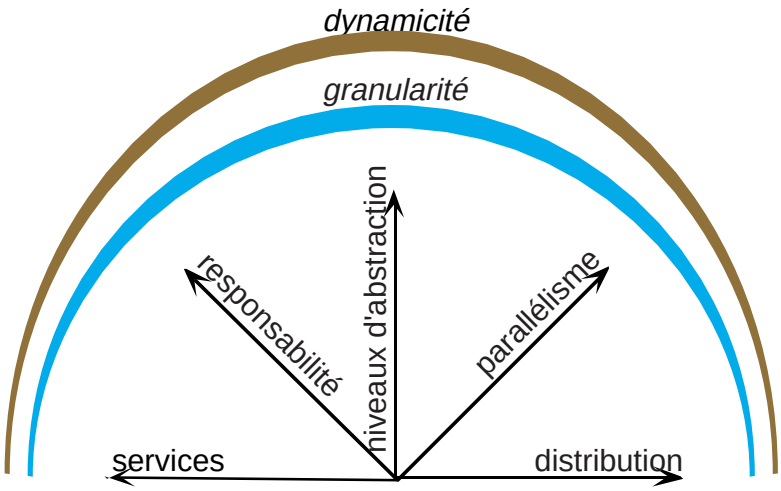


Figure 6.2. Espace problème des modèles d'architecture pour les systèmes multi-utilisateurs. Les arcs représentent des modificateurs qui s'appliquent à chacun des cinq axes.

Nous avons identifié cinq dimensions pour l'espace problème des modèles d'architecture des systèmes multi-utilisateurs : services, responsabilité, niveaux d'abstraction, parallélisme et distribution. A ces cinq axes sont accolés deux modificateurs : granularité et dynamicité. La combinaison d'un modificateur et d'un axe donné permet de moduler le champ d'application de l'axe considéré. La figure 6.2 présente notre espace problème.

6.4.1. Services

La première dimension représente les *services* qu'une architecture d'un système multi-utilisateur doit prendre en compte. Cet axe correspond à la perspective fonctionnelle de

l'architecture. Pour préciser la dimension des services, nous empruntons à Ellis une classification fonctionnelle des services des systèmes multi-utilisateurs. [Ellis 1994] distingue quatre classes de services : les dépositaires, les synchronisateurs, les communicateurs et les agents actifs ("keepers, synchronizers, communicators and agents"). Nous ne conservons que les trois premières catégories, la quatrième étant une combinaison des trois précédentes modulée par la dimension responsabilité de notre espace problème. Nous y reviendrons lorsque nous détaillerons cette dimension.

Les *dépositaires* sont des composants d'une architecture dont le rôle est de maintenir des données. Des exemples typiques de dépositaires sont les systèmes de gestion de fichiers ou les bases de données. Rapprochés du modèle du trèfle du chapitre 1, les dépositaires implémentent ce qui a trait à l'espace de production. On peut aussi faire l'analogie entre les dépositaires et le noyau fonctionnel des systèmes interactifs. Pour spécifier complètement un dépositaire, il convient de s'interroger sur la structuration des données maintenues, du partage de ces données et du contrôle d'accès. La granularité des données maintenues par un dépositaire est aussi un aspect à considérer.

Les *synchronisateurs* assurent la synchronisation des tâches du groupe. Un synchronisateur peut être par exemple la procédure régissant un système de type workflow, ou un composant gérant le contrôle d'accès à un dépositaire. Vis-à-vis du modèle du trèfle, les synchronisateurs implémentent l'espace de coordination. On peut aussi faire le parallèle entre les synchronisateurs et le contrôleur de dialogue des systèmes interactifs. Les synchronisateurs doivent être examinés en lien étroit avec les dimensions responsabilité et parallélisme de notre espace problème.

Les *communicateurs* ont la charge de la communication entre les utilisateurs. Les systèmes de courrier électronique ou les mediaspaces reposent essentiellement sur des communicateurs. Un communicateur peut aussi être vu comme un connecteur particulier qui ne modifie pas le contenu du message qu'il fait transiter : il assure une simple fonction "copie". Les communicateurs implémentent l'espace de communication du modèle du trèfle. Le tableau de la figure 6.3 résume les liens entre ces trois classes de services, le modèle fonctionnel du trèfle et les systèmes interactifs mono-utilisateurs.

Les dépositaires, synchronisateurs et communicateurs catégorisent les services utiles pour un système multi-utilisateurs. Pour être complet, il faut toutefois rajouter les services de l'interface utilisateur. Nous sommes donc amenés à ajouter la classe des *éléments d'interaction* qui est constituée par les composants avec lesquels l'utilisateur interagit. Les services de cette classe sont analogues à ceux de la composante interaction ou présentation des systèmes mono-utilisateurs. A ces services, il faut toutefois ajouter de nouveaux

mécanismes pour garantir des propriétés spécifiques : par exemple un télépointeur ou des barres de défilement (*scrollbars*) multi-utilisateurs.

Classes de services	Espace du trèfle	Systèmes interactifs
Dépositaires	Production	Noyau fonctionnel
Synchronisateurs	Coordination	Contrôleur de dialogue
Communicateurs	Communication	(Pas d'équivalent)

Figure 6.3. Équivalence entre les classes de services des modèles d'architecture, les espaces du modèle fonctionnel du trèfle et les systèmes interactifs mono-utilisateurs.

6.4.2. Responsabilité

La deuxième dimension de notre espace problème définit la *responsabilité* de la prise en charge d'un service donné. Un service peut être réalisé par un des deux types d'acteurs : acteur humain ou acteur logiciel. Dans son acception pour les systèmes mono-utilisateurs, cette distinction traduit le fait qu'une tâche peut être déléguée par l'utilisateur au système. Nous avons trouvé l'exemple suivant dans le Finder du Macintosh [Apple 1994] : si l'on veut copier un élément vers un disque et que la place nécessaire ne peut être obtenue qu'en vidant la corbeille, le système propose de le faire pour l'utilisateur (figure 6.4). Si l'utilisateur répond "OK", il délègue au système la tâche de vider la corbeille. On trouve déjà cette délégation dans Eager lorsque Eager est certain qu'il à détecté des tâches répétitives de l'utilisateur [Cypher 1991].

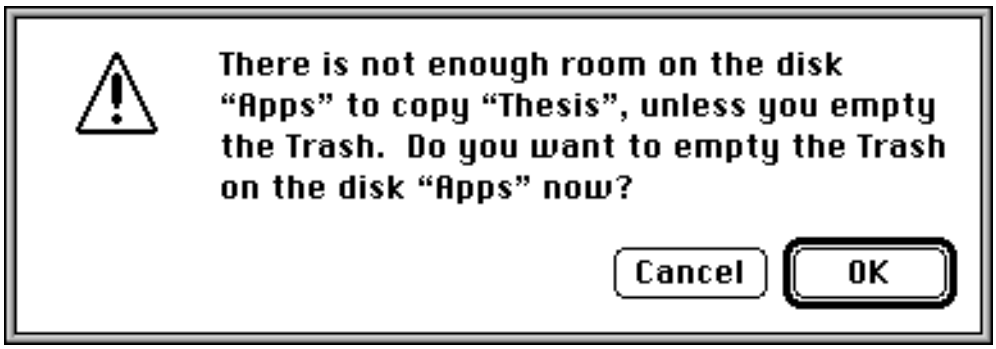


Figure 6.4. Une boîte de dialogue du Finder du Macintosh. Le système demande à l'utilisateur de confirmer qu'il lui délègue la tâche de vider la corbeille.

En fait, la délégation est un phénomène général qui peut se produire entre différentes entités : entre l'utilisateur et le système, entre composants du système et entre différents utilisateurs.

La délégation entre l'utilisateur et le système peut prendre plusieurs formes. A un extrême, l'utilisateur peut se voir déléguer une tâche par le système (par exemple, "insérez la disquette n° 2"). Un exemple commun de délégation est fourni par Eudora, logiciel de courrier électronique [Dorner 1995] : on peut modifier les réglages pour déléguer à Eudora la tâche de relever la boîte-aux-lettres à intervalles réguliers (figure 6.5).

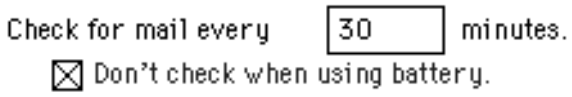


Figure 6.5. Dans Eudora, l'utilisateur délègue au système la tâche de relever la boîte-aux-lettres toutes les 30 minutes. La délégation peut être faite de façon conditionnelle suivant la valeur d'une variable du contexte (l'ordinateur fonctionne sur batterie ou sur le secteur électrique).

A l'autre extrême, les agents actifs (ou agents "intelligents"), informés par l'utilisateur, exécutent pour lui une tâche parfois complexe. Knowledge Navigator [Apple 1988] laisse entrevoir les possibilités de ce type d'outils. Les agents actifs d'Ellis relèvent de ce type de délégation. Les trieurs de courrier électronique tels MAXIMS [Metral 1994] ou Information Lens [Malone 1986] font aussi partie de cette catégorie.

Le phénomène de délégation se produit aussi entre les composants logiciels constituant le système. La délégation sémantique définie par [Coutaz 1990] est un cas particulier dans lequel une partie du rôle du noyau fonctionnel est déléguée au contrôleur de dialogue. Cet enrichissement du contrôleur de dialogue en connaissances sémantiques permet de réutiliser un noyau fonctionnel sans le modifier. La délégation sémantique remplit aussi un autre rôle : la réparation sémantique, c'est-à-dire l'introduction dans les composants dialogue et présentation de concepts inconnus du noyau fonctionnel. Appliquée aux composants logiciels, la notion de délégation est un élément important pour définir la perspective allocation d'un modèle d'architecture.

Pour les systèmes multi-utilisateurs, la délégation existe aussi entre utilisateurs. Cette *délégation sociale* nécessite un support système plus ou moins spécifique. Les mécanismes requis peuvent aller d'un mécanisme général comme le changement des droits d'accès à un fichier jusqu'à des mécanismes plus ad hoc comme la possibilité de se mettre à la place d'un autre utilisateur, par exemple dans les systèmes de réalité virtuelle. Il faut aussi distinguer la délégation inter- et intra-rôles qui peuvent demander des mécanismes différents. Enfin, la dimension responsabilité permet de prendre en compte les différents rôles que doit implémenter le système.

6.4.3. Niveaux d'abstraction

Les services se situent à différents *niveaux d'abstraction*, depuis les primitives du système d'exploitation jusqu'aux mécanismes du noyau fonctionnel. Cette dimension rappelle la règle de validité du modèle d'architecture vis-à-vis du monde réel, c'est-à-dire la plate-forme d'accueil. Tout comme le modèle Arch [Bass 1992] pour les systèmes mono-utilisateurs prend en compte explicitement les boîtes à outils graphiques, un modèle pour les systèmes multi-utilisateurs doit tenir compte des possibilités offertes par l'environnement. Typiquement, un système d'exploitation à objets distribués comme GUIDE [Krakowiak 1990] offre des services de plus haut niveau qu'un système classique comme Unix. En relation avec l'axe responsabilité et modulé par le modificateur dynamicité, la dimension niveaux d'abstraction permet d'exprimer pour un service, la capacité de migration dynamique entre les différents niveaux d'abstraction du système.

6.4.4. Parallélisme

Déjà importante dans les systèmes mono-utilisateurs, en particulier multimodaux [Coutaz 1993], la dimension *parallélisme* prend une nouvelle importance pour les systèmes multi-utilisateurs. Elle permet de décrire les relations temporelles non seulement pour un utilisateur donné, mais pour l'ensemble des utilisateurs. Ces relations temporelles, en relation avec l'axe niveaux d'abstraction, peuvent porter sur les groupes de tâches, les tâches ou les actions des utilisateurs (suivant le modificateur granularité). Cette dimension englobe aussi la traditionnelle distinction entre systèmes multi-utilisateurs synchrones et asynchrones.

6.4.5. Distribution

La dimension *distribution* est à rapprocher de la perspective structurelle d'une architecture. Elle exprime la localisation physique des composants logiciels sur l'ensemble des sites[1] constituant le système multi-utilisateur. Combinée au modificateur granularité, cette dimension demande de considérer quelles entités seront distribuées : objet ou même partie d'objet pour un système d'exploitation à objets, composants logiciels ou combinaisons de composants. Combinée au modificateur dynamicité, elle nous permet de prendre en compte les déplacements d'objets d'un site vers un autre qui se produisent dans les systèmes à objets. Cette dimension permet aussi de réfléchir à la réplication des composants et répond donc au besoin de mise en évidence de la factorisation que nous avons introduite au paragraphe 6.3. A un plus haut niveau de granularité, la dimension distribution permet de prendre en compte le nombre de sites

[1] Nous utilisons ici "site" dans le sens "unité de traitement informatique indépendante". En général, un site est une station de travail.

participant au système multi-utilisateur. Du point de vue de l'architecture, il est souvent plus pertinent de considérer le nombre de sites plutôt que le nombre d'utilisateurs, mais ces deux caractéristiques sont bien sûr liées. On peut toutefois trouver des cas où plusieurs utilisateurs partagent le même site.

Les cinq axes de notre espace-problème, services, responsabilité, niveaux d'abstraction, parallélisme et distribution, modulés par les modificateurs granularité et dynamicité, nous donnent un cadre de réflexion pour les modèles d'architecture pour les systèmes multi-utilisateurs. Cet espace-problème dont nous avons explicitées les dimensions va nous permettre d'évaluer les modèles usuels proposés pour les systèmes multi-utilisateurs.

6.5. Modèles d'architecture usuels pour les systèmes multi-utilisateurs

Après avoir identifié ce que doit communiquer une architecture et les concepts pertinents pour les systèmes multi-utilisateurs, nous passons maintenant en revue dans cette double perspective quelques modèles usuels proposés dans la littérature. Nous commençons par le triple modèle classique centralisé/répliqué/hybride. Puis nous présentons trois modèles pour les systèmes synchrones qui privilégient l'information partagée (l'espace de production) comme les éditeurs partagés. Ces trois modèles, ALV, le modèle à états partagés et les User Display Agents, ont des couvertures fonctionnelles différentes. Nous détaillons ensuite deux modèles plus adaptés à des systèmes privilégiant la communication.

6.5.1. Le triple modèle classique : centralisé/répliqué/hybride

Le modèle triple centralisé/répliqué/hybride fait quasiment office de modèle de référence dans la littérature sur les architectures des systèmes multi-utilisateurs. Il identifie trois cas présentés figures 6.6, 6.7 et 6.8.

Dans le cas centralisé, un processus unique s'exécute sur un site serveur et gère l'ensemble de l'application multi-utilisateur, c'est-à-dire le noyau fonctionnel et l'ensemble des interfaces des utilisateurs. Ce modèle est bien adapté à un système graphique client-serveur comme X-Window. La figure 6.6 présente une extension de ce modèle dans laquelle différentes interfaces correspondant à différents rôles sont pris en compte.

Le *modèle centralisé* présente l'avantage de la simplicité : comme l'application est réduite à un seul processus, les questions de communication et de synchronisation entre

processus ne se posent pas. Or elles constituent une difficulté de la conception logicielle des systèmes multi-utilisateurs. Mais cette simplicité se paie en termes de fiabilité et de flexibilité de l'architecture. Le problème de fiabilité est connu : comme toute l'architecture repose sur un seul processus, sa défaillance ou son manque de performance ont des conséquences gênantes pour l'ensemble des utilisateurs. D'autre part, un processus central ne permet pas de garantir pour tous les utilisateurs un temps de réponse acceptable ni même constant. Les propriétés de stabilité et de conformité du temps de réponse sont compromises. Cette architecture présente aussi l'inconvénient de ne pas prendre en compte tous les types de systèmes. Un système qui requiert un parallélisme important tel MMM (Multi-device, Multi-user, Multi-editor) [Bier 1991] est limité par une architecture centralisée. Notons que cette limitation sur le parallélisme peut être levée en utilisant des processus légers ou threads comme en proposent de nombreux systèmes d'exploitation modernes. Cependant, les problèmes de fiabilité et de vivacité de l'interface persistent. Notons aussi que la grosse granularité de l'architecture centralisée, qui raisonne au niveau du processus, limite son intérêt pour un système comme MMM qui requiert un grain plus fin, de l'ordre de l'objet, voire de la méthode d'objet.

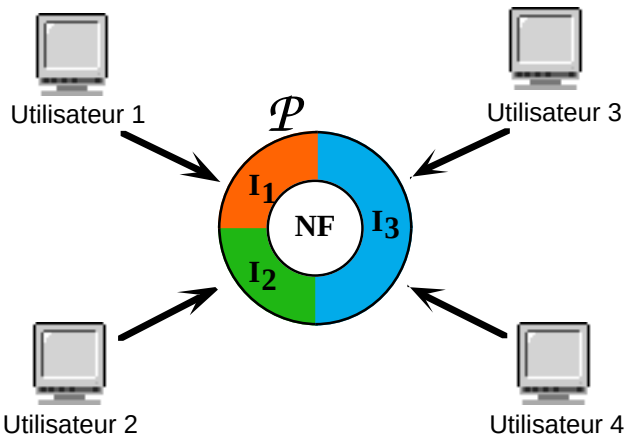


Figure 6.6. Le modèle centralisé. Exemple de configuration avec quatre utilisateurs, quatre terminaux et un serveur. Les flèches signifient "est client de". $\mathcal{P}$ est l'unique processus du système. Il gère le noyau fonctionnel **NF** et une ou plusieurs interfaces utilisateur { $\mathbf{I_i}$ }. Une interface I_i présente et permet d'utiliser les services correspondant au rôle i. Sur la figure, les utilisateurs 3 et 4 ont le même rôle. Dans le modèle centralisé originel, i = 1.

En résumé, vis-à-vis de notre grille d'analyse, l'architecture centralisée privilégie les dépositaires au détriment des autres classes de services. Remarquons aussi que des systèmes qui sont apparemment des systèmes de communication, comme les messageries multi-utilisateurs des serveurs Minitel ou les MUDs (Multi-User Dimensions) adoptent une architecture centralisée. En fait, du point de vue de l'architecture, ces systèmes sont bien des systèmes orientés vers la production : les MUDs par exemple, permettent une communication de chaque utilisateur vers l'ensemble des utilisateurs sous forme d'un

espace texte partagé. Ils sont en fait plus proches des éditeurs partagés. Lorsque des communications entre utilisateurs un à un sont possibles, l'architecture assigne un processus à chaque utilisateur et l'on retrouve le modèle répliqué que nous présentons maintenant.

Le *modèle répliqué* est une perspective radicalement opposée. Cette fois, un processus est alloué à chaque utilisateur et les éventuelles données partagées sont dupliquées sur tous les sites. La figure 6.7 présente une généralisation du modèle répliqué qui prend en compte différents rôles.

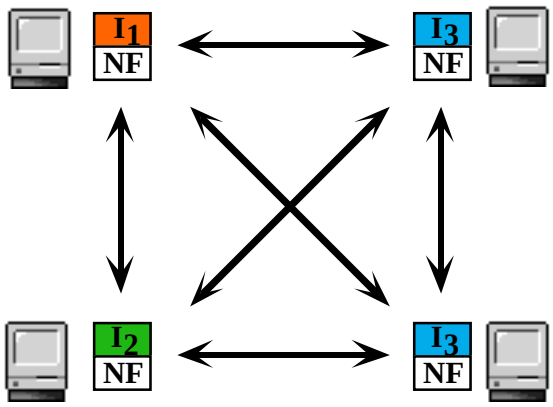


Figure 6.7. Le modèle répliqué. Exemple de configuration à quatre utilisateurs. Un processus par utilisateur gère une copie du noyau fonctionnel **NF** répliqué et l'interface locale $\mathbf{I_i}$ adaptée au rôle de l'utilisateur. Chaque processus peut communiquer directement avec n'importe quel autre. Les connecteurs représentés par des flèches sont un protocole réseau de communication inter-applications. Ils relient deux à deux les noyaux fonctionnels et les interfaces.

Ce modèle semble a priori mieux adapté aux systèmes multi-utilisateurs : la duplication des composants de l'architecture permet d'améliorer la vivacité de l'interface. La duplication permet aussi une fiabilité accrue : en cas de défaillance de l'un des processus, les autres processus continuent leur activité. Au cas où le processus défaillant doit communiquer avec d'autres processus, des techniques classiques (communication asynchrone, délai de garde—ou "timeout") permettent de détecter la défaillance et de mettre "hors circuit" le processus fautif. Cependant, l'architecture répliquée, si elle est bien adaptée aux services de la classe communicateurs, présente l'inconvénient de dupliquer les dépositaires. Les données étant répliquées sur tous les sites, il devient indispensable de mettre en place des mécanismes de maintien de la cohérence des données répliquées. Cette approche présente les inconvénients suivants.

- Les communications n'étant pas instantanées, des phénomènes de "zone grise" peuvent se produire : la modification d'une donnée sur un site ne sera répercutée sur les autres sites qu'après un intervalle de temps Δt dépendant du réseau de

communication. Un réseau de communication (tel Ethernet) étant non-déterministe, Δt est en général imprévisible. Des conflits peuvent donc apparaître si plusieurs utilisateurs modifient la même donnée simultanément. Pour résoudre ce problème, des protocoles de communication garantissant l'ordonnancement global des messages, tels les protocoles multicast sont envisageables. L'inconvénient de cette technique est qu'un protocole de ce type a un temps de réponse qui croît exponentiellement en fonction du nombre de sites. Cette solution n'est donc pas envisageable pour un grand nombre de sites. D'autres protocoles, comme ceux fondés sur des algorithmes optimistes [Karsenty 1994], sont toutefois des solutions moins coûteuses et prometteuses.

- En cas de défaillance d'un processus ou du réseau, l'intégrité des données locales au processus fautif (ou de l'ensemble des données dans le cas du réseau) est compromise. Il est indispensable d'envisager des mécanismes robustes de reprise après panne. Ces mécanismes peuvent être analogues aux mécanismes de fusion de versions de documents et nécessitent souvent dans ce cas l'intervention des utilisateurs pour résoudre les conflits. D'autres solutions reposant sur l'utilisation d'un historique des modifications sont aussi utilisées, mais requièrent le maintien d'un historique local à chaque processus.

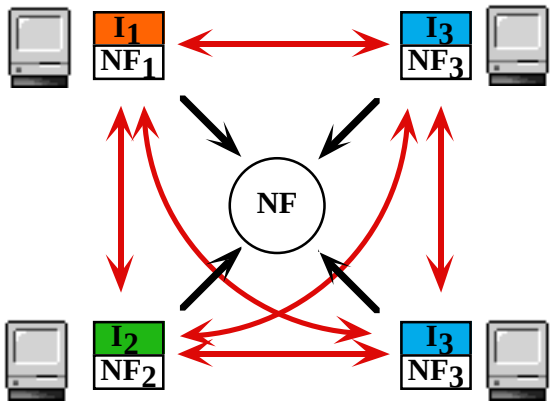


Figure 6.8. Le modèle hybride. Exemple de configuration à quatre utilisateurs. Un processus gère la station de chaque utilisateur (noyau fonctionnel local $\mathbf{NF_i}$ et interface locale $\mathbf{I_i}$) et un processus central gère le noyau fonctionnel commun **NF**. Les communications entre les processus combinent les possibilités du cas centralisé avec le processus central et celles du cas répliqué pour les processus utilisateurs.

Le *modèle hybride* apporte une solution au problème du maintien de la cohérence des données sans sacrifier la vivacité du temps de réponse de l'interface. Cette approche combine les deux cas précédents en décomposant un même système en deux parties, l'une centralisée, l'autre répliquée. La figure 6.8 présente le modèle hybride étendu au cas d'un système à plusieurs rôles.

Cette solution permet d'isoler dans la partie centralisée les données partagées et conserve le modèle répliqué pour la gestion locale. Notons qu'il n'est pas nécessaire de centraliser les données partagées. On peut utiliser le composant centralisé comme un simple synchronisateur et garder des dépositaires locaux à chaque processus utilisateur. Le synchronisateur central est alors un "chef d'orchestre" qui gère le contrôle d'accès aux données partagées répliquées. Cette solution garantit de meilleurs temps de réponse du système lorsqu'un utilisateur accède aux données partagées, comme souligné par [Karsenty 1994]. Mais en contrepartie, les données partagées doivent être maintenues cohérentes.

A l'examen de ce triple modèle fondamental centralisé/répliqué/hybride, plusieurs constatations s'imposent : la validité de ces modèles doit être appréciée selon les dimensions de notre espace problème et selon leur couverture fonctionnelle du modèle du trèfle du chapitre 1.

Les deux modèles extrêmes (centralisé et répliqué) tiennent plus du cas d'école que du modèle utilisable en pratique. Ces modèles présentent toutefois un intérêt dans des cas particuliers : lorsqu'il faut tenir compte de contraintes logicielles ou matérielles ou si l'on s'intéresse à un système multi-utilisateur limité. Un environnement logiciel particulier peut conduire à adopter un modèle centralisé pour faciliter l'implémentation et si les contraintes du cahier de charges (par exemple, on souhaite du WYSIWIS strict) s'y prêtent. On peut citer comme exemple SharedX qui repose sur X-Window [Garfinkel 1989] ou le serveur mediaspace IIIF [Buxton 1990]. Dans ce deuxième exemple, l'utilisation d'un réseau audio/vidéo analogique construit autour d'un répartiteur piloté par un serveur conduit à une architecture centralisée. Nous présentons au chapitre 7 l'architecture du système NEIMO dans lequel d'autres contraintes ont imposé de centraliser certains services. Les deux modèles centralisé et répliqué sont chacun bien adaptés à un extrême de l'éventail des systèmes multi-utilisateurs. Le modèle centralisé répond bien aux exigences de systèmes permettant principalement des tâches de production. De plus, l'utilisation de ce modèle pour un système asynchrone lève la restriction que nous avons émise sur la satisfaction de la propriété de vivacité. S'il n'y a pas de parallélisme entre les tâches des utilisateurs, on retrouve le cas d'un système utilisé successivement par différents utilisateurs. Si l'on admet du parallélisme entre les tâches des utilisateurs, et si la synchronisation est entièrement de la responsabilité du système, on retrouve un autre cas commun : les systèmes du type base de données multi-utilisateurs. Le modèle répliqué, lui, convient mieux aux systèmes privilégiant les tâches de communication. Nous reviendrons au chapitre 7 sur l'architecture de notre mediaspace VideoPort qui s'inspire de l'architecture répliquée.

D'un point de vue pratique et pour un système multi-utilisateur quelconque, le modèle hybride est le plus réaliste. Mais ce modèle est schématique et n'offre qu'une aide limitée pour guider la décomposition des services de haut niveau comme le noyau fonctionnel. Le modèle hybride est aussi simplificateur : il ne représente les systèmes hybrides existants qu'à un haut niveau d'abstraction. En pratique, une partie centralisée peut être complexe (par exemple une base de données répartie gérée par une collection ou une hiérarchie de serveurs comme l'annuaire électronique sur Minitel ou un réseau hypertexte comme World-Wide Web).

En fait, vis-à-vis de notre espace problème, le triple modèle de référence centralisé/répliqué/hybride ne couvre correctement que la dimension distribution pour une granularité de l'ordre de la combinaison de composants. Ce modèle envisage clairement les différentes localisations possibles des composants noyau fonctionnel et interface. En revanche, il montre des lacunes pour toutes les autres dimensions. En ce qui concerne les services, il n'identifie pas de services spécifiques aux systèmes multi-utilisateurs, à part peut-être un mécanisme de contrôle de la cohérence des données répliquées. Mais il ne positionne pas ce service dans les composants logiciels qu'il préconise et ne dit rien de sa réalisation. Malgré l'existence de modèles de référence simples comme le modèle Arch, le modèle centralisé/répliqué/hybride se contente de la distinction élémentaire entre interface et noyau fonctionnel. Nous verrons plus loin avec ALV ou au chapitre suivant avec notre modèle CoPAC que cette distinction peut être affinée. Les trois modèles ne permettent pas de prendre en compte de façon utile les dimensions responsabilité ou niveaux d'abstraction de notre espace problème. Le parallélisme peut être envisagé en termes de fonctionnement simultané au niveau du processus, mais sans plus de détail. Remarquons que les lacunes de ce modèle et d'une façon générale son trop haut niveau d'abstraction le rendent vite inopérant. Dans les figures 6.6, 6.7 et 6.8, nous avons essayé d'étendre le modèle en introduisant la notion de rôle. Nous avons distingué différentes combinaisons de composants de l'interface correspondant à différents rôles. Cette extension n'est toutefois pas entièrement satisfaisante : le noyau fonctionnel ne doit-il pas être également adapté au rôle ? Un composant d'interface ne peut-il pas être partagé entre plusieurs rôles ? La "grosse" granularité des composants dans ce modèle d'architecture empêche une analyse précise. Le modèle CoPAC que nous présentons au chapitre 7 vise à répondre à ces questions.

Nous avons vu que le modèle précédent pêche par manque de détail dans la définition et la structuration des composants. Le modèle ALV que nous présentons maintenant est une tentative pour y remédier.

6.5.2. Le modèle ALV

Le modèle d'architecture ALV (Abstraction-Link-View) [Hill 1992], utilisé dans le système RendezVous™, vise les systèmes multi-utilisateurs synchrones construits autour d'un noyau fonctionnel centralisé. Ce composant central est dénommé "Shared Abstraction" dans le modèle. Chaque utilisateur dispose d'un composant interface qui lui est propre et adapté à son rôle ("Personal View"). Des composants de liaison, "Link", font le lien entre l'abstraction partagée et chaque vue. Les Link maintiennent des systèmes de contraintes et propagent les modifications de l'un des deux composants vers l'autre. La figure 6.9 présente une configuration du modèle ALV pour quatre utilisateurs.

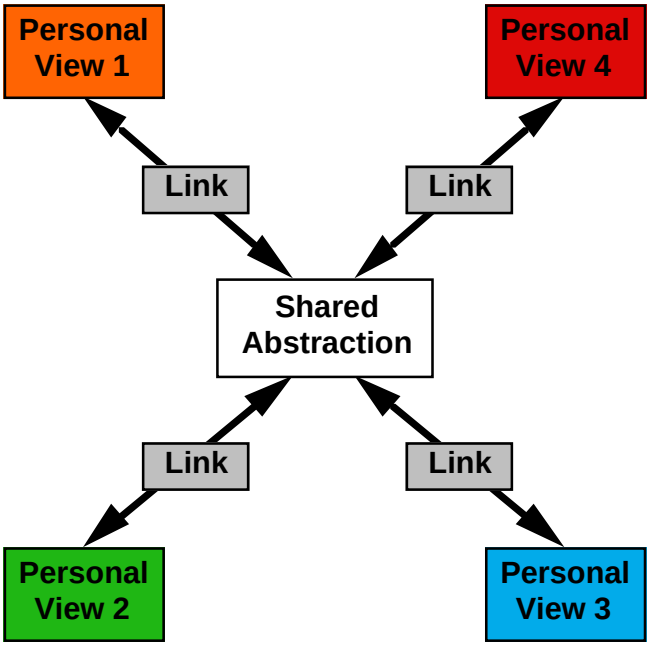


Figure 6.9. Le modèle ALV. Les flèches représentent des contraintes liant un composant abstraction partagé à des vues propres à chaque utilisateur. Les composants Link maintiennent et propagent les contraintes entres les composants abstraction et vue. Sur la figure quatre utilisateurs partagent une même abstraction.

Le modèle ALV rappelle le modèle PAC pour les interfaces mono-utilisateur : tous deux distinguent un composant abstraction et un composant d'interface et tous deux insistent sur la nécessité d'introduire un composant intermédiaire qui maintient la cohérence. Ce composant rend indépendants abstraction et interface conformément au principe de séparation [Coutaz 1990]. Toutefois, on peut noter des différences entre les deux conceptions de cette séparation.

Le modèle PAC préconise une organisation des agents PAC en une hiérarchie à l'intérieur du contrôleur de dialogue. ALV ne semble pas identifier clairement un composant de contrôle du dialogue à part le Link. Or le rôle d'un contrôleur de dialogue est double : il maintient la cohérence entre abstraction et présentation et il gère aussi l'ordonnancement

des tâches. Si le premier aspect est bien pris en compte par les Link de ALV, le deuxième aspect n'est pas explicite dans le modèle. D'autre part, ALV ne recommande pas une vraie structuration des composants. A part pour les composants vue où il est dit explicitement qu'ils sont composés d'une hiérarchie de vues, les composants constituant un Link en particulier ne sont pas organisés et sont tous au même niveau. En revanche, la hiérarchie PAC exprime la coordination entre agents. Cette analyse met en évidence deux points faibles du modèle ALV : sa couverture fonctionnelle est incomplète puisqu'il n'identifie qu'une partie du contrôle du dialogue. Du point de vue structurel, l'organisation des composants n'est indiquée que pour les vues.

Par rapport à notre espace problème, ALV insiste sur un service spécifique des systèmes multi-utilisateurs : le partage de composants abstraction. Notons qu'ALV, au moins dans une version préliminaire [Patterson 1990], identifiait un service de mise en place de session reposant sur un serveur de sessions. Ce type de service, que nous verrons plus en détail avec l'exemple de GroupKit, augmente notre éventail de services spécifiques aux systèmes multi-utilisateurs. ALV distingue deux niveaux d'abstraction (abstraction et vue) et précise comment les relier. ALV permet aussi de réfléchir à la distribution des composants, et dans une moindre mesure, aux possibilités de parallélisme. Mais ALV est un modèle qui privilégie le partage de composants abstraction ; il est donc bien adapté aux systèmes privilégiant l'aspect production et peut être vu comme un affinement du modèle centralisé.

Notons que [Croisy 1994] propose un affinement du modèle ALV dans lequel le contrôleur de dialogue est explicite. Plus précisément, ce modèle distingue un composant "multi-dialogue" géré par un superviseur et composé de "micro-dialogues". Un micro-dialogue est dédié à l'abstraction et communique avec autant de micro-dialogues de présentation qu'il y a de vues. Afin d'assurer la rétroaction de groupe, les micro-dialogues peuvent communiquer directement entre eux. Mais comme ALV, ce modèle est adapté aux systèmes privilégiant l'espace de production. D'autre part, l'indépendance des micro-dialogues n'est pas clairement garantie dans le modèle. En l'absence d'un mécanisme d'abonnement comme GroupKit (voir paragraphe 6.5.5), l'indépendance des micro-dialogues n'est pas assurée sauf s'ils sont tous identiques. La flexibilité du modèle risque donc d'être compromise. Toutefois ce modèle a l'avantage d'intégrer explicitement le contrôle du dialogue et en propose un affinement intéressant.

6.5.3. Le modèle à états partagés

Patterson, un des auteurs de RendezVous d'où est issu ALV, propose un autre modèle qui vise les mêmes objectifs mais de façon différente. Le domaine d'application est

toujours les systèmes multi-utilisateurs synchrones dans lesquels le partage est important. Le modèle à états partagés identifie quatre niveaux d'états d'une application : File, Model, View et Display [Patterson 1994]. Il distingue deux modes de partage : par états partagés et par états synchronisés. Sur la figure 6.10, l'état du composant Model est partagé et l'état des composants View est synchronisé.

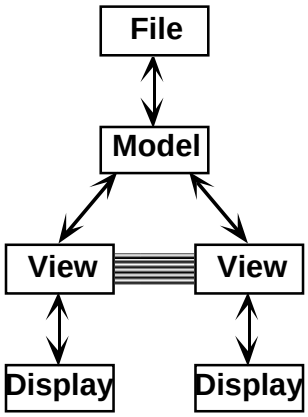


Figure 6.10. Le modèle à états partagés, d'après [Patterson 1994]. Pour faciliter la lecture, ce schéma montre une configuration à deux utilisateurs. Le composant Model est partagé ainsi que le composant File. Les deux composants View sont synchronisés (barres horizontales). Les composants Display sont indépendants.

Dans le partage d'états, si un composant est partagé, tous les composants de niveau plus abstrait sont aussi partagés (sur la figure, Model est partagé par états, donc File l'est aussi). Dans le partage par synchronisation, les deux copies des composants utilisent un protocole pour synchroniser leurs états.

A première vue, ce modèle peut apparaître comme un affinement de ALV qui prend en compte la possibilité de maintenir la cohérence entre composants répliqués, comme le permet le modèle hybride. En fait il faut noter une différence : le modèle repose sur un partage des états des composants et non sur un partage des données comme le modèle hybride du paragraphe 6.5.1. D'autre part, les quatre niveaux d'abstraction identifiés par le modèle à états partagés sont différents de ceux d'ALV. L'abstraction d'ALV est décomposée en File et Model, et la vue d'ALV est précisée en View et Display. Mais au contraire d'ALV, le modèle ne prend pas du tout en compte le contrôle du dialogue. Or l'état du dialogue devrait pouvoir aussi être partagé. Il est probable qu'une partie du dialogue est diluée dans les mécanismes de synchronisation et de partage sous-jacents. Mais le dialogue en tant que tel n'apparaît pas.

Comme ALV, le modèle à états partagés souffre d'une couverture fonctionnelle incomplète. Le contrôle du dialogue n'est pas explicite. Il semble qu'une partie du dialogue soit présente dans le mécanisme de synchronisation sous-jacent. Mais dans ce

cas, la synchronisation devrait apparaître comme un composant à part entière et non comme un simple connecteur. En fait, ce modèle nous semble supposer l'existence d'une plate-forme matérielle d'accueil spécifique, comportant en particulier des mécanismes de synchronisation de haut niveau d'abstraction. Dans ce cas, la synchronisation peut être un service offert par le système et peut être en effet vue comme un type particulier de connecteur. Cette constatation est à double tranchant : elle montre une limitation du modèle en termes de validité vis-à-vis de l'environnement de l'application. Mais elle montre aussi que le modèle distingue explicitement un service spécifique aux systèmes multi-utilisateurs : le partage et la synchronisation d'états entre composants. Vis-à-vis des autres axes de notre espace problème, le modèle n'est pas beaucoup plus complet que ALV : il présente l'avantage de mettre en évidence le service de synchronisation et affine les niveaux d'abstraction de ALV. Il permet donc de considérer plus complètement les possibilités de parallélisme et de distribution des composants. Il met aussi en évidence la migration du lieu de partage et de synchronisation.

Pour le cas des systèmes reposant sur le partage d'informations, nous avons vu plusieurs décompositions : du cas simpliste du modèle centralisé jusqu'à la décomposition à quatre niveaux du modèle à états partagés. Il convient ici de citer le modèle SLICE [Karsenty 1994] qui identifie sept niveaux et est à notre connaissance le plus complet. Sans être à proprement parler un modèle d'architecture, SLICE (Sharing Layers In Cooperative Editing) propose une structuration en sept couches : document abstrait, représentation du document, manipulation directe, représentation des vues, manipulation des vues, manipulation indirecte, curseurs. Chacune des couches peut être partagée entre les utilisateurs de façon indépendante. Cette décomposition fournit un cadre de réflexion pour le partage d'informations des systèmes d'édition coopérative.

Notons que ni ALV, ni le modèle à états partagés ne tiennent compte des services de la classe des communicateurs. ALV privilégie les dépositaires et le modèle à états partagés privilégie, quoiqu'incomplètement, les synchronisateurs. Cette limitation est peut-être induite par le domaine d'application qui conduit d'abord à s'interroger sur le partage de l'information. Le modèle des User Display Agents que nous présentons maintenant est une autre approche de ce même domaine.

6.5.4. Le modèle des User Display Agents

Le modèle des User Display Agents [Bentley 1994], comme les modèles précédents, s'intéresse aussi au cas des systèmes synchrones. Il décompose un système en une partie centralisée et une partie répliquée. La partie centralisée s'articule autour de l'Object Store, un dépositaire d'objets partagés. L'Object Store Server est le composant qui gère l'Object

Store : il est composé de l'Update Handler qui transmet toutes les modifications des objets et gère l'accès aux objets. Ses autres constituants sont des User Display Agents (UDA). Un UDA a la charge de gérer un élément d'interaction en relation avec l'Object Store. Ce type d'agent est proche d'un agent de type PAC ou MVC, avec la différence que sa partie présentation ou vue est un ensemble de composants répliqués, les UDAs délégués. Les UDAs délégués sont répliqués sur chacune des stations utilisateur.

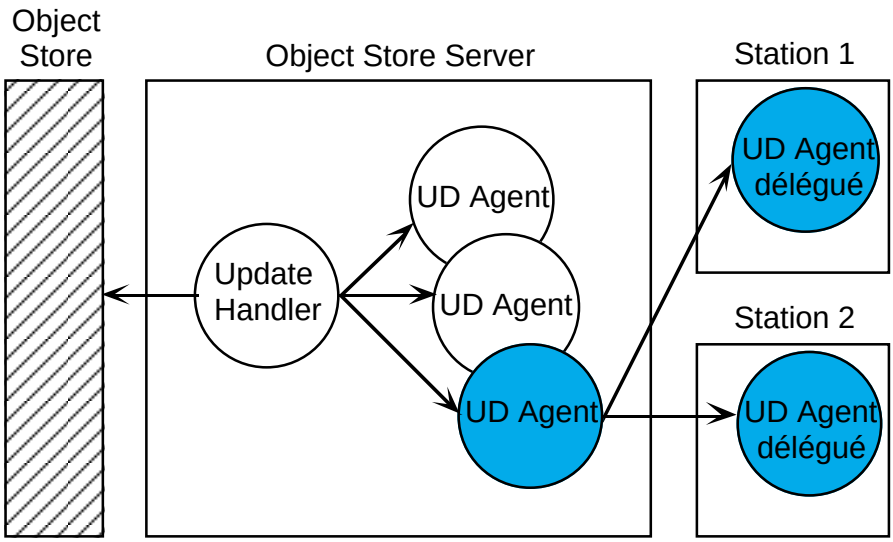


Figure 6.11. Le modèle des User Display Agents (UDA). Un UDA est localisé sur le serveur central et gère un composant d'interface. Sur chaque station utilisateur, des agents UDA délégués reflètent à l'interface les changements d'état de leur UDA "maître". Le Update Handler répartit les changements des objets de l'Object Store aux UDAs concernés.

Ce modèle nous fait penser à la gestion des vues multiples d'une interface mono-utilisateur avec le modèle d'architecture PAC : ce modèle recommande d'assigner à un agent la gestion d'une vue donnée et de chapeauter ces agents vue par un agent père chargé de maintenir la cohérence entre les vues. Le modèle des UDAs est une sorte de généralisation de la gestion des vues multiples.

Le modèle des UDAs permet de raisonner à un niveau d'abstraction plus fin que les modèles que nous avons vus précédemment. Il introduit des agents qui modélisent des composants en charge d'un élément d'interaction donné. Cette décomposition plus fine permet aussi de réfléchir à la délégation entre composants. Le modèle fait intervenir un service spécifique aux systèmes multi-utilisateurs : l'Object Store Server gère la session et permet l'abonnement et le désabonnement des UD délégués. Dans ce modèle encore, l'existence d'une infrastructure système particulière (système à objets) est supposée. Mais au regard de notre espace problème, le modèle des UDAs est le plus complet de ceux que nous avons présentés. Toutefois, lui non plus ne prend pas en compte les composants de la classe communicateurs.

6.5.5. GroupKit

Les modèles que nous avons vus jusqu'à présent s'intéressent au cas des systèmes multi-utilisateurs qui privilégient la dimension production. Nous présentons maintenant le modèle d'architecture de GroupKit [Roseman 1992], qui repose sur la notion de conférence et met en avant la dimension coordination. GroupKit est une boîte à outils de haut niveau pour la construction des systèmes multi-utilisateurs et non pas un modèle d'architecture proprement dit. Mais nous verrons au chapitre 8 que les outils de haut niveau reposent sur un modèle d'architecture implicite sous-jacent. Nous explicitons maintenant le modèle d'architecture sous-jacent de GroupKit, présenté figure 6.12.

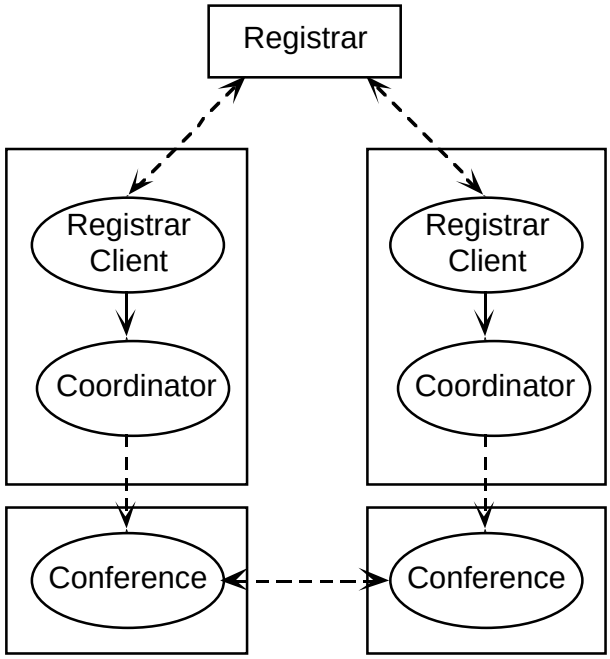


Figure 6.12. Le modèle sous-jacent de GroupKit. La figure présente une configuration pour deux utilisateurs. Les rectangles représentent des processus et les ovales représentent des objets. Les flèches pleines matérialisent les envois de messages entre objets et les flèches pointillées sont des communications inter-applications via des sockets.

En haut de la figure 6.12, le Registrar est un composant centralisé avec lequel les Registrar Client communiquent pour initier, se joindre à, ou quitter une conférence. Le composant Coordinator gère des processus Conference. Sur la figure, nous n'avons fait figurer qu'un seul processus Conference attaché à chaque Coordinator, mais un Coordinator a la capacité de gérer plusieurs Conference qui peuvent être de natures différentes. Une Conference est une application partagée et peut être par exemple, un module d'édition de texte ou de dessin partagé ou un module de communication audio ou vidéo. Une Conference contient à la fois le noyau fonctionnel, le dialogue et l'interface utilisateur.

On peut regretter dans ce modèle qu'il ne détaille pas plus l'objet Conference. Le modèle de ce fait ne garantit pas l'observation de principes élémentaires comme la distinction entre noyau fonctionnel et interface utilisateur. Cependant, le modèle de GroupKit a le mérite de mettre en évidence un service important pour les systèmes multi-utilisateurs : l'abonnement des utilisateurs à une session. Vis-à-vis de notre découpage fonctionnel, les composants Registrar, Registrar Client et Coordinator assurent l'essentiel de la facette coordination. Les aspects production et communication sont couverts par le composant Conference. Un autre aspect intéressant du modèle de GroupKit est qu'il distingue deux types de connecteurs : les messages de contrôle entre objets représentés par des flèches pleines et les communications entre processus représentées par des flèches pointillées. Toutefois, nous pensons que cette distinction n'a été faite que pour renforcer la distinction entre processus et objets sur le schéma d'architecture. Par exemple, la communication entre deux objets Conference sera toujours matérialisée par des flèches pointillées, quelle que soit la nature des échanges : messages adressés au noyau fonctionnel distant dans le cas d'un éditeur de dessin ou messages dont le contenu n'est pas compris par le système comme dans le cas d'une vidéoconférence. Dans le modèle CoPAC présenté au chapitre suivant, nous proposons de distinguer explicitement les messages destinés au noyau fonctionnel de ceux dont la sémantique n'est pas interprétée par le système comme les flots audio/vidéo.

Contrairement aux modèles que nous avions vus jusqu'ici, le modèle de GroupKit détaille peu les composants en charge de l'espace de production et insiste particulièrement sur la facette coordination. Nous allons voir maintenant un modèle qui recouvre l'aspect communication.

6.5.6. Le modèle de communication de Gemma

Gemma est une architecture pour un système de fenêtrage distribué qui étend X-Window [Freeman 1993]. Gemma dans son ensemble s'intéresse particulièrement à des systèmes faisant un usage intensif de multiples dispositifs d'entrée. L'architecture globale proposée rappelle le modèle à états partagés de Patterson mais se limite à une synchronisation au niveau View ou Display. Un intérêt de ce modèle est qu'il prend aussi en compte la communication entre utilisateurs via des médias continus comme le son ou la vidéo. Le modèle adopté pour la communication est assez classique : on retrouve une architecture similaire dans [Koegel Buford 1994]. Notre bibliothèque de communication UserLink que nous présentons au chapitre 8 adopte une approche analogue. Le principe directeur de cette architecture repose sur la distinction entre informations à communiquer et contrôle du canal de communication (figure 6.13).

Une source génère un flot d'un média continu, comme un système de capture vidéo. Un puits est un récepteur d'un flot continu, comme une fenêtre d'affichage. Un gestionnaire de capture contrôle la source. Par exemple, c'est ce composant qui a la charge d'initier le flot vidéo ou de le stopper. Le gestionnaire de présentation a la responsabilité d'initier le puits, c'est à dire de lui indiquer sa surface d'affichage, et gère les actions de l'utilisateur sur la surface d'affichage. Par exemple un clic sur la fenêtre vidéo peut signifier qu'il faut stopper le flot vidéo. C'est le gestionnaire de présentation qui effectue ce traitement. Le gestionnaire de présentation et le gestionnaire de capture communiquent directement entre eux pour échanger des informations de contrôle et de synchronisation. Le gestionnaire de synchronisation va, par exemple, répercuter sur le gestionnaire de capture une modification de la taille de la surface d'affichage.

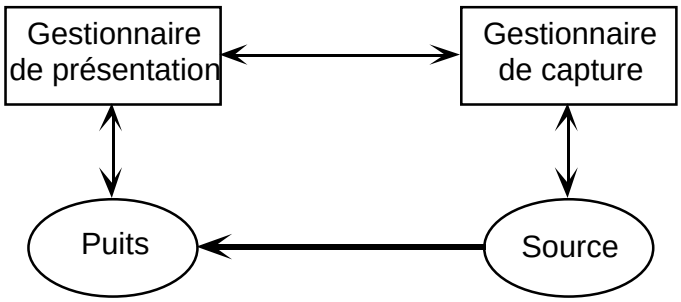


Figure 6.13. Le modèle de communication de Gemma. Le trait gras indique un flot de données. Les autres flèches indiquent des flots de contrôle.

Ce modèle usuel est bien adapté à l'intégration des médias continus dans une architecture. La séparation qu'il impose entre flots de données et contrôle traduit les propriétés différentes qui sont requises des deux types de communication. En particulier, des mécanismes adaptés au niveau des sources et des puits permettent d'assurer par exemple les propriétés de régularité des flots ou de synchronisation. On peut regretter que l'architecture de Gemma ne propose pas d'intégration entre la communication et les autres services pour les systèmes multi-utilisateurs. La communication est vue comme un aspect indépendant. La plupart des modèles d'architecture pour les systèmes multi-utilisateurs qui intègrent l'aspect communication adoptent aussi cette approche. Or elle rend plus difficile une meilleure intégration des services de communication avec les autres services des systèmes multi-utilisateurs. Nous avons tenté avec le modèle CoPAC présenté au chapitre suivant de mieux intégrer la communication avec les deux autres facettes de l'espace fonctionnel des systèmes multi-utilisateurs, production et coordination.

6.6. Conclusion

Dans ce chapitre, nous avons vu qu'une architecture logicielle communique quatre perspectives sur le système qu'elle décrit : fonctionnelle, structurelle, allocation et

coordination. Une architecture n'est pas bonne ou mauvaise dans l'absolu mais répond à des critères et un contexte particulier. Pour réfléchir sur les modèles d'architecture pour les systèmes multi-utilisateurs, nous avons donc énoncé des caractéristiques générales attendues d'un modèle d'architecture pour ce type de systèmes, puis nous avons défini un espace-problème pour servir de cadre à notre analyse de modèles d'architecture existants. Nous avons passé en revue quelques modèles d'architecture représentatifs des modèles d'architecture pour les systèmes multi-utilisateurs et avons constaté que ceux-ci ne couvrent qu'incomplètement notre espace-problème. En termes de couverture fonctionnelle au sens du modèle du trèfle du collecticiel, les modèles d'architecture favorisent généralement l'un des trois espaces, production, coordination ou communication au détriment des deux autres.

Références

[Abowd 1994] G. D. Abowd. *Defining reference models and software architectural styles for cooperative systems*, Workshop on Software architectures for cooperative systems at CSCW'94, ACM Conference on Computer-Supported Cooperative Work, Chapel Hill, North Carolina, USA, 1994.

[Apple 1988] *The Knowledge Navigator*, vidéo. Apple Computer Inc., Cupertino, California, USA, 1988.

[Apple 1994] *Finder 7.5*. Logiciel pour Macintosh. Apple Computer Inc., Cupertino, California, USA, 1994.

[Bass 1994] L. Bass et G. Abowd. *Software Architecture: A Tutorial Introduction*, Software Engineering Institute, Carnegie-Mellon University, Tutorial, 1994.

[Bass 1992] L. Bass, R. Little, R. Pellegrino, S. Reed, R. Seacord, S. Sheppard et M. R. Szczur. *The UIMS Tool Developers' Workshop: A Metamodel for the Runtime Architecture of an Interactive System*, in *SIGCHI Bulletin*, 24(1), janvier 1992. pp. 32-37.

[Bentley 1994] R. Bentley, T. Rodden, P. Sawyer et I. Sommerville. *Architectural Support for Cooperative Multiuser Interfaces*, in *IEEE Computer*, 27(5), mai 1994. pp. 37-46.

[Bier 1991] E. A. Bier et S. Freeman. *MMM: A User Interface Architecture for Shared Editors on a Single Screen*, UIST'91, ACM Symposium on User Interface Software and Technology, Hilton Head, South Carolina, USA, 1991. pp. 79-86.

[Buxton 1990] W. Buxton et T. Moran. *EuroPARC's Integrated Interactive Intermedia Facility (IIIF): Early Experiences*, IFIP Conference on Multi-User Interfaces and Applications, Heraklion, Crète, 1990.

[Coutaz 1987] J. Coutaz. *PAC, an Implementation Model for Dialog Design*, Interact'87, IFIP Conference on Human-Computer Interaction, Stuttgart, 1987. pp. 431-436.

[Coutaz 1990] J. Coutaz. *Interface homme-ordinateur : conception et réalisation*, Dunod, Paris, France, 1990.

[Coutaz 1993] J. Coutaz, L. Nigay et D. Salber. *The MSM Framework: A Design Space for Multi-Sensori-Motor Systems*, in *Human-Computer Interaction, Third International Conference, EWHCI'93, Moscow, Russia, August 3-7, 1993, Selected Papers*. Lecture Notes in Computer Science n° 753, L. J. Bass, J. Gornostaev et C. Unger, (eds.). Springer-Verlag, Berlin, 1993. pp. 231-241.

[Croisy 1994] P. Croisy. *Modèle multi-agent et conception d'applications coopératives interactives*, IHM'94, Sixièmes journées sur l'Ingénierie des Interfaces Homme-Machine, Lille, France, 1994. pp. 139-144.

[Cypher 1991] A. Cypher. *EAGER: Programming Repetitive Tasks by Example*, CHI'91, ACM Conference on Human Factors in Computing Systems, New Orleans, Louisiana, USA, 1991. pp. 33-39.

[Dorner 1995] *Eudora 1.5.1*. Logiciel pour Apple Macintosh. Steve Dorner & Qualcomm, 1995.

[Ellis 1994] C. Ellis. *Keepers, Synchronizers, Communicators and Agents*, Workshop on Software architectures for cooperative systems at CSCW'94, ACM Conference on Computer-Supported Cooperative Work, Chapel Hill, North Carolina, USA, 1994.

[Freeman 1993] S. M. G. Freeman. *An Architecture for Distributed User Interfaces*. Ph.D Thesis, Darwin College, University of Cambridge, 1993.

[Garfinkel 1989] D. Garfinkel. *The Shared X Multiuser Interface User's Guide*, Hewlett-Packard, Research Report, STL-TM-89-07, 1989.

[Garlan 1993] D. Garlan et M. Shaw. *An introduction to software architecture*, in *Advances in Software Engineering and Knowledge Engineering.* World Scientific Publishing Company, 1993.

[Hill 1992] R. D. Hill. *The Abstraction-Link-View Paradigm: Using Constraints to Connect User Interfaces to Applications*, CHI'92, ACM Conference on Human Factors in Computing Systems, Monterey, California, USA, 1992. pp. 335-342.

[Karsenty 1994] A. Karsenty. *GroupDesign : un collecticiel synchrone pour l'édition partagée de documents.* Thèse de doctorat, Université d'Orsay Paris-Sud, 1994.

[Kazman 1994] R. Kazman, L. Bass, G. Abowd et M. Webb. *SAAM: A Method for Analyzing the Properties of Software Architectures*, ICSE-16, Sorrento, Italy, 1994.

[Koegel Buford 1994] J. F. Koegel Buford. *Middleware System Services Architecture*, in *Multimedia Systems*. J. F. Koegel Buford, (ed.) Addison-Wesley, New York, New York, USA, 1994. pp. 221-244.

[Krakowiak 1990] S. Krakowiak, M. Meysembourg, H. Nguyen Van, M. Riveill et C. Roisin. *Design and Implementation of an object-oriented strongly typed language for distributed applications*, in *Journal of Object-Oriented Programming*, septembre 1990.

[Malone 1986] T. W. Malone, K. R. Grant et F. A. Turbak. *The Information Lens: An Intelligent System for Information Sharing in Organizations*, CHI'86, ACM Conference on Human Factors in Computing Systems, 1986. pp. 1-8.

[Metral 1994] M. Metral. *MAXIMS: A Learning Interface Agent For Eudora (A User's Guide to the System)*, MIT Media Lab, Technical report, 1994.

[Nigay 1994] L. Nigay. *Conception et réalisation des systèmes interactifs: Application aux Interfaces Multimodales.* Thèse de doctorat, Université Joseph Fourier Grenoble I, 1994.

[Patterson 1994] J. F. Patterson. *A Taxonomy of Architectures for Synchronous Groupware Applications*, Workshop on Software architectures for cooperative systems at CSCW'94, ACM Conference on Computer-Supported Cooperative Work, Chapel Hill, North Carolina, USA, 1994.

[Patterson 1990] J. F. Patterson, R. D. Hill, S. L. Rohall et W. S. Meeks. *Rendezvous: An Architecture for Synchronous Multi-User Applications*, CSCW'90, ACM Conference on Computer-Supported Cooperative Work, Los Angeles, California, USA, 1990. pp. 317-328.

[Roseman 1992] M. Roseman et S. Greenberg. *GROUPKIT: A Groupware Toolkit for Building Real-Time Conferencing Applications*, CSCW'92, ACM Conference on Computer-Supported Cooperative Work, Toronto, Canada, 1992. pp. 43-50.

[Shaw 1995] M. Shaw et D. Garlan. *Software Architecture. Perspectives on an Emerging Discipline*, Prentice Hall, 1995.

CoPAC, notre modèle d'architecture pour les systèmes multi-utilisateurs

C'est dans les vieilles marmites
qu'on fait les meilleures soupes.

Dicton populaire

CoPAC, notre modèle d'architecture
pour les systèmes multi-utilisateurs

7.1. Introduction

Notre présentation au chapitre précédent des modèles d'architecture pour les systèmes multi-utilisateurs nous a montré plusieurs insuffisances. Vis-à-vis de notre espace problème, aucun modèle n'apporte une solution couvrant toutes les dimensions. Vis-à-vis du modèle du trèfle du chapitre 1, les modèles tendent à privilégier une des trois dimensions du trèfle, souvent l'espace de production, au détriment des autres. Nous nous fixons donc comme objectif dans ce chapitre l'élaboration d'un modèle d'architecture qui intègre harmonieusement les trois dimensions du trèfle et qui réponde le plus largement possible aux dimensions de notre espace-problème des modèles d'architecture. Comme les interfaces multi-utilisateurs sont aussi en un sens des interfaces utilisateur classiques, nous partons de modèles proposés pour les systèmes mono-utilisateurs. Nous verrons que cette approche nous permet de préserver les propriétés de ces modèles tout en les étendant aux systèmes multi-utilisateurs. Nous présentons d'abord PAC et PAC-Amodeus qui servent de base à notre modèle et nous justifions ce choix. Nous présentons ensuite CoPAC, extension du modèle PAC-Amodeus aux systèmes multi-utilisateurs. Puis nous détaillons plusieurs exemples de mise en œuvre du modèle. Chaque exemple met en avant une dimension particulière du trèfle et nous permet de vérifier la validité du modèle pour chacune des composantes de l'espace fonctionnel des systèmes multi-utilisateurs.

7.2. Nos modèle de base : PAC, Arch et PAC-Amodeus

Notre modèle CoPAC repose sur le modèle PAC-Amodeus, lui-même issu des modèles PAC et Arch. Ces trois modèles d'architecture sont adaptés aux systèmes mono-utilisateurs. Nous avons ainsi voulu profiter des acquis, en particulier afin de bénéficier des propriétés de ces modèles. Nous présentons maintenant brièvement ces trois modèles et les propriétés de qualité du logiciel qu'ils permettent d'assurer.

7.2.1. Le modèle PAC

Le modèle PAC est un modèle multi-agent pour la conception logicielle des systèmes interactifs [Coutaz 1987]. Il repose sur deux principes directeurs : le concept d'agents réactifs à facettes, et l'organisation hiérarchique de ces agents.

Un agent réactif est un système de traitement de l'information. Il est constitué d'un ensemble d'opérations, de mécanismes d'entrée/sortie et d'un état interne. A la différence des agents cognitifs de l'intelligence artificielle, un agent réactif n'a pas à un but qu'il cherche à satisfaire. Il réagit à des stimuli, génère des réponses et modifie son état en

fonction de ces stimuli. Un agent PAC est constitué de trois facettes, chacune ayant une compétence particulière (figure 7.1).

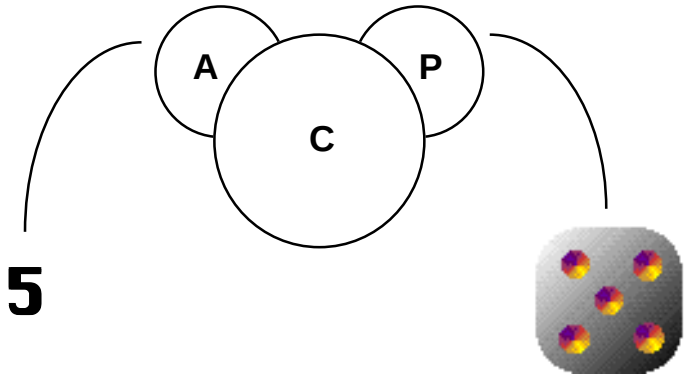


Figure 7.1. Un agent PAC. L'agent gère ici un dé à jouer. Son abstraction (A) maintient la valeur courante sous forme d'un entier. Sa présentation (P) gère la représentation visible du dé. Dans ce cas il s'agit d'une image stockée en ressource et représentant une face du dé. Le contrôleur de dialogue (C) assure la traduction de formalismes entre A (un entier, valeur du dé) et P (un entier, numéro de ressource de l'image à afficher).

- La facette P, Présentation, définit le comportement perceptible de l'agent pour l'utilisateur. Cette facette gère à la fois les communications de l'utilisateur vers l'agent (entrées) et de l'agent vers l'utilisateur (sorties).

- La facette A, Abstraction, définit la compétence propre de l'agent, indépendamment de toute considération de présentation.

- La facette C, Contrôle, a deux rôles : il exprime les dépendances et assure la traduction de formalismes entre les facettes A et P. Il gère aussi la communication de l'agent avec son environnement, c'est-à-dire d'autres agents PAC.

Un système interactif est constitué d'un ensemble d'agents PAC organisés en une hiérarchie. Cette hiérarchie traduit des relations de coopération dynamique entre les agents et reflète un continuum de niveaux d'abstraction depuis le noyau fonctionnel jusqu'aux éléments d'interaction (figure 7.2).

[Nigay 1994] identifie trois propriétés de qualité du logiciel que l'utilisation du modèle PAC permet de satisfaire : l'indépendance entre le noyau fonctionnel et la présentation qui est un précepte fondamental de la conception d'interfaces, la réutilisabilité des composants de l'interface grâce à l'indépendance des facettes P et A, la modifiabilité indispensable à la conception itérative des systèmes interactifs. Notons aussi que la granularité fine des agents PAC apporte deux avantages : elle permet une conception

logicielle détaillée et il est possible de réfléchir de façon précise à la modifiabilité et à la réutilisabilité.

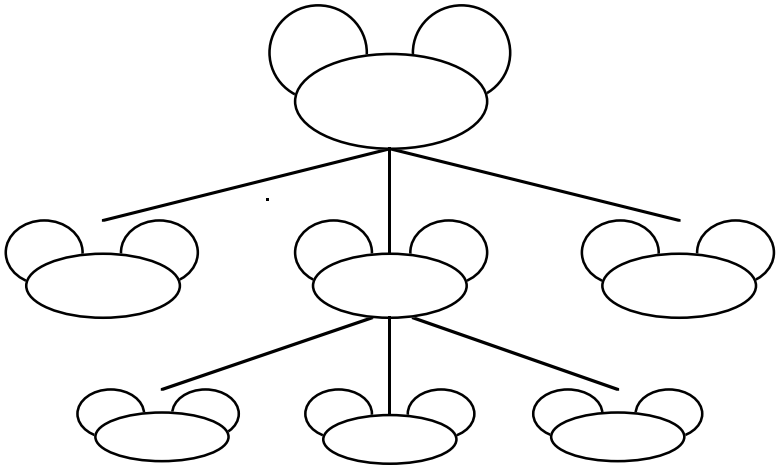


Figure 7.2. Une hiérarchie d'agents PAC. Les agents communiquent par leur composant Contrôle.

7.2.2. Le modèle Arch

Le modèle Arch [Bass 1992] est un affinement du classique modèle de Seeheim. Il distingue cinq composants organisés sous forme d'une arche : noyau fonctionnel, adaptateur de domaine, contrôleur de dialogue, présentation et interaction (figure 7.3).

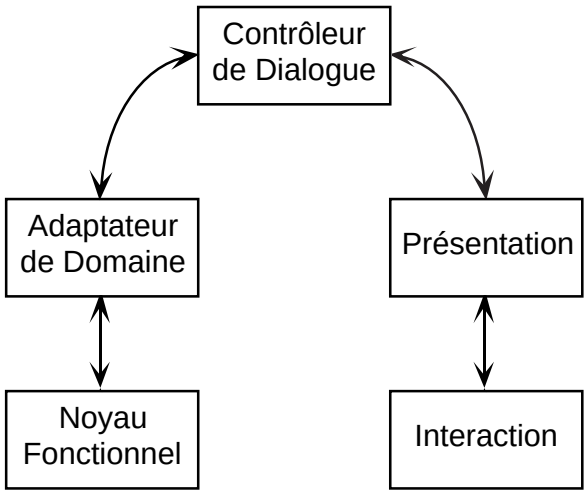


Figure 7.3. Le modèle Arch.

- Le noyau fonctionnel implémente les concepts du domaine.

- Le composant interaction est en contact direct avec l'utilisateur et peut être mis en œuvre grâce à une boîte à outils.

- Le contrôleur de dialogue est la clé de voûte du modèle : il gère l'enchaînement des tâches et assure le lien entre les composants présentation et adaptateur de domaine.

- L'adaptateur de domaine établit un pont entre le contrôleur de dialogue et le noyau fonctionnel. Ce composant permet d'ajuster les différences de modélisation des objets conceptuels entre les deux composants qui l'entourent.

- Le composant présentation a un rôle similaire à l'adaptateur de domaine pour l'autre branche de l'arche : il permet de définir une boîte à outils virtuelle qui est concrétisée par le composant interaction.

Nous retenons deux points forts du modèle Arch : la présence de l'adaptateur de domaine et le composant présentation.

L'*adaptateur de domaine* joue un rôle de médiateur entre le contrôleur de dialogue et le noyau fonctionnel. En tant que tel, il implémente un protocole de communication entre ces deux composants. Un tel protocole est caractérisé par une stratégie temporelle et la nature des données échangées. La *stratégie temporelle* définit les règles de coordination entre les deux entités communicantes : la communication peut être synchrone, asynchrone ou alterner entre ces deux modes. Dans le mode synchrone, la communication est du style requête-réponse. De ce fait, du point de vue de l'utilisateur, le dialogue est soit à fil unique, soit dans le meilleur des cas à fils multiples entrelacés. Dans le mode asynchrone en revanche, le dialogue à fils multiples parallèle est possible. Ces deux modes correspondent donc à deux interprétations différentes de la propriété de dialogue à fils multiples. Notons que pour implémenter le mode asynchrone, il est nécessaire que le noyau fonctionnel et le contrôleur de dialogue soient implémentés dans deux processus séparés, ou au minimum, qu'ils soient exécutés dans deux *threads* distincts d'un même processus. Dans le cas multi-utilisateur, cette stratégie temporelle prend une nouvelle importance comme nous le verrons au paragraphe 7.3.1. Les *données échangées* via l'adaptateur de domaine sont des objets du domaine de l'application, connus par le noyau fonctionnel. Si la nature de ces objets n'est pas adaptée à leur présentation à l'utilisateur, l'adaptateur du noyau fonctionnel permet d'enrichir la sémantique de ces objets de façon à les faire correspondre aux attentes de l'utilisateur. Il peut s'agir de combiner plusieurs structures du noyau fonctionnel en un unique objet du domaine, ou au contraire de segmenter un concept du noyau fonctionnel en un ensemble d'objets du domaine. Cette notion de délégation sémantique confiée à l'adaptateur du noyau fonctionnel, dont nous avons déjà parlé au chapitre 6, participe à la satisfaction de deux propriétés : la

configurabilité pour laquelle on peut envisager différents types de délégation sémantique adaptés aux différents rôles, et l'observabilité publiée. Les filtres requis pour la satisfaction de l'observabilité publiée sont typiquement implémentés par l'adaptateur du noyau fonctionnel.

Le *composant présentation* garantit l'indépendance du contrôleur de dialogue vis-à-vis du composant interaction qui recouvre la boîte à outils de la plate-forme d'accueil. Ce composant permet de définir une boîte à outils virtuelle qui permet d'assurer la propriété de portabilité. Certains outils, comme les squelettes d'application intègrent explicitement ce composant et masquent au réalisateur les particularités de la plate-forme d'accueil. Nous avons dit toutefois au chapitre 4 que la satisfaction de la propriété de portabilité impose des contraintes au réalisateur. Elle exige souvent l'utilisation exclusive des services qui constituent le plus petit dénominateur commun entre les plates-formes d'accueil et interdit l'exploitation de toute la richesse d'une plate-forme donnée. Avec le composant présentation, il est cependant possible d'enrichir une boîte à outils suivant un mécanisme analogue à la délégation sémantique pour l'adaptateur du noyau fonctionnel.

Le modèle Arch est un modèle de référence : avec le métamodèle Slinky qui permet de faire varier l'importance relative de ses cinq composants, il peut être adapté aux contraintes d'un environnement particulier. Un des intérêts du modèle Arch consiste en la présence des composants adaptateur de domaine et présentation qui permettent de garantir la satisfaction de propriétés.

7.2.3. Le modèle PAC-Amodeus

PAC-Amodeus combine la décomposition fonctionnelle canonique du modèle Arch et affine le contrôleur de dialogue en agents PAC [Nigay 1994] (figure 7.4).

Dans PAC-Amodeus, le contrôleur de dialogue d'Arch est organisé en une hiérarchie d'agents PAC. Chaque agent PAC établit plusieurs liaisons :

- la facette A de chaque agent est en relation avec un objet du domaine situé dans l'Adaptateur du Noyau Fonctionnel (ANF, équivalent de l'adaptateur de domaine de Arch). La facette P d'un agent pointe sur un objet de présentation dans le Composant Techniques de Présentation (CTP, équivalent du composant présentation de Arch). Comme dans PAC, le contrôle de chaque agent réalise la traduction entre le objets de présentation et les objets du domaine.

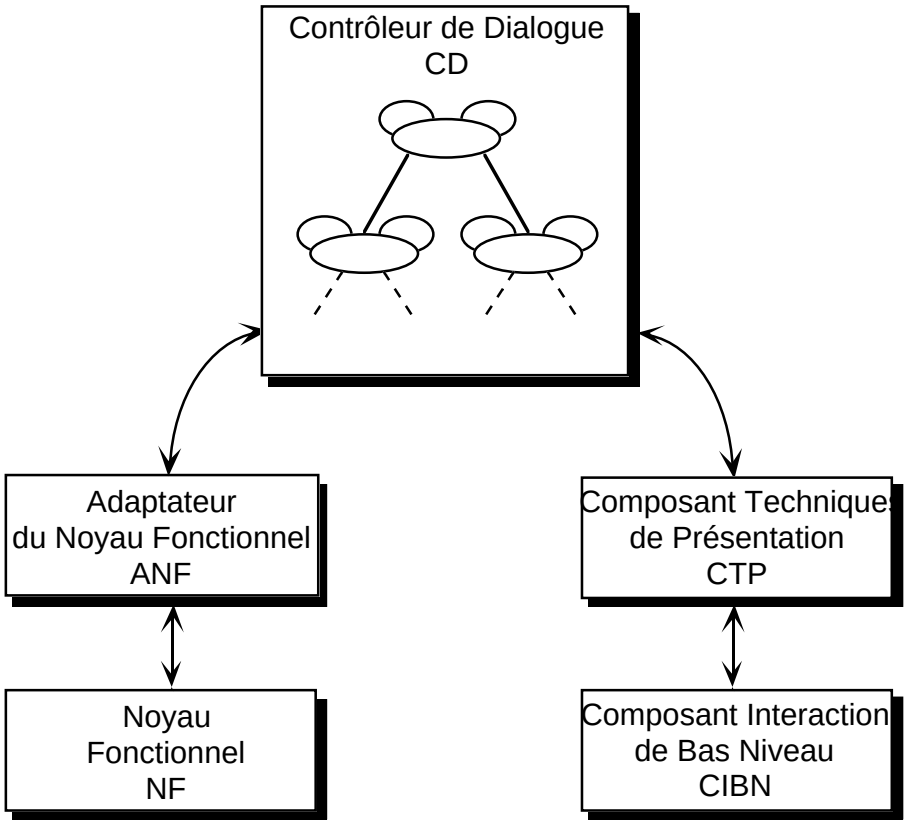


Figure 7.4. Le modèle PAC-Amodeus. Reposant sur la structure du modèle Arch, il affine le contrôleur de dialogue en une hiérarchie d'agents PAC.

- Chaque agent est relié à des agents fils et parents dans la hiérarchie. Comme dans PAC, la hiérarchie définit les relations de coopération entre agents. Cette décomposition appliquée au Contrôleur de Dialogue (CD) de Arch est particulièrement adaptée au rôle d'enchaînement des tâches du CD. On peut identifier dans la hiérarchie d'agents des grappes d'agents qui auront la charge d'une tâche donnée. La composition de ces grappes d'agents alliée au fonctionnement indépendant des agents de la hiérarchie permet d'implémenter les relations entre tâches comme le parallélisme ou l'entrelacement. Le modèle PAC-Amodeus permet ainsi de garantir la satisfaction de la propriété de dialogue à fils multiples.

Le modèle PAC-Amodeus conserve les propriétés intrinsèques du modèle Arch comme la portabilité et la décomposition fonctionnelle qu'il préconise. Grâce à la décomposition du Contrôleur de Dialogue en agents PAC, il permet aussi de satisfaire les propriétés de qualité du logiciel de modifiabilité et réutilisabilité, ainsi que les propriétés d'utilisabilité comme le dialogue à fils multiples.

7.3. Le modèle CoPAC

CoPAC est l'extension du modèle PAC-Amodeus aux systèmes multi-utilisateurs. Il augmente à la fois les agents PAC et le modèle Arch. CoPAC est un modèle d'architecture conceptuel et nous verrons comment il répond aux différentes dimensions de l'espace-problème que nous avons présenté au chapitre précédent. CoPAC étend PAC-Amodeus de deux façons :

- il prend en compte les systèmes multi-utilisateurs en répondant à l'espace-problème des modèles d'architecture des systèmes multi-utilisateurs. Il apporte aussi des réponses aux caractéristiques requises d'un modèle d'architecture pour les systèmes multi-utilisateurs énoncées au chapitre précédent,

- Il introduit de nouveaux composants et connecteurs pour répondre aux demandes particulières des services de communication. En effet, ces services exigent de manipuler de nouveaux types de données, les flots de médias continus, pour lesquels les modèles multi-agents n'ont pas de solution explicite.

7.3.1. Le partage des composants dans CoPAC

Un moyen simple de prendre en compte les systèmes multi-utilisateurs avec PAC-Amodeus pourrait consister à répliquer le modèle pour chaque utilisateur et à introduire des mécanismes de synchronisation pour garantir par exemple la cohérence des données. Mais cette approche mène à une duplication inutile de composants logiciels et à des difficultés pour identifier les mécanismes de synchronisation.

L'approche que nous avons choisie consiste à donner la possibilité de partager de différentes façons chacun des cinq composants du modèle Arch sur lequel repose PAC-Amodeus. Plusieurs cas sont envisageables : chacun des composants peut être découplé, couplé, ou commun. La figure 7.5 résume les différents modes de partage d'un composant de l'arche.

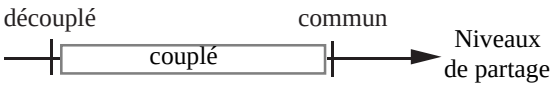


Figure 7.5. Les différents niveaux de partage d'un composant d'Arch dans le modèle CoPAC. Entre le niveaux découplé et le niveau commun, on trouve un continuum de possibilités de couplage.

Un composant du modèle est mis en *commun* lorsqu'il est partagé par plusieurs utilisateurs. C'est par exemple le cas du noyau fonctionnel pour les éditeurs partagés : tous les utilisateurs manipulent le même document et le noyau fonctionnel qui gère ce

document est alors partagé. On retrouve ici l'approche d'ALV [Hill 1992]. Cependant, la granularité plus fine des composants de Arch permet une plus grande souplesse. L'adaptateur du noyau fonctionnel (ANF) peut être chargé de gérer différentes politiques d'accès au document en fonction de différents rôles. Par exemple, si l'on doit implémenter deux rôles, lecteur et rédacteur, on utilisera deux composants ANF, chacun implémentant l'un des rôles. Tous les utilisateurs ayant un rôle de lecteur partageront l'ANF correspondant. Les utilisateurs rédacteurs partageront l'ANF rédacteur. Les rôles doivent aussi être pris en compte au niveau du contrôleur de dialogue mais celui-ci peut être mis en commun entre différents rôles. Cette configuration permet en particulier la modification dynamique des rôles en cours de session par la reconfiguration de la partie de l'arche contenant l'ANF et la modification des connexions entre composants, à l'initiative du contrôleur de dialogue. Cette possibilité permet de satisfaire la propriété de liaison dynamique entre le noyau fonctionnel et l'interface énoncée par [Dewan 1992]. La stratégie temporelle, que nous avons définie au paragraphe 7.2.2, doit être ici considérée. Elle définit la façon dont l'ANF fait communiquer le noyau fonctionnel et le contrôleur de dialogue. Lorsque l'ANF est mis en commun entre plusieurs utilisateurs, la stratégie temporelle choisie a une influence sur la façon dont les utilisateurs peuvent accéder au noyau fonctionnel. Le mode de communication synchrone impose des accès séquentiels au noyau fonctionnel, tandis qu'un mode asynchrone permet des accès parallèles. Cet aspect doit être examiné en liaison avec les compétences du noyau fonctionnel, en particulier afin d'éviter le classique problème de l'interblocage. Si le noyau fonctionnel inclut des mécanismes de contrôle d'accès, l'ANF peut utiliser un mode de communication asynchrone. Dans le cas contraire, par exemple lorsque l'on veut réutiliser le noyau fonctionnel d'un système mono-utilisateur, la délégation sémantique permet de prendre en compte le contrôle d'accès dans l'ANF.

A l'autre extrême de l'arche, un composant comme le composant interaction de bas niveau (CIBN) peut aussi être partagé : dans le mode souris partagée de Timbuktu par exemple, les utilisateurs partagent le même pointeur. La mise en commun du CIBN est un moyen simple d'assurer le partage des dispositifs physiques. La mise en commun d'éléments d'interaction, tels des formulaires ou des formes graphiques peut être réalisé en mettant en commun le composant techniques de présentation (CTP). Dans ce cas cependant, on pourra souhaiter affiner à un niveau de détail plus précis. Un modèle tel que SLICE [Karsenty 1994] fournit une décomposition plus fine que le couple CIBN/CTP et pourrait être utilisé pour structurer plus précisément cette partie de l'arche. La conception logicielle des applications multi-utilisateurs permettant la manipulation directe et le partage fin d'éléments d'interaction bénéficierait particulièrement de la combinaison de SLICE et CoPAC.

Il faut remarquer que lorsqu'un composant est mis en commun dans l'architecture conceptuelle, cela ne se traduit pas nécessairement par un composant unique dans l'implémentation. Bien que l'existence d'un composant unique dans l'implémentation soit justifiée pour le noyau fonctionnel (comme dans le modèle centralisé), une telle approche risque de poser des problèmes de performance pour le CIBN. Dans l'implémentation, le CIBN peut en fait être répliqué et la synchronisation entre les différents CIBN peut être assurée par le partage d'états entre les CIBN. Cette approche ressemble au modèle à états partagés de Patterson que nous avons présenté au chapitre précédent.

Les composants d'un niveau de l'arche sont *couplés* s'ils doivent maintenir une forme de cohérence entre eux. La cohérence requise ici est plus lâche que dans les cas précédents et ne nécessite pas une mise en commun comme précédemment. Un bon exemple en est la rétroaction de groupe, lorsque les actions d'un utilisateur sur un élément d'interaction doivent être répercutées vers les autres utilisateurs. Par exemple, dans l'éditeur partagé SASSE [Baecker 1992], chaque utilisateur peut suivre la position des autres utilisateurs dans le document partagé grâce à un ensemble de scrollbars. Chacune des scrollbars reflète la position d'un autre utilisateur et chaque utilisateur ne peut déplacer que sa propre scrollbar. Si l'effet de la manipulation des scrollbars est partagé, la manipulation des scrollbars ne l'est pas. Les scrollbars ne sont partagées que pour l'affichage et pas pour la manipulation. Il s'agit de ce que nous appelons cohérence lâche d'un élément d'interaction. Dans ce cas, il n'est pas nécessaire de mettre en commun le composant CTP mais les différents composants CTP doivent quand même respecter un couplage. La modification de la position d'une scrollbar implique la répercussion de cette modification sur les "images" de cette scrollbar chez les autres utilisateurs. Dans ce cas, la cohérence lâche est assurée par l'intermédiaire de l'agent PAC du contrôleur de dialogue en charge de la gestion de la scrollbar. Nous verrons au paragraphe suivant comment cet agent PAC répercute son changement d'état vers ses "cousins", c'est-à-dire les agents PAC des autres utilisateurs qui gèrent l'affichage de l'image de la scrollbar concernée. Notons qu'ici encore, il s'agit d'une synchronisation par partage d'état.

Enfin, les composants de différents utilisateurs peuvent être *découplés*. C'est le cas lorsqu'il n'y a aucun partage et que les composants sont indépendants. Les composants sont répliqués et ne communiquent pas. Par exemple dans un éditeur graphique partagé, le document est géré par un noyau fonctionnel commun comme expliqué plus haut, mais les composants CIBN sont découplés : chaque utilisateur a une copie locale du CIBN.

Par rapport à notre espace-problème des modèles d'architecture, ce mécanisme de partage permet de considérer la dimension distribution et en partie la dimension responsabilité. Le partage permet en effet de réfléchir à la réplication des composants et aux éventuels

mécanismes de synchronisation qu'ils requièrent. Toutefois la granularité du partage pourra être modulée suivant les possibilités du système d'exploitation utilisé. En particulier un système d'exploitation à objets permettant un partage fin nécessitera d'affiner les parties des composants qui doivent être partagées. La dimension responsabilité considérée au niveau de granularité des utilisateurs correspond à la notion de rôle. Dans ce cas, le partage ou le découplage de composants et en particulier de l'adaptateur du noyau fonctionnel, comme nous l'avons vu, permet d'implémenter différents rôles.

Les stratégies de partage et de réplication de composants exposées dans ce paragraphe nous permettent aussi de répondre à la règle ❶ sur la factorisation que nous avons énoncée au paragraphe 6.3 du chapitre 6. Le modèle CoPAC permet d'identifier les composants répliqués afin de factoriser la réalisation.

Nous allons maintenant détailler la façon dont le modèle CoPAC permet à différents composants sur différents sites de communiquer et la façon dont il intègre les services de communication homme-homme médiatisée.

7.3.2. La communication dans CoPAC

Deux aspects de la communication importants pour les systèmes multi-utilisateurs sont pris en compte par le modèle CoPAC :

- la communication entre composants, que ceux-ci soient mis en commun et répliqués, ou qu'ils doivent maintenir une cohérence lâche. En général, ces composants communiquent en synchronisant leurs états.

- La facette communication du modèle du trèfle du chapitre 1, c'est-à-dire la communication homme-homme médiatisée. Dans ce cas, l'architecture doit pouvoir intégrer des flots de données correspondant aux médias continus comme le son et la vidéo.

7.3.2.1. Communication entre composants

En règle générale dans CoPAC, la communication entre composants s'effectue par l'intermédiaire du contrôleur du dialogue (CD). En effet, le CD, constitué d'une hiérarchie d'agents PAC, est le seul composant de l'architecture dans lequel on peut identifier précisément l'élément qui doit communiquer : c'est un agent PAC et nous verrons que nous distinguons dans ce cas une nouvelle facette dans un agent PAC pour gérer la communication. Cette règle permet de localiser précisément les éléments communicants et permet de conserver la modularité du modèle PAC-Amodeus. Pour

prendre en compte la coordination entre les rôles, un contrôleur de dialogue peut être privilégié. Dans ce cas, les autres contrôleurs lui sont subordonnés. Nous en verrons un exemple avec la mise en œuvre du jeu $(421)^n$. La partie du contrôleur de dialogue qui gère la coordination entre les rôles peut aussi être isolée dans un "super-contrôleur de dialogue" qui contrôle tous les autres contrôleurs de dialogue. Nous détaillerons l'exemple du super-contrôleur de dialogue de NEIMO au paragraphe 7.4.3.

Cependant, dans certains cas et principalement pour des raisons de performance (c'est là la règle de "validité vis-à-vis du monde réel" qui joue), des composants de l'arche de CoPAC peuvent communiquer directement entre eux sans passer par le contrôleur de dialogue. C'est le cas par exemple lorsque le CIBN est mis en commun comme dans Timbuktu. Dans les autres cas, la communication passe par un agent PAC et nous distinguons alors explicitement dans l'agent une facette communication, COM, comme le montre la figure 7.6. Lorsque la communication est établie directement entre composants, le facette COM joue un rôle légèrement différent que nous expliquons plus loin.

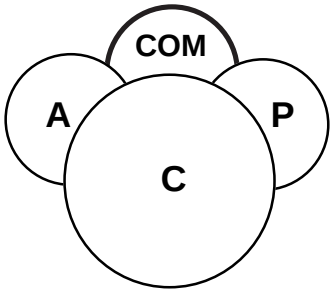


Figure 7.6. Les agents du modèle CoPAC. La facette communication (COM) rajoutée à l'agent PAC a la charge de la communication avec les agents d'autres sites.

La nouvelle facette, COM, est exactement sur le même plan que les facettes A et P. De même que la facette A d'un agent PAC communique avec l'ANF du modèle PAC-Amodeus et la facette P est en relation avec le CTP, la facette COM est en relation avec un nouveau composant spécialisé dans la communication (figure 7.7). De même que le contrôle (C) assure la communication entre les facettes A et P, c'est encore le contrôle qui a la charge des échanges entre COM et les facettes A et P. En règle générale, une modification de la facette P (respectivement A) sera répercutée par C à la fois vers A (respectivement P) et COM. L'arrivée d'un message en provenance d'un autre site sur la facette COM sera répercutée à la fois vers A et P par le contrôle.

Le modèle CoPAC ne détaille pas le contenu du Composant Communication avec lequel la facette COM est en relation. Ceci permet de préserver la diversité des mécanismes de communication existants. Toutefois, il existe quelques règles simples pour structurer le Composant Communication. Avec la plupart des systèmes d'exploitation, le Composant

Communication s'appuiera sur les services de communication fournis par le système (par exemple, les sockets ou les Apple Events [Apple 1991]). Si nécessaire, on pourra envisager une structuration classique en couches comme le modèle OSI pour organiser le Composant Communication. Certains mécanismes de communication inter-applications nécessitent toutefois une approche particulière. C'est par exemple le cas du protocole de communication des Apple Events : en effet, les messages inter-applications envoyés sous forme d'Apple Events sont reçus par la queue d'événements de l'application, c'est-à-dire dans le CIBN. Dans ce cas particulier, le Composant Communication communique avec le CIBN. Nous verrons des exemples de cette mise en œuvre au paragraphe 7.4.

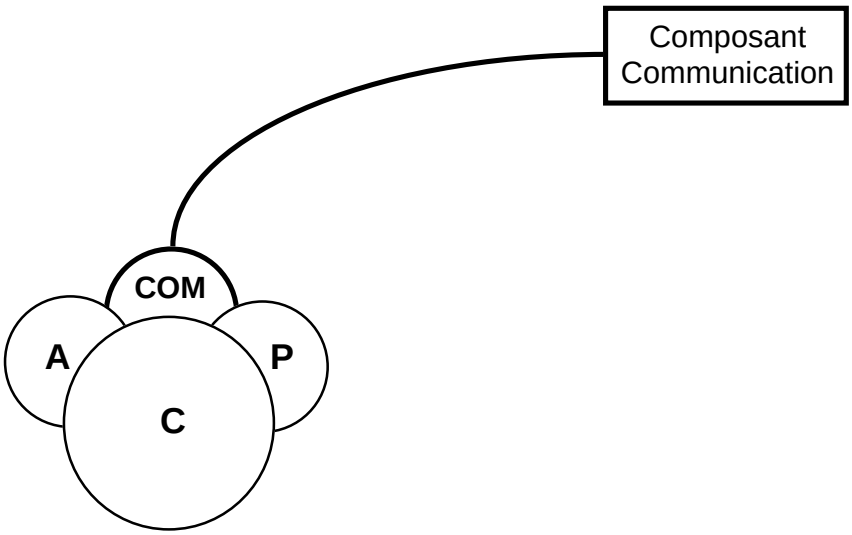


Figure 7.7. La facette COM de l'agent PAC communique avec les autres sites via le Composant Communication.

Notons que l'idée de rajouter des facettes spécialisées à un agent PAC n'est pas une idée nouvelle. Cette approche permet de détailler la structure d'un agent PAC et participe au respect des propriétés de modifiabilité et de réutilisabilité. Par exemple, [Ouadou 1994] avec le modèle AMF propose d'affiner les agents PAC dans le cas d'une application mono-utilisateur en rajoutant des facettes gérant l'aide à l'utilisateur ou les erreurs. La particularité de la facette COM que nous proposons est qu'elle réalise un service qui ne peut, au contraire de l'aide ou des erreurs, être pris en charge par les autres facettes. Ainsi la facette COM ne précise pas seulement la structure de l'agent PAC, mais elle lui rajoute une compétence qui n'était pas requise pour les systèmes mono-utilisateurs.

7.3.2.2. Communication homme-homme médiatisée

Le second type de communication modélisé par CoPAC correspond à la communication homme-homme médiatisée via des médias continus comme l'audio ou la vidéo. Cette communication impose de considérer un nouveau type de données : les *flots*. Les flots sont particuliers à plusieurs titres : ils sont produits de façon constante par une *source*

(par exemple une caméra vidéo) et sont consommés par un *puits* (par exemple une région d'affichage sur un écran). Leur nature continue pose problème dans une architecture multi-agents : en effet, les agents savent bien manipuler des informations discrètes, sous formes de messages ou d'événements. Mais ils ne peuvent manipuler des données continues sans les tronçonner. Pour assurer la propriété de régularité des flots, il est nécessaire d'envisager des communications directes entres sources et puits, comme nous l'avons vu au chapitre précédent avec le modèle Gemma. Ces communications directes court-circuitent les agents et peuvent donc être établies directement entre des composants de l'architecture, typiquement les Composants Techniques de Présentation. Mais la mise en relation a toujours lieu à l'initiative d'un Contrôleur de Dialogue d'un site partie prenante de la communication. Plus précisément, un agent du Contrôleur de Dialogue, via sa facette COM, peut ouvrir et fermer une communication et gérer l'aspect contrôle du flot de médias continus. Cet aspect contrôle inclut en particulier la gestion de la qualité de service (QoS) du réseau utilisé. Via sa facette abstraction et en faisant appel au noyau fonctionnel, l'agent peut aussi effectuer des traitements sur les médias continus reçus ou envoyés. Un traitement peut être par exemple un retournement d'image pour satisfaire la propriété de réversibilité vidéo. Nous verrons un exemple de mise en œuvre au paragraphe 7.4.2 avec le mediaspace VideoPort. Notons enfin que pour tenir compte de la diversité des outils de communication existants, on peut observer un phénomène analogue au "slinky" du modèle Arch : suivant la plate-forme d'accueil, l'importance relative du Composant Communication et du Composant Techniques de Présentation doit être modulée. Par exemple, pour utiliser un outil de communication qui fait circuler une partie des informations de contrôle dans le flot de médias continus, il convient de donner une compétence accrue au Composant Techniques de Présentation au détriment du Composant Communication.

En résumé, la communication dans le modèle CoPAC est prise en compte de deux façons. La première repose sur la facette COM qui communique avec le Composant Communication. Cette première possibilité est adaptée à l'échange de messages entre agents situés sur différents sites. La communication proprement dite s'appuie sur les compétences du Composant Communication qui reposent sur les possibilités de la plate-forme d'accueil. Le deuxième type de communication est adapté aux médias continus utilisés principalement dans la communication homme-homme médiatisée. Dans ce cas, la communication distingue le transfert d'informations de contrôle et le transfert des médias continus. Le transfert des flots de médias est réalisé par le Composant Techniques de Présentation sous le contrôle d'un agent. Les informations de contrôle sont échangées entre les agents via la facette COM et le Composant Communication.

7.3.3. Évaluation du modèle CoPAC

Après la présentation du modèle d'architecture CoPAC, il nous faut maintenant l'évaluer vis-à-vis de notre espace problème et des caractéristiques attendues d'une architecture que nous avons présentées au chapitre précédent.

L'espace problème que nous avons proposé comporte cinq dimensions : services, responsabilité, niveaux d'abstraction, parallélisme, distribution. En ce qui concerne les services, le modèle CoPAC les considère au niveau conceptuel, tels que définis par le modèle du trèfle du chapitre 1. L'espace de production est à la charge du noyau fonctionnel. L'espace de coordination est géré principalement par le contrôleur de dialogue. La communication est prise en compte par le composant communication. Nous avons déjà remarqué au chapitre 6 ces équivalences entre les composantes de l'espace fonctionnel du trèfle et les composants du modèle Arch pour la production et la coordination. Avec CoPAC, nous avons montré que cette équivalence pouvait être étendue aux systèmes multi-utilisateurs, et qu'un nouveau composant chargé de la communication permet de couvrir la facette communication du trèfle. Nous verrons plus précisément dans les exemples de mise en œuvre comment CoPAC prend en compte chacune des composantes de notre espace fonctionnel. Pour la dimension responsabilité, CoPAC bénéficie de l'apport de Arch pour la délégation entre composants fonctionnels. Nous avons notamment mentionné la délégation sémantique au niveau de l'adaptateur du noyau fonctionnel. Nous avons aussi vu que l'adaptateur du noyau fonctionnel permet d'implémenter différents rôles et que la stratégie de partage contribue également à la dimension responsabilité. En ce qui concerne les niveaux d'abstraction, là encore, CoPAC exploite les niveaux d'abstraction présents dans Arch. Cependant, CoPAC ne propose pas de structuration en niveaux d'abstraction pour le composant communication, à part la séparation entre communication de contrôle et communication de médias continus. Mais il est possible d'utiliser un modèle classique comme le modèle OSI pour organiser ce composant. Pour la dimension parallélisme, deux aspects interviennent : la structuration du contrôleur de dialogue en une hiérarchie d'agents PAC fonctionnant de façon indépendante et les stratégies temporelles que nous avons expliquées en 7.2.2 et 7.3.1 permettent toutes deux de considérer la dimension parallélisme. Enfin nous avons vu que la stratégie de partage permet de réfléchir à la dimension distribution.

La satisfaction des caractéristiques attendues d'une architecture est également assurée, à une nuance près. Les quatre premières règles, la modifiabilité, la réduction de la complexité, la décomposition du travail et la règle de validité vis-à-vis du monde réel sont satisfaites principalement grâce au modèle PAC-Amodeus, sous-jacent dans CoPAC. La règle de validité vis-à-vis du monde réel a fait l'objet d'une attention particulière pour le

composant communication. Elle a en particulier inspiré la séparation entre messages de contrôle et médias continus. Elle a aussi conduit à ne pas préconiser une structuration du composant communication pour tenir compte de la diversité des outils de communication. Les deux autres règles que nous nous sommes fixées concernent la factorisation de la réalisation des composants et la prise en compte du facteur d'échelle. Nous avons vu au paragraphe 7.3.1 que la première règle est vérifiée. En revanche, il nous faut être prudents avec la règle de facteur d'échelle. Nos expérimentations avec le modèle n'ont concerné jusqu'à présent que des petits groupes d'utilisateurs (4-5 personnes) et nous manquons ici d'expériences à plus grande échelle.

En résumé, CoPAC couvre de façon satisfaisante les dimensions de notre espace problème. On note en particulier qu'il intègre de façon équilibrée les trois facettes de notre espace fonctionnel du trèfle. Pour la satisfaction des caractéristiques attendues d'une architecture, CoPAC bénéficie de l'apport du modèle PAC-Amodeus et garantit aussi la possibilité de factoriser la réalisation. En revanche, nous manquons d'éléments pour apprécier sa validité au-delà d'un petit groupe d'utilisateurs.

7.4. Exemples de mise en œuvre de CoPAC

Nous présentons maintenant trois exemples de mise en œuvre du modèle d'architecture CoPAC. Le premier exemple, un jeu multi-utilisateur, met l'accent sur la synchronisation. Le second, le mediaspace numérique VideoPort, privilégie la facette communication. Enfin le troisième exemple d'application, la plate-forme d'utilisabilité NEIMO que nous avons présentée au chapitre 4, est plus orientée vers la production, mais inclut aussi des composantes synchronisation et communication importantes. Tous ces exemples ont été réalisés sur Macintosh et nous verrons comment nous avons tenu compte des particularités de cet environnement.

7.4.1. $(421)^n$, un jeu de 421 multi-utilisateur

Les jeux sont traditionnellement un terrain privilégié pour les systèmes multi-utilisateurs. Non seulement ils sont un exemple d'application pertinent pour ces systèmes car ils permettent de sortir du classique paradigme du "jouer contre l'ordinateur", mais ils constituent un mode d'interaction sociale particulier. Un jeu repose en effet sur des règles écrites et connues par tous les joueurs, même s'il accepte des variantes. Le contrôle des interactions entre les joueurs est ainsi clairement défini et est approprié à une mise en œuvre informatique. La facette coordination du modèle du trèfle est privilégiée. Notons toutefois que la mise en œuvre d'un jeu sous forme informatique conduit souvent à perdre un élément social important : tricher n'est plus possible. Par exemple, dans notre jeu de 421, il n'est pas possible de passer le tour de quelqu'un "par inadvertance" ou de

modifier subrepticement la valeur d'un dé. On retrouve ici une caractéristique du contrôle technique par rapport au contrôle social : une plus grande rigidité et l'impossibilité de transgresser les règles intégrées au système.

Le jeu de 421 est un classique des bistros français et se joue avec trois dés et un nombre illimité de joueurs. Le but est de faire sortir des combinaisons de dés, la plus recherchée étant 421 (un dé sur 4, un autre sur 2, le dernier sur 1). Chaque joueur joue à tour de rôle. Les dés sont lancés tous en même temps au premier lancer, le joueur a droit à deux lancers supplémentaires et peut reprendre tout ou partie des dés jetés à chaque lancer. L'interface de notre réalisation informatique du jeu de 421 est présentée figure 7.8.

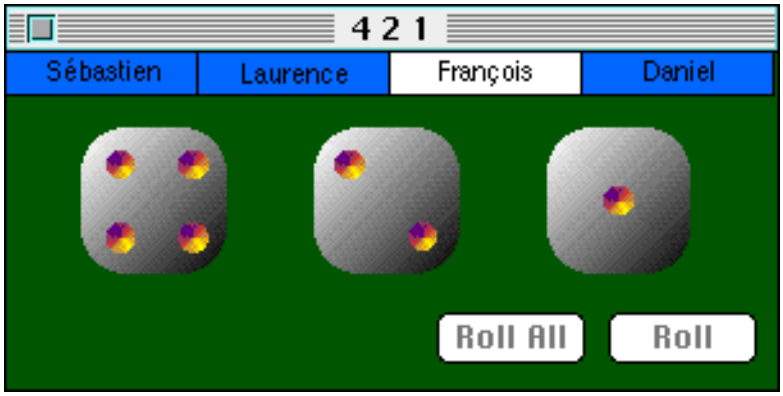


Figure 7.8. L'interface de $(421)^n$. Les trois dés viennent d'être lancés et sont affichés au centre de la fenêtre. Les noms des joueurs sont affichés en haut et le nom du joueur qui a la main (ici, François), est mis en évidence. Les boutons en bas à droite permettent de lancer tous les dés ou uniquement les dés sélectionnés. Lorsqu'un autre joueur a la main (comme ici), ces boutons sont grisés.

L'architecture choisie privilégie la réplication de composants. Deux rôles sont identifiés : un joueur est l'hôte, les autres joueurs sont les invités. L'architecture correspondante est présentée figure 7.9.

Le noyau fonctionnel (NF) est répliqué pour tous les joueurs : il maintient la liste des joueurs. Cette liste est affichée sur tous les sites dans le bandeau visible sur la figure 7.8. Lorsqu'un nouveau joueur se présente, il établit une connexion avec le joueur hôte et l'état du NF hôte est transmis au NF invité. Les rôles sont distingués grâce à deux ANF différents. L'ANF hôte permet la modification de la liste des joueurs, l'ANF invité ne permet l'accès à la liste des joueurs qu'en mode lecture. Chaque modification de l'état du NF hôte est répercuté vers les NF invités. Ce sont les contrôleurs (eux aussi répliqués) qui gèrent la coordination des différents joueurs. Le joueur dont c'est le tour de jouer est marqué par le NF hôte dans la liste des joueurs. Lorsqu'un NF voit son état modifié, il transmet son changement au contrôleur de dialogue qui vérifie si c'est le tour de son site. Dans ce cas, il active les boutons de jeu et donne la main au joueur.

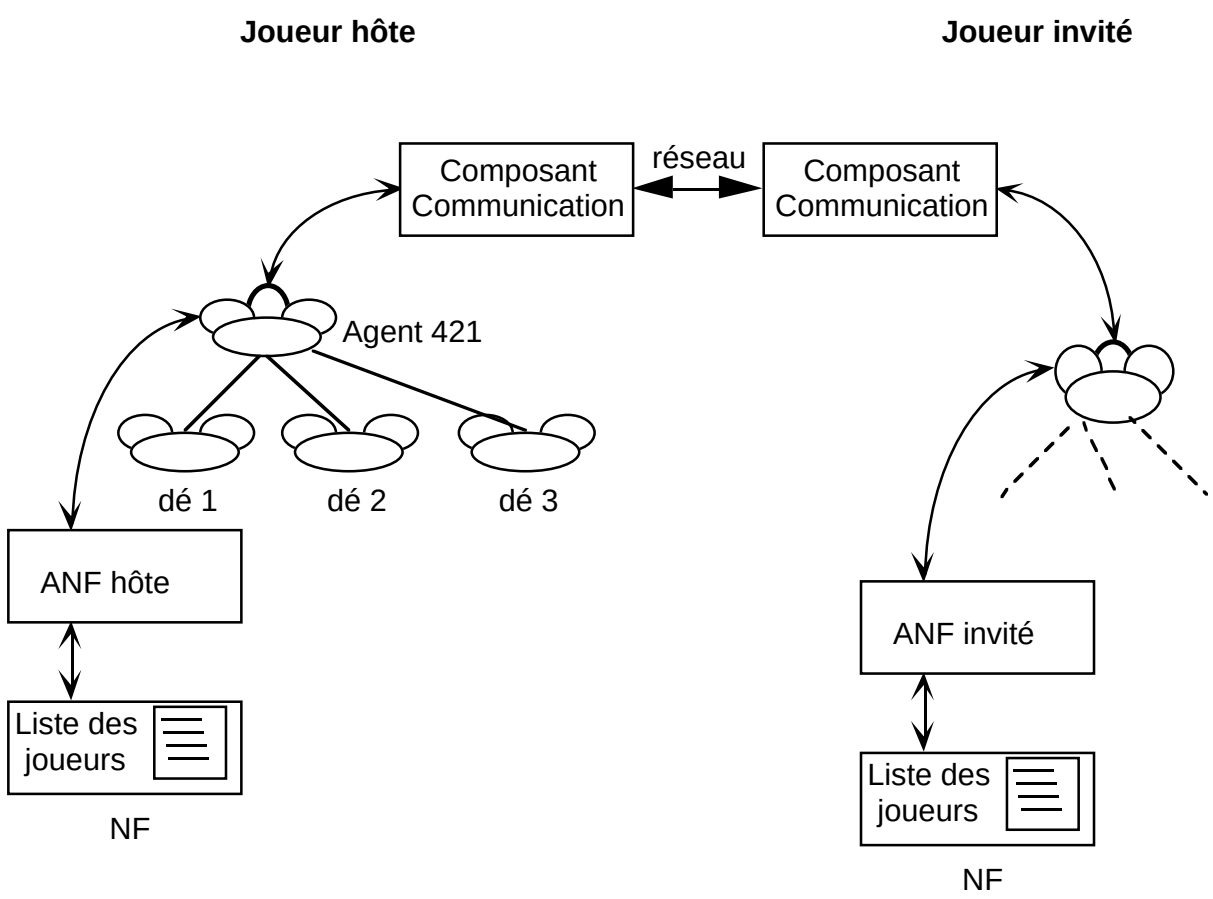


Figure 7.9. Architecture logicielle du jeu $(421)^n$. Deux sites sont représentés. L'un des joueurs (à gauche) joue le rôle d'hôte, les autres celui d'invités.

Cet exemple montre que l'on peut assurer la coordination avec le modèle CoPAC à l'aide du contrôleur de dialogue. Il montre aussi que les ANF permettent d'implémenter simplement différents rôles avec le même NF.

7.4.2. VideoPort, un mediaspace numérique

VideoPort est notre mediaspace que nous avons déjà mentionné au chapitre 2. Cette réalisation nous a permis d'étudier la mise en œuvre de la communication de médias continus dans le modèle CoPAC. L'interface utilisateur est très simple : les connexions vidéo sont affichées dans une fenêtre, un menu permet d'ouvrir et de fermer une connexion et de modifier des réglages comme le choix du son joué lors de la réception d'un appel (figure 7.10). Dans les premières versions dont nous présentons ici l'architecture, le seul média de communication implémenté est la vidéo. En effet, le son est plus exigeant en termes de propriété de régularité des flots et la première version qui ne traitait pas explicitement les médias continus n'était pas à même de gérer le son. Le son est maintenant en cours d'intégration dans la seconde version dont nous présentons aussi

l'architecture, grâce à la bibliothèque UserLink de gestion des médias continus (présentée au chapitre 8).

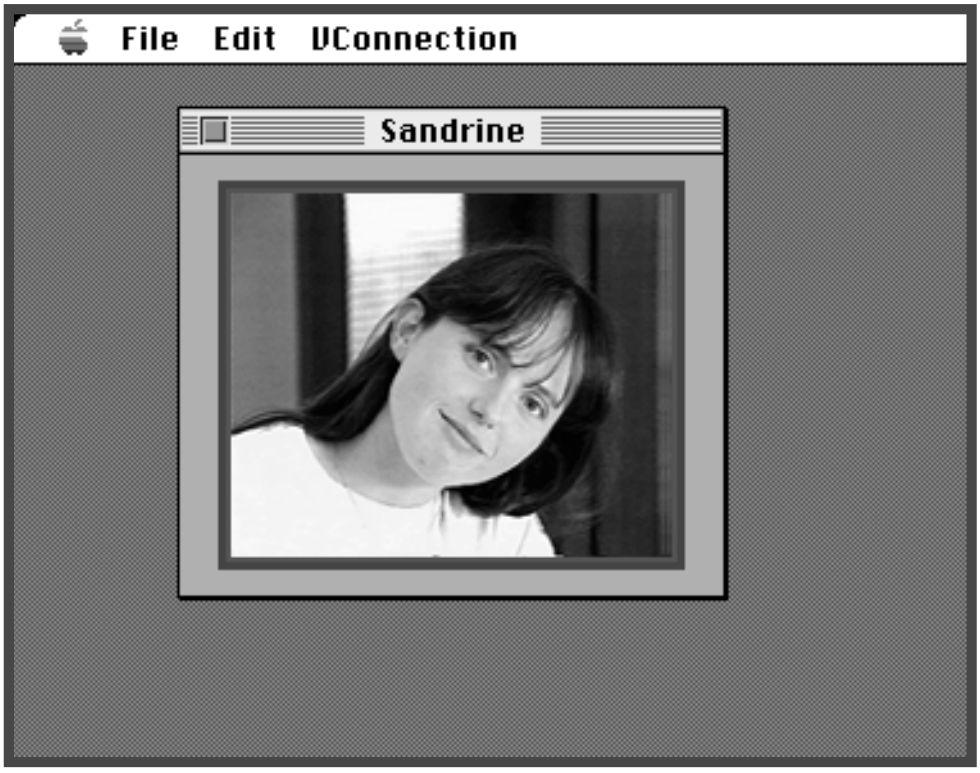


Figure 7.10. Copie d'écran de l'interface de VideoPort.

Deux versions successives de VideoPort ont été réalisées, toutes deux conservant la même interface utilisateur. La première version utilise comme mécanisme de communication les Apple Events. Il s'agit d'une communication reposant sur des messages qui sont indifféremment des messages de contrôle ou qui contiennent des données comme par exemple des images.

La figure 7.11 montre une vue partielle de l'architecture de deux applications VideoPort en cours de communication. Pour alléger le schéma, nous avons conservé de l'architecture uniquement les composants significatifs : le composant communication, le composant techniques de présentation (CTP, représenté pour un seul des deux sites) et, à l'intérieur du contrôleur de dialogue, les agents VP qui gèrent les fenêtres comme celle montrée figure 7.10.

L'acquisition vidéo se fait au niveau du CIBN. Dans cette première version de VideoPort, nous ne transmettons pas un flot vidéo, mais juste des images successives. Dans notre cas, nous utilisons en fait la boîte à outils QuickTime qui fournit une interface standardisée et qui permet de s'affranchir des particularités du dispositif d'acquisition

vidéo. Nous considérons donc que QuickTime nous fournit une surcouche garantissant l'indépendance vis-à-vis du matériel vidéo et l'acquisition se fait donc au niveau du CTP et non du CIBN. Notons qu'il s'agit là d'un cas typique d'application du métamodèle "slinky" de Arch, tenant compte des particularités de la plate-forme d'accueil. La facette P de l'agent VP (pour VideoPort) acquiert l'image capturée et la transmet, via le contrôleur de dialogue, à la facette COM. Si l'on voulait afficher cette image "en local", par exemple pour fournir une vue miroir (au sens du chapitre 4), la facette P ferait appel au CTP pour l'affichage de l'image dans une fenêtre.

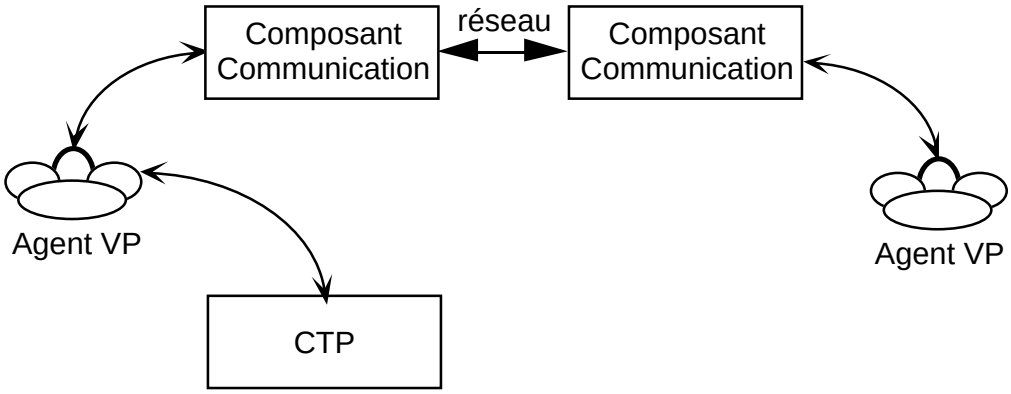


Figure 7.11. Architecture logicielle de VideoPort version Apple Events. Chaque site dispose d'un agent VP. Un agent VP gère une fenêtre comme celle de la figure 7.10.

La facette COM transmet l'image au composant communication qui se charge de l'encapsuler dans un Apple Event. Le composant communication se charge ensuite de l'envoi de l'Apple Event à travers le réseau vers l'autre site VideoPort.

Le mécanisme de réception est analogue. Un Apple Event est reçu par le CIBN (non représenté pour ne pas alourdir le dessin), comme déjà expliqué pour l'exemple de $(421)^n$. Il est immédiatement transmis au composant communication qui en extrait l'information pertinente, c'est-à-dire l'image reçue, et la transmet à la facette COM de l'agent VP. Celle-ci est alors transmise à la facette P par le contrôle et P la transmet au CTP pour affichage.

Nous n'avons pas parlé du rôle de la facette A de l'agent VP. Son rôle est en fait extrêmement réduit : elle se contente de conserver le titre de la fenêtre, c'est-à-dire le nom du correspondant. Mais elle pourrait être le lieu de traitements sur l'image, par exemple un retournement pour garantir la propriété de réversibilité vidéo.

La deuxième version de VideoPort vise à intégrer les médias continus. En effet nous avons vu que la première version ne gère pas des flots vidéo, mais juste des images

successives et qui sont traitées séquentiellement. La gestion des flots vidéo a été rendue possible par l'utilisation de la bibliothèque de communication UserLink décrite au chapitre suivant. Cette évolution de VideoPort a été aussi l'occasion de vérifier les propriétés de l'architecture en suivant les recommandations de la méthode SAAM (Software Architecture Analysis Method) [Kazman 1994]. Cette méthode recommande notamment de choisir une tâche concrète et d'éprouver les propriétés de l'architecture dans le cadre de la réalisation de cette tâche. Cette tâche est typiquement une modification ou une extension du logiciel. Dans notre cas, il s'agissait de modifier le moyen de communication utilisé dans VideoPort : de l'envoi d'images successives par Apple Events, nous voulions passer aux médias continus fournis par la bibliothèque UserLink. La propriété de modifiabilité a donc pu être étudiée.

L'architecture de la deuxième version de VideoPort est présentée figure 7.12.

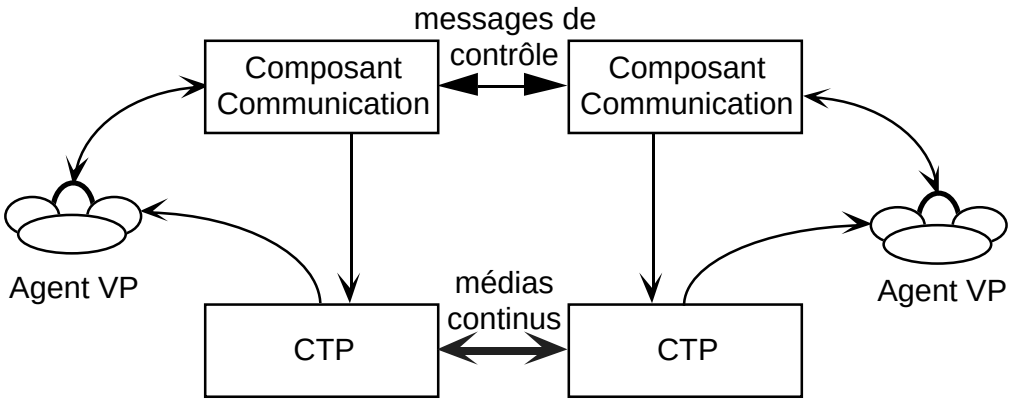


Figure 7.12. Architecture logicielle de VideoPort version UserLink. la communication a désormais lieu à deux niveaux dans l'architecture : les messages de contrôle au niveau du composant communication, les médias continus au niveau du CTP.

UserLink prend en charge plusieurs services qui étaient réalisés par l'architecture dans la première version : la communication, mais aussi l'acquisition vidéo et l'affichage. Le rôle de l'agent VP se voit donc réduit. En fait, dans cette deuxième version, son rôle consiste surtout à faire appel à UserLink pour contrôler le flot vidéo. Avec UserLink nous avons introduit une couche supplémentaire qui nous masque la plupart des services. Lors de l'établissement d'une connexion, l'agent VP contacte le composant communication, c'est-à-dire la partie contrôle de UserLink. UserLink établit ensuite la connexion entre les deux sites et instancie le flot vidéo entre les CTP. UserLink se charge aussi de l'acquisition du flot et de son affichage via le CTP.

Les modifications à réaliser ont pu être localisées précisément. Sur le schéma d'architecture de la figure 7.12, on note les modifications suivantes :

- la facette P d'un agent VP se contente de recevoir des informations depuis le CTP. En effet, l'affichage est géré par UserLink. Le lien du CTP vers l'agent VP ne fait que recopier l'affichage réalisé par UserLink vers la facette P de l'agent. Cette façon de faire permet de prévoir d'éventuels traitements qui devront être réalisés par l'agent, par exemple la réversibilité vidéo.

- Le composant communication contrôle directement le CTP. Ce contrôle a lieu au sein de UserLink.

En résumé, les modifications introduites par l'utilisation de UserLink ont été au nombre de cinq :

- suppression du code d'aiguillage des Apple Events dans le CIBN,

- suppression du code d'acquisition vidéo dans le CTP et des appels correspondants dans la facette P de l'agent VP,

- suppression du code d'affichage dans le CTP et des appels correspondants dans la facette P de l'agent VP,

- modification de l'établissement et de la rupture des connexions dans la facette COM de l'agent VP. Cette modification a été la plus importante à cause du changement du mode d'adressage des sites entre les Apple Events et UserLink qui utilise les adresses IP,

- modification du contrôle de dialogue. Il est maintenant réduit au contrôle de la facette COM.

L'architecture nous a ainsi permis de localiser précisément les modules à modifier. On peut donc considérer que la propriété de modifiabilité est vérifiée, au moins pour la partie communication du modèle CoPAC.

La mise en œuvre des deux versions successives de VideoPort nous a permis d'expérimenter divers moyens de communication pour la vidéo et d'éprouver le modèle CoPAC pour la communication. Il s'est révélé adapté aussi bien à la prise en compte de succession d'images que de flots vidéo en utilisant la bibliothèque spécialisée UserLink. De plus, l'évolution de VideoPort a permis de vérifier la propriété de modifiabilité de l'architecture prônée par le modèle.

7.4.3. NEIMO, notre plate-forme d'observation du comportement des utilisateurs

NEIMO est la plate-forme d'étude de l'utilisabilité présentée au chapitre 5. L'architecture globale de NEIMO est complexe et nous n'en donnerons qu'un aperçu en faisant ressortir les points intéressants dans la mise en œuvre de CoPAC.

L'architecture de NEIMO repose sur un serveur central, NeimoCom. La raison de ce choix est que les participants à une session d'expérimentation contribuent tous à la production d'un unique fichier historique de la session. D'autre part, tous les enregistrements de ce fichier sont datés. La nécessité d'avoir une horloge unique fiable a imposé la solution centralisée. Notons que des problèmes de performances nous amènent maintenant à considérer une solution où la capture est répartie. Mais cette nouvelle solution étant en cours de réalisation, nous nous intéressons ici à la solution centralisée. Chacun des rôles identifié dans NEIMO dispose d'une application locale qui lui est propre et se connecte à NeimoCom via une bibliothèque adaptée à son rôle. Ces bibliothèques jouent le rôle d'ANF et NeimoCom joue le rôle du NF (figure 7.13). Le principal service de NeimoCom vu comme NF est la capture vers le fichier historique.

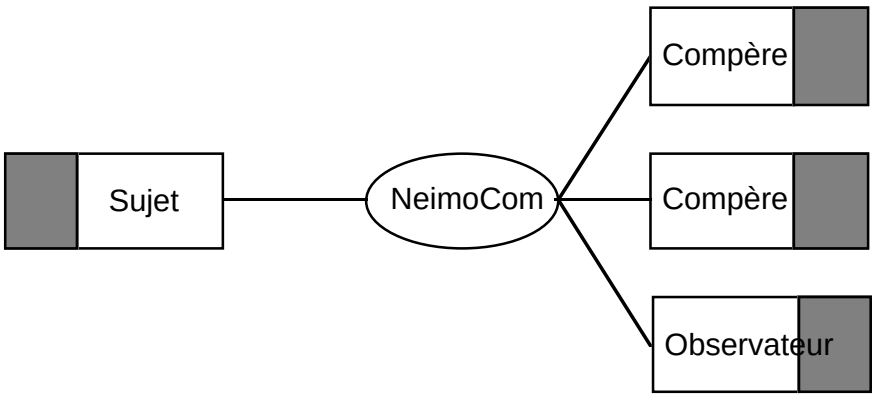


Figure 7.13. Architecture globale de NEIMO. Au centre le processus serveur central NeimoCom. A droite, trois applications : deux sont dédiées à un compère, l'autre à un observateur. A gauche, l'application du sujet. Les parties grisées représentent les interfaces et contrôleurs de dialogue locaux développés pour l'expérimentation considérée. Les rectangles blancs sont des bibliothèques d'accès à NeimoCom et jouent le rôle d'ANF vis-à-vis du NF NeimoCom.

Mais NeimoCom a aussi un rôle de contrôleur de dialogue global. A ce titre, cette architecture est proche de celle de Suite qui distingue aussi un contrôleur de dialogue global pour la coordination entre les rôles et des contrôleurs de dialogue locaux pour la gestion de l'interface utilisateur [Dewan 1992]. Ce contrôleur de dialogue est assez particulier : il n'établit pas une communication entre un noyau fonctionnel et une interface mais entre un noyau fonctionnel et un ensemble de contrôleurs de dialogue locaux, répartis sur différents sites. Il peut être vu comme un "super-contrôleur de dialogue"

auquel tous les contrôleurs de dialogue locaux sont subordonnés. La figure 7.14 détaille l'architecture du contrôleur de dialogue de NeimoCom pour un sujet et deux compères.

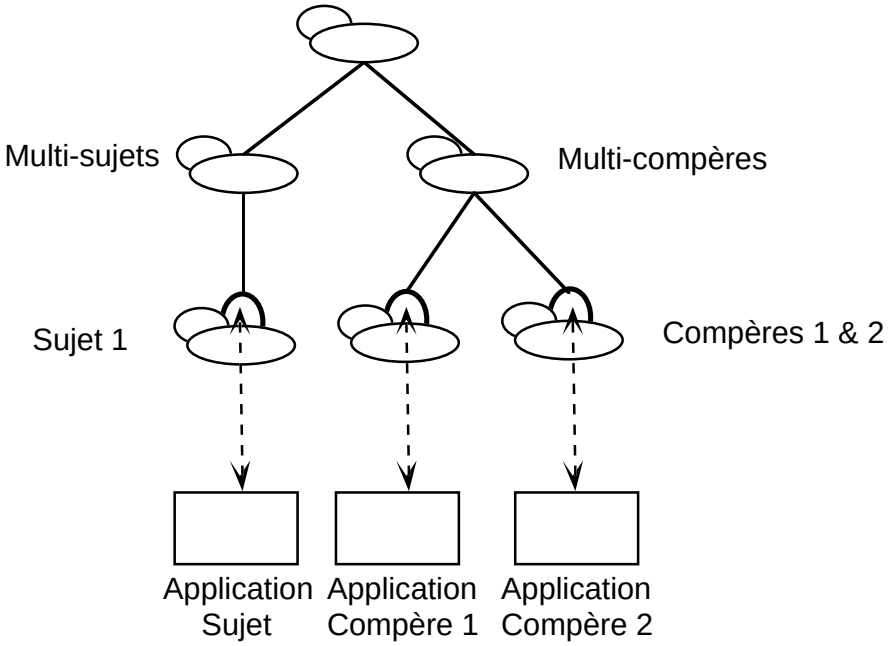


Figure 7.14. Architecture logicielle de NeimoCom. Les flèches pointillées indiquent des communications réseau. Sur la figure, seuls les agents du contrôleur de dialogue global sont représentés pour le cas d'un sujet et deux compères. Les agents multi-sujets et multi-compères gèrent l'enchaînement des tâches des sujets et des compères sous le contrôle de l'agent de plus haut niveau.

Les agents du super-contrôleur de dialogue (super-CD) ont une caractéristique remarquable : ils ne comportent pas de facette P. En effet, ils ne remplissent que des services de coordination, et l'interface de NEIMO ne permet pas d'agir sur la coordination. Celle-ci est "câblée" dans NeimoCom. Chaque sujet et chaque compère est représenté dans le super-CD par un agent qui communique avec le contrôleur de dialogue local du sujet ou du compère par sa facette COM. Les agents multi-sujets et multi-compères ainsi que l'agent de plus haut niveau gèrent la coordination entre les rôles des sujets et des compères. Par exemple, on peut considérer l'échange suivant : le sujet émet une commande qui nécessite une action de simulation de la part de chacun des compères (par exemple, un compère simule la reconnaissance d'une phrase dite par le sujet, et l'autre compère vérifie la cohérence de la réponse). L'agent multi-sujets reçoit la requête du sujet, et la transmet, via l'agent de plus haut niveau et l'agent multi-compères, aux deux compères. Le premier compère effectue l'action correspondant à la requête du sujet et sa réponse est transmise à l'agent compère 1, puis à l'agent compère 2 via le multi-compères. Le deuxième compère vérifie la cohérence de la réponse et la valide. L'agent compère 2 reçoit la validation et la transmet à l'agent multi-compères qui envoie alors la réponse à l'agent de plus haut niveau. Cette réponse est ensuite transmise au sujet.

Notons deux possibilités d'extension de l'exemple que nous avons présenté. L'agent multi-sujets permet potentiellement d'assurer la coordination entre plusieurs sujets, mais cette possibilité n'a pas encore été mise en œuvre. On peut envisager de fournir une interface pour contrôler la coordination : par exemple, le deuxième compère pourrait désactiver la validation. Dans ce cas, l'agent compère 2 aurait une facette P.

L'exemple de NeimoCom nous a permis de voir la conception d'un noyau fonctionnel centralisé et d'une coordination forte entre plusieurs rôles. Dans ce cas, un super-contrôleur de dialogue assure la coordination entre les rôles. En règle générale, les agents de ce super-CD sont regroupés sous un agent contrôle de plus haut niveau et les agents feuilles de la hiérarchie communiquent avec les contrôleurs de dialogue locaux via leur facette COM. Les agents du super-CD n'ont pas de facette P, sauf si l'interface permet d'agir sur la coordination.

7.5. Synthèse

Dans ce chapitre, nous avons présenté le modèle d'architecture CoPAC pour les systèmes multi-utilisateurs et nous avons détaillé des exemples de mise en œuvre. CoPAC repose sur le modèle PAC-Amodeus, lui-même composé de Arch et PAC. Il préserve les qualités de ces modèles quant à la structuration et la vérification de propriétés comme la modifiabilité, la portabilité et la réutilisabilité. Il répond aussi à l'espace problème des architectures des systèmes multi-utilisateurs et aux caractéristiques attendues d'une architecture, au moins pour des groupes de petite taille. CoPAC distingue plusieurs niveaux de partage des composants de Arch, et étend à la fois les agents PAC et le modèle Arch. Il rajoute aux agents PAC une facette COM pour gérer la communication et un composant communication au modèle Arch. L'ajout de ces nouveaux composants permet à CoPAC de prendre explicitement en compte les médias continus pour la communication homme-homme médiatisée.

Références

[Apple 1991] Apple. *The Apple Event Manager*, Apple Computer Inc., Cupertino, California, USA, 1991.

[Baecker 1992] R. M. Baecker, D. Nastos, L. R. Posner et K. L. Mawby. *The user-centred iterative design of collaborative writing software*, Workshop on Real Time Group Drawing and Writing Tools (CSCW'92, ACM Conference on Computer-Supported Cooperative Work), Toronto, Canada, 1992.

[Bass 1992] L. Bass, R. Little, R. Pellegrino, S. Reed, R. Seacord, S. Sheppard et M. R. Szczur. *The UIMS Tool Developers' Workshop: A Metamodel for the Runtime Architecture of an Interactive System*, in *SIGCHI Bulletin*, 24(1), janvier 1992. pp. 32-37.

[Coutaz 1987] J. Coutaz. *PAC, an Implementation Model for Dialog Design*, Interact'87, IFIP Conference on Human-Computer Interaction, Stuttgart, 1987. pp. 431-436.

[Dewan 1992] P. Dewan. *Principles of Designing Multi-User User Interfaces Development Environments*, IFIP TC2/WG2.7 Working Conference on Engineering for Human-Computer Interaction, Ellivuori, Finland, 1992. pp. 35-50.

[Hill 1992] R. D. Hill. *The Abstraction-Link-View Paradigm: Using Constraints to Connect User Interfaces to Applications*, ACM CHI'92 Conference on Human Factors in Computing Systems, Monterey, California, USA, 1992. pp. 335-342.

[Karsenty 1994] A. Karsenty. *GroupDesign : un collecticiel synchrone pour l'édition partagée de documents.* Thèse de doctorat, Université d'Orsay Paris-Sud, 1994.

[Kazman 1994] R. Kazman, L. Bass, G. Abowd et M. Webb. *SAAM: A Method for Analyzing the Properties of Software Architectures*, ICSE-16, Sorrento, Italie, 1994.

[Nigay 1994] L. Nigay. *Conception et réalisation des systèmes interactifs: Application aux Interfaces Multimodales.* Thèse de doctorat, Université Joseph Fourier Grenoble I, 1994.

[Ouadou 1994] K. E. Ouadou. *AMF : un modèle d'architecture multi-Agents Multi-Facettes pour interfaces homme-machine et les outils associés.* Thèse de Doctorat, Ecole Centrale de Lyon, 1994.

Outils pour la communication homme-homme médiatisée

No network can beat the bandwidth
of a truck full of CD-ROMs

Anonyme (lu sur USENET)

Outils pour la communication homme-homme médiatisée

8.1 Introduction

Nous avons vu que les connecteurs sont des éléments importants d'une architecture. Pour la communication homme-homme médiatisée, différents types de connecteurs peuvent être utilisés. Dans ce chapitre, nous présentons la classification IMPACT qui offre un espace d'analyse des mécanismes de connexion pour la communication homme-homme médiatisée. Cette classification identifie les dimensions pertinentes pour le choix d'un type de connecteurs. Nous présentons ensuite notre bibliothèque de communication homme-homme médiatisée, UserLink, et nous l'étudions dans le cadre de la classification IMPACT.

8.2. La classification IMPACT pour les mécanismes de connexion

Pour aider les choix de conception liés à la définition des connecteurs pour la communication homme-homme médiatisée, nous proposons une classification des mécanismes de connexion. Cette classification a en fait trois objectifs :

- structurer l'espace de conception de outils de communication homme-homme médiatisée,
- guider les choix de conception et de réalisation auxquels peuvent être confrontés les développeurs de tels outils,
- aider à la compréhension et à l'analyse des outils existants en permettant de classer leurs caractéristiques.

La classification IMPACT adopte une perspective système, et structure l'espace de conception suivant six dimensions : **I**nteraction, **M**edia, **P**rocessing, **A**uthentication, **C**ooperation, **T**ime (figure 8.1).

La dimension *Interaction* caractérise le nombre d'utilisateurs qui peuvent être impliqués dans une communication. Cette dimension permet de clarifier les services de base du réseau nécessaires. *Media* indique le nombre et le type des médias de communication utilisés et permet d'exprimer des contraintes sur les possibilités du réseau. *Processing* exprime les traitements effectués sur les informations à communiquer. Cette caractéristique aide à préciser les services système requis et a des influences sur l'architecture logicielle. *Authentication* s'intéresse à l'authentification des utilisateurs et

l'on peut en dériver des conséquences sur l'architecture du système. *Cooperation* indique les liens de l'outil de communication avec les éventuelles autres dimensions fonctionnelles du système, production et coordination. Enfin, *Time* reprend en la précisant la dimension temps de la classification espace-temps (présentée au chapitre 1).

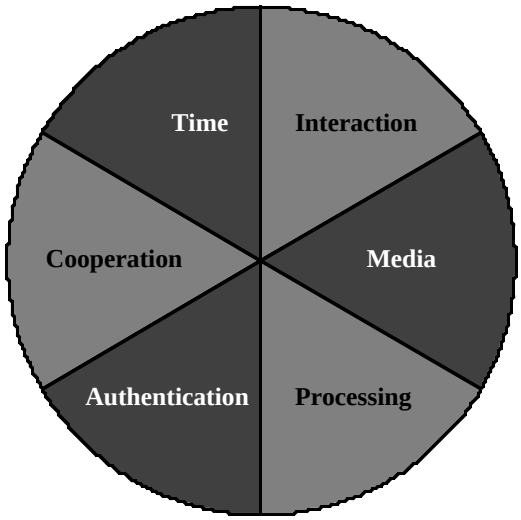


Figure 8.1. Les six dimensions de la classification IMPACT.

Nous détaillons maintenant chacune de ces dimensions et montrons comment elles peuvent être utilisées pour analyser des systèmes existants et guider des choix de conception.

- Interaction

  La dimension Interaction concerne le nombre d'utilisateurs qui sont mis en communication par le système. Les valeurs possibles pour cette dimension sont les suivantes : 1-1, 1-plusieurs, plusieurs-1, plusieurs-plusieurs. Pour chacune de ces possibilités, les communications peuvent être uni- ou bi-directionnelles. Une interaction 1-1 met en communication un utilisateur-source et un utilisateur-destinataire. Notons que l'initiateur de la connexion peut éventuellement être un tiers, ou bien comme dans le mediaspace Cruiser [Fish 1992] le système lui-même. Le téléphone ou le mail par exemple permettent notamment des interactions 1-1. Une interaction 1-plusieurs met en jeu un utilisateur-source et plusieurs destinataires, comme le font les mailing-lists ou les newsgroups d'Internet. L'interaction plusieurs-1 est une communication de plusieurs utilisateurs-sources vers un autre utilisateur. Un bon exemple de ce type d'interaction est fourni par l'accès World-Wide Web/mediaspace du LRI[1] : un utilisateur externe se connectant à ce serveur voit les quatre images des utilisateurs

[1] Laboratoire de Recherche en Informatique, Orsay Paris-Sud, http://www-ihm.lri.fr

du mediaspace local. Enfin l'interaction plusieurs-plusieurs permet la communication de plusieurs utilisateurs-sources à la fois vers plusieurs utilisateurs-destinataires à la fois à l'intérieur d'un même groupe. C'est par exemple le cas des systèmes de vidéoconférence individuels à plusieurs participants, ainsi que celui du système Portholes [Dourish 1992]. C'est aussi le cas de systèmes qui regroupent plusieurs communications et les envoient vers plusieurs destinataires. Les "digests" de mailing-lists en sont un exemple.

La dimension Interaction permet d'envisager certains choix de conception. Pour l'interaction 1-plusieurs par exemple, on peut réfléchir à l'utilisation d'un service multicast ou broadcast. Le choix entre les deux possibilités sera guidé par la dimension Authentication (utilisateurs identifiés pour le multicast, par exemple le mail et utilisateurs non identifiés pour le broadcast, par exemple les newsgroups). Dans les cas plusieurs-1 et plusieurs-plusieurs, une information composite venant de plusieurs sources est transmise vers un ou plusieurs destinataires. On peut ici réfléchir à l'architecture du système et choisir une architecture centralisée de préférence à une architecture répliquée.

- Media

  La dimension Media précise le nombre de médias utilisés dans la communication et leurs types : texte, audio, vidéo, etc. Chacun des médias est aussi caractérisé par un ensemble d'attributs indiquant : sa nature (continu ou discret), son support (analogique ou numérique), sa synchronisation éventuelle avec un autre média, la bande passante qu'il requiert. Ces informations permettent d'exprimer les caractéristiques matérielles requises (bande passante souhaitée, donc type de réseau à mettre en œuvre) ainsi que des services logiciels requis (mécanisme de synchronisation, gestion des flots de médias continus, gestion de la qualité de service, etc.).

- Processing

  La dimension Processing caractérise les traitements effectués sur l'information à communiquer. Cette dimension prend en compte les services de codage et de compression ou le cryptage. D'autres types de traitements peuvent aussi être effectués et ce à différents niveaux dans l'architecture logicielle. C'est en analysant cette dimension qu'il faudra se poser la question de la vérification de la propriété d'intégrité définie au chapitre 4. Par exemple, la communication dans un système de vidéoconférence peut utiliser un algorithme de compression qui ne perd pas d'information ou un algorithme avec perte (*lossy*) comme MPEG. Incidemment, cette dimension doit aussi prendre en compte des dispositifs matériels particuliers qui ne sont pas directement gérés par le système. On pense par exemple au

dispositif Hydra [Buxton 1993], ensemble d'unités audio/vidéo autonomes qui permettent de simuler la disposition spatiale des participants ou au système MAJIC [Okada 1994] qui utilise des écrans semi-transparents pour préserver le contact visuel entre les interlocuteurs. Une autre approche visant à préserver le contact visuel repose sur la combinaison d'un dispositif matériel et d'un traitement par le système : elle utilise deux caméras placées face à l'utilisateur en haut et en bas du moniteur et un système de traitement d'images qui recrée une vue de face à a partir des deux images [Ott 1993].

- Authentication

  La dimension Authentication prend en compte l'identification des utilisateurs. Les valeurs possibles pour cette dimension sont : anonyme, identifié, identifié avec droits. Les utilisateurs d'un système peuvent être anonymes : les newsgroups d'Internet par exemple ne demandent par l'identification de l'utilisateur pour la lecture. Le rédacteur d'un article d'un newsgroup (qui lui est identifié) poste sa contribution sans indiquer de destinataire autre que le nom du newsgroup. Le modérateur d'un newsgroup a la possibilité de valider les messages. On retrouve le lien entre la définition du rôle et les droits qui lui sont attribués. En revanche, le mail nécessite une identification, à la fois de la source et du destinataire. Des systèmes font aussi intervenir des droits liés à chaque utilisateur. Ces droits constituent un mécanisme de contrôle d'accès. Le mediaspace RAVE [Gaver 1992] par exemple permet à chaque utilisateur de définir les droits qu'il donne aux autres utilisateurs d'ouvrir une connexion avec lui. Le système de droits participe à la satisfaction des propriétés liées au respect de la vie privée que nous avons vues au chapitre 2. La propriété d'observabilité publiée doit aussi être examinée en liaison avec cette dimension. Un système de droits d'accès, qui peut tenir compte des rôles, permet en effet de la satisfaire.

- Cooperation

  L'axe Cooperation précise les liens entre l'outil de communication homme-homme médiatisée et les éventuels services de production et de coordination du système (au sens du chapitre 1). Dans le cas courant des systèmes de vidéoconférence individuelle, les facettes production et coordination sont absentes. Pour les systèmes incluant des services de production et de coordination, il convient de réfléchir à l'intégration du service de communication. Nous avons vu qu'un mediaspace comme CAVECAT [Mantei 1991] intègre des outils d'édition partagée. Un lien entre les aspects production ou coordination et l'aspect communication a une influence sur l'architecture. Si par exemple pour un éditeur de dessin partagé, on veut teinter de la même couleur l'image de chaque participant

et son télépointeur, les agents de gestion de la connexion et de télépointeur doivent partager de l'information. Un service de communication lié à outil de dessin partagé comme celui que nous avons utilisé pour l'expérimentation Garden Movie (chapitre 3) exigera de considérer la propriété de réversibilité vidéo.

- Time
  La dimension Time reprend en l'étendant et en la précisant la dimension temps de la classification espace-temps. Deux aspects temporels sont pris en compte : l'occurrence et la durée de la communication. L'occurrence indique quand la communication a lieu à partir du moment où elle est initiée par l'utilisateur. Les valeurs possibles sont : immédiate, dès que possible, différée. La durée de la communication peut prendre les valeurs suivantes : instantanée, finie, indéterminée ou potentiellement infinie. Une communication de durée instantanée correspond au transfert d'informations discrètes : l'envoi d'un mail par exemple peut être considéré comme instantané. Dans un mediaspace, un "glance" a une durée finie ; en revanche, une connexion "office-share" a une durée indéterminée. Cette distinction, en conjonction avec l'analyse de tâche, permet de réfléchir au débit souhaitable pour les médias continus : lorsqu'une connexion est de durée potentiellement infinie, on pourra sans doute se satisfaire par moments d'une qualité moindre, donc par exemple d'un débit vidéo plus faible. On remarque que les deux aspects temporels occurrence et durée permettent de retrouver la distinction classique synchrone/asynchrone. Une communication synchrone a une occurrence immédiate et une durée finie ou indéterminée. Une communication asynchrone a une occurrence différée (éventuellement dès que possible) et une durée instantanée ou finie.

On remarque que la classification IMPACT ne reprend pas la dimension espace de la classification espace-temps. En effet, l'éloignement des utilisateurs n'a pas de véritable influence sur les mécanismes logiciels mais plus sur des aspects matériels (type de réseau) qui seront déterminés par la bande passante requise par les médias utilisés. L'éloignement des utilisateurs a des conséquences sur l'usage du système : le délai des communications longue distance, les décalages horaires, les différences culturelles sont des aspects qui doivent être pris en compte dans la politique d'utilisation qu'implémente le système.

En résumé, la classification IMPACT détaille les mécanismes à mettre en œuvre pour la communication homme-homme médiatisée. Elle peut servir de cadre de réflexion à la fois pour l'analyse des systèmes existants et pour la conception de nouveaux systèmes. Nous montrons son application à la bibliothèque UserLink au paragraphe 8.3.2.

8.3. Réalisation : UserLink

UserLink est une bibliothèque de gestion des médias continus pour la communication homme-homme médiatisée. UserLink répond à deux objectifs :

- fournir une solution de communication pour les médias continus qui ne soit pas dépendante d'un matériel ou d'un système d'exploitation. En effet, la plupart des solutions actuelles pour le transport des médias continus requiert des services spécifiques du système d'exploitation comme les capacités "temps réel" du système d'exploitation ou utilise des solutions ad hoc non généralisables [Tokuda 1994],

- fournir une solution extensible et portable vis-à-vis des plates-formes matérielles et des protocoles de communication réseau utilisés.

8.3.1. Architecture de la bibliothèque UserLink

L'architecture de la bibliothèque UserLink distingue les communications de contrôle par messages et les communications de médias continus pour chaque média (figure 8.2).

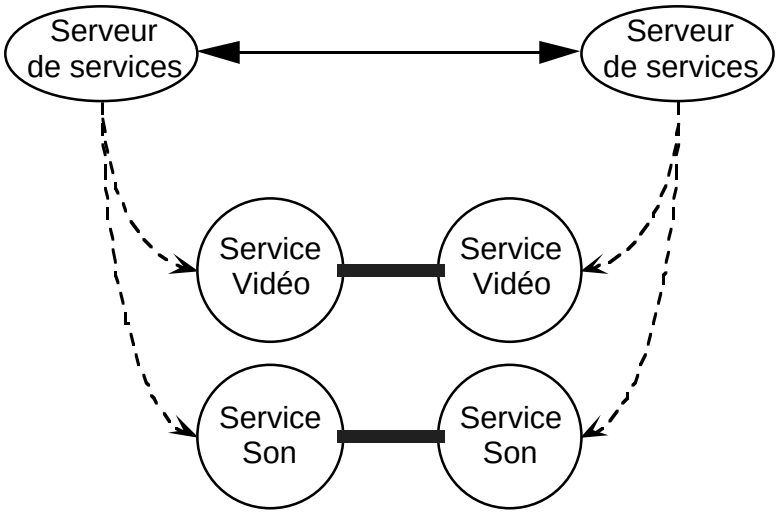


Figure 8.2. L'architecture de la bibliothèque UserLink. Ici, deux sites sont en communication. Les flèches pointillées indiquent des relations "instancie". La double flèche pleine indique la communication de messages de contrôle. Les traits épais sont des communications de flots de médias continus.

Le pivot de UserLink est le composant "Serveur de Services". On trouve un serveur de services sur chaque site. Il gère les messages de contrôle échangés avec d'autres serveurs de services sur d'autres sites. Sur demande, il instancie des services adaptés à la transmission d'un média continu donné. Par exemple sur la figure 8.2, chaque serveur de

services a instancié un service pour la vidéo et un service pour le son. Les services échangent les données continues directement entre eux. Après réalisation de UserLink, nous nous sommes rendus compte que cette approche est similaire à celle adoptée par Middleware System Services [Koegel Buford 1994].

Vis-à-vis du modèle d'architecture CoPAC présenté au chapitre précédent, la bibliothèque UserLink trouve sa place dans deux composants : le composant communication et le composant techniques de présentation. Le composant communication recouvre le serveur de services. Il est typiquement sous le contrôle d'un agent via la facette COM. Via le serveur de services, un agent peut ainsi ouvrir et fermer une connexion et modifier son débit. Les services vidéo ou son font partie du composant techniques de présentation. Ainsi, deux composants techniques de présentation communiquent directement comme indiqué dans le modèle CoPAC. La figure 8.3 indique les composants sur lesquels repose un service.

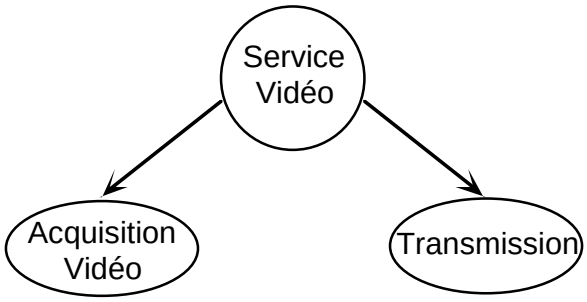


Figure 8.3. Les composants utilisés par un service. Ici l'exemple du service vidéo qui utilise un composant d'acquisition vidéo et un composant de transmission réseau. Les flèches indiquent la relation "utilise les services de".

Un service utilise deux composants : un service d'acquisition et un service de transmission. Chacun de ces deux composants est spécifique au matériel et au réseau utilisé. Par exemple, le composant d'acquisition vidéo est spécifique à la carte de numérisation utilisée. Le composant transmission est spécifique au protocole réseau, dans notre cas TCP. Cette organisation permet d'isoler les composants spécifiques à un matériel ou un protocole réseau donné et garantit la portabilité des composants de type service et serveur de services.

UserLink est écrit en C++ et fonctionne sur Macintosh. Ses performances pour la transmission vidéo permettent d'obtenir un débit de 15 images par seconde sur un réseau Ethernet et pour une image de taille 160 sur 120 pixels en 256 niveaux de gris. Ce débit ne permet pas une haute qualité d'image, mais est acceptable pour la communication homme-homme médiatisée. La propriété de régularité des flots est vérifiée expérimentalement mais est fortement dépendante de la charge du réseau.

8.3.2. UserLink dans la classification IMPACT

Nous analysons maintenant l'outil UserLink dans le cadre de la classification IMPACT. Cette analyse va nous permettre de préciser les possibilités et les limitations de UserLink.

- Interaction
  UserLink permet toutes les possibilités répertoriées dans la dimension Interaction. UserLink simule la diffusion multicast en ouvrant plusieurs connexions pour plusieurs sites. De même UserLink peut recevoir plusieurs connexions simultanément. Notons que cette approche permet de fixer des caractéristiques différentes pour différentes connexions. Il est possible pour un site UserLink d'envoyer par exemple deux flots vidéo à deux destinataires avec des débits différents et des tailles d'image différentes sur chaque flot.

- Media
  Les seuls médias utilisés sont le son et la vidéo. La synchronisation n'est pas prise en compte explicitement.

- Processing
  Aucun traitement n'est effectué sur les images ou le son envoyés. Aucun mécanisme de compression n'est intégré. L'architecture choisie dans UserLink qui place l'acquisition directement sous le contrôle du service rend d'ailleurs difficile les traitements à l'émission autrement que par UserLink lui-même. Par exemple, la réversibilité vidéo est assurée par le récepteur comme nous l'avons montré dans la présentation de l'architecture de VideoPort au chapitre précédent.

- Authentication
  Il n'y a pas dans UserLink de système d'authentification des utilisateurs. Seuls les sites, sous forme de leurs adresses IP, sont connus.

- Cooperation
  UserLink ne permet que des services de communication. L'intégration avec les aspects coordination ou coopération est à la charge de l'application qui intègre UserLink.

- Time
  UserLink permet uniquement des connexions immédiates de durée finie ou indéterminée.

En résumé, nous avons présenté UserLink, notre outil de gestion des médias continus pour la communication homme-homme médiatisée. Cette bibliothèque nous a permis d'expérimenter une architecture qui distingue les messages de contrôle et la transmission des médias continus. Son analyse dans le cadre de la classification IMPACT confirme que cet outil ne propose qu'un service de communication élémentaire. Il pourrait être enrichi dans différentes directions indiquées par la classification comme : l'identification des utilisateurs, l'intégration avec les aspects coordination et communication des systèmes multi-utilisateurs ou encore la possibilité de différer des connexions.

8.4. Synthèse

La classification IMPACT est un espace d'analyse et de conception des mécanismes de connexion pour la communication homme-homme médiatisée. Elle comporte six dimensions : Interaction, Media, Processing, Authentification, Cooperation, Time. Nous avons présenté la bibliothèque UserLink de gestion des médias continus pour la communication homme-homme médiatisée. Cet outil repose sur la séparation de la communication des messages de contrôle et des flots de médias continus. L'architecture de UserLink permet de satisfaire la propriété de portabilité. L'analyse de UserLink avec la classification IMPACT nous a confirmé que cet outil procure des services de communication élémentaire qui peuvent être enrichis.

Références

[Buxton 1993] W. A. S. Buxton. *Telepresence: Integrated Shared Task and Person Spaces,* in *Readings in Groupware and Computer-Supported Work.* R. M. Baecker, (ed.) Morgan Kaufman, San Mateo, California, USA, 1993. pp. 816-822.

[Dourish 1992] P. Dourish et S. A. Bly. *Portholes: Supporting Awareness in a Distributed Work Group*, CHI'92, ACM Conference on Human Factors in Computing Systems, Monterey, California, USA, 1992. pp. 541-547.

[Fish 1992] R. S. Fish, R. E. Kraut, R. W. Root et R. E. Rice. *Evaluating Video as a Technology for Informal Communications*, CHI'92, ACM Conference on Human Factors in Computing Systems, Monterey, California, USA, 1992. pp. 32-48.

[Gaver 1992] W. W. Gaver, T. Moran, A. MacLean, L. Lövstrand, P. Dourish, K. Carter et W. Buxton. *Realizing a Video Environment: EuroPARC's RAVE System*, CHI'92, ACM Conference on Human Factors in Computing Systems, Monterey, California, USA, 1992. pp. 27-36.

[Koegel Buford 1994] J. F. Koegel Buford. *Middleware System Services Architecture*, in *Multimedia Systems.* J. F. Koegel Buford, (ed.) Addison-Wesley, New York, New York, USA, 1994. pp. 221-244.

[Mantei 1991] M. Mantei, R. M. Baecker, A. Sellen, W. Buxton, T. Milligan et B. Wellman. *Experiences in the use of a Media Space*, CHI'91, ACM Conference on Human Factors in Computing Systems, New Orleans, Louisiana, USA, 1991. pp. 203-208.

[Okada 1994] K.-i. Okada, F. Maeda, Y. Ichikawaa et Y. Matsushita. *Multiparty Videoconferencing at Virtual Social Distance: MAJIC Design*, CSCW'94, ACM Conference on Computer-Supported Cooperative Work, Chapel Hill, North Carolina, USA, 1994. pp. 385-393.

[Ott 1993] M. Ott, J. P. Lewis et I. Cox. *Teleconferencing Eye Contact Using a Virtual Camera*, InterCHI'93, ACM/IFIP Conference on Human Factors in Computing Systems, Amsterdam, Pays-Bas, 1993. pp. 109-110.

[Tokuda 1994] H. Tokuda. *Operating System Support for Continuous Media Applications,* in *Multimedia Systems.* J. F. Koegel Buford, (ed.) Addison-Wesley, New York, New York, USA, 1994. pp. 201-220.

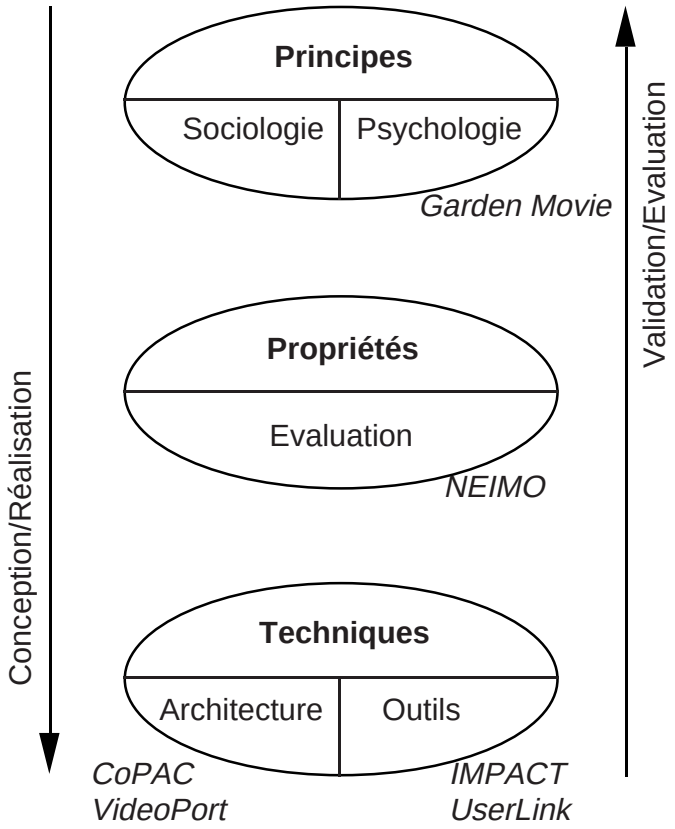
Conclusion

"Would you tell me, please,
which way I ought to go from here?"
"That depends a great deal
on where you want to get to," said the Cat

Lewis Carroll

Résumé de notre contribution

La figure ci-dessous reprend notre cadre structurant de la figure 1.8 (chapitre 1). Elle montre comment nous avons incarné chacun des trois niveaux et les contributions conceptuelles et techniques que nous y apportons.



Notre travail contribue à l'étude des systèmes multi-utilisateurs et de la communication homme-homme médiatisée selon quatre aspects complémentaires :

- face au nombre et à la diversité des disciplines dont l'étude de notre domaine doit tenir compte, nous avons proposé une structuration de l'analyse des apports des sciences non-informatiques à la conception et à la réalisation des systèmes multi-utilisateurs,

- à l'aide de ce cadre structurant, nous avons étudié l'extension aux systèmes multi-utilisateurs des acquis issus de l'étude de l'interaction homme-machine individuelle,

- pour chacun des aspects du domaine que nous avons étudiés, nous avons proposé des outils scientifiques comme les propriétés, les taxinomies, les espaces problèmes ou les modèles pour guider la réflexion sur les propositions existantes et développer de nouveaux concepts,

- enfin, des réalisations ont mis en œuvre et validé les concepts que nous avons développés.

Pour analyser l'intégration des apports des sciences sociales et humaines au processus de développement informatique, nous proposons une approche à trois niveaux : principes, propriétés et techniques (chapitre 1). Cette approche structurée nous permet de synthétiser les apports des sciences sociales et humaines sous forme de principes dont nous dérivons des propriétés. Nous montrons ensuite comment ces propriétés peuvent être vérifiées par l'utilisation de techniques appropriées d'évaluation et de réalisation.

Nous identifions plusieurs aspects significatifs de la conception et de la réalisation des systèmes multi-utilisateurs : les sciences sociales et humaines (chapitres 2 et 3), l'évaluation ergonomique (chapitre 5), la conception logicielle (chapitres 6 et 7) et les outils logiciels (chapitre 8). Pour chacun de ces aspects, nous évaluons et critiquons l'acquis provenant de l'étude des systèmes mono-utilisateurs dans la perspective des systèmes multi-utilisateurs. Nous proposons ensuite l'extension des concepts et des techniques présentés au cas des systèmes multi-utilisateurs et de la communication homme-homme médiatisée.

Nous proposons l'utilisation systématique des propriétés pour prendre en compte les enseignements des disciplines non-informatiques qui informent la conception et la réalisation logicielles. Les propriétés (chapitre 4) constituent un outil scientifique et permettent de jeter un pont entre les sciences "douces" comme les sciences humaines et sociales et la science "dure" qu'est l'informatique. Pour guider l'évaluation des travaux existants et pour élucider les composantes du domaine, nous proposons d'abord le trèfle fonctionnel, un espace de définition des systèmes multi-utilisateurs (chapitre 1), puis un espace problème pour les modèles d'architecture logicielle (chapitre 6), et la taxinomie IMPACT des mécanismes de connexion pour la communication homme-homme médiatisée (chapitre 8). Le modèle d'architecture logicielle CoPAC (chapitre 7) rassemble de façon équilibrée les requis fonctionnels des systèmes multi-utilisateurs que nous avons identifiés avec le modèle du trèfle.

Nous examinons la validité des concepts développés en les mettant en œuvre dans des réalisations. L'expérimentation Garden Movie, menée en collaboration avec des

psychologues, étudie la communication humaine gestuelle et parlée par l'intermédiaire d'un lien audio/vidéo. Ses résultats font apparaître une difficulté de l'utilisation de la communication homme-homme médiatisée. Nous en déduisons une propriété d'après notre cadre structurant à trois niveaux. Nous avons étudié l'utilisation des propriétés pour l'évaluation ergonomique avec la plate-forme d'évaluation NEIMO en l'appliquant à l'évaluation de l'outil de communication Supratel. Nous avons examiné la validité du modèle d'architecture CoPAC en l'appliquant à trois réalisations : le jeu $(421)^n$, la plate-forme d'observation des utilisateurs NEIMO et le mediaspace VideoPort. Enfin nous avons appliqué l'espace de classification IMPACT pour évaluer les possibilités et les limitations de UserLink, notre bibliothèque de communication pour les médias continus.

Perspectives

Les perspectives de développement de notre travail s'organisent suivant quatre directions : l'exploration systématique des possibilités des mediaspaces pour la communication homme-homme médiatisée, la validation de notre approche structurée par de nouvelles collaborations avec des sciences non-informatiques, la validation extensive du modèle CoPAC et la prise en compte de la mobilité de l'utilisateur.

Par la construction d'espaces d'analyse, un de nos objectifs est l'exploration systématique des composantes d'une question. Pour la communication homme-homme médiatisée, nous avons fait un premier pas dans cette direction avec la classification IMPACT. D'autres espaces d'analyse nous permettront d'étudier de façon systématique les possibilités des mediaspaces. Le but de cette recherche est la construction d'un "espace des mediaspaces" qui rassemble les possibilités de ces outils de communication. Nous progressons dans cette direction avec l'expérimentation de caméras mobiles montées sur tourelle pivotante. L'utilisation d'un ensemble de caméras pivotantes afin de donner une vision globale et continue d'un espace distant est un de nos objectifs à court terme. Ce travail reprend en les étendant au cas de la communication homme-homme médiatisée les possibilités de vision à 360° d'outils comme Surround Video ou QuickTime VR.

Nous avons éprouvé la validité de notre vision structurée de l'apport des sciences non-informatiques (sous forme de principes, propriétés et techniques) avec notre collaboration pour l'expérimentation Garden Movie. Mais il n'est pas facile de concilier des méthodes d'analyse et des cultures scientifiques différentes. Nous voulons éprouver notre vision à trois niveaux en travaillant avec par exemple des spécialistes de l'éthique ou des sociologues. De notre point de vue, l'éthique informatique est amenée à prendre une importance croissante dans les années qui viennent. Collaborer tôt avec cette discipline

permettra une compréhension mutuelle qui ne peut être que profitable. La participation de ces disciplines à l'évaluation doit aussi être étudiée.

Le modèle d'architecture logicielle CoPAC doit être validé de façon plus approfondie. Il nous faut réfléchir par exemple à l'application du modèle à des systèmes plus conséquents que ceux que nous avons étudiés. L'étude d'un système multi-utilisateur plus complet, intégrant de façon harmonieuse les trois facettes fonctionnelles, production, coordination et communication nous permettrait d'éprouver le modèle et de mettre en évidence ses limitations. Nous n'avons pas non plus abordé l'intégration du modèle avec des technologies comme les plates-formes à objets distribués.

L'interface homme-machine a jusqu'à présent principalement considéré un utilisateur assis devant un ordinateur de bureau. Mais les avancées technologiques permettent maintenant à l'utilisateur de profiter de son environnement : en utilisant des dispositifs mobiles, il peut se déplacer et utiliser l'informatique et les outils de communication dans de nouveaux lieux. Cette ubiquité augmente les possibilités de l'utilisateur mais lui impose peut-être de nouvelles contraintes. Nous pensons que cette étude peut donner lieu à l'élaboration de nouveaux principes et de nouvelles propriétés adaptés à ces outils novateurs. Ces outils posent aussi des questions originales à la communication homme-homme médiatisée.

Annexe A

The System Modelling Glossary

Salber D., Coutaz J. & Nigay L. (editors)
Laboratoire de Génie Informatique, University of Grenoble
Faconti G. & Paterno F.
CNUCE
Duke D. & Harrison M.
University of York

1st March 1994

Amodeus Project Document: System Modelling/WP 26

Une version hypertexte du System Modelling Glossary est consultable à l'adresse :

`http:://www-lgi.imag.fr/Les.Groupes/IHM/AMODEUSGlossary/GlossaryMain.html`

Abstract

The System Modelling Glossary gathers definitions for the vocabulary that is commonly used within the RP1 System Modelling Package. Its purpose is to serve as a support/companion for reading RP1 papers. It is also aimed at providing the Amodeus project with a reference vocabulary for discussing system modelling issues.

Instructions

Entries in the glossary may be words or expressions (e.g., "level of abstraction"). They are listed in alphabetical order. Some of the definitions may draw upon or replicate widely-accepted meanings. In this case, the source reference is denoted as [ref] and is listed in the "References" section at the beginning of the document.

Updates

As the vocabulary used within RP1 may evolve, some entries and definitions might become obsolete. Such entries are marked {Obsolete} and might be amended with a newer definition. Every member of the Amodeus project is welcome to suggest revisions to the RP1 glossary.

Flags

The three following entries are marked with an asterisk [*]: Affordance, Real-world validity and Task domain concept. The asterisk indicates that we strongly feel the need for other packages' input and comments on these definitions.

References

[CCS] A.J.R.G.Milner, *A Calculus of Communicating Systems*, Lecture Notes in Computer Science, Vol. 92, Springer Verlag, 1980.

[CGRM] Information technology - Computer graphics and image processing- *Computer Graphics Reference Model*, ISO 11072, International Organisation for Standardisation, 1992.

[Dictionary] *The American Heritage Dictionary of the English Language*, William Morris, editor, Houghton Mifflin Company, 1969.

[LOTOS] T. Bolognesi, H. Brinska,*Introduction to the ISO Language LOTOS*, Computer Networks and ISDN Systems 14, pp 25-59.

Abstraction [CGRM]
Processing element within a layer that transforms an information type. The transformed information is transferred to the next higher layer in a hierarchy or to another subsystem in the system when that layer is the highest one in that hierarchy.

Abstraction
(a) Process that transforms information into information whose semantic content and scope are richer/higher than the content and scope of the initial information.
(b) Result of the process of abstracting.

Abstraction function
Concept to denote the capacity of abstracting within a system.
See Rendering function.

Action [CCS or LOTOS]
Fact happening within a system identified by a name and by an information. The information can be either explicit (an instance of an information type is communicated) or implicitly associated to the event name (an information may be inferred from the event). Synonym for Event [CCS or LOTOS].

Action
(a) Transformation between two states of a system.
(b) Specification structure that defines a state transition by means of such a predicate.
(c) Indivisible operation performed by an agent whose effect can be perceived by another agent (in terms, for example, of events received through communication channels).
See Physical action, User physical action, System physical action, System conceptual action as subclasses of actions.

Action-tree [CCS or LOTOS]
Tree-like representation of a behaviour expression.

Affordance [*]
Property that the presentation of a system conveys information about the actions that can be performed by the user of that system. These properties relate to the "real world" understandings that the user has.

Agent
Object capable of initiating the performance of actions.

Alphabet
Union set of event names of a behaviour. Set of events in which some agent can engage.

Autonomous System
System capable of autonomous behaviour.

Backward recoverability
Property that the system provides the user with an undo facility to return to a previous state.
See Recovery, Recoverability.

Behaviour
(a) Set of all possible traces of a process. [CCS or LOTOS]
(b) An ordered set of states. See Trace.

Behaviour expression
Description of a behaviour within some formalism (e.g., action-tree, temporal logic, or process algebra).

Bias
Structure in a model that is not essential for characterising the behaviour or properties of a system.

Browsability
Property that the system provides the user with commands to make perceivable different portions of the system functional state. (By modifying the presentation state, the user may access different portions of the state of the functional core.)

Channel
(a) Common path shared by two or more components of a system.
(b) System structure/component ued for the communication of information.
See Physical channel, Digital channel, Human channel.

Command
(1) Set of physical actions transformed by the system into a system conceptual action.
(2) System conceptual action.
See Inspection command, Task domain command.

Communication [Dictionary]
Act of transmitting information.

Computation
Ordered set of operations that transforms the state of a system.

Component
Part of a whole. Depending on the context, can be instantiated as a software module, a subsystem, an agent, an interactor, an abstraction, a device, etc.

Conceptual action
Action performed by the system at the highest level of abstraction, i.e., a primitive function in the functional core.
See Level of abstraction.

Conceptual unit
Part of the state of the functional core that models a task domain concept at the highest level of abstraction.

Conformance
Property that the presentation of a system mirrors the underlying behaviour of the system.
See Layer conformance, Observability, Honesty.

Connectedness
Property of presentation that requires that notions of continuity in the state are faithfully rendered in the presentation.

Context
Set of state vectors used by a system in a computation. Examples of state vectors: state vector about the user, state vector about the presentation, state vector about the dialogue, state vector about the functional core, state vector about the environment.
See Dynamic context, Static context.

Continuity
The property that given a function f:A>B between ordered sets A and B, the neighbourhood around any point in set B corresponds to some neighbourhood around the inverse image of that point under f. This is a topological definition that generalises the

usual notion of continuity found for example in real analysis, and is useful in that it also applies in the case of discrete sets.

Deontic logic
A logic for reasoning about permissible or obligatory actions.

Detour
Interaction that is not directly involved in achieving a goal.

Device
(1) Physical artefact necessary to a system to acquire (input device) or deliver (output device) information. Examples include keyboard, loudspeaker, ears and mouth.
(2) Lowest level component of a system whose state changes result from physical actions.

Device assignment
Relation between a device over a state and a non empty subset of expressions of an interaction language. A device d is assigned in state s to a set E of expressions of a language l, if it does not exist any device equivalent to d over s and E. Assignment is permanent if the relation holds for any state. Assignment is total if the relation holds for E equals to the set of expressions that define l. For example, in Matis, the mouse is permanently assigned to the expression of window resizing in the direct manipulation interaction language.

Device equivalence
Relation between a non empty set of devices over a state and a non empty set of expressions in an interaction language. Devices in a set D are equivalent over a state s and a non empty set E of expressions in an interaction language L, if all of the expressions of E can be elaborated using either one of the devices in D. Equivalence is permanent if the relation holds for any state. Equivalence is total if the relation holds for E equals to the set of expressions that define L. For example, in Matis, keyboard and microphone are totally and permanently equivalent over natural language.

Device redundancy
Relation between a set of devices over a state and an expression of an interaction language. Devices of a set D are used redundantly in some state s for an expression e of a language l, if these devices are equivalent over s and e, and if they are used simultaneously to express e. For example, the user can spell a character using the microphone and type in the same character.

Device complementarity
Relation between a set of devices over a state and a non empty subset of expressions of an interaction language. Devices of a set D are complementary over a state s and a non empty set E of expressions of a language l, if E can be partitionned such that for each partition Ep of E, it exists a device d of D assigned over s and Ep. Complementarity is permanent if the relation holds for any state. Complementarity is total if the relation holds for E equals to the set of expressions of l. Language complementarity is best illustrated by spoken natural languages where concept names must be typed in. For example, in Munix, a multimodal user interface for Unix, commands that involve a file name such as remove, can be expressed using the microphone for the command name and options while file names must be elaborated with the keyboard.

Dialogue
(1) Same as interaction.
(2) Restriction on the behaviour of the interactive system to the set of actions performed with the participation of both human and computer processes.

Dialogue strategy
Set of behaviour expressions enforcing constraints over a dialogue. Examples of dialogue strategies include: reactive strategy (the system supplies the missing parameters of a command = default values), cooperative strategy (the system engages in a subdialogue to propose the set of possible values for a missing parameter and the user disposes), directive strategy (the system engages in a subdialogue to ask for the value of a missing parameter), negociated strategy (the system engages in a subdialogue to propose a value for the missing parameter and the user disposes), intentional strategy (the system engages in a subdialogue to check with the user that the value for the missing parameter is correct).

Dynamic context
Portion of the context that may vary over time, e.g., during a session. This variation can affect the values of the set of state variables, as well as the actual set of state variables (as in learning systems).

Dynamic system
See Autonomous system.

Equivalence
A relation that is reflexive, symmetric and transitive.
See Language equivalence, Device equivalence.

Event [CCS or LOTOS]
see Action [CCS or LOTOS]. The two terms have the same meaning in the CCS or LOTOS theory.

Event
(a) Data structure used to transfer information between components of a system.
(b) The observation that some action has occurred.

Event class
Name of an event.

Event instance
An instance of an event of a particular class.

Event occurrence
Same as Event instance.

Fan-in [CGRM]
Gathering of information units from multiple source processes to a single process.

Fan-out [CGRM]
Distribution of information units from one source process to multiple processes.

Feedback
See Response.

Fission
(a) Computation of a process abstracting/presenting an information type into a collection of different information types to be transferred to a set of processes.
(b) Decomposition of an information type at some level of abstraction into multiple information types of the same level of abstraction.
See Fan-out, Fusion.

Formal Method
Collection of mathematical structures, together with a precise syntax for defining instances of those structures and organising them into domain-specific abstractions. It should be associated with a method for eliciting these abstractions and transforming them.

Frame
A collection of variables that are referenced or modified by an action.

Functional core
Component of a system that implements task domain concepts.

Functional state
State vector of the functional core.

Fusion
(a) Computation of a process abstracting/concretizing a collection of information types received from distinct processes into a different information type to be transferred to another process.
(b) Composition of multiple information types at some level of abstraction into a single information type of the same level of abstraction.
See Fan-in, Fission.

Goal
Desired property of a system; usually expressed as a state or set of states that a user intends to achieve. Particularly for reactive or control systems, goals may also be expressed as predicates over the behaviour or trace of the system.

Guard
See Precondition.

Homomorphism
In general, structure preserving transformation where `structure' can be interpreted within many fields of mathematics. For example, a homomorphism between ordered sets <S,R> and <Q,T> is a function f from S to Q such that for all x,y in S, if (x,y) in R then (f(x), f(y)) is in T.

Honesty
Property that the presentation of the system renders its functional state appropriately (e.g., does not distort the functional state). Honesty is a necessary but not always sufficient condition for the user to be able to elaborate a correct mental representation of the functional state.
See Connectedness as a special case.

Human channel
Physical channel used by a human being to acquire (sensori channel) or deliver (motor channel) information.

Initiative
Denotes the agent that leads an interaction. In many cases the interactional initiative is clear. For example, in the case of a system like UNIX the initiative is with the user, whereas in many expert systems the system takes the initiative (the user simply has to respond to questions supplied by the expert system).

Input physical channel
A physical channel involved in the acquisition of information.

Interaction
Non-empty ordered set of events involving more than one agent.

Interaction dissonance
Occurs when an agent interacts incorrectly in the sense that an appropriate communication is directed to an inappropriate agent. This notion is intended to express the situations where two agents interact at cross purposes. A typical example of this occurs when a user interacts in a way that is appropriate through one window of a window based system with the wrong window selected.

Interaction language
Language used by the user or the system to exchange information. A language defines the set of all possible well-formed expressions, i.e., the conventional assembly of symbols, that convey meaning. The generation of a symbol or a set of symbols, results from a physical action. It is modelled as an event produced via some physical channel carrying a message which is this symbol or set of symbols.

Interaction object
An agent that participates in an abstraction and/or presentation transformation. It may interact directly with a human agent.
See also: Interactor.

Interaction trajectory
See Interaction.

Interactional invariance
Measure of the vulnerability of an interaction to the actions of other agents that are not taking part directly in the interaction. The issue is the way that interactive system goals are influenced by autonomous behaviour as would be a concern if we were talking of autonomous systems. An interactive system should be capable of supporting certain objectives regardless of the other activities that are going on autonomously within the system.

Interactionally-rich
(a) Qualifies interactive systems that support multiple input interaction languages and/or multiple output interaction languages and these languages can be conveyed via multiple input devices and/or multiple output devices.
(b) is used to refer to interactive systems which engage user or users in a "natural" interactive environment. Users can employ the multiple interaction languages or modalities to have a more natural interaction with the objects of the work domain.
In this sense, therefore, a hammer which uses few modalities and interaction languages is interactionally rich because of the texture and directness of the interaction.

Interactive system
System for which a subsystem is explicitly defined to be a human.

Interactor
See Interaction object.

Language assignment
Relation between an interaction language over a state and a non empty subset of conceptual units of a system. An interaction language l is assigned in state s to a set of conceptual units C, if it does not exist any interaction language equivalent to l over s and c. Assignment is permanent if the relation holds for any state. Assignment is total if the relation holds for C equals to the set of conceptual units of the system.

Language complementarity
Relation between a set of interaction languages over a state and a non empty subset of conceptual units. Interaction languages of a set L are complementary over a state s and a non empty set C of conceptual units of the system, if C can be partitionned such that for each partition Cp of C, it exists a language l of L assigned over s and Cp. Complementarity is permanent if the relation holds for any state. Complementarity is total if the relation holds for C equals to the set of conceptual units of the system. Language complementarity is best illustrated by coreferential expressions. For example, in Matis, natural language and direct manipulation are complementary over the conceptual unit "city" and any state where the specification of a city name is possible : "flights from this city" and selection of a city name through direct manipulation.

Language equivalence
Relation between a set of interaction languages over a state and a non empty subset of conceptual units of a system. Interaction languages of a set L are equivalent over a state s and a non empty set C of conceptual units of the system, if all of the conceptual units in C can be represented using either one of the language in L. Equivalence is permanent if the relation holds for any state. Equivalence is total if the relation holds for C equals to the set of conceptual units of the system. For example, in Matis, direct manipulation language and natural language are permanently equivalent for specifying requests.

Language redundancy
Relation between a set of interaction languages over a state and a conceptual unit of a system. Interaction languages of a set are used redundantly in some state s for a conceptual unit c, if these languages are equivalent over s and c, and if they are used simultaneously to represent c. For example, a wall is represented redundantly by the system via a red line (graphics interaction language) and the message "mind the red wall!" (natural language).

Layer conformance
(a) Relation between the abstraction/presentation transformations and the information types bound to two adjacent layers with respect to some invariants.
(b) Generalisation of conformance between any pair of layers.

Layer
Hierarchical partition of a subsystem into elements for which specific processing and/or transferring entities as well as information types are precisely defined.

Level of abstraction
(a) Layer within a system whose information types are characterized by a given semantic content and scope. The lowest level of abstraction corresponds to the poorest information type with regard to scope and content. The highest level of abstraction corresponds to the richest information type with regard to scope and content. These levels as well as any level in-between depend on the perspective or the objective of the modeller and/or the modelling technique.
(b) different perspectives in the design process of a system.

Media
Same as physical device type or set of physical devices types used for communication.

Message
Any value transmitted on a channel.

Metaphor
Representation of one system model in terms of another that has some real world significance or analogy, compare affordance. The classic example is the metaphor by which the file and directory structure of an operating system is understood in terms of a desktop with windows (folders) and icons.
See Homomorphism.

Migratability
Ability of the interactive system to dynamically transfer performance between agents.

Modal
Having possible modes or worlds; modal logic is a system for reasoning about possible and necessary truth.

Modality
Method characterized by the type of usage of interaction languages and physical devices for communicating information at the user interface.

Mode
Same as Context.

Model
Any representation of a real or imagined system.

Multidevice
Property of the system that provides the user with more than one physical device (simultaneously or not) to communicate information to the system and/or uses more than one physical device (simultaneously or not) to communicate information to the user.
See Multimedia.

Multilanguage
Property of the system that provides the user with more than one interaction language (simultaneously or not) to communicate information to the system and/or uses more than one interaction language (simultaneously or not) to communicate information to the user.

Multimedia
(a) Use of more than one medium for communication.
(b) Same as multidevice.

Multimediality
Property of a system that supports multimedia interaction.

Multimodal
Which provides the user with more than one modality (simultaneously or not) to communicate information to the system and/or uses more than one modality (simultaneously or not) to communicate information to the user.

Multimodality
Property of a system that supports multimodal interaction.

Objective
In general we shall be interested in the extent to which a system supports a set of objectives. An objective is considered to be a pre- and post-condition on state which may be linked with the agent who owns the objective.

Observability
Property that the presentation of a system contains sufficient information to allow the user to determine the functional state of the system.
See Conformance, Browsability, Honesty.

Ordered Set
A set S and relation R over S such that R is reflexive, transitive, and maybe antisymmetric .

Output physical channel
Physical channel that delivers information.

Parallelism
Simultaneity in the activity of multiple agents. Two events performed by different agents engaging in parallel activities may occur in overlapping time intervals.

Passive command
Command whose effect does not modify the state of the functional core of a system.

Performance
Interaction trajectory that achieves an objective.

Physical action
Action performed either by the user or by the system on a physical device.

Physical channel
Channel involving a physical device.
See Input physical channel, Output physical channel, Channel, Human channel.

Poset
Ordered set <S,R> where the relation R is a antisymmetric.

Postcondition
Predicate that describes the state of a system after the performance of some action. Actions are often specified by pre- andpost-condition pairs.

Potential
Scope that exists for exploiting the functionality of the interactive system at any stage in an interaction irrespective of any interactional initiative. "Event" potential is the user's scope for engaging in alternative actions yet continuing to achieve the same objective (for example filling a form by entering the fields in any possible permutation). "Objective" potential is the user's scope for achieving alternative objectives at a particular stage in the interaction. Hence low objective potential occurs when a user is locked into a single objective at a stage in the interaction.

Precondition
Predicate on the state of a system that determines those states from which an action can be carried out.

Predicate
A boolean-valued function of the state, behaviour, or trace of a system. Apredicate may represent a property.

Prefix
(of an ordered set <S,R>) is a subset T of S such that for every x in S and y in T, if (x,y) is in R then x is in T. Note that the empty set <0,R> and the full set <S,R> are always prefixes of<S,R>.

Prefix Closure
(of an ordered set <S,R>) is the set of all ordered sets <T,R> such that <T,R> is a prefix of <S,R>.

Presentation [CGRM]
Processing element within a layer that transforms an information type. The transformed information is transferred to the next lower layer in a hierarchy or to another sub-system in the system, when that layer is the lowest one in that hierarchy.

Presentation
(a) That part of the system or component of a system whose state can be perceived and potentially perceived by the user through browsing commands.
(b) Perceivable state of the system or component of a system or potentially perceivable state of the system or component of a system through inspection commands.

Presentation state
State vector of the presentation component of the system.

Process
Same as Agent.

Property
An observable of a system that can be described by a predicate and measurable (if used to assess the usability of a system).

Proportionality
Property that holds when measurable properties of a presentation are proportional (in the mathematical sense) to some value or function in the functional state.

Reachability
Property that requires that some state or set of states can be reached from a given state through the actions defined on a system.

Real world validity [*]
that a metaphor corresponds to the users intuitions about what the system will do, for example the mapping of left in (visual) space to an appropriate notion in the underlying model.
See Metaphor, Affordance.

Recovery
Performance of actions that take a system from some `unsafe' or undesired state to one satisfying some safety property.
See Recoverability.

Recoverability
Property that the system provides the user with commands to undo the effect of some action.
See Recovery, Undoability.

Refinement
(1) A model C of a system that includes the observables of a model A and whose behaviour is consistent with that of A - the actual consistency requirements vary between refinement notions. C is called the concrete system and A is the abstract system. Usually C will introduce new observables.
(2) The process of developing a refinement (1) for some system.

Rendering, a
Some perceivable presentation.

Render, to
To make perceivable.

Rendering function
Concept to denote the concretization activity within a system.
See Abstraction function.

Rendering relation
A relation between the functional state and presentation of a system or component.

Repair
See Recoverability.

Response
An event generated or caused by some agent.

Sequential process
Process whose behaviour is a totally ordered set of events, i.e., a process that receives at most one event at a time (e.g., if two events arrive at the same time, one is processed, one is lost or delayed).

Semantic delegation
Transfer of task domain concepts from the functional core into the user interface.
See Semantic repair.

Semantic repair
Semantic delegation used to augment the functional core with concepts that were forgotten in a previous implementation of the functional core and that cannot be incorporated in the functional core for technical reasons (e.g., source code is not available).

Session, user
Step between the initial state and the terminal state used by the system for the user.

Software architecture, a
A particular set of software components satisfying a set of relationships and described at some level of refinement (does not necessarily derive from an architecture model).

Software Architecture Model
Guide (model) for developing a software architecture.

Software Architecture Model, Conceptual
Guide (model) for developing a software architecture whose components are described at a conceptual level.

Software Component
Software unit that encapsulates a set of logically connected operations and data.

State
Assignment of values to names representing the observables of a system.

State vector, of a component, of a system
Names that define the state of a component, of a system.

Static context
Portion of the context that does not vary over session but may be modified between sessions.

Step
Any pair of states in a behaviour. For action A, an A-step is a pair of states that are related by A.

Stimulus
Event received by some component of a system from another.

Stuttering step
Step that leaves some subset of the observables unchanged.

Subsystem
Component of a system that behaves like a system.

System
A whole or collection of components capable of performing information processing and/or information transfer, and whose behaviour can be observed and possibly measured.

System physical action
Action performed by the system whose effect is made perceivable through output physical channels and possibly captured by human input channels.

System state
Same as State.
See Functional state, Presentation state.

Task
(a) A goal, together with some procedure or ordered set of actions that will achieve the goal.
(b) An ordered set of actions that will achieve a given goal when performed from a state satisfying some precondition.

Task domain concept [*]
Concept identified by task analysis as relevant to the user to accomplish the tasks in that domain.

Task domain command
Command whose effect modifies the functional core component of the system.

Template
Subset of the observables of a system that are relevant to the performance of some task.

Trace
An ordered set of events.
See Trajectory.

Transition
A pair of states, usually defining the effect of an action on a system.

Undoability
Same as Backward recoverability.

User interface system
Component of a system that implements the abstraction and/or rendering process of information between the functional core and the physical devices of the system. It includes the presentation component.

User interface
Common shortcut for User interface system.

User interface state
State vector of the user interface subsystem.

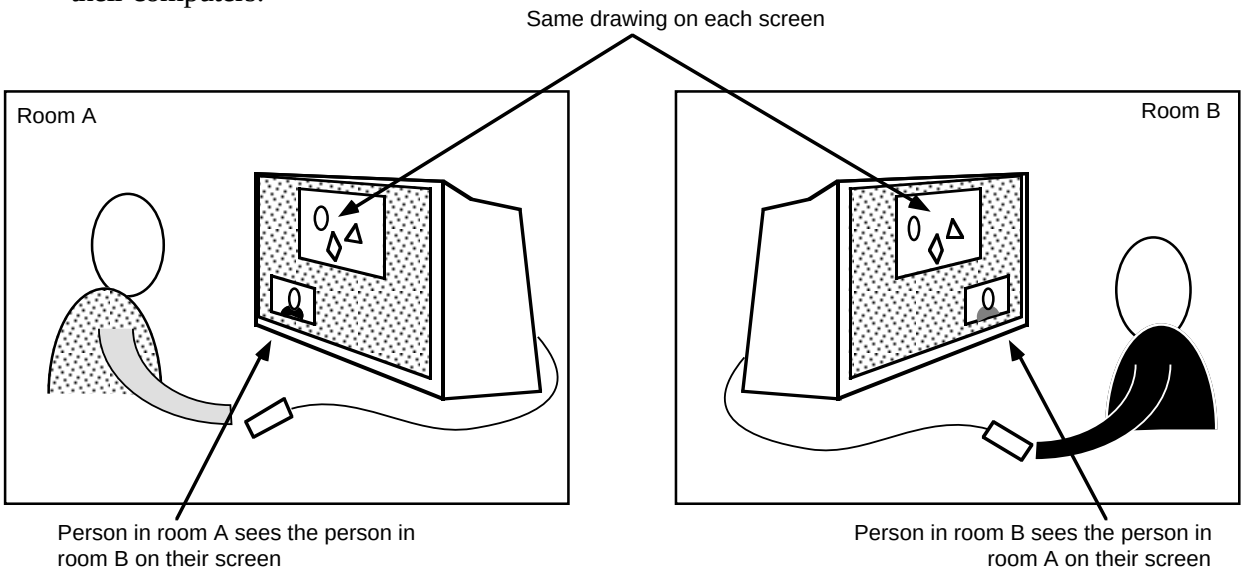
User physical action
Physical action initiated by the user.

Annexe B

Instructions de l'expérimentation "Garden Movie"

Ces instructions étaient lues aux sujets pour leur présenter l'expérimentation.

Modern computer systems are being developed so that people in different offices can see each other on their computer screens.
This can help them when they are discussing a piece of work that they can both see on their computers:



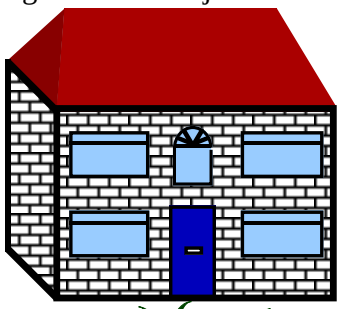
This experiment is looking at the usefulness of different camera positions.
You are playing the role of one of the people in the figure. You will see a 'window' from a drawing program, and your task is to move two of the objects in the window to new positions.
Your 'colleague' in another room will tell you which two objects to move, and where to move them to. You will be able to hear him and see a video of him on your screen.
You will now be shown an example…

Here is a 'drawing' that you and your colleague can both see. Both of you can see exactly the same picture.
Here is the 'video camera' that will show your colleague. Since this is just an experiment, we are using a recording instead of a 'live' picture. At the start of each clip, you will be able to see your colleague.
He will ask you to move two of the objects in the drawing. Listen and watch carefully, because the clip will vanish when he has finished talking.
As soon as he has finished talking, please move the objects he has indicated to their new positions.
You can move objects by moving the mouse pointer over them, and holding the button down while you drag them around. When you release the button, you 'let go' of the objects and they stay where you have left them.
You can move all of the objects as many times as you want, but you cannot move them out of the drawing window.
Try it now…

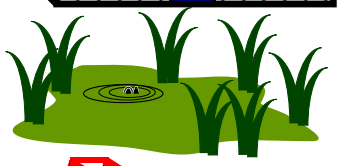
If you forget what he wanted you to do, or you were not clear about his instructions, you can press the 'play again' button with the mouse pointer. He will repeat his instructions, and the objects will be returned to their original places.
As soon as you have finished moving the objects, click the 'OK' button.
Please work as quickly as you can without making mistakes. Do you have any questions?
The rest of the experiment is divided into several sections.
In each section you will be given a different view of your 'colleague', or a different 'drawing' to work on.
There will be several clips for you to work on in each section.

In this section, you will see your colleague from the front. The picture you will see of him has been taken by a video camera placed just above his computer screen.
The drawing that you are working on is a plan of a garden. The objects it includes are:

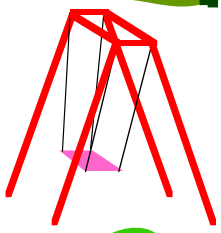
a house



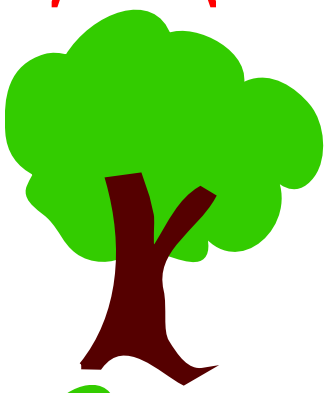
a pond



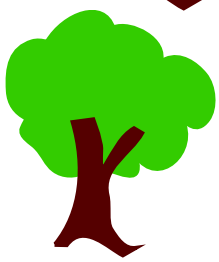
a swing



a big tree



a small tree

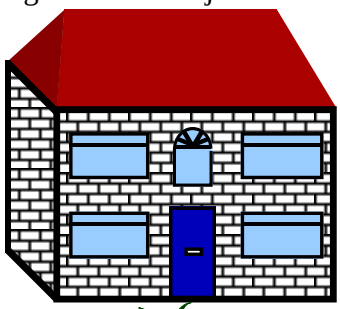


You can move these objects to the left or to the right, and you can move them behind each other, or in front of each other.

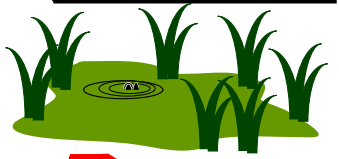
In this section, you will see your colleague from the front. The picture you will see of him has been taken by a video camera placed just above his computer screen, and has been reversed to form a mirror image.

The drawing that you are working on is a plan of a garden. The objects it includes are:

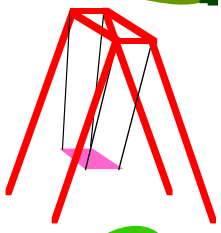
a house



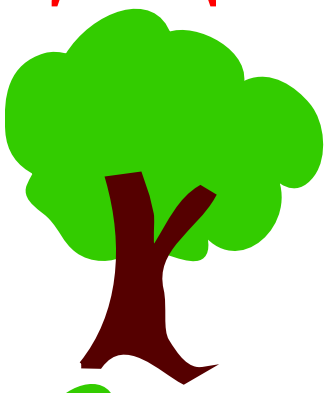
a pond



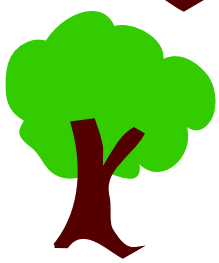
a swing



a big tree



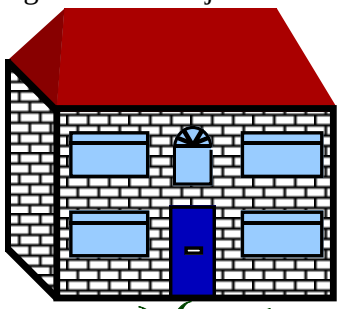
a small tree



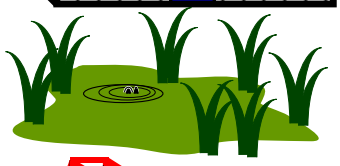
You can move these objects to the left or to the right, and you can move them behind each other, or in front of each other.

In this section, you will see your colleague from the side. The picture you will see of him has been taken by a video camera placed slightly behind him.
The drawing that you are working on is a plan of a garden. The objects it includes are:

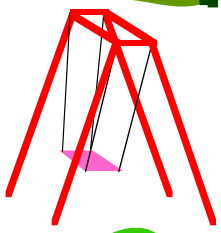
a house



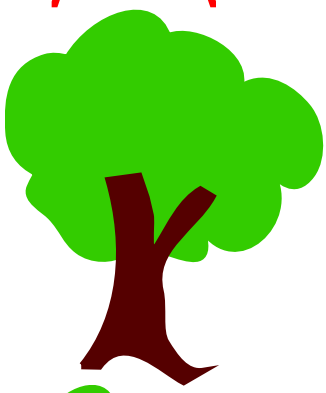
a pond



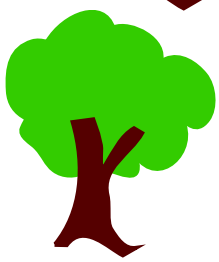
a swing



a big tree



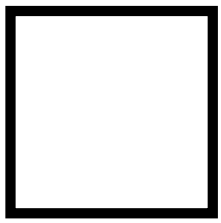
a small tree



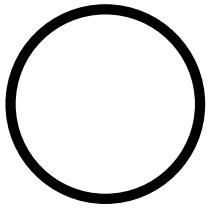
You can move these objects to the left or to the right, and you can move them behind each other, or in front of each other.

In this section, you will see your colleague from the front. The picture you will see of him has been taken by a video camera placed just above his computer screen.
The drawing that you are working on is a geometrical design. The objects it includes are:

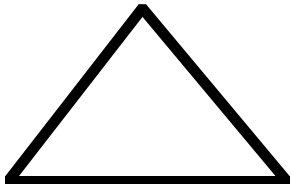
a square



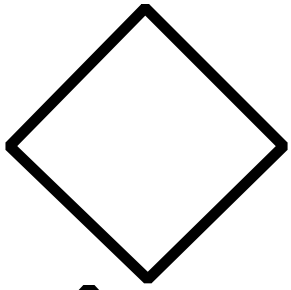
a circle



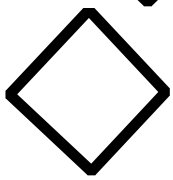
a triangle



a big diamond



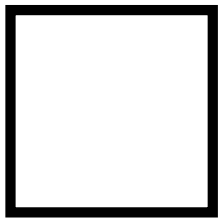
a small diamond



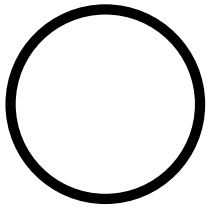
You can move these objects to the left or to the right, and you can move them above each other, or below each other.

In this section, you will see your colleague from the front. The picture you will see of him has been taken by a video camera placed just above his computer screen, and has been reversed to form a mirror image.
The drawing that you are working on is a geometrical design. The objects it includes are:

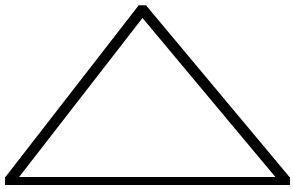
a square



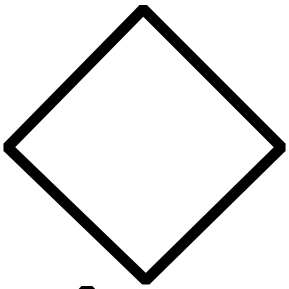
a circle



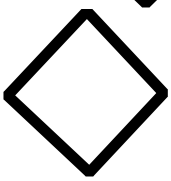
a triangle



a big diamond



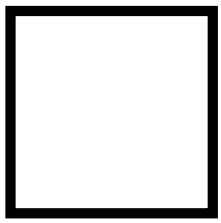
a small diamond



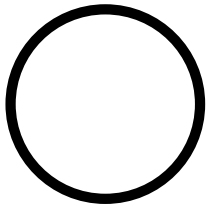
You can move these objects to the left or to the right, and you can move them above each other, or below each other.

In this section, you will see your colleague from the side. The picture you will see of him has been taken by a video camera placed slightly behind him.
The drawing that you are working on is a geometrical design. The objects it includes are:

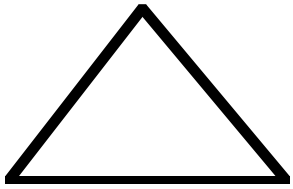
a square



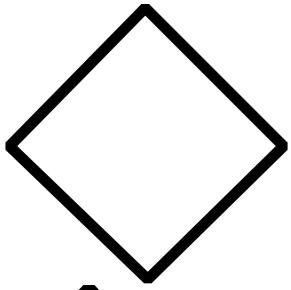
a circle



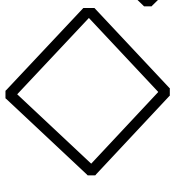
a triangle



a big diamond



a small diamond



You can move these objects to the left or to the right, and you can move them above each other, or below each other.

Références bibliographiques

[Abowd 1994]
G. D. Abowd. *Defining reference models and software architectural styles for cooperative systems*, Workshop on Software architectures for cooperative systems at CSCW'94, ACM Conference on Computer-Supported Cooperative Work, Chapel Hill, North Carolina, USA, 1994.

[Andler 1992]
D. Andler. *Introduction,* in *Introduction aux sciences cognitives.* D. Andler, (ed.) Gallimard, Paris, France, 1992.

[Apple 1988]
The Knowledge Navigator, vidéo. Apple Computer Inc., Cupertino, California, USA, 1988.

[Apple 1991]
Apple. *The Apple Event Manager*, Apple Computer Inc., Cupertino, California, USA, 1991.

[Apple 1993]
Apple. *OpenDoc Human Interface Guidelines (preliminary),* Apple Computer Inc., 1993.

[Apple 1994]
Finder 7.5. Logiciel pour Macintosh. Apple Computer Inc., Cupertino, California, USA, 1994.

[Baecker 1992]
R. M. Baecker, D. Nastos, L. R. Posner et K. L. Mawby. *The user-centred iterative design of collaborative writing software*, Workshop on Real Time Group Drawing and Writing Tools (CSCW'92, ACM Conference on Computer-Supported Cooperative Work), Toronto, Canada, 1992.

[Balbo 1994]
S. Balbo. *Evaluation ergonomique des interfaces utilisateur: un pas vers l'automatisation.* Thèse de doctorat, Université Joseph Fourier, Grenoble I, 1994.

[Ballé 1992]
C. Ballé. *Sociologie des organisations,* Presses Universitaires de France, Paris, France, 1992.

[Barlow 1994]
J. P. Barlow. *The Economy of Ideas,* in *Wired*, 2(03), mars 1994.

[Barnard 1985]
P. J. Barnard. *Cognitive Resources and the Learning of Computer Dialogs*, in *Interfacing Thought*, Cognitive aspects of Human Computer Interaction. J. M. Carroll, (ed.) MIT Press, 1985. pp. 112-158.

[Barnard 1992]
P. J. Barnard et J. May. *Real time blending of data streams: a key problem for the cognitive modelling of user behaviour with multimodal systems,* Amodeus Project, Working Paper, UM/WP 26, 1992.

[Barnard 1994]
P. J. Barnard, J. May et D. Salber. *Deixis and Points of View in Media Spaces*, Amodeus Project, Working Paper, UM/WP 19, 1994.

[Bass 1992]
L. Bass, R. Little, R. Pellegrino, S. Reed, R. Seacord, S. Sheppard et M. R. Szczur. *The UIMS Tool Developers' Workshop: A Metamodel for the Runtime Architecture of an Interactive System,* in *SIGCHI Bulletin*, 24(1), janvier 1992. pp. 32-37.

[Bass 1994]
L. Bass et G. Abowd. *Software Architecture: A Tutorial Introduction*, Software Engineering Institute, Carnegie-Mellon University, Tutorial, 1994.

[Bellotti 1994]
V. Bellotti et A. MacLean. *Integrating and Communicating Design Perspectives with QOC Design Rationale*, Amodeus Project, Working Paper, ID/WP29, 1994.

[Bentley 1992]
R. Bentley, J. A. Hughes, D. Randall, T. Rodden, P. Sawyer, D. Shapiro et I. Sommerville. *Ethnographically-informed systems design for air traffic control*, CSCW'92, ACM Conference on Computer-Supported Cooperative Work, Toronto, Canada, 1992. pp. 123-129.

[Bentley 1994]
R. Bentley, T. Rodden, P. Sawyer et I. Sommerville. *Architectural Support for Cooperative Multiuser Interfaces*, in *IEEE Computer*, 27(5), mai 1994. pp. 37-46.

[Berne 1995]
Y.-H. Berne et B. Hocq. *SIDECOM (Suivi d'individus destiné à l'extension de la communication médiatisée)*, Equipe IHM, Laboratoire de Génie Informatique, Institut IMAG, Rapport de stage, 1995.

[Bernoux 1985]
P. Bernoux. *La sociologie des organisations*, Editions du Seuil, Paris, France, 1985.

[Bernsen 1994]
N. O. Bernsen et J. Ramsay. *An executive summary of the DSD framework illustrated by two worked exemplars*, Amodeus Project, 1994.

[Bier 1991]
E. A. Bier et S. Freeman. *MMM: A User Interface Architecture for Shared Editors on a Single Screen*, UIST'91, ACM Symposium on User Interface Software and Technology, Hilton Head, South Carolina, USA, 1991. pp. 79-86.

[Bly 1993]
S. A. Bly, S. R. Harrison et S. Irwin. *Media Spaces: Bringing People Together in a Video, Audio, and Computing Environment*, in *Communications of the ACM*, 36(1), janvier 1993.

[Boersma 1994]
P. Boersma. *Experimental Research into Usability and Organisational Impact of Workflow Software*. Master's Thesis, Université de Twente, Pays-Bas, 1994.

[Bourguet 1992]
M.-L. Bourguet. *Conception et réalisation d'une interface de dialogue personne-machine multimodale*. Thèse de doctorat, Institut National Polytechnique de Grenoble, 1992.

[Brothers 1990]
L. Brothers, V. Sembugamoorthy et M. Muller. *ICICLE: Groupware for Code Inspection*, CSCW'90, ACM Conference on Computer Supported Cooperative Work, Los Angeles, California, USA, 1990. pp. 169-181.

[Bruckman 1994]
A. Bruckman. *Approaches to Managing Deviant Behaviour in Virtual Communities (Panel)*, CHI 95, ACM Conference on Human Factors in Computing Systems, Boston, Massachussets, USA, 1994. pp. 183-184.

[Bulick 1989]
S. Bulick, M. Abel, D. Corey, J. Schmidt et S. Coffin. *The US WEST Advanced Technologies Prototype Multi-media Communications System*, GLOBECOM'89, IEEE Global Telecommunications Conference, Dallas, Texas, USA, 1989.

[Buxton 1990]
W. Buxton et T. Moran. *EuroPARC's Integrated Interactive Intermedia Facility (IIIF): Early Experiences*, IFIP Conference on Multi-User Interfaces and Applications, Heraklion, Crète, 1990.

[Buxton 1993]
W. A. S. Buxton. *Telepresence: Integrated Shared Task and Person Spaces*, in *Readings in Groupware and Computer-Supported Work.* R. M. Baecker, (ed.) Morgan Kaufman, San Mateo, California, USA, 1993. pp. 816-822.

[Buxton 1994]
W. Buxton. Integrating the Periphery and Context: A New Taxonomy of Telematics, ERGO-IA'94, Biarritz, France, 1994.

[Card 1983]
S. K. Card, T. P. Moran et A. Newell. *The Psychology of Human-Computer Interaction*, Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1983.

[Charon 1993]
J.-P. Charon, L. Tézier, M. Carli et S. Liska. *Le projet Supratel : étude de l'utilisabilité d'un terminal de communication multiservices*, DESS Génie Informatique, Laboratoire de Génie Informatique, Université Joseph Fourier Grenoble 1, Rapport de stage, 1993.

[Clark 1991]
H. H. Clark et S. E. Brennan. *Grounding in Communication*, in *Perspectives on Socially Shared Cognition.* L. B. Resnick, J. M. Levine et S. D. Teasley, (eds.). American Psychological Association, Washington, DC, USA, 1991. pp. 127-149.

[Coiffier 1990]
E. Coiffier, Y. Crozet, D. Dehoux-Grafmeyer, F. Faure et J.-F. Renaud. *Sociologie basique*, Nathan, Paris, France, 1990.

[Coutaz 1987]
J. Coutaz. *PAC, an Implementation Model for Dialog Design*, Interact'87, IFIP Conference on Human-Computer Interaction, Stuttgart, 1987. pp. 431-436.

[Coutaz 1990]
J. Coutaz. *Interface homme-ordinateur : conception et réalisation*, Dunod, Paris, France, 1990.

[Coutaz 1992]
J. Coutaz, G. Abowd et L. Nigay. *Refining Software Engineering Quality Factors with HCI Factors*, HCI'92, UK, 1992.

[Coutaz 1993]
J. Coutaz, L. Nigay et D. Salber. *The MSM Framework: A Design Space for Multi-Sensori-Motor Systems*, in *Human-Computer Interaction, Third International Conference, EWHCI'93, Moscow, Russia, August 3-7, 1993, Selected Papers*. Lecture Notes in Computer Science n° 753, L. J. Bass, J. Gornostaev et C. Unger, (eds.). Springer-Verlag, Berlin, 1993. pp. 231-241.

[Coutaz 1995]
J. Coutaz, L. Nigay, D. Salber, A. E. Blandford, J. May et R. M. Y. Young. *Four Easy Pieces for Assessing the Usability of Multimodal Interaction*, INTERACT'95, IFIP Fifth International Conference on Human Computer Interaction, 25-29 juin 1995, Lillehammer, Norvège (à paraître), 1995.

[Croisy 1994]
P. Croisy. *Modèle multi-agent et conception d'applications coopératives interactives*, IHM'94, Sixièmes journées sur l'Ingénierie des Interfaces Homme-Machine, Lille, France, 1994. pp. 139-144.

[CU-SeeMe 1995]
CU-SeeMe . Logiciel pour Apple Macintosh. Cornell University, 1995.

[Cypher 1991]
A. Cypher. *EAGER: Programming Repetitive Tasks by Example*, CHI'91, ACM Conference on Human Factors in Computing Systems, New Orleans, Louisiana, USA, 1991. pp. 33-39.

[Dahlbäck 1988]
N. Dahlbäck et A. Jönsson. *Talking to a computer is not like talking to your best friend*, SCAI-88, Scandinavian Conference on Artificial Intelligence, 1988. pp. 53-68.

[Dahlbäck 1989]
N. Dahlbäck et A. Jönsson. *Empirical studies of discourse representations for natural language interfaces*, Fourth conference of the european chapter of the ACL, 1989. pp. 291-298.

[Daly-Jones 1994]
O. Daly-Jones. *Characterising the Social Salience of Electronically Mediated Communication*, CHI'94, ACM Conference on Human Factors in Computing Systems, Boston, Massachussets, USA, 1994. pp. 93-94.

[Decouchant 1994]
D. Decouchant. *Rétroaction de groupe et édition coopérative de documents structurés*, IHM'94, Sixièmes Journées sur l'Ingénierie des Interfaces Homme-Machine, Lille, France, 1994. pp. 145-150.

[Dewan 1992]
P. Dewan. *Principles of Designing Multi-User User Interfaces Development Environments*, IFIP TC2/WG2.7 Working Conference on Engineering for Human-Computer Interaction, Ellivuori, Finland, 1992. pp. 35-50.

[Diaper 1989]
D. Diaper. *The Wizard's Apprentice: A Program to Help Analyse Natural Language Dialogues*, 5th Conference of the British Computer Society HCI SIG, 1989.

[Dix 1993]
A. Dix, J. Finlay, G. Abowd et R. Beale. *Human-Computer Interaction*, Prentice Hall, New York, 1993.

[Dorner 1995]
Eudora 1.5.1. Logiciel pour Apple Macintosh. Steve Dorner & Qualcomm, 1995.

[Dourish 1992]
P. Dourish et S. A. Bly. *Portholes: Supporting Awareness in a Distributed Work Group*, CHI'92, ACM Conference on Human Factors in Computing Systems, Monterey, California, USA, 1992. pp. 541-547.

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

[Dowell 1989]
J. Dowell et J. Long. *Towards a conception for an engineering discipline of human factors*, in *Ergonomics*, 32(11), pp. 1513-1535.

[Duke 1995]
D. J. Duke, (editor), A. Aboulafia, A. E. Blandford, S. J. Buckingham-Shum, J. Darzentas, J. May, L. Nigay, J. Ramsay, D. Salber et S. Verjans. *Integration Techniques for Multi-Disciplinary HCI Modelling: A Survey*, Projet européen ESPRIT BRA 7040 AMODEUS 2, RP3 Working Paper (en cours de soumission), ID/WP, 1995.

[Egido 1988]
C. Egido. *Videoconferencing as a Technology to Support Group Work: A Review of its Failures*, CSCW'88, ACM Conference on Computer-Supported Cooperative Work, 1988. pp. 13-24.

[Ellis 1991]
C. A. Ellis, S. J. Gibbs et G. L. Rein. *Groupware: some issues and experiences*, in *Communications of the ACM*, 34(1), janvier 1991. pp. 38-58.

[Ellis 1994] (chapitre 1)
C. Ellis et J. Wainer. *A Conceptual Model of Groupware*, CSCW 94, ACM Conference on Computer Supported Cooperative Work, Chapel Hill, North Carolina, USA, 1994. pp. 79-88.

[Ellis 1994] (chapitre 6)
C. Ellis. *Keepers, Synchronizers, Communicators and Agents*, Workshop on Software architectures for cooperative systems at CSCW'94, ACM Conference on Computer-Supported Cooperative Work, Chapel Hill, North Carolina, USA, 1994.

[Fish 1992]
R. S. Fish, R. E. Kraut, R. W. Root et R. E. Rice. *Evaluating Video as a Technology for Informal Communications*, CHI'92, ACM Conference on Human Factors in Computing Systems, Monterey, California, USA, 1992. pp. 32-48.

[Flores 1988]
F. Flores, M. Graves, B. Hartfield et T. Winograd. *Computer Systems and the Design of Organizational Interaction*, in *ACM Transactions on Office Information Systems*, 6(2), avril 1988. pp. 153-172.

[France 1995]
E. France. *UK Data Protection Registrar, Keynote speech*, Ethicomp'95, An International Conference on the Ethical Issues of Using Information Technology, Leicester, UK, 1995.

[Freeman 1993]
S. M. G. Freeman. *An Architecture for Distributed User Interfaces.* Ph.D Thesis, Darwin College, University of Cambridge, 1993.

[Garfinkel 1989]
D. Garfinkel. *The Shared X Multiuser Interface User's Guide*, Hewlett-Packard, Research Report, STL-TM-89-07, 1989.

[Garlan 1993]
D. Garlan et M. Shaw. *An introduction to software architecture*, in *Advances in Software Engineering and Knowledge Engineering.* World Scientific Publishing Company, 1993.

[Gaver 1992] (chapitres 2 et 8)
W. W. Gaver, T. Moran, A. MacLean, L. Lövstrand, P. Dourish, K. Carter et W. Buxton. *Realizing a Video Environment: EuroPARC's RAVE System*, CHI'92, ACM Conference on Human Factors in Computing Systems, Monterey, California, USA, 1992. pp. 27-36.

[Gaver 1992] (chapitre 4)
W. W. Gaver. *The Affordances of Media Spaces for Collaboration*, Proceedings of ACM CSCW'92 Conference on Computer-Supported Cooperative Work, Toronto, Canada, 1992. pp. 17-24.

[Gaver 1993]
W. W. Gaver, A. Sellen, C. Heath et P. Luff. *One is not Enough: Multiple Views in a Media Space*, InterCHI'93, ACM/IFIP Conference on Human Factors in Computing Systems, Amsterdam, Pays-Bas, 1993. pp. 335-341.

[Gaver 1995]
W. W. Gaver, G. Smets et K. Overbeeke. *A Virtual Window on Media Space*, CHI'95, ACM Conference on Human Factors in Computing Systems, Denver, Colorado, USA, 1995. pp. 257-264.

[Gibson 1979]
J. J. Gibson. *The ecological approach to visual perception*, Houghton Mifflin, New York, New York, USA, 1979.

[Godwin 1994]
M. Godwin. *How to make virtual communities work*, in *Wired*, 2(6), juin 1994. pp. 92.

[Gould 1987]
J. D. Gould, S. J. Boies, S. Levy, J. T. Richards et J. Schoonard. *The 1984 Olympic message system: a test of behavioral principles of system design*, in *Communications of the ACM*, 30(9),

[Grudin 1989]
J. Grudin. *Why groupware applications fail: problems in design and evaluation*, in *Office: Technology and People*. Elsevier, 1989. pp. 245.

[Grudin 1994]
J. Grudin. *CSCW: History and Focus*, in *IEEE Computer*, 27(5), mai 1994. pp. 19-26.

[Hammontree 1992]
M. L. Hammontree, J. J. Hendrickson et B. W. Hensley. *Integrated data capture and analysis tools for research and testing on graphical user interfaces*, CHI'92, ACM Conference on Human Factors in Computing Systems, Monterey, California, USA, 1992. pp. 431-432.

[Harper 1992]
R. Harper. *Looking at Ourselves: An Examination of the Social Organization of Two Research Laboratories*, CSCW'92, ACM Conference on Computer-Supported Cooperative Work, Toronto, Canada, 1992. pp. 330-337.

[Harrison 1994]
B. Harrison, M. Mantei, G. Beirne et T. Narine. *Communicating About Communicating: Cross-Disciplinary Design of a Media Space Interface*, CHI'94, ACM Conference on Human Factors in Computing Systems, Boston, Massachussets, USA, 1994. pp. 124-130.

[Heath 1991]
C. Heath et P. Luff. *Collaborative Activity and Technological Design: Task Coordination in the London Underground Control Rooms*, ECSCW'91, European Conference on Computer-Supported Cooperative Work, Amsterdam, Pays-Bas, 1991.

[Heath 1992]
C. Heath et P. Luff. *Media Space and Communicative Asymmetries: Preliminary Observations of Video-Mediated Interaction*, in *Human-Computer Interaction*, 7(3), 1992. pp. 315-246.

[Hill 1992]
R. D. Hill. *The Abstraction-Link-View Paradigm: Using Constraints to Connect User Interfaces to Applications*, CHI'92, ACM Conference on Human Factors in Computing Systems, Monterey, California, USA, 1992. pp. 335-342.

[Hiltz 1978]
S. R. Hiltz et M. Turoff. *The Network Nation*, MIT Press, Cambridge, Massachussets, USA, 1978.

[Hix 1993]
D. Hix et H. R. Hartson. *Developing User Interfaces, Ensuring Usability Through Product & Process*, John Wiley & Sons, New York, New York, 1993.

[Hollan 1992]
J. Hollan et S. Stornetta. *Beyond Being There*, CHI'92, ACM Conference on Human Factors in Computing Systems, Monterey, California, USA, 1992. pp. 119-125.

[Insignia 1994]
SoftWindows 1.0. Logiciel pour Macintosh. Insignia Solutions, 1994.

[Ishii 1992]
H. Ishii et M. Kobayashi. *ClearBoard: A Seamless Medium for Shared Drawing and Conversation with Eye Contact*, CHI'92, ACM Conference on Human Factors in Computing Systems, Monterey, California, USA, 1992. pp. 525-532.

[Jambon 1994]
F. Jambon et L. Karsenty. *Formalisation des interfaces et travail coopératif : quelles conséquences ?*, IHM 94, Sixièmes Journées sur l'Ingénierie des Interfaces Homme-Machine, Lille, France, 1994. pp. 163-168.

[Johnson 1994]
D. G. Johnson. *Computer Ethics*, Prentice Hall, 1994.

[Karsenty 1994]
A. Karsenty. *GroupDesign : un collecticiel synchrone pour l'édition partagée de documents*. Thèse de doctorat, Université d'Orsay Paris-Sud, 1994.

[Kazman 1994]
R. Kazman, L. Bass, G. Abowd et M. Webb. *SAAM: A Method for Analyzing the Properties of Software Architectures*, ICSE-16, Sorrento, Italy, 1994.

[Koegel Buford 1994]
J. F. Koegel Buford. *Middleware System Services Architecture*, in *Multimedia Systems*. J. F. Koegel Buford, (ed.) Addison-Wesley, New York, New York, USA, 1994. pp. 221-244.

[Kosbie 1994]
D. S. Kosbie. *Hierarchical Events in Graphical User Interfaces*, CHI'94, ACM Conference on Human Factors in Computing Systems, Boston, Massachussets, USA, 1994. pp. 131-132.

[Krakowiak 1990]
S. Krakowiak, M. Meysembourg, H. Nguyen Van, M. Riveill et C. Roisin. *Design and Implementation of an object-oriented strongly typed language for distributed applications*, in *Journal of Object-Oriented Programming*, septembre 1990.

[Lewis 1991]
C. Lewis, P. Polson, C. Wharton et J. Rieman. *Testing a Walkthrough Methodology for Theory-Based Design of Walk-Up-and-Use Interfaces*, CHI'91, ACM Conference on Human Factors in Computing Systems, New Orleans, Louisiana, USA, 1991. pp. 235-242.

[Lewis 1994]
P. H. Lewis. *An Ad (Gasp!) in Cyberspace*, in *The New York Times*, 1994(19/4), avril 1994. pp. D1:2.

[Lischetti 1994]
N. Lischetti. *SupraAnalyse : un outil d'aide à l'analyse de l'utilisabilité. Application à un terminal de télécommunication multiservices*. Mémoire CNAM, Laboratoire de Génie Informatique, Université Joseph Fourier Grenoble 1, 1994.

[Long 1989]
J. Long et J. Dowell. *Conceptions of the Discipline of HCI: Craft, Applied Science, and Engineering*, Fifth Conference of the British Computer Society HCI SIG, 1989.

[Lövstrand 1991]
L. Lövstrand. *Being Selectively Aware with the Khronika System*, ECSCW'91, European Conference on Computer Supported Cooperative Work, Amsterdam, Pays-Bas, 1991.

[Mackay 1991]
W. E. Mackay. *Triggers and Barriers to Customizing Software*, Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems, New Orleans, Louisiana, USA, 1991. pp. 153-160.

[Mackay 1995]
W. E. Mackay. *Ethics, Lies and Videotape…*, CHI'95, ACM Conference on Human Factors in Computing Systems, Denver, Colorado, USA, 1995. pp. 138-145.

[Mackinlay 1991]
J. D. Mackinlay, G. G. Robertson et S. K. Card. *The Perspective Wall: Detail and Context Smoothly Integrated*, CHI'91, ACM Conference on Human Factors in Computing Systems, New Orleans, Louisiana, USA, 1991. pp. 173-179.

[MacLean 1991]
A. MacLean, R. M. Young, V. Bellotti et T. Moran. *Questions, Options and Criteria: Elements of Design Space Analysis*, in *Human-Computer Interaction*, 6(3 & 4), 1991. pp. 201-250.

[MacLeod 1993]
M. MacLeod. *DRUM, Diagnostic Recorder for Usability Measurement*, NPL, DITC HCI Group, Teddington, UK, 1993.

[MacroMedia 1993]
Director 3.1.1. Logiciel pour Apple Macintosh. MacroMedia Inc., 1993.

[Malone 1986]
T. W. Malone, K. R. Grant et F. A. Turbak. *The Information Lens: An Intelligent System for Information Sharing in Organizations*, CHI'86, ACM Conference on Human Factors in Computing Systems, 1986. pp. 1-8.

[Maner 1995]
W. Maner. *Unique Ethical Problems in Information Technology*, Ethicomp'95, An International Conference on the Ethical Issues of Using Information Technology, Leicester, UK, 1995.

[Mantei 1989]
M. Mantei. *Observation of Executives Using a Computer Supported Meeting Environment*, in *Readings in Groupware and Computer-Supported Cooperative Work*. R. M. Baecker, (ed.) Morgan Kaufman, San Mateo, California, US, 1989. pp. 695-708.

[Mantei 1991]
M. Mantei, R. M. Baecker, A. Sellen, W. Buxton, T. Milligan et B. Wellman. *Experiences in the use of a Media Space*, CHI'91, ACM Conference on Human Factors in Computing Systems, New Orleans, Louisiana, USA, 1991. pp. 203-208.

[Mason 1986]
R. O. Mason. *Four Ethical Issues of the Information Age*, in *MIS Quarterly*, 10(1), janvier 1986. pp. 486-498.

[Maulsby 1993]
D. Maulsby, S. Greenberg et R. Mander. *Prototyping an Intelligent Agent through Wizard of Oz*, InterCHI'93, ACM/IFIP Conference on Human Factors in Computing Systems, Amsterdam, Pays-Bas, 1993. pp. 277-284.

[McCall 1977]
J. McCall. *Factors in Software Quality*, General Electric Eds., 1977.

[McCarthy 1994]
J. C. McCarthy et A. F. Monk. *Channels, conversation, cooperation and relevance: all you wanted to know about communication but were afraid to ask*, in *Collaborative Computing*, 1(1), mars 1994. pp. 35-60.

[McGurk 1976]
H. McGurk et J. MacDonald. *Hearing lips and seeing voices*, in *Nature*, numéro(264), 1976. pp. 746-748.

[McQuail 1987]
D. McQuail. *Mass Communication Theory, An Introduction*, SAGE Publications, London, UK, 1987.

[Metral 1994]
M. Metral. *MAXIMS: A Learning Interface Agent For Eudora (A User's Guide to the System)*, MIT Media Lab, Technical report, 1994.

[Meyer 1990]
B. Meyer. *Conception et Programmation par Objets*, InterEditions, Paris, France, 1990.

[Mignot 1993]
C. Mignot, C. Valot et N. Carbonell. *An Experimental Study of Future 'Natural' Multimodal Human-Computer Interaction*, InterCHI'93, ACM/IFIP Conference on Human Factors in Computing Systems, Amsterdam, Pays-Bas, 1993. pp. 67-68.

[Minneman 1991]
S. L. Minneman et S. A. Bly. *Managing a trois: A Study of a Multi-User Drawing Tool in Distributed Design Work*, CHI'91, ACM Conference on Human Factors in Computing Systems, New Orleans, Louisiana, USA, 1991. pp. 217-224.

[Moor 1985]
J. H. Moor. *What is Computer Ethics ?*, in *Metaphilosophy*, 16(4), octobre 1985. pp. 226-275.

[Myst 1993]
Myst 1.0. Logiciel pour Macintosh (CD-ROM). 1993.

[Neumann 1995]
P. G. Neumann. *Computer-related Risks*, Addison-Wesley, 1995.

[Nielsen 1994]
J. Nielsen. *Usability Engineering*, Academic Press, Boston, Massachussets, USA, 1994.

[Nigay 1994] (chapitre 1)
L. Nigay, J. Coutaz et D. Salber. *Initial draft preliminary on-the-fly PAC analysis for the ECOM Interface*, Projet européen ESPRIT BRA 7040 AMODEUS 2, Integration Report, SM/IR 4, 1994.

[Nigay 1994] (chapitres 3, 4, 5, 6 et 7)
L. Nigay. *Conception et réalisation des systèmes interactifs: Application aux Interfaces Multimodales*. Thèse de doctorat, Université Joseph Fourier Grenoble I, 1994.

[Norman 1988]
D. Norman. *The Psychology of Everyday Things*, Basic Books Publishing, 1988.

[Okada 1994]
K.-i. Okada, F. Maeda, Y. Ichikawaa et Y. Matsushita. *Multiparty Videoconferencing at Virtual Social Distance: MAJIC Design*, CSCW'94, ACM Conference on Computer-Supported Cooperative Work, Chapel Hill, North Carolina, USA, 1994. pp. 385-393.

[Orwell 1950]
G. Orwell. *1984,* Gallimard, Paris, France, 1950.

[Ott 1993]
M. Ott, J. P. Lewis et I. Cox. *Teleconferencing Eye Contact Using a Virtual Camera*, InterCHI'93, ACM/IFIP Conference on Human Factors in Computing Systems, Amsterdam, Pays-Bas, 1993. pp. 109-110.

[Ouadou 1994]
K. E. Ouadou. *AMF : un modèle d'architecture multi-Agents Multi-Facettes pour interfaces homme-machine et les outils associés.* Thèse de Doctorat, Ecole Centrale de Lyon, 1994.

[Pagani 1993]
D. Pagani et W. E. Mackay. *Bringing Media Spaces into the Real World*, ECSCW'93, European Conference on Computer-Supported Cooperative Work, Milan, Italie, 1993.

[Patterson 1990]
J. F. Patterson, R. D. Hill, S. L. Rohall et W. S. Meeks. *Rendezvous: An Architecture for Synchronous Multi-User Applications*, CSCW'90, ACM Conference on Computer-Supported Cooperative Work, Los Angeles, California, USA, 1990. pp. 317-328.

[Patterson 1994]
J. F. Patterson. *A Taxonomy of Architectures for Synchronous Groupware Applications*, Workshop on Software architectures for cooperative systems at CSCW'94, ACM Conference on Computer-Supported Cooperative Work, Chapel Hill, North Carolina, USA, 1994.

[Platt 1995]
R. G. Platt et B. Morrison. *Ethical and Social Implications of the Internet*, Ethicomp'95, An International Conference on the Ethical Issues of Using Information Technology, Leicester, UK, 1995.

[Polity 1990]
Y. Polity, J.-M. Francony, R. Palermiti, P. Falzon et S. Kazma. *Recueil de dialogues homme-machine en langue naturelle écrite,* Criss, Les cahiers du Criss n°17, 1990.

[Poltrock 1995]
S. Poltrock et J. Grudin. *Groupware and Workflow: A Survey of Systems and Behavioral Issues*, Tutorial at CHI'95, ACM Conference on Human Factors in Computing Systems, Denver, Colorado, USA, 1995.

[Reinhardt 1993]
A. Reinhardt. *Video Conquers the Desktop,* in *BYTE*, 18(9), septembre 1993. pp. 64-80.

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

[Roseman 1992]
M. Roseman et S. Greenberg. *GROUPKIT: A Groupware Toolkit for Building Real-Time Conferencing Applications*, CSCW'92, ACM Conference on Computer-Supported Cooperative Work, Toronto, Canada, 1992. pp. 43-50.

[Rouncefield 1994]
M. Rouncefield, J. A. Hughes, T. Rodden et S. Viller. *Working with "Constant Interruption": CSCW and the Small Office*, CSCW 94, ACM Conference on Computer-Supported Cooperative Work, Chapel Hill, North Carolina, USA, 1994. pp. 275-286.

[Salber 1994] (chapitre 1)
D. Salber et J. Coutaz. *Taxinomie des mécanismes de connexion pour la communication homme-machine-homme*, IHM'94, Sixièmes Journées sur l'Ingénierie des Interfaces Homme-Machine, Lille, France, 1994. pp. 183-189.

[Salber 1994] (chapitre 2)
D. Salber et J. Coutaz. *Fenêtres sur groupes: des MediaSpaces pour collaborer et communiquer*, 3èmes Journées Internationales sur L'Interface des Mondes Réels et Virtuels '94, Montpellier, France, 1994. pp. 309-318.

[Salber 1994] (chapitre 3)
D. Salber, J. Coutaz, L. Nigay, (editors), G. Faconti, F. Paterno', D. Duke et M. Harrison. *The System Modelling Glossary*, Amodeus Project, System Modelling Working Paper, SM/WP 26, 1994.

[Salber 1995]
D. Salber, J. Coutaz, D. Decouchant et M. Riveill. *De l'observabilité et de l'honnêteté : le cas du contrôle d'accès dans la Communication Homme-Homme Médiatisée*, soumis à IHM'95, Conférence sur l'Ingénierie des Interfaces Homme-Machine, Toulouse, France, 1995.

[Savetz 1994]
K. Savetz. *Internet Fax FAQ*, disponible sur Internet, FAQ, 1994.

[Scapin 1990]
D. L. Scapin. *Des critères ergonomiques pour l'évaluation et la conception d'interfaces utilisateur*, XXVIè Congrès de la SELF, Montréal, Canada, 1990.

[Schneiderman 1995]
B. Schneiderman. *The Info Superhighway: For The People*, in *Communications of the ACM*, 38(1), janvier 1995. pp. 162.

[Sellen 1992]
A. J. Sellen. *Speech Patterns in Video-Mediated Conversations*, CHI'92, ACM Conference on Human Factors in Computing Systems, Monterey, California, USA, 1992. pp. 49-59.

[Senach 1990]
B. Senach. *Evaluation ergonomique des interfaces homme-machine : une revue de la littérature*, INRIA, Programme 8, Communication Homme-Machine, n° 1180, 1990.

[Shackel 1991]
B. Shackel et S. Richardson. *Human Factors for Informatics Usability*, Cambridge University Press, 1991.

[Shaw 1995]
M. Shaw et D. Garlan. *Software Architecture. Perspectives on an Emerging Discipline*, Prentice Hall, 1995.

[Smith 1986]
S. L. Smith et J. N. Mosier. *A design evaluation checklist for user-system interface software*, The MITRE Corporation, Bedford, Massachussets, USA, #MTR-9480 EDS_TR_84-358, 1986.

[Smolensky 1992]
P. Smolensky. *IA connexionniste, IA symbolique et cerveau*, in *Introduction aux sciences cognitives*. D. Andler, (ed.) Gallimard, Paris, 1992. pp. 77-106.

[Stefik 1988]
M. Stefik, G. Foster, D. G. Bobrow, K. Kahn, S. Lanning et L. Suchman. *Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings*, in *Computer-Supported Cooperative Work: A Book of Readings*. I. Greif, (ed.) Morgan Kaufman, San Mateo, California, 1988. pp. 335-366.

[Stults 1986]
R. Stults. *MediaSpace*, Xerox PARC, Rapport technique, 1986.

[Suchman 1987]
L. Suchman. *Plans and Situated Actions*, Cambridge University Press, Cambridge, 1987.

[Tang 1991]
J. C. Tang et S. L. Minneman. *VideoWhiteboard: Video Shadows to Support Remote Collaboration*, CHI'91, ACM Conference on Human Factors in Computing Systems, New Orleans, Louisiana, USA, 1991. pp. 315-322.

[Tang 1994]
J. C. Tang et M. Rua. *Montage: Providing Teleproximity for Distributed Groups*, CHI'94, ACM Conference on Human Factors in Computing Systems, Boston, Massachussets, USA, 1994. pp. 37-43.

[Thompson 1967]
J. D. Thompson. *Organizations in Action*, McGraw-Hill, New York, New York, USA, 1967.

[Tognazzini 1991]
B. Tognazzini. *On High-Altitude Computing*, in *Apple Directions*, 1991.

[Tokuda 1994]
H. Tokuda. *Operating System Support for Continuous Media Applications*, in *Multimedia Systems*. J. F. Koegel Buford, (ed.) Addison-Wesley, New York, New York, USA, 1994. pp. 201-220.

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

[Watabe 1990]
K. Watabe, S. Sakata, K. Maeno, H. Fukuoka et T. Ohmori. *Distributed Multiparty Desktop Conferencing System: MERMAID*, CSCW'90, ACM Conference on Computer Supported Cooperative Work, Los Angeles, California, USA, 1990. pp. 27-38.

[Weber 1947]
M. Weber. *The Theory of Social and Economic Organization*, Oxford University Press, New York, New York, USA, 1947.

[Wylie 1970]
L. Wylie. *Chanzeaux, village d'Anjou*, Gallimard, Paris, France, 1970.

[Young 1994]
R. M. Young et G. D. Abowd. *Multi-perspective Modelling of Interface Design Issues: Undo in a Collaborative Editor*, in *People and Computers IX: Proceedings of HCI'94*. G. Cockton, S. W. Draper et G. R. S. Weir, (eds.). Cambridge University Press, Cambridge, UK, 1994. pp. 249-260.

[Zimmerman 1993]
PGP (Pretty Good Privacy) 2.3ui. Logiciel pour Macintosh. Phil Zimmerman, 1993.

Table des figures

Table des matières

Résumé

Cette thèse s'inscrit dans le domaine de l'ingénierie des systèmes multi-utilisateurs. Dans notre modèle conceptuel, un système multi-utilisateur repose sur la combinaison de trois espaces fonctionnels : l'espace de production, l'espace de coordination et l'espace de communication. Ce dernier espace, la communication entre individus, fait l'objet de notre étude et définit la communication homme-homme médiatisée. L'approche adoptée s'articule en trois niveaux : les principes, issus des sciences non-informatiques (psychologie, sociologie, éthique, etc.), les propriétés issues des principes et destinées à guider la conception et évaluer la réalisation, et enfin les techniques de mise en œuvre informatique.
Les *principes* traduisent les contributions de sciences non-informatiques comme les sciences sociales ou la psychologie cognitive à notre domaine d'étude. L'expérimentation psychologique Garden Movie, qui étudie l'influence de la disposition des caméras et de la surface de travail dans une tâche collecticielle, illustre l'utilisation des principes. Les *propriétés* sont des caractéristiques objectives et vérifiables d'un système informatique dont le choix est guidé par les principes. Nous proposons des propriétés pour les systèmes multi-utilisateurs et les systèmes de communication homme-homme médiatisée. Les propriétés permettent de guider la conception ainsi que l'étude de l'utilisabilité d'un système. Nous présentons la plate-forme d'observation du comportement des utilisateurs et Magicien d'Oz NEIMO pour l'étude expérimentale de l'utilisabilité. Nous illustrons son utilisation pour les systèmes de communication homme-homme médiatisée avec l'expérience Supratel. Les *techniques* comprennent deux volets : les modèles d'architecture logicielle guident la réalisation et les outils permettent la réalisation effective. Le choix des techniques est guidé par les propriétés que le système doit vérifier. Nous présentons une grille d'analyse des modèles d'architecture logicielle pour les systèmes multi-utilisateurs que nous utilisons pour évaluer les modèles proposés dans la littérature. Constatant qu'aucun n'intégre harmonieusement les trois espaces de notre modèle conceptuel et l'insuffisance des modèles pour la communication homme-homme médiatisée, nous présentons CoPAC, un modèle d'architecture logicielle pour les systèmes multi-utilisateurs et la communication médiatisée. Nous illustrons sa mise en œuvre avec la réalisation de notre mediaspace VideoPort. En ce qui concerne les outils, nous décrivons la réalisation de la bibliothèque UserLink pour la communication de médias continus. Nous proposons aussi la taxonomie IMPACT pour l'analyse des outils de communication homme-homme médiatisée.

Mots clés

Système multi-utilisateur, Interface Homme-Machine, Communication homme-homme médiatisée, Modèle d'architecture logicielle, Mediaspace, Propriétés, Évaluation, Collecticiel

Abstract

This thesis focuses on the design and implementation of multi-user systems. Our conceptual model describes a multi-user system as relying on the combination of three functional spaces: the production space, the coordination space, and the communication space. We concentrate on the latter, which covers communication between users and defines computer-mediated communication. Our approach is made up of three levels: principles are defined by social and psychological sciences, properties are derived from principles and drive the design and evaluation phases, techniques are used to effectively build the system.
Principles express contributions from the social and psychological sciences to our field of study. The Garden Movie psychology experiment studies the influence of camera and work area positions for a cooperative task. We use it to illustrate the use of principles. *Properties* are features of a system that can be verified. The choice of a relevant set of properties is driven by principles. We propose properties for multi-user and computer-mediated communication systems. Properties drive the design and the usability study of a system. We present the NEIMO user behaviour observation and Wizard of Oz platform. We illustrate its use for the study of computer-mediated communication systems with the Supratel system. *Techniques* may be software architecture models used to drive the implementation or implementation tools. Properties that the system should verify drive the choice of techniques. We present a problem space for software architecture models for multi-user systems and we use it to evaluate models proposed in the litterature. We note that no model actually covers harmoniously the three spaces of our conceptual model and that the models are not satisfying for computer-mediated communication. Thus, we present CoPAC, our software architecture model for multi-user systems and computer-mediated communication. We illustrate the use of CoPAC with the implementation of VideoPort, our mediaspace system. We also describe the UserLink library for continuous media. Finally, we propose the IMPACT classification for computer-mediated communication tools.

Keywords

Multi-user system, Human-computer interaction, Computer-mediated communication, Software architecture model, Mediaspace, Properties, Evaluation, CSCW, Groupware