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.

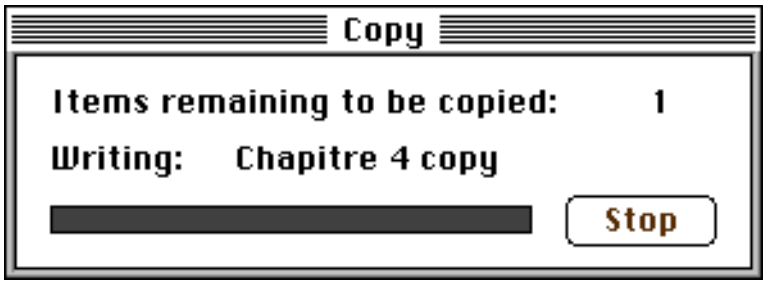

Figure 4.1. Boîte de dialogue affichant la progression d'une opération de copie. En haut, le système rend observable le nombre d'éléments restant à copier. Cependant la barre de progression est ambiguë car l'utilisateur n'a pas de moyen de savoir quelle variable interne elle représente.

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.

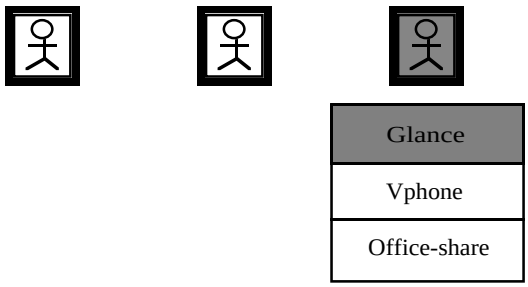

Figure 4.2. Interface permettant d'établir des connexions audio/vidéo dans un mediaspace. Lorsque l'utilisateur clique sur l'icône représentant un autre utilisateur, un menu lui indique les types de connexions que l'utilisateur distant a autorisé, ici le vphone et l'office-share.

  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.