Vers des outils formels de description des taches orientes conception d'interfaces

Dominique Scapin

INRIA

UNITÉ DE RECHERCHE
INRIA-ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
BP 105
78153 Le Chesnay Cedex
France
Tél. (1) 39 63 55 11

Rapports de Recherche

N° 893

VERS DES OUTILS FORMELS DE DESCRIPTION DES TACHES ORIENTES CONCEPTION D'INTERFACES

Programme 8

Dominique Louis SCAPIN

Septembre 1988

RR_0893

Vers des Outils Formels de Description des Tâches Orientés Conception d'Interfaces

Towards Interface-Design-Oriented Tasks Description Tools

Dominique Louis Scapin

...scap@ergo-archie.inria.fr

Programme 8

Septembre 1988

Résumé

Afin d'explorer le problème général de la prise en compte de l'ergonomie dans la conception d'interfaces utilisateurs, et en particulier les aides que peut apporter l'ergonomie aux concepteurs d'interfaces, cet article tente de faire le point sur les diverses approches existantes et de proposer quelques directions d'études. Après quelques notions sur les méthodes informatiques et sur le processus de conception, quelques uns des problèmes posés par les recommandations ergonomiques seront évoqués, puis, après un examen rapide des modèles en ergonomie, on montrera la nécessité d'une meilleure prise en compte des caractéristiques des tâches, préalablement à la conception. Enfin, à partir de ces diverses considérations, on présentera une ébauche de formalisme de description des tâches, en particulier pour des tâches de bureau. Ce formalisme, proche de la planification hiérarchique donnera lieu à une méthode de recueil de données et sera discuté quant à ses extensions possibles et à son évaluation.

Mots clefs: *ergonomie du logiciel - interfaces utilisateurs - conception et évaluation d'interfaces - modèles de l'interaction homme-ordinateur - modélisation des tâches - méthodes d'analyse*

Abstract

This paper attempts first to review and evaluate the various contributions in the field of user interfaces design, and second, to offer some research directions and identify tools for a better involvement of human factors engineering in the interface design process. Following a few remarks about computer science design methods, and about the design process as it is described in the literature, the various drawbacks of human factors recommendations and guidelines are analyzed, and a short review of existing cognitive models in the area of human-computer interaction is provided. These various considerations support the conclusion that a major difficulty in the involvement of human factors in the design of user interfaces concerns the analysis and description of tasks, prior to the design. A preliminary description model is provided, using office tasks.as an example domain, for the conceptual analysis of tasks and interfaces. This model, partly based on hierarchical planning, is discussed for further evaluation and improvements.

Key-words *: software psychology - user interfaces - design and evaluation of interfaces - human-computer interaction models - task modeling - analysis methods*

1. Introduction

L'objectif de ce papier est de poser le problème des outils conceptuels et techniques pour une meilleure insertion de l'ergonomie des logiciels dans la conception d'interfaces utilisateurs. Bien que le domaine de l'ergonomie des logiciels ait beaucoup progressé ces dernières années, il demeure encore insuffisamment pris en compte dans la conception d'interfaces. En effet, si le corpus des résultats disponibles s'est beaucoup élargi, si on est passé de l'identification de problèmes de recherche et de quelques résultats (cf. Scapin, 1981) à un ensemble très large de résultats applicables (cf. Smith and Mosier, 1984a, Scapin, 1987), il reste que cet effort a été plutôt d'ordre empirique, guère accompagné de théorisations satisfaisantes, mais surtout (et ceci explique peut être cela), la prise en compte de l'ergonomie des logiciels au sein même du processus de conception d'interfaces n'en est qu'à ses débuts.
Nous nous sommes attaché ici à identifier les obstacles (et quelques directions de recherche de solutions) à une bonne insertion de l'ergonomie en conception d'interfaces. Notre hypothèse essentielle est que ces obstacles tiennent en partie aux lacunes de théorisation, de formalisation, et par voie de conséquence de méthodes, en particulier dans le domaine de la description des tâches. En effet, on (et surtout le concepteur d'interface) ne dispose pas d'outils permettant, à partir de techniques d'analyse[1] du travail centrées sur les représentations d'utilisateurs individuels, de disposer de descriptions stables et complètes de l'activité, base primordiale pour la conception d'interfaces. Pour comprendre la situation de l'ergonomie des logiciels vis-à-vis du processus de conception d'interfaces, et pour ensuite envisager de concevoir des aides appropriées, il convient de s'interroger sur la prise en compte de l'ergonomie dans les méthodes informatiques, sur la manière dont les concepteurs identifient et règlent les problèmes liés à l'utilisateur, ainsi que d'identifier les apports de l'ergonomie du point de vue des recommandations applicables et du point de vue des théories et modèles existants. Une fois ce constat établi, on présentera quelques concepts et définitions d'un formalisme de description des tâches orienté conception d'interfaces.

2. La conception d'interfaces

2.1 Les méthodes informatiques

Le fait que l'utilisateur (du point de vue de ses caractéristiques propres et du point de vue des tâches) ne soit pas pris en compte dans les méthodes de conception informatiques est clairement démontré dans (Barthet, 1986). L'auteur identifie deux groupes de méthodes

[1] Il ne s'agit pas d'analyse au sens de l'analyse informatique traditionnelle, en termes de systèmes d'information, mais d'analyse des représentations de l'utilisateur, dans l'objectif de les traduire en termes d'interfaces utilisateurs.

utilisées actuellement: l'un (MERISE et Structured Design) qui donne peu ou pas de spécifications sur l'interface; l'autre (AXIAL, Faulle, APEL et USE) qui, s'il aboutit à des spécifications complètes de l'interface, le fait sans utiliser de critères explicites liés à l'utilisateur, et en identifiant une et une seule interface, sans variabilité aucune. Les travaux de Barthet (1986, op. cit.), Barthet et Sibertin-Blanc (1986), s'ils n'ont pas encore donné lieu à une méthode largement utilisée par les concepteurs, représentent une tentative intéressante de mise en relation de l'ergonomie et de la conception en informatique. La méthode distingue en effet les aspects représentation conceptuelle, représentation externe et représentation interne, ce qui est proche des distinctions introduites par Norman (1983). Cependant, les distinctions entre procédures prescrites, procédures effectives et procédures globales semblent peu claires. En particulier, aucun lien n'est présenté avec une méthode de recueil de données. Par ailleurs, cette méthode a pour extension prometteuse, en particulier pour le prototypage, une modélisation de la structure de contrôle d'une application (Sibertin-Blanc, 1988) par le moyen de Réseaux de Petri à Objets. Si donc les méthodes informatiques ne prennent pas encore en compte explicitement les caractéristiques de l'utilisateur et sa représentation des tâches, comment les concepteurs s'y prennent-ils pour concevoir les interfaces? Quelques études ergonomiques ont porté sur cette question.

2.2 Les études sur l'activité de conception d'interfaces

Hammond et al. (1983) ont constaté que, contrairement à l'idée reçue, les concepteurs prennent au sérieux la définition et la formulation de la tâche. Ils cherchent donc à tenir compte des problèmes liés aux facteurs humains mais dans la plupart des cas, ignorent les ressources disponibles pour les résoudre[2]. Ainsi, ils sont dépendants de leur analyse logique de la tâche qui sera accomplie avec le système qu'ils vont concevoir et de leurs intuitions sur les besoins et les capacités logiques des utilisateurs, ainsi que sur la façon dont ces derniers structurent leurs buts. Le problème majeur reste que les intuitions des différents concepteurs varient considérablement sur ce qui est nécessaire dans une situation particulière. Ceci est illustré par une comparaison de systèmes différents destinés à servir aux mêmes buts pour les mêmes utilisateurs (Roberts and Moran, 1983). Mais l'analyse logique de la tâche par les concepteurs n'est pas obligatoirement en accord avec les conceptualisations des utilisateurs. Les concepteurs semblent plus rigoureux sur la décomposition de la tâche logique que sur la considération des connaissances, des procédures et des buts des utilisateurs. Dans la plupart des cas, les concepteurs possèdent des "modèles de l'utilisateur", qui ne leur permettent que de répondre à des questions très générales (e.g., quels termes techniques utilisent-ils, quelle sorte de disposition d'écran préfèrent-ils, ont-ils déjà travaillé avec un micro-ordinateur?). Cependant, les concepteurs

[2] e.g., les techniques d'analyse du travail, les recommandations, etc.

ont souvent très peu de contact avec les utilisateurs et généralement sous-estiment leur diversité (e.g., l'éventail des niveaux d'expertise)[3] . Une autre étude (Malhotra et al., 1981) montre que les processus de conception sont rarement issus d'une activité linéaire ou séquentielle dans laquelle les problèmes seraient décomposés et les sous-problèmes résolus indépendamment les uns des autres: les processus tendraient plutôt à être cycliques et évolutifs. Les auteurs notent aussi l'importance de la communication et de la collaboration entre les concepteurs et les utilisateurs futurs du système lors du processus de conception, nécessaires pour la prise en compte des difficultés rencontrées par les sujets; aspects qui sont habituellement oubliés dans la conception de nouveaux systèmes. Dans un tout autre registre, Visser (1987) montre le caractère opportuniste des stratégies de conception de systèmes automatisés commandés par automates programmables, alors qu'il est habituel de penser, chez les concepteurs, qu'ils travaillent de façon systématique. Mais que peut, à l'heure actuelle, apporter l'ergonomie aux concepteurs? On examinera successivement deux domaines de contribution de l'ergonomie du logiciel: les recommandations ergonomiques pour la conception d'interfaces et les modèles de l'interaction homme-ordinateur.

3. Apports de l'ergonomie des logiciels

3.1 Recommandations ergonomiques

Les difficultés rencontrées par les informaticiens (mais aussi parfois par les ergonomes eux-mêmes) lors de leur utilisation de recommandations ergonomiques sont de plusieurs types (Smith and Mosier, 1984a, Smith and Mosier, 1986; Reisner, 1986; et pour une discussion plus large, Rouse, 1987; Simon, 1987; Smith, 1987):

- le concepteur d'interfaces n'est pas toujours en mesure d'accorder le temps nécessaire (très long) à la lecture complète d'un manuel de recommandations;
- les recommandations sont souvent trop générales;
- elles posent des problèmes d'accès très complexes. En effet, leur organisation ne reflète pas forcément l'organisation telle que se la représente l'ingénieur, et surtout elles supposent que le lecteur soit en mesure de poser les bonnes questions, ce qui requiert déjà un spécialiste;
- un problème essentiel est celui des interactions entre critères ergonomiques. En effet, très souvent, il ne suffit pas d'envisager un seul critère de conception (e.g., performance d'utilisation vs performance d'apprentissage). Cet aspect conduit à des compromis difficiles à apprécier pour le non-spécialiste. La forme livresque n'est évidemment pas idéale pour cela.

---

[3] On a même rencontré, dans certaines entreprises de technologie que l'on ne citera pas, des équipes de conception dont aucun des membres ne s'était rendu sur le terrain afin d'y observer les tâches devant être informatisées.

Par ailleurs, on constate que certaines recommandations sont des sujets de controverse, même pour les spécialistes. En effet, toutes les recommandations ne sont pas "vérifiées" (e.g., elles peuvent même être incorrectes dans certains contextes), il arrive qu'elles soient contradictoires, et l'on ne s'accorde pas toujours sur les compromis. Cependant, on peut considérer qu'elles sont utiles au moins comme aide-mémoire pour les concepteurs d'interfaces, et pour leur formation, afin de montrer la variété des problèmes à prendre en compte.

Enfin, un problème crucial nous paraît être celui de la mise en relation entre les caractéristiques des tâches et la conception des interfaces. Un effort particulier est nécessaire dans ce domaine. En effet, il est assez difficile d'effectuer les bonnes inférences (dans les cas où cela est possible) entre les recommandations et les caractéristiques de la tâche et de l'utilisateur. Deux raisons nous semblent causer ce type de difficultés: d'une part, les caractéristiques pertinentes des tâches à prendre en compte pour appliquer telle ou telle recommandation ne sont pas toujours explicites, et d'autre part, ce type de questions apparaît trop tard dans la conception. En effet, c'est en examinant tel ou tel écran, ou codage que le concepteur peut éventuellement être amené à se poser des questions sur la tâche, au lieu que ces éléments soient déjà considérés lors de l'analyse conceptuelle de l'interface. Ceci pose d'ailleurs un problème méthodologique de recueil des données: quelles informations recueillir et sous quelle forme? L'examen de l'applicabilité des recommandations ergonomiques issues de la littérature permet de se faire une idée sur les informations dont il faut disposer, en particulier celles qui concernent la manière dont l'utilisateur conduit et se représente sa tâche.

3.2 Quelques données sur la tâche pour la conception d'interfaces

Pour illustrer ce constat avec quelques exemples, on a sélectionné quelques recommandations ergonomiques dans Smith et Mosier (1984b) et Scapin (1987, op. cit.), concernant l'affichage des écrans, pour lesquelles on a recensé quelques-uns des items d'informations liés à la tâche et nécessaires pour l'application des recommandations à la conception des interfaces.

Recommandation (R): Le choix d'une ou plusieurs fenêtres (ou curseurs, ou modes, etc.) est déterminé par le découpage de la tâche en sous-activités.

Données Tâche (DT): Dans quelle mesure deux activités sont suffisamment distinctes, même si elles peuvent éventuellement être effectuées de façon parallèle (ou au moins de façon intriquée)? Comment ces tâches se distinguent-elles, en temps, séquentialité, etc.?

R: les informations affichées doivent être seulement celles nécessaires, et de façon utilisable immédiatement.

DT: Il faut donc déterminer, pour toute séquence de tâches les informations qui sont utilisées, et vérifier sous quelle forme elles le sont, en particulier en repérant les transformations apportées à ces informations pour être utilisables (e.g., changements d'unités).

R: Les listes d'informations doivent être organisées selon un ordre facilement reconnaissable.

DT: Il convient donc d'identifier la façon dont les informations sont utilisées lors de la tâche. Comment sont-elles regroupées? Selon leur fréquence, selon une certain disposition spatiale, selon leur importance, etc.

R: (spécialisation de la règle précédente): la définition des proximités d'affichage dépend des relations de ces informations, pour une activité particulière (e.g., comparaison). Si activité de comparaison entre items (e.g., comparer deux adresses), alors afficher ces champs l'un au-dessus de l'autre.

DT: identification des rapports entre items d'information selon les types d'opérations effectuées. Par exemple, y-a-t-il des comparaisons caractère par caractère?

R: Utiliser des codes, dénominations, etc. familiers pour l'affichage des identificateurs d'écrans, de commandes, de labels, etc.

DT: quels codes, dénominations, etc., sont utilisés dans les tâches actuelles? Existe-il une terminologie familière?

On peut évidemment dénombrer d'autres règles, en particulier pour la conception des structures de dialogues, des séquences de commandes (organisation des options de menus, des commandes, des fonctions permanentes/spécifiques à une sous-tâche, etc.). Ces considérations ne sont que des exemples, mais montrent bien ce dont il est nécessaire de disposer, suite à une analyse de tâches, afin d'envisager la conception d'interfaces. Simplement au vu de ces quelques cas, on voit qu'il faut examiner les tâches du point de vue:

- des rapports logiques, fonctionnels, de séquentialité entre plusieurs sous-tâches d'une même tâche;
- des listes d'informations affectées à chaque opération élémentaire des tâches et leur pertinence;
- des regroupements d'informations (listes, groupements, associations, etc.);
- des listes détaillées d'activités (e.g., comparaison caractère par caractère). Une typologie de ce type d'activités serait d'ailleurs très utile;
- des termes utilisés dans les tâches actuelles.

Il faut être en mesure de disposer des informations pertinentes sur la tâche avant la conception d'interfaces, de manière précise et exhaustive, et non pas se demander, a posteriori, pour appliquer une recommandation ergonomique, quelles sont les informations requises.

Une méthode d'analyse des tâches et un formalisme de description doivent donc au moins permettre, pour l'analyse conceptuelle ultérieure, mais aussi pour des décisions d'ordre syntaxique et lexical, de faire apparaître notamment: le découpage en sous-tâches (ou sous-buts), les relations d'existence (et/ou), de précédence (avant/après/même temps), les états du monde pour chaque sous-tâche (i.e., quelle information est utilisée), les dénominations habituelles, etc.

3.3 Les modèles cognitifs de l'interaction homme-ordinateur

Avec Reisner (1986, op. cit.), on peut s'accorder sur le fait qu'en raison de manques de formalisations théoriques pour la recherche, le domaine de l'interaction homme-ordinateur est un domaine pour lequel les résultats de la recherche ne peuvent se cumuler. En effet, des études empiriques d'interfaces spécifiques ne permettent pas nécessairement de faire avancer l'état de l'art. Il est donc essentiel de disposer de canevas théoriques pour la recherche. Un des domaines pour lesquels des outils de modélisation paraissent essentiels est celui de la formalisation des tâches. En effet, comme on l'a vu précédemment, c'est à partir de cette dernière que l'on peut tenter de poser les bonnes questions de conception d'interfaces. De plus, on peut aussi remarquer que la plupart des articles scientifiques et des guides ergonomiques s'accordent pour souligner l'exigence d'une bonne prise en considération des tâches de l'utilisateur (Moran, 1981b; Smith and Aucella, 1983, op. cit.; Nickerson, 1986; Norman and Draper, 1986; Scapin, 1987, op. cit.; Shneiderman, 1987; etc.). En général, il est suggéré de porter attention à la manière dont les utilisateurs se représentent leur tâche. Cependant, dispose-t-on d'outils, de modèles à cet effet, pour prendre en compte l'utilisateur et sa tâche? Sans se livrer à une revue critique des modèles existants, pour laquelle on renverra le lecteur à Fountain et Norman (1985), Jacobs (1985), et surtout à Hoppe et al. (1986) pour une vue plus complète, on s'intéressera à la pertinence des modèles existants, de notre point de vue, c'est-à-dire, en fonction de leur applicabilité pour la conception d'interfaces ergonomiques, et de leur capacité à rendre compte des aspects conceptuels des tâches. On ne s'intéressera pas ici aux modèles à niveaux (layered models) (Foley and Van Dam, 1973; Oberquelle et al., 1983; Gaines and Shaw, 1984, etc.). Bien que ceux-ci offrent des perspectives intéressantes, en particulier en structurant les couches de l'interface et en permettant d'envisager la conception à différents niveaux d'examen (les distinctions devenues classiques entre niveaux conceptuel / sémantique / syntaxique / lexical (Foley and Van Dam, op. cit.), l'utilisation d'un niveau "pragmatique" (Buxton, 1983), ou la prise en considération d'un niveau "Buts" et d'un niveau "Tâche" (Nielsen, 1986)), ils restent cependant très généraux, en particulier quant au contenu des couches "conceptuelles". Leur utilité étant essentiellement d'ordre classificatoire, on s'intéressera plutôt ici aux modèles de l'interaction homme-ordinateur qui se donnent pour objectif d'expliciter le comportement de l'utilisateur, en relation avec celui de l'interface. On a subdivisé les divers modèles utilisés dans le domaine de l'interaction homme-ordinateur en 5 groupes: Grammaires formelles, CLG, Systèmes de production, Modèles de la connaissance, enfin Réseaux de transition (d'états) et réseaux de Petri.

Grammaires formelles: dans les grammaires formelles telles que les a utilisées Reisner (1981, 1984), ou Richards et al. (1986) pour décrire une interface à base d'icônes, il s'agit plutôt d'effectuer a posteriori des comparaisons entre des interfaces, à un niveau assez bas, sans prendre en considération les aspects conceptuels (les buts sont plutôt du genre "déplacer un caractère" que "gérer la comptabilité").

Les extensions de l'approche BNF (Backus-Naur-Form) (e.g., TAG (Task-Action Grammar), Payne and Green, 1983; Payne, 1984) constituent un effort vers ces aspects conceptuels. Cependant, elles concernent des tâches dites "simples", ne donnant pas lieu à des activités de résolution de problèmes, ou de planification, et des activités de très bas niveau (e.g., déplacer un curseur). TAG ne permet pas de mettre en relation les diverses tâches simples, et donc de disposer d'un modèle conceptuel complet, pour une tâche de plus haut niveau. De plus, la méthode TAG ne distingue pas entre un objet effectivement perceptible par l'utilisateur et un objet imaginé. Cependant, un aspect intéressant de TAG est celui de l'organisation des concepts en plans, à partir de Brewer et Dupree (1983), idée que l'on retrouve à travers de nombreux formalismes.

On notera que dans les grammaires d'attributs étendues (extended attribute grammars), utilisées d'abord pour les langages de programmation (Watt & Madsen, 83), on trouve des applications intéressantes pour l'interaction homme-ordinateur (Sugaya et al., 84). En particulier, ces grammaires, plus concises que TAG, introduisent la notion d'héritage.

CLG (Command Language Grammar; Moran, 1981a) a pour objectif de modéliser les aspects conceptuels de l'interaction. Cependant, CLG ne permet aucune prédiction, mais surtout, ses quatre niveaux constituants ne sont pas explicitement mis en relation. De plus, comme le soulignent Carroll et Rosson (1984), CLG a l'inconvénient d'être une décomposition uniquement "top-down" et de ne pas définir la notion d'itération. Sharratt (1987) propose quelques solutions à ces problèmes (e.g., vérification de l'homogénéité et de leur mise en correspondance ("mapping") entre niveaux. Enfin, la mise en oeuvre de CLG nécessiterait des outils formels d'analyse de tâches, comme le souligne Davis (1983). L'intérêt essentiel de CLG est d'expliciter le modèle conceptuel de l'utilisateur et donc de permettre la conception de systèmes d'aide ("help").

Systèmes de production: dans la catégorie Systèmes de production, Kieras et Polson (1983, 1985) ont montré qu'une description de type CLG pouvait être traduite en règles de production et donc être implémentable. Leur théorie de la Complexité Cognitive représente un système comme un réseau de transition généralisé et la connaissance qu'a l'utilisateur des procédures d'opérations comme un système de production.TAG se définit comme une instanciation du modèle "Task Actions Mappings" de Young (1981, 1983). Cependant cette théorie, tout comme les travaux de Reisner, permet uniquement d'effectuer des comparaisons a posteriori d'interfaces. En particulier cette méthode permet d'apprécier les transferts d'apprentissage.

Modèles de la connaissance: sous ce vocable, on entend les modèles dont l'objectif est d'expliciter la connaissance utilisée dans une tâche à partir de la description de cette dernière. C'est le cas de TAKD (Tasks, Skills and Knowledge, Johnson et al. (1984)), qui génère des descriptions de tâches, à partir de comportements orientés selon un but, qui requièrent l'exécution de certaines actions sur des objets particuliers, puis les traduit en termes de connaissance. La description de la tâche peut nous renseigner sur les connaissances nécessaires à son accomplissement, par la spécification de concepts et de plans,

idée que l'on retrouve chez Morgenstern (1986). La connaissance est divisée en connaissances déclaratives et connaissances procédurales. Son analyse (cf. Hayes-Roth et al., 1981) repose sur la définition de concepts (éléments d'un domaine problème, i.e., actions ou objets et leurs propriétés), de contraintes comportementales (les règles qui gouvernent le comportement dans un domaine problème), et des exécutions d'heuristiques (stratégies). A partir de la description de ces tâches, on voit donc que TAKD permet d'identifier des actions individuelles et des objets conceptuels, qui sont ensuite classés pour les premières en actions génériques et pour les secondes en liste d'objets. Cette méthode ne fait pas référence au processus de conception, mais permet une bonne analyse de l'existant (informatisé) en termes de connaissances.

Réseaux de transition et réseaux de Petri: les réseaux de transition ont été utilisés notamment pour décrire des interfaces, en particulier, du point de vue des réactions du système face aux opérations de l'utilisateur (Green, 1985, Jacobs, 1985, op. cit.). Les réseaux de Petri ont quant à eux été utilisés pour la représentation de systèmes virtuels (Oberquelle et al., 1983). Ils restent très anthropomorphiques; cependant, ce type de structure peut être utile pour les architectures de prototypage (e.g., Sibertin-Blanc, 1988, op. cit.).

4. Discussion

A la lecture de l'ensemble de ces travaux liés, de près ou de loin, à la conception des interfaces, on a donc constaté tout d'abord la non prise en compte de l'ergonomie dans les méthodes de conception informatiques, en particulier le peu d'examen systématique de l'utilisateur (du point de vue de ses caractéristiques propres et du point de vue des tâches). C'est en explorant plus en détail le processus de conception, la manière dont les questions ergonomiques sont posées, que l'on pourra envisager de fournir des outils d'aide à la prise en compte des facteurs humains. Une étude est en cours actuellement sur ce thème. Cette étude a pour objectif, par le moyen d'interviews et de participation à des réunions de conception, d'identifier la manière dont les questions ergonomiques sont posées (à quel moment, en fonction de quels critères, etc.), comment les réponses sont obtenues (expérience, recours à des guides, expérimentation), et comment les compromis sont évalués et réglés.

Mais parallèlement, lorsqu'on s'intéresse, comme le font maintenant nombre d'informaticiens, au domaine de l'ergonomie des logiciels, et en particulier à ses méthodes, on constate les difficultés rencontrées par les informaticiens pour prendre en compte l'ergonomie à partir de ce que les ergonomes leur fournissent.

D'un côté, si les habituelles recommandations ergonomiques encourageant à prendre en compte l'utilisateur, sa tâche, etc. sont souvent présentées de façon anecdotique ou floue, et surtout sans lien avec des méthodes de recueil d'information, il reste cependant envisageable de tirer de celles-ci des idées sur les informations à prendre en compte avant la

conception (comme on l'a vu sur des exemples de recommandations), au moyen d'outils d'aide à la description de tâches.

D'un autre côté, parmi les modèles qui ont été proposés pour l'interaction homme-ordinateur, rares sont ceux qui font référence spécifiquement au problème de la conception d'interface, et nombreux sont ceux qui se préoccupent essentiellement de phénomènes de comportement de bas niveau.

De plus, les modèles dont on dispose sont difficilement applicables et a fortiori implémentables. Ainsi, pour Coutaz (1988), les méthodes de conception sont inopérantes ou presque, et les modèles examinés décrits par l'auteur sont pour l'un trop réducteur (GOMS, Card and Moran, 1980; Card et al., 1983), pour un autre trop informel (Norman, 1983), et enfin, pour un dernier, trop difficile (ACT, Anderson, 1983) (en fait, surtout trop général).

Cependant, en considérant que la conception d'interfaces est un processus itératif et qu'à un moment ou un autre l'évaluation est utile, nombre de ces modèles sont appropriés. De plus, avec l'objectif de concevoir des outils d'aide très tôt dans la conception, ces modèles comportent certains aspects à retenir, ainsi que quelques idées à suivre ou des lacunes à combler.

On a aussi relevé un certain nombre d'aspects issus de l'ergonomie des logiciels, du point de vue recommandations et du point de vue modèles.

Il reste à définir une architecture adéquate permettant de rendre compte de ces divers aspects. La planification hiérarchique semble être un bon candidat pour ce rôle. Fikes (1981) estime ainsi que les techniques d'intelligence artificielle, et en particulier la planification hiérarchique (Sacerdoti, 1974, 1977) permettent de bien rendre compte de l'exécution des procédures de bureau. Sans vouloir détailler la planification hiérarchique, en particulier pour l'analyse de tâches de bureau pour laquelle on renverra à Sebillotte (1987, 1988), retenons le concept-clef qui sera utilisé par la suite: hiérarchie de représentation de plans à différents niveaux d'abstraction, du plus général (racine de l'arbre) au plus détaillé (feuilles de l'arbre, opérations élémentaires). Ce formalisme d'arbre hiérarchique de buts et sous-buts sera en effet retenu pour notre formalisme. Le sujet humain est considéré comme élaborant un plan d'action à partir du but qu'il s'est fixé. Ce plan d'action lui permet de raisonner à des niveaux de détail différents en passant par la définition de sous-buts intermédiaires nécessaires à l'accomplissement du but. Selon Kieras et Polson (1983, 1985), la structure en buts et sous-buts peut être utilisée comme une description concise de la structure d'une tâche du point de vue de l'utilisateur. Comme l'indique justement Moran (1983, op. cit.), le découpage en buts et sous-buts présente un caractère plus constant que la description d'une tâche par ses méthodes d'exécution qui, elles, peuvent subir des changements au gré des personnes ou des machines, au moins à un certain niveau.

Ceci étant dit, il est important de souligner que l'aspect qui nous semble le plus important n'est pas tant le formalisme lui-même, mais ce que l'on y met. Il convient en effet de

dépasser les modèles existants concentrés sur des niveaux très bas d'interaction, applicables essentiellement aux niveaux dits syntaxique et lexical, pour prendre en compte les aspects conceptuels, de façon systématique et rigoureuse. De plus, cette revue de question des problèmes d'insertion de l'ergonomie dans le processus de conception d'interfaces permet de penser l'analyse conceptuelle comme dirigée par la conception, i.e., par un souci d'identifier les items d'information liés à la tâche, nécessaires à prendre en compte pour une activité ultérieure de conception d'interfaces. En fonction de cet objectif et des caractéristiques souhaitables d'un modèle, nous présentons maintenant une ébauche de formalisme, pour des tâches de bureau.

5. MAD: une Méthode Analytique de Description des Tâches (de Bureau) Orientée Conception d'Interfaces Utilisateur

5.1 Un formalisme de description pour quoi faire?

Quels sont les problèmes dont un modèle doit rendre compte? Un modèle aura une architecture différente selon ce qu'il privilégie. Ainsi, son but peut-être de décrire une tâche de telle façon qu'un novice puisse aisément l'exécuter (par exemple pour un mode d'emploi) ou de déléguer une partie de la tâche à un autre système, à un autre opérateur, pour la formation, etc. Avec Reisner (1986, op. cit.), on s'accordera pour estimer qu'un des objectifs de la modélisation est clair: obtenir des descriptions explicites. La part descriptive de tels modèles est déjà une forme d'analyse de tâches, mais elle va au-delà: en effet, ces modèles nous aident à penser en termes de conception. On mettra l'affirmation suivante au conditionnel, tout en estimant que c'est une issue souhaitable: " .. il serait utile que ces modèles puissent rendre explicite le type de connaissance intuitive que les meilleurs concepteurs ont toujours mis en oeuvre..". Ce que nous devons privilégier est donc un modèle permettant de disposer de connaissances suffisantes sur la tâche et la représentation que s'en fait l'opérateur pour permettre de faire de la conception. Cela implique un fort degré de précision et d'exhaustivité.
Les objectifs plus spécifiques d'un tel outil de formalisation sont les suivants:
(1) permettre la formalisation de tâches pour la conception d'interfaces utilisateurs (méthodologie de conception) et disposer d'un ensemble représentatif de descriptions de tâches de bureau [4]. En effet, même pour des tâches simples et structurées, comme les tâches de bureau, on a besoin d'un formalisme de description sophistiqué et puissant;
(2) permettre un dialogue plus facile entre différents spécialistes (concepteurs, ergonomes, informaticiens);
(3) permettre une implémentation sur laquelle travailler, notamment lors de la mise en relation avec d'autres logiciels d'intelligence artificielle, en particulier ceux qui tentent de

[4] A partir desquelles seront menées des études de conception et de prototypage d'interfaces.

modéliser l'environnement de bureau (e.g., AMS (Tueni et al., 1988); OTM (Lochowsky et al., 1988); POLYMER (Croft and Lefkowitz, 1988));
(4) permettre, après test et évaluation, de définir une méthode de recueil de données, dans le cadre de l'analyse du travail.[5]

5.2 Critères privilégiés

Tout d'abord, il s'agit de se donner les moyens de prendre en compte l'utilisateur et sa tâche, dès la conception. Ensuite, une des caractéristiques d'un tel formalisme est de faire référence à la représentation mentale qu'a l'utilisateur de son travail et non pas à la logique du traitement informatique (si informatisation), ou à la tâche prescrite. Par ailleurs, ce formalisme doit être en mesure de décrire une tâche de manière telle qu'en deçà d'un certain niveau, il n'est aucunement fait référence au système sur lequel elle va s'appliquer. A l'inverse de la plupart des modèles d'interaction homme-ordinateur, on tentera d'expliciter les différences entre niveau tâche et niveau système[6] .
Un tel formalisme doit :

- être rigoureux et stable (i.e., laisser peu de place à l'interprétation, ou en d'autres termes, donner lieu au même résultat indépendamment de celui ou celle qui applique le formalisme),
- conduire à l'exploration systématique de tous les éléments du formalisme, i.e., donner lieu à des descriptions complètes (ou du moins permettre l'intégration complète des représentations, objets, opérations au niveau d'explicitation considéré (e.g., éventuellement jusqu'aux opérations élémentaires, aussi bien pour des tâches manuelles que pour des tâches informatisées)). On peut penser que ceci doit permettre une plus grande exhaustivité que le simple interview non- ou semi-directif. En effet, on peut penser que si un formalisme est rigoureux, alors on pourra en tirer une méthode de recueil rigoureuse et exhaustive,
- autoriser la description aussi bien d'un point de vue déclaratif (état du monde) que procédural (façon d'arriver à cet état),
- autoriser la prise en compte du parallélisme et pas seulement du séquentiel,
- représenter tous les concepts de façon hiérarchique et uniforme.

---

[5] Par ailleurs, ce travail s'insère dans un projet plus large (ARCHIE, i.e., Advice-giving Routines for Computer-Human Interaction Ergonomics) dont l'objectif est de fournir des outils logiciels d'aide à la conception ergonomique d'interfaces. Ce projet de recherche consiste notamment en la création d'une base d'objets des éléments de l'interface, l'organisation et la mise en place (sous forme de système expert) de recommandations en ergonomie des logiciels, la définition d'un modèle à niveaux de l'interaction homme-ordinateur permettant d'organiser la base de connaissance, et la definition d'une méthode de conception de l'interface.

[6] Sans adopter de codage particulier pour différencier ces deux niveaux, les différences seront explicites dans la mesure ou elles apparaîtront dans le contenu des Items et dans leur décomposition (e.g., Transmettre une information à quelqu'un vs Utiliser le "Reply" du "Mail).

5.3 Concepts et Définitions

En fonction des critères précédents, le formalisme MAD repose essentiellement sur la planification hiérarchique, à quelques différences près, et avec un certain nombre d'ajouts, que l'on pourra observer au fur et à mesure

Dans le formalisme MAD, on utilisera le concept de hiérarchie de buts de la planification hiérarchique, en définissant l'accomplissement d'une tâche par la transformation d'un état initial en un état-But. Cependant, on élargira cette notion de hiérarchie de buts à la notion de **hiérarchie d'Items**. Par ailleurs, le formalisme MAD fait apparaître non seulement le découpage en sous-tâches (ou sous-buts), mais aussi la synchronisation, i.e., les relations booléennes (et/ou), de précédence (avant/après/même temps), ainsi que, par le moyen de la décomposition, les états du monde (e.g., état des objets de départ et de fin) pour chaque élément descriptif de la tâche (i.e., quelle information est utilisée, sous quelle forme?), les dénominations habituelles, etc.

> ***Définition (1) Item:*** *unité descriptive de niveau n dans l'arbre. Item peut prendre les valeurs suivantes: Tâche, But, Pré-Requis.*

MAD représente sous forme d'arbre hiérarchique une série d'items descriptifs d'un Item-tâche. Chaque item se caractérise par un ensemble de buts et un ensemble de pré-requis (qui sont eux-mêmes des items), l'un et/ou l'autre de ces ensembles pouvant être vide(s): un item peut être décrit par n Buts et son déclenchement peut être conditionné par n Pré-Requis. On y ajoutera une décomposition à structure d'objets.

> ***D (2) Décomposition:*** *chaque composant de l'item peut se définir (en particulier pour son intitulé par une décomposition en 2 composants:* ***opérations*** *et* ***objets,*** *lesquels comportent une série d'attributs.*
>
> ***D (3) Héritage:*** *chaque élément décomposé (opération ou objet) hérite des attributs de l'élément des niveaux supérieurs.*

Ceci revient à décomposer au fur et à mesure les items de l'arbre en éléments de plus en plus spécifiques. Par exemple, l'opération ENVOYER se caractérise par les attributs (émetteur/ destinataire/ objets (lettre/ commande)); l'objet COMMANDE se caractérise par les attributs (référence/ type formulaire/ prix, etc.). A un niveau inférieur, on pourra avoir une opération REPONDRE qui héritera des attributs d'ENVOYER, auxquels s'ajoutera l'attribut (destinataire=émetteur lettre reçue). De même pour les objets, on aura un objet COMMANDE de PETIT MATERIEL qui héritera des attributs de COMMANDE, auxquels s'ajoutera l'attribut (prix < 5000FF), etc.

> ***D (4) Tâche:*** *la racine de l'arbre hiérarchique.*
>
> ***D (5) But:*** *état à atteindre, i.e., l'état du monde tel qu'il sera une fois l'activité mise en oeuvre).*

Le But peut donc être considéré comme l'état résultant de la suite des actions. Un But peut être précédent, ultérieur ou concomitant d'un autre But, sans être pour autant un Pré-

Requis (comme c'est le cas dans Sacerdoti (op. cit.)), sauf si certaines conditions de précédence sont respectées.

Le problème des préconditions, prérequis et conditions déclenchantes est ardu. Les différences de terminologie entre ces trois termes ne sont pas toujours très claires dans les formalismes, il convient donc d'adopter une définition explicite.

*D (6) **Pré-requis:** condition nécessaire (strictement) au déclenchement d'un item.*

Un pré-requis représente donc l'état du monde à satisfaire afin de lancer tout item (tâche, but ou pré-requis). Le pré-requis est donc considéré comme une condition déclenchante qui se situe, comme son nom l'indique, avant le départ de la procédure proprement dite. Cependant, il convient de préciser deux notions importantes.

La première est la notion de nécessité. En effet, dans MAD, il ne s'agit pas de lister au départ l'ensemble des pré-requis d'une activité, comme dans les "scripts" (Shank and Abelson, 1977), mais de retenir comme pré-requis uniquement celui ou ceux strictement nécessaires au lancement d'un item. Par exemple, "avoir une facture" qui est une condition nécessaire pour déclencher l'activité de "Rédaction de chèques" est considérée comme un pré-requis, alors que "disposer d'une calculette", qui n'est nécessaire que pour "vérifier les sommes indiquées sur une facture" est un pré--requis de cette dernière activité, mais pas un pré-requis de "Rédaction de chèques".

La seconde notion concerne la différence entre pré-requis et items reliés par des connecteurs de précédence. En effet, par définition, tout item Ii de même niveau qu'un autre item de même classe In (Tâche, But ou Pré-requis) comportant une relation "ET" avec "Avant" (Ii (et,avant) In) est préalable à In, mais n'est pas considéré comme pré-requis. C'est ce type d'item qui est utilisé en planification hiérarchique de façon indifférenciée avec les pré-requis comme ils sont définis ici (condition nécessaire strictement).

Cette notion de pré-requis permet de rendre compte du parallélisme des activités. Enfin, le Pré-requis est de type déclaratif (le fait d'être dans tel état), mais peut déclencher (se décomposer en) une suite (en fait un sous arbre) de type procédural (la manière d'atteindre cet état).

Un modèle doit être capable de prendre en compte la définition des **primitives**, qu'elles soient nommées unit tasks (Kieras et Polson, 1983, op. cit.), simples tâches ou bien encore actions de base (Mayer and Bayman, 1981). Ainsi selon ces auteurs, une action de base est une transformation opérée sur un objet en un lieu. Selon Sebillotte (1987, op. cit.) les primitives sont les sous-tâches correspondant au plus bas niveau d'abstraction et au-delà desquelles le sujet est incapable de décrire sa tâche. Selon Richard (1986, op. cit.), une primitive est une action à laquelle un but peut être assigné alors que cela n'est pas possible pour les micro-actions qui la composent.

*D (7) **Action:** But (éventuellement issu d'un Pré-requis) dont l'ensemble des pré-requis est vide et dont la décomposition n'occasionne par d'autre but que le but précédent, ou une forme plus spécifique. Il s'agira donc des opérations terminales directement exécutables.*

Hewitt (1986) a montré que le travail de bureau réside souvent en un traitement d'informations contradictoires et asynchroniques. Il est donc essentiel de prendre en compte les aspects de **synchronisation**, même à un niveau élémentaire. Dans cet objectif, on a introduit la notion d'items (buts et pré-requis) reliés par des connecteurs booléens (et, ou) et des connecteurs de précédence (avant/après/même temps/sans contrainte). Il pourra cependant être utile dans l'avenir, si des analyses plus précises des tâches de bureau en montrent la nécessité, d'envisager des cas plus complexes du type: début et fin d'une activité, interruptabilité, deadlines, notions floues du temps (e.g., dans une quinzaine, avant ce soir, etc.).

> ***D (8) Synchronisation:*** *(contraintes d'exécution): tout Item pouvant donner lieu à n Buts et n Pré-Requis, ceux-ci (Buts et Pré-Requis) sont organisés en fonction de:*
>
> - ***D (8.1) connecteurs booléens*** *(et/ ou), ainsi que de*
> - ***D (8.2) connecteurs de précédence*** *(avant/ après/ même temps/ sans contrainte).*

Enfin, dans le travail de bureau, comme dans d'autres, certaines opérations sont facultatives, ce qui est différent de la relation logique "ou" introduite précédemment. Ce type d'opérations sera donc distingué par un codage.

> ***D (9) items facultatifs:*** *par défaut, et en fonction des relations introduites par les connecteurs précédents, tout Item sera considéré comme obligatoire, sauf s'il est codé (F), c'est-à-dire s'il est déterminé comme pouvant être effectué ou pas, sans conséquence immédiate sur les Items ultérieurs.*

Par ailleurs, il peut se présenter assez fréquemment dans des tâches de bureau qu'une activité soit répétée pour une classe quelconque comportant n objets (e.g., effectuer la procédure de règlement par chèque de toute une pile de factures). Il sera donc utile de permettre de distinguer les séquences d'opérations individuelles des routines, i.e., des opérations répétées en fonction de l'état de certaines valeurs des éléments de l'environnement. La notion de Boucle est introduite à cet effet et peut être considérée en fait comme un item plus général qui lance une série d'items jusqu'à ce qu'un état soit atteint (e.g., pour Tâche "Faire Chèques"; pile de factures = vide). Dans les tâches de bureau, il semble que soient en fait très fréquentes les boucles d'activité sur un seul Item, en particulier un but (e.g., signer toutes les lettres, préparer toutes les adresses des enveloppes, insérer toutes les lettres dans les enveloppes, etc., plutôt que, pour chacune des lettres: signer, mettre l'adresse, introduire la lettre dans l'enveloppe).

> ***D (10) Boucle*** *boucle itérative sur un Item ou un ensemble d'Items.*

Il est par ailleurs important de pouvoir illustrer les cas où le cheminement dans l'arbre dépend de valeurs de variables. Ainsi, aussi bien pour les buts que pour les pré-requis, il sera possible, comme on le verra plus loin, d'illustrer les conditionnelles de type "Si...Alors".

Enfin, il est souhaitable de rendre compte des interruptions de l'activité, des abandons de plans, des changements de buts, i.e., des modifications de chemins dans l'arbre de l'activité. En effet, ceci pourra être utile pour prendre en compte le problème des erreurs, des interruptions et des changements de stratégie, en somme pour tenir compte du caractère opportuniste de l'activité.

> ***(D8) Chemins** = la poursuite d'un chemin dans l'arbre peut être interrompue et reprise à tout moment pourvu que les contraintes d'exécution (synchronisation) soient respectées.*

On admettra donc que la lecture d'un graphe (ou en fait sa simulation, i.e., l'exécution d'une tâche) peut se faire en passant d'une branche à l'autre de l'arbre, ou dit autrement, en passant d'un chemin à un autre, les seules contraintes à respecter étant bien entendu d'une part la satisfaction des pré-requis et la mise en oeuvre préalable des items caractérisés par des connecteurs et/avant. Donc, il est admis qu'un opérateur puisse suivre un chemin A (se donner des buts, suivre des procédures) puis l'abandonner pour se livrer à une autre activité (un autre chemin B dans l'arbre), et revenir plus tard au précédent, si et seulement s'il n'existe pas de relation du type: A est un pré-requis de B, ou bien les relations de A, B sont du type (A, et/avant, B), auquel cas le chemin A ne peut être abandonné pour B.

Pour envisager la conception d'interfaces, en particulier du point de vue de l'allocation des tâches homme-ordinateur et de l'affichage d'informations sur écran, on sera amené à utiliser les codages (ad hoc, mais utiles) suivants:

> ***C (1) Evénements extérieurs:** (Eo) événement issu d'un autre opérateur, (Em) événement matériel.*

Il paraît en effet utile de distinguer au sein de la notion d'événements extérieurs les événements attribuables à un autre opérateur (coopération) des événements matériels, en particulier pour les aspects pragmatiques de la conception d'interfaces (e.g., gestion des événements).

> ***C (2) Mémoire/Perceptif:** ce qui est du registre de la mémoire et ce qui est du domaine perceptif (référence disponible vs à mémoriser) seront différenciés pour chaque item. **(Mm / Pe)***

En effet, ce type d'information est nécessaire pour une meilleure définition de la permanence des objets visualisés sur écran.

> ***C (3) Actions Mentales/ (Mt) vs Actions Physiques (Py)***

La distinction entre Actions Mentales et Actions Physiques semble importante, notamment en ce qui concerne l'exécution de la tâche. En effet, dans MAD, la notion de connaissance au sens de Morgenstern (1986, op. cit.) est importante, en particulier les actions mentales. Cependant, il est tout aussi important d'identifier les actions physiques mises en oeuvre lors de l'accomplissement de la tâche, en particulier pour que le modèle puisse rendre compte de délégations de tâches. On adoptera donc un codage différencié selon qu'il s'agit d'actions mentales ou physiques.

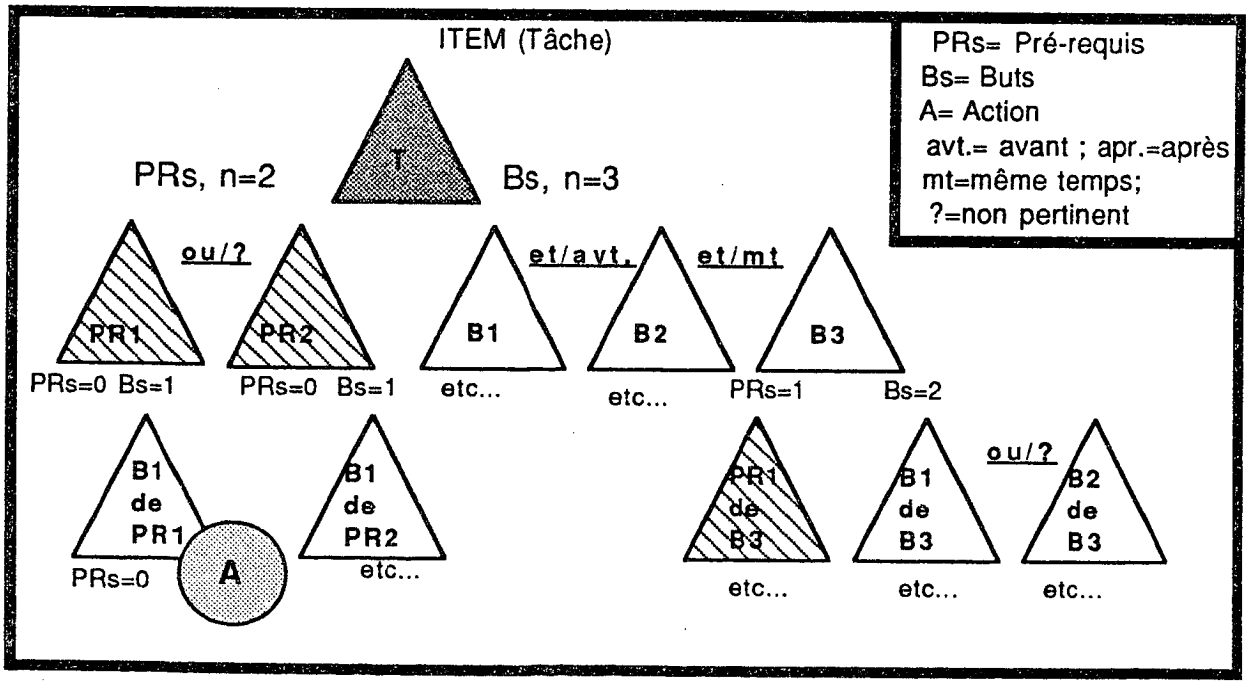
5.4 Lecture des descriptions et représentation graphique

Le formalisme doit permettre une lecture différenciée, selon qu'il s'agit de la liste des actions à effectuer, du cheminement procédural, ou d'items de plus haut niveau. En particulier, cette représentation doit fournir niveau par niveau la liste des divers items (buts et pré-requis), le cheminement descendant des procédures, et de façon ultime et transversale, la liste des opérations élémentaires.

La description graphique d'une tâche selon le formalisme MAD est un arbre hiérarchique, dont chaque élément est constitué d'un item, lui même un sous-arbre, de manière récursive. Chaque item est issu d'un item précédent dans l'arbre (la racine étant l'item Tâche décrite) et se décompose en items inférieurs, jusqu'aux feuilles de l'arbre, i.e. les actions. Ainsi, un item d'une part se décrit par un ensemble de Buts, lesquels ont eux-mêmes des Pré-Requis et des Buts, et d'autre part ne déclenche l'ensemble des Buts qui lui sont affectés qui si l'ensemble de ses Pré-Requis sont satisfaits. Dans le cas où ils ne le sont pas, les Pré-Requis renvoient eux-mêmes à une série de buts et de Pré-Requis.

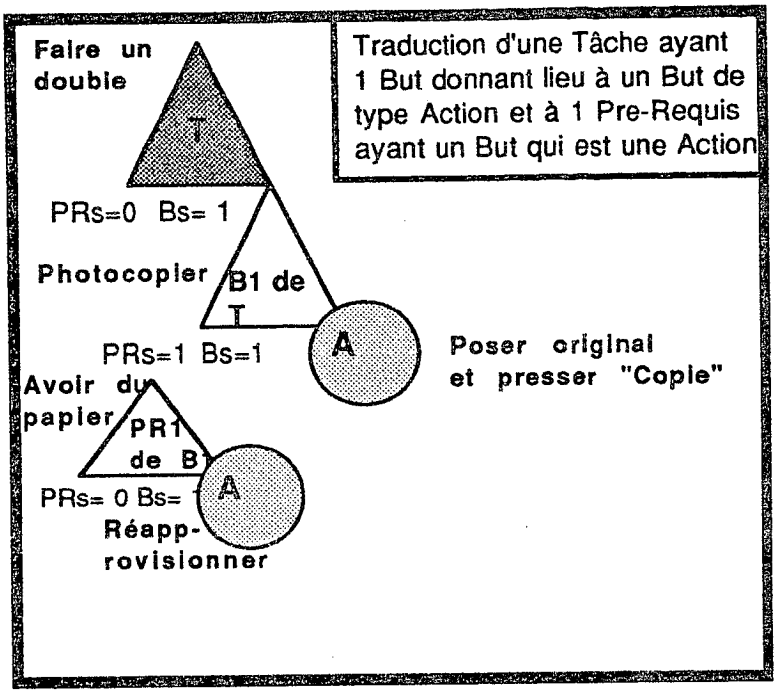
La lecture des intitulés des noeuds du graphe donne donc l'ensemble des états possibles; la lecture descendante d'un sous-arbre quelconque identifie une procédure particulière (e.g., l'ensemble des items successifs pour un but "faire un double"); la lecture des feuilles de l'arbre identifie l'ensemble des actions terminales.

Graphiquement, on peut représenter la structure d'un arbre de la manière suivante (Fig. 1):



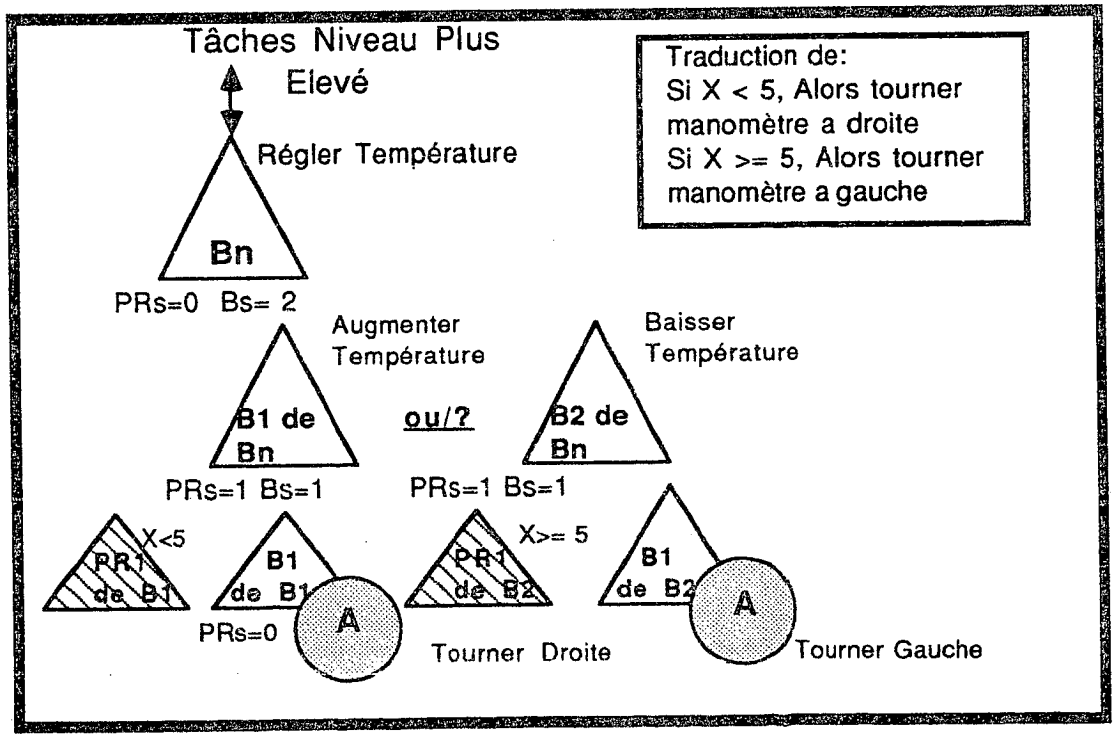
-- Fig. 1 --

Dans la figure 1, on a vu un exemple de tâche hypothétique. Un séquencement complet (i.e., un seul chemin dans l'arbre) traduirait une procédure simple comportant un seul but ou un seul prérequis à chaque noeud de l'arbre; voir par exemple (Fig. 2)



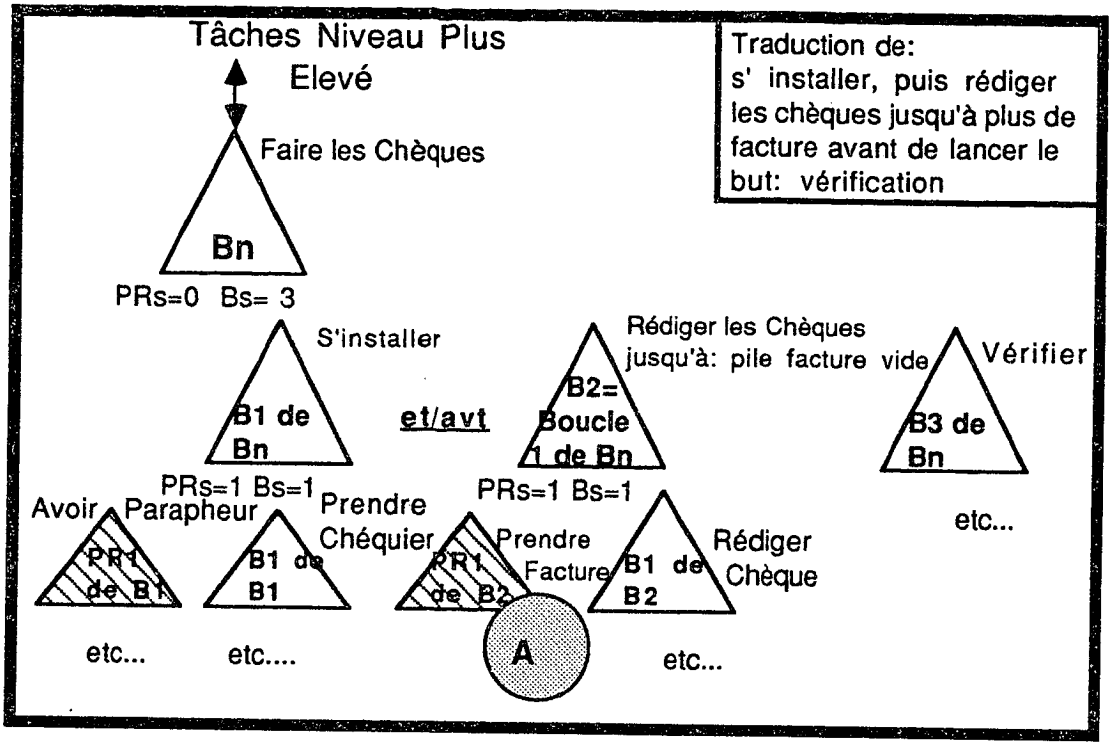
-- Fig. 2 --

De plus, ce formalisme permet de traduire les boucles Si ... Alors. Par exemple (Fig. 3):



-- Fig. 3 --

Enfin, un exemple d'utilisation de l'Item "Boucle" est illustré dans la Figure 4.



-- Fig 4 --

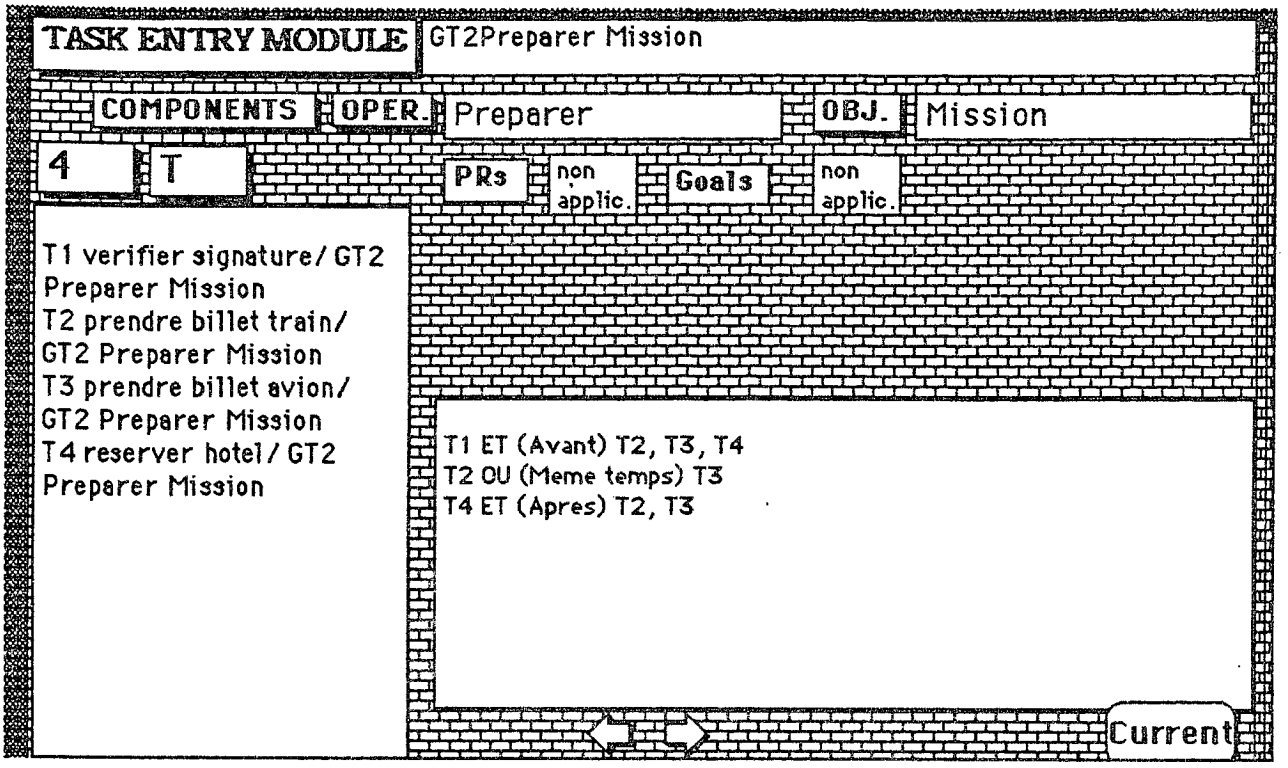
5.5 Prototype d'aide à la description des tâches

5.5.1 Architecture et utilisation de l'application

En utilisant le langage HyperTalk de l'application HyperCard™, nous avons construit un prototype d'aide à la description des tâches, basé sur le formalisme MAD. Le programme est constitué d'un module d'entrée permettant de décrire de façon récursive l'ensemble des caractéristiques de chaque Item de tâche (en réalité de groupes d'items de même nature). Ce module guide l'entrée par l'analyste des dénominations des Items, de leur décomposition, de leur rapport avec les Items (Buts et Pré-requis) précédents et ultérieurs. Par ailleurs, il vérifie la cohérence des divers Items et des connecteurs logiques et de précédence. Chaque Item ainsi décrit est stocké sous forme de carte, au sein de la pile ("stack") d'une Tâche. Le programme permet ensuite, à partir de l'ensemble des Items de créer automatiquement un "browser" qui construit l'arbre hiérarchique et permet d'accéder aux items individuels. Le programme comporte environ 700 lignes de code et représente environ 100K de mémoire (compactée). La formalisation d'une tâche particulière est donc disponible sur une pile HyperCard de n +1 cartes correspondant à n items ainsi qu'une carte "browser". Des exemples de cartes sont présentés Figures 5, 6 et 7, représentant respectivement le module d'entrée, un item particulier, et un "browser", pour une tâche particulière.

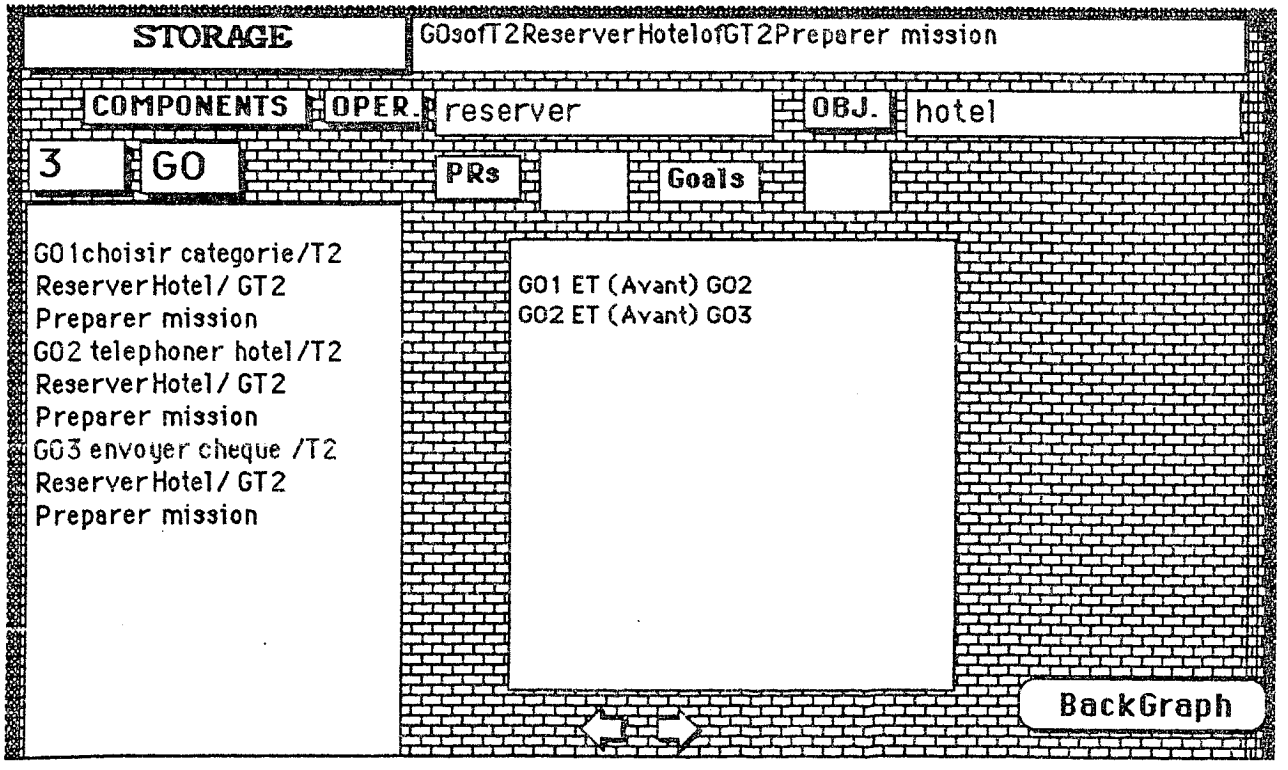
TM: HyperCard and HyperTalk are a Trademark of Apple Computers Inc.

Module d'entrée



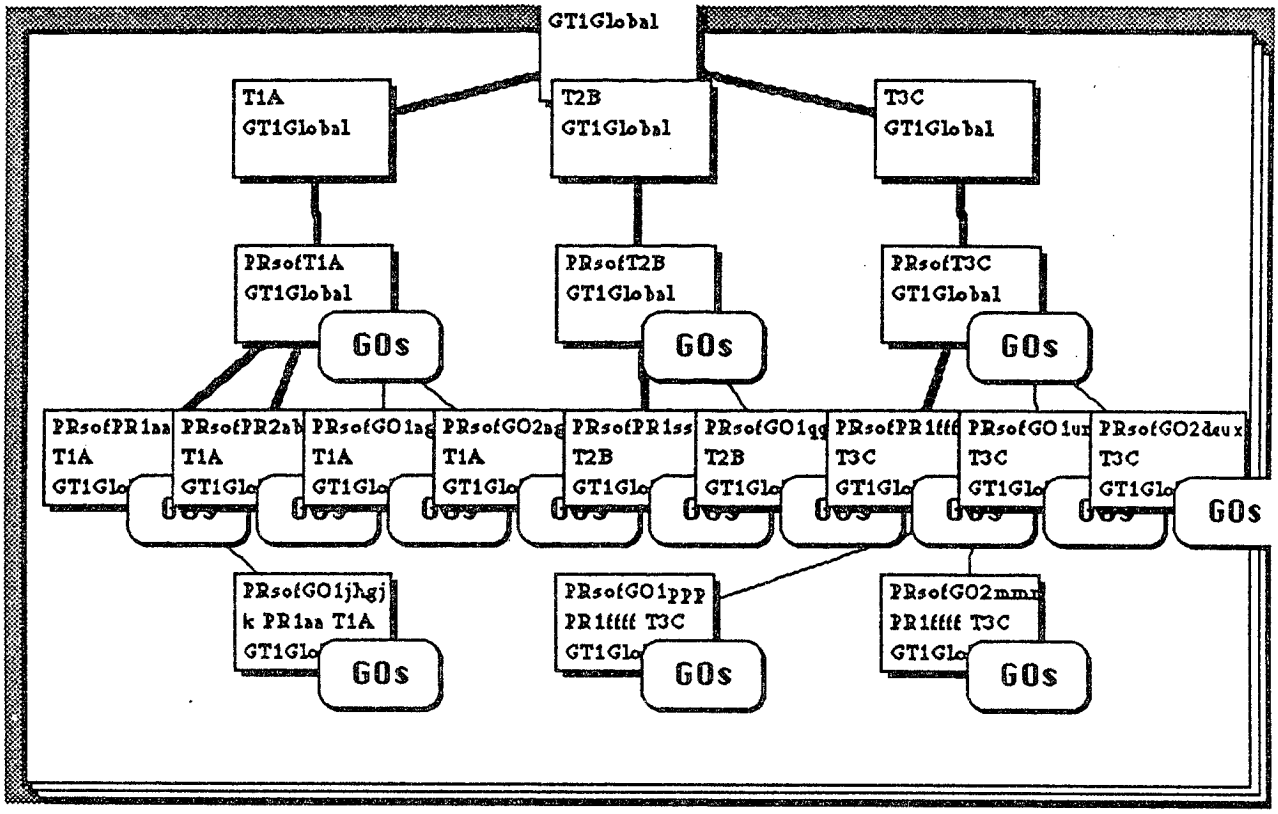
-- Fig. 5 --

Un item (Buts)



-- Fig. 6 -

Le browser pour une tâche



-- Fig. 7 --

5.5.2 Améliorations prévues

Ce prototype n'en est qu'à une phase préliminaire de démonstration de faisabilité. Il devra être amélioré de plusieurs manières. D'une part, en raison des limitations inhérentes à l'application HyperCard, en particulier la taille limitée des cartes, l'impossibilité de travailler sur plusieurs cartes en même temps, et les difficultés de maintenance, il est envisagé de porter ce programme en C ou Lisp sur une station de travail de type Sun 3-60, ou sur l'application Notecards [TM], quand celle-ci sera disponible sur des stations de travail autres que dédiées (Xerox). D'autre part, un certain nombre d'améliorations sont envisagées:

- autoriser l'édition, en particulier graphique des représentations obtenues. Pour l'instant, le programme ne permet que la reprise, après interruption, d'une description de tâches, pas son édition a posteriori,
- autoriser la description de manière plus flexible. En effet, le module de description guide l'opérateur dans une description par niveaux, en explorant de manière exhaustive chacun des items d'un même niveau (Top-down et transversal). Il serait souhaitable de permettre

[TM] Notecards is a trademark of Xerox PARC

à l'opérateur de décrire une tâche aussi bien à partir des feuilles de l'arbre (Bottom-up) qu'en suivant chaque branche de l'état initial aux actions élémentaires,

- enfin, il conviendra d'examiner la mise en correspondance d'un tel programme, un fois écrit en Lisp avec d'autres logiciels de type intelligence artificielle (e.g., AMS (Tueni et al., 1988, op. cit.)).

6. Etude en cours

Pour évaluer ce formalisme, une étude est en cours, afin d'explorer concrètement les problèmes posés par ce formalisme (facilité d'utilisation, cohérence, différences avec planification hiérarchique classique, etc.). Cette étude comporte trois étapes, chacune correspondant à un objectif particulier:

- illustrer les différences obtenues entre des représentations classiques de type planification hiérarchique pour certaines tâches de bureau et la formalisation de ces tâches avec le formalisme MAD. Ceci sera mis en oeuvre à partir de formalisations classiques déjà effectuées (tirées de Sebillotte (1988, op. cit.) mais aussi à partir des données brutes (interviews) correspondant à ces tâches.

  Dans la mesure où il apparaît que MAD répond aux exigences que l'on se sera fixé[7], on en tirera une technique d'interviews dirigés (ou même en fait d'observation armée) afin d'effectuer :

- des interviews avec la méthode MAD des mêmes sujets que pour l'étape précédente;
- de nouveaux interviews méthode MAD, pour un ensemble unitaire de tâches (soit un ensemble de tâches différentes au sein d'un même service, soit un même type de tâches au sein de plusieurs services différents). Cette dernière étape aura pour autre objectif d'envisager la constitution d'une petite base de données tâches.

7. Conclusion

Ce papier représente une tentative d'examen critique des problèmes posés par la mise en relation de l'ergonomie avec le processus de conception d'interface. Cet examen a suggéré la prise en compte de nombreux éléments liés aux tâches et a permis de proposer une ébauche de formalisme répondant à cette exigence. Il est certain que le formalisme MAD devra considérer un certain nombre d'aspects non évoqués jusqu'à présent, par exemple:

- il sera utile de pouvoir situer dans la description des tâches, les niveaux d'abstraction tels que les suggère Sebillotte (1987, op. cit.). Celle-ci a montré que lorsque des secrétaires décrivent leur tâche, elles fournissent une représentation planifiée de leur tâche. Elles donnent d'abord une vue schématique de leur activité, en termes de tâches devant être

---

[7] e.g., exhaustivité, fidélité, limitation des erreurs d'interprétation, réduction du cycle interview / suivi pour complément d'information, niveau approprié d'information pour a conception d'interfaces.

accomplies, ou en termes de buts à atteindre. Elles décrivent leurs procédures uniquement si cela leur est demandé. Il conviendra d'être en mesure de pouvoir repérer et nommer ces divers niveaux et regroupements d'opérations, en particulier dans un objectif de définition d'environnements de bureau adaptables ("taylorable"), par exemple pour autoriser l'utilisateur à appeler des fonctionnalités, des procédures, ou les modifier;

- d'autres aspects devront être explorés plus précisément, en particulier les notions de prototypes, d'opérations générales/spécifiques, de buts et pré-requis identiques au sein d'un même arbre;
- il conviendra, si nécessaire, d'affiner les notions de temps, pour prendre en compte des temps absolus, y compris flous, et pas seulement des précédences;
- on pourra définir un langage non graphique de description selon formalisme MAD, ce qui requerrera un codage particulier des relations de précédence et des booléens.

De plus, et c'est peut être le plus important, le formalisme MAD, ne pourra démontrer son utilité éventuelle qu'à l'usage, d'une part lors de l'étude évaluative précédemment évoquée, mais aussi dans les travaux futurs qui consisteront à définir des outils de conception ergonomique, à partir de descriptions de tâches. Dans ce cadre, on espère que le formalisme MAD permettra de mieux opérationnaliser, dans la conception de l'interface, la différence entre logique de fonctionnement et logique d'utilisation (Richard, 1983), de mieux tenir compte des caractéristiques de l'opérateur et de sa tâche au lieu de tenir compte essentiellement de la logique fonctionnelle de l'application et des contraintes imposées par l'ordinateur. Un intérêt annexe de cette approche est à terme, dans le cas de simulations, de fournir des informations à l'utilisateur sur ce qu'il peut faire, et donc faciliter la conception des modules d'aide et d'apprentissage.

En conclusion, on peut estimer qu'il faut non seulement fournir des outils de formalisation puissants afin de disposer des informations sur les tâches, mais qu'il faudra aussi disposer de méthodes, d'un cadre théorique, permettant d'insérer dans le processus de conception les éléments d'ergonomie pertinents. L'objectif final est donc bien d'aboutir à des outils d'ingénierie, en considérant l'ergonomie du logiciel comme partie prenante de la conception, dans toutes ses étapes, y compris l'analyse initiale, au même titre que l'architecture système, le choix d'un type de processeur, ou d'un réseau. Bien entendu, et ce sera l'objet d'une étude à venir, il devient essentiel selon cette perspective d'être en mesure de définir, en collaboration avec les informaticiens de toutes spécialités, une méthode générique qui englobe harmonieusement tous les aspects de la conception.

BIBLIOGRAPHIE

- Anderson,J.R.(1983).The architecture of cognition .Havard University Press.
- Barthet, M-F. (1986). Conception d'applications conversationnelles adaptées à l'utilisateur. Thèse d'Etat, Institut National Polytechnique de Toulouse, France.

- Barthet, M-F. et Sibertin-Blanc, C. (1986) La modélisation d'applications interactives adaptées aux utilisateurs par des réseaux de Petri à structure de données. Congrès AFCET, Génie Logiciel, Versailles, France.
- Brewer, W. F. and Dupree, D. A. (1983) Use of plan schemata in the recall and recognition of goal directed actions. Journal of Experimental Psychology, Learning, Memory and Cognition, 9, 1, 117-129.
- Buxton, W. (1983) Lexical and pragmatic considerations of input structures. ACM SIGRAPH: Computer Graphics 17 (1), 31, 37.
- Card, S.and Moran,T.P. (1980) The Keystoke-Level model for user performance time with interactive systems.Communications of the ACM, 23 (7), 396-410.
- Card, S., Moran, T.P., and Newell (1983) The psychology of human-computer interaction, Lawrence Erlbaum.
- Carroll, J. M. and Rosson, M. B.. Usability specifications as a tool in iterative development. In: Hartson, H.R. (ed.), Advances in Human-Computer Interaction, Vol. 1, Ablex, Norwood, N.J., 1-28.
- Croft, W. B. and Lefkowitz, L.S. (1988) Using a planner to support office work. In Proceedings of COIS '86, Palo Alto, CA, USA, 55-62.
- Coutaz, J. (1988) De l'ergonome à l'informaticien: pour une méthode de conception et de réalisation des systèmes interactifs. Colloque ERGO-IA, Biarritz, 4-6 Octobre, France.
- Davis, R. (1983) Task analysis and user errors: a methodology for assessing interactions. International Journal of Man Machine Studies, 19, 561-574.
- Fikes, R.E. (1981) Automating the problem solving in procedural office work. Comm. AFIPS Office Automation Conference, Houston, TX, USA.
- Foley, J. D. and Van Dam, A. (1984) Fundamentals of interactive computer graphics. Addison Wesley.
- Foutain, A. J., and Norman, M. A. (1985) Modelling user behaviour with formal grammars. In P. Johnson and S. Cook (eds.): People and Computers: designing the interface. Cambridge University Press, 3-12.
- Gaines, B. R., and Shaw, M. (1984) Principles of dialog engineering. NATO Workshop on Research needs in computer-user interaction. Loughborough, September 1984.
- Green, M. (1985) Design notations and user interface management systems. In G.E. Pfaff (ed.) User interface management systems. Springer Verlag.
- Hammond, N., Jourgensen, A., MacLean, A., Barnard, P., and Long, J. (1983) Design practice and interface usability: evidence from interviews with designers. In Proceedings of CHI'83, A. Janda (ed.), New York, ACM.
- Hayes-Roth B. and Hayes-Roth, F. (1979) A cognitive model for planning. Cognitive Science, 3, 275-310.

- Hayes-Roth, F.,Klahr, P.,Mostow, J. (1981) Advice taking and knowledge refinement. in Anderson, J.R.(ed) Cognitive skills and their acquisitions, Erlbaum., N.J.
- Hewitt, C. (1986) Offices are open systems.ACM Tansactions on Office Information Systems.Vol 4 , n°3, 271-287.
- Hoppe, H. U., Tauber, M., and Ziegler, J. E. (1986) A survey of models and formal description methods in HCI with example applications. ESPRIT Project HUFIT, Report B3.2a.
- Jacobs, R.J.K. (1983) Using formal specifications in the design of a human-computer interface. Communications of the ACM, 26, 4, 259-264.
- Johnson, P., Diaper, D., and Long, J. (1984) Tasks, skills and knowledge: task analysis for knowledge based descriptions. INTERACT 84', B. Shackel (ed.), North Holland, 499-503.
- Kieras, D., and Polson P.G. (1983) A generalized transition network representation of interactive systems. Proceedings of CHI '83, Boston, MA, USA.
- Kieras, D., and Polson P.G. (1985) An approach to the formal analysis of user complexity. International Journal of Man-Machine, 22, 365-394.
- Lochowsky, F. H., Hogg, J. S., Weiser, S. P., and Mendelzon, A. O. (1988) OTM: specifying office tasks. In Proceedings of COIS '86, Palo Alto, CA, USA, 46-53.
- Malhotra, A. Thomas, J. C., Carroll, J. M., and Miller, L. A. (1980) Cognitive processes in design. International Journal of Man-Machine Studies, 12, 119-140.
- Mayer, R.E.and Bayman, P.(1981) Psychology of calculator langage:a framework for describing differences in user's knowledge. Communications of the ACM, 24, 8, 511-520.
- Moran, T.P. (1981a) The Command Langage Grammar: a representation for the user interface of interactive computer systems. International Journal of Man Machine Studies, 15, 3-50.

Moran, T.P. (1981b) An applied psychology of the user. In ACM Computing Surveys, Special Issue on The Psychology of Human-Computer Interaction, Vol. 13, No. 1, 1-11.

- Moran, T.P. (1983) Getting into a system: External-Internal Task Mapping Analysis.Human factors in Computers Systems, Gaithersburg, MD, USA.
- Morgenstern,L.A. (1986) First order theory of Planning,Knowledge and Action.Conference on theoricals aspects of learning about knowledge.Ed Halpern N.Y.
- Nickerson, R. S. (1986) Using computers: the human factors of information systems. The MIT Press, Cambridge, MA, USA.
- Nielsen, J. (1986) A virtual protocol model for computer-human interaction. International Journal of Man-Machine Studies, 24, 301-312.
- Norman, D. A. (1983) Some observations on mental models. In D. Gentner and L. A. Stevens (eds.) Mental models. Hillsdale, N.J., Lawrence Erlbaum.

- Norman, D. A., and Draper, S. W. (1986) User-centered system design. Lawrence Erlbaum Associates.
- Oberquelle, H., Kupka, I, and Maass, S. (1983) A view of human-machine communication and co-operation. International Journal of Man-Machine Studies, 19, 309-333.
- Payne, S. J. (1984) Task-Action Grammars. Proceedings of Interact '84, London, U.K., 139-144.
- Payne, S. J., and Green, T. R. G. (1983) The user's perception of the interaction language: a two-level model. Proceedings of CHI '83, Boston, MA, USA, 202-206.
- Reisner, P. (1981) Formal grammar and human factors design of an interactive graphic system. IEE Transactions on Software Engineering , 5, 229-240.
- Reisner, P. (1984) Formal grammars as a tool for analysing ease of use: some fundamental concepts. In J. C. Thomas and M. L. Schneider (eds.). Human Factors in Computer Systems. Norwood, N.J.: Ablex.
- Reisner, P. (1986) Human-computer interaction: what is it and what research is needed? IBM Research Report RJ 5308.
- Richard, J.F. (1983) Logique du fonctionnement et logique de l'utilisation. Rapport de recherche INRIA n° 202.
- Richard, J.F. (1986) The semantics of action: its processing as a function of the task. Rapport de recherche INRIA n°542.
- Richards, J. N. J., Bez, H. E., Gittings, D. T., and Cooke, D. J. (1986) On methods for interface specification and design. International Journal of Man-Machine Studies, 24, 545-568.
- Roberts, T. R., and Moran, T. P., (1983) The evaluation of text editors: methodology and empirical results. Communications of the ACM, 26, 265-283.
- Rouse, W. B. (1987) Much ado about data. Human Factors Society Bulletin, 30(9), 1-3.
- Sacerdoti, E. D. (1974) Planning a hierarchy of abstraction spaces. Artificial Intelligence, 5, 115-135.
- Sacerdoti, E. D. (1977) A structure for plans and behavior. Elsevier Computer Science Library: New York, London, Amsterdam.
- Scapin, D. L. (1981) Ergonomie des dialogues homme-ordinateur: revue de question. Le Travail Humain, 42 (2), 275-292.
- Scapin, D. L. (1987) Guide ergonomique de conception des interfaces homme-machine. Rapport INRIA N° 77.
- Sebillotte, S. (1987) La plannification hiérarchique comme méthode d'analyse de la tâche; analyse des tâches de bureau.Rapport de recherche INRIA n° 599.
- Sebillotte, S. (1988) Hierarchical planning as a method for task analysis: the example of office task analysis. Behaviour and Information Technology, Vol. 7, No. 3, 275-293.
- Shank and Abelson (1977) Scripts, plans, goals and understanding. Hillsdale, New Jersey: Erlbaum.

- Sharratt, B. D. (1987) Top-down interactive systems design: some lessons learnt from using command language grammar. Proceedings of INTERACT'87, H.J. Bullinger and B. Shackel (eds.), North-Holland, 395-399.
- Shneiderman, B. (1987) Designing the user interface: strategies for effective human-computer interaction. Addison-Wesley.
- Sibertin-Blanc, C. (1988) Le prototypage des applications interactives à l'aide de réseaux de Petri. (soumis pour publication).
- Simon, C. W. (1987) Will egg-sucking ever become a science? Human Factors Society Bulletin, 30(6), 1-4.
- Smith, L. L. (1987) Whyfore human factors? Human Factors Society Bulletin, 30(2), 6-7.
- Smith, S.L., and Mosier, J. N. (1984a) The user interface of computer-based information systems: a survey of current software design practice. Behaviour and Information Technology, Vol. 3, No. 3, 195-203.
- Smith, S.L., and Mosier, J. N. (1984b) Design guidelines for the user interface software. Technical Report ESD-TR-84-190, U.S.A.F. Electronic Systems Division, Hanscom Air Force Base, MA, USA (NTIS No. AD A154 907).
- Smith, S.L., and Mosier, J. N. (1986) Standards versus guidelines for designing user interface software. Behaviour and Information Technology, Vol. 5, No. 1, 47-61.
- Sugaya, H, Stelovsky, J., Nievergelt, J., and Biagioni, E. S. (1984) XS-2: an integrated interactive system. BBC Forshungsbericht KLR 84-73 C. Baden, Suisse.
- Tueni, M. , Li, J. and Fares, P. (1988) AMS: a knowledge-based approach to tasks representations, organization and coordination. In Proceedings of COIS '86, Palo Alto, CA, USA, 78-87.
- Visser, W. (1987) Abandon d'un plan hiérarchique dans une activité de conception. In Proceedings Cognitiva '87, 18-22 Mai, Paris, France, 366-371.
- Watt, D. A. and Madsen, O. L. (1983) Extended attribute grammars. The Computer Journal, Vol. 26, No 2, 142-386.
- Young, R.M.(1981). The machine inside the machine: user's model of pocket calculators. International Journal of Man Machine Studies, 15, 51-85.
- Young, R. M. (1983) Surrogates and mappings: two kinds of conceptual models for interactive devices. In A. L. Stevens and D. Gentner, (eds.) Mental Models, Hillsdale, N.J.: Erlbaum, 35-52.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique