De la modelisation des taches a la specification d'interfaces utilisateur

Hamid Hammouche

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

De la modélisation des tâches à la spécification d'interfaces utilisateur

Hamid HAMMOUCHE

N° 1959
Juillet 1993

PROGRAMME 3

Intelligence artificielle,
systèmes cognitifs et
interaction homme-machine

Programme 3

Intelligence artificielle, Systèmes cognitifs et Interaction homme-machine

DE LA MODELISATION DES TACHES A LA SPECIFICATION D'INTERFACES UTILISATEUR

From Task Modelling To User Interfaces Specification

Hamid Hammouche

Mai 1993

Cette étude s'inscrit dans le cadre d'une convention de recherche entre le CENA (Centre d'Etudes de la Navigation Aérienne) et l'INRIA (Institut National de Recherche en Informatique et en Automatique). *Convention n°90/C.0008.*

DE LA MODÉLISATION DES TÂCHES A LA SPÉCIFICATION D'INTERFACES UTILISATEUR

RESUME

Parce que la compatibilité entre la représentation utilisateur (cognitive) et la représentation machine (fonctionnelle) est insuffisante aux niveaux de la représentation et de l'organisation des tâches (objectifs) et des procédures associées, et ce malgré les récents progrès dans les domaines de l'ingénierie des interfaces, l'objectif aujourd'hui consiste à compléter les méthodes actuelles du génie logiciel ainsi que les architectures de spécification d'interfaces utilisateur par des apports de nouveaux modèles et concepts d'autres disciplines telles que les sciences cognitives, l'ergonomie, etc. En effet, notre approche consiste à formaliser et appliquer les résultats de l'ergonomie mettant en rapport les caractéristiques conceptuelles de la tâche avec celles de l'interface. Pour ce faire, il nous est apparu nécessaire de définir d'abord la notion de modèle conceptuel de l'interface et les concepts qui y participent afin d'expliciter et de justifier le passage d'un modèle de tâches à un modèle (structure) approprié de l'interface et ce d'une façon incrémentale et cohérente.
Trois points sont traités dans ce rapport. Le premier situe les définitions des concepts de l'interaction homme-machine. Le deuxième point concerne la situation actuelle dans le domaine de la conception d'interfaces et le domaine de l'analyse des tâches et ses diverses contributions par rapport à la conception et l'évaluation des interfaces. Dans le troisième point, nous situons la perspective de la modélisation du raisonnement orienté tâche vis à vis de la conception d'interfaces, puis nous définissons et discutons des étapes et procédés (heuristiques) que nous pensons affiner et approfondir dans l'objectif de concevoir une méthode formelle de spécification d'interfaces ergonomiques.

Mots-clés: interaction homme-machine, modèles de tâches, méthodes d'évaluation et de conception, règles et heuristiques ergonomiques, objets intermédiaires et spécification de l'interface utilisateur.

FROM TASK MODELLING TO USER INTERFACES SPECIFICATION

ABSTRACT

The compatibility between the user (cognitive) representation and the computer (functional) representation remains not sufficiently related to the levels of the tasks (goals) representation and organisation, and their associated procedures, despite the recent progress in the domain of interface engineering. Given this state of affairs, efforts have been devoted to applying the new models and concepts taken from other disciplines such as cognitive sciences and ergonomics, to actual software engineering methods and to interface specification architectures. Our approach consists in the formalization and the application of ergonomic knowledge to relate the conceptual characteristics of the tasks with the interface. To do so, it appeared necessary to first define the notion of a conceptual model of the interface and also the concepts that one part of it, in order to make explicit and to justify a coherent and incremental path from a task model to an appropriate model of the interface.
This report addresses three main points. The first one concerns the definitions of the concepts of the human-computer interaction. The second point is related to the actual state of the art in the domain of interface design and to tasks models and their relative contributions to the design and evaluation of interfaces. In the last point, we position the perspective of the modelisation of the task-based reasoning with respect to interface design, and we define and discuss the steps and procedures (heuristics) that will be refined in the order to develop a formal method for ergonomic interface specification.

Key words: human-computer interaction, tasks models, design and evaluation methods ergonomics, rules and heuristics, intermediate objects and user interface specification.

TABLE DES MATIÈRES

1 INTRODUCTION

1.1 INTERFACE UTILISATEUR : POUR UNE AVANCÉE

On assiste aujourd'hui à une évolution très rapide dans le domaine de l'interaction homme-machine. En effet, de nombreux travaux de recherche et de l'industrie sont menés (e.g. sur la modélisation de l'utilisateur, sur les outils de construction d'interfaces). Cependant, la compatibilité entre la représentation utilisateur (cognitive) et la représentation machine (fonctionnelle) demeure insuffisante aux niveaux de la modélisation et de l'organisation des tâches (objectifs utilisateur) et des procédures associées. Par conséquent, les utilisateurs ne s'adaptent que partiellement, ou pas du tout, aux différents outils et applications développés pour les assister dans leurs tâches et accroître leur productivité — cela malgré les récents progrès dans les domaines de l'ingénierie des interfaces.

L'objectif aujourd'hui consiste à compléter les méthodes actuelles du génie logiciel (e.g. Merise, Sadt) ainsi que les architectures (e.g. Seeheim) de spécification d'interfaces utilisateur par des apports de nouveaux modèles et concepts (e.g. les modèles de l'interaction, les recommandations) d'autres disciplines telles que les sciences cognitives, l'ergonomie, etc. En effet, ces théories tentent non seulement de comprendre et d'analyser les documents et/ou outils utilisés par les opérateurs mais aussi d'étudier l'activité réelle[1] de ces derniers, les difficultés ainsi que les incidents auxquels ils sont confrontés lors de la réalisation de leurs tâches, afin de réduire l'écart et/ou la complexité entre la représentation mentale de l'opérateur : le modèle cognitif, et la représentation abstraite de l'application : le modèle conceptuel, i.e. concevoir l'application en tenant compte des connaissances d'utilisation et pas seulement de la logique fonctionnelle de l'information.

1.2 INTERACTION HOMME-ORDINATEUR : UN DOMAINE PLURIDISCIPLINAIRE

De fait, l'ergonomie des logiciels dans la conception d'applications utilisateur — ou la prise en compte des divers aspects ergonomiques dans les méthodes de conception de systèmes interactifs — constitue depuis quelques années, un domaine de recherche pluridisciplinaire (e.g. modélisation des connaissances, modélisation de l'interface). En particulier, on constate un manque de formalisation des connaissances au niveau tâche (sémantique), e.g. pour la modélisation du contrôle, dans le domaine de l'intelligence artificielle (IA) ; l'absence de la prise en compte explicite des aspects utilisateur, c'est-à-

[1] Par opposition à la tâche prescrite utilisée dans les méthodes du génie logiciel. L'étude de l'activité des opérateurs (e.g. [Johnson et al.91; Sebillotte91]), i.e. en situation, permet une modélisation formelle, par le biais d'un formalisme de description, des représentations qu'ont les utilisateurs de leurs tâches.

dire de méthodologie de conception d'interfaces, dans le domaine du génie logiciel ; etc. Par conséquent, il apparaît de plus en plus nécessaire, dans le cadre de la conception de systèmes utiles et utilisables, de coordonner les efforts et résultats de divers domaines de recherche tels que l'IA, l'ergonomie, les sciences cognitives, etc.

Dans cette perspective, de nombreux travaux sont menés dans chacun de ces domaines, en particulier dans les domaines de l'ergonomie et des sciences cognitives où des résultats commencent à s'affirmer depuis quelques années. En ergonomie, ces travaux sont notamment concernés par la modélisation de l'utilisateur [e.g. Card et al.83]. Parallèlement, des études ont porté sur la modélisation de l'interaction homme-machine [e.g. Norman et al.86, Shneiderman87]. D'autre part, d'autres travaux se sont focalisés sur la modélisation de l'interface considérant que la couche interface doit être distincte de la couche application (noyau fonctionnel) [Green85]. On trouve dans [Simon88 ; Hoppe91] d'autres classifications des différents modèles et approches concernant la prise en compte des aspects tâches et utilisateur dans la conception et/ou l'évaluation d'interfaces homme-machine.

Un des problèmes rencontrés en psychologie, en ergonomie et en ingénierie est celui de la représentation et la manipulation des connaissances. En ce domaine, l'apport de l'informatique, en particulier de l'IA, a porté sur l'étude de nouveaux modèles de représentation des connaissances (e.g. les modèles à objets [Masini et al.89]) et, par conséquent, de nouveaux environnements de développement (e.g. les boîtes à outils) qui ont fortement influencé les nouvelles approches de conception (e.g. le modèle serveur-clients de X-windows [Scheifler et al.86]) du fait de leurs possibilités de spécification (maquettage) et de réutilisation.

Il nous semble en outre que, partant des apports et concepts de ces différents travaux, l'un des objectifs ultimes dans le domaine de l'interaction homme-machine concerne la représentation et la mise en correspondance de façon formelle, au sens informatique, des modèles cognitifs de l'interaction (e.g. la tâche, l'utilisateur) et des résultats de l'ergonomie (e.g. les critères, les recommandations, etc) de manière à permettre la spécification et la conception d'interfaces qui répondent aux objectifs et caractéristiques des utilisateurs. Il s'agit dans ce cas de concevoir des méthodes et des outils d'ingénierie englobant les divers concepts des diverses disciplines dans toutes les étapes du cycle de conception, depuis l'analyse initiale, les spécifications jusqu'à la réalisation (e.g. [Scapin et al.88 ; Carter90a]). Toutefois, en l'absence de méthodes formelles qui répondent au problème concret de la prise en compte des aspects utilisateur, les concepteurs de systèmes interactifs ont tendance à s'appuyer sur les approches de l'ingénierie et utiliser voire développer des outils qui facilitent la réalisation et la réutilisation de prototypes d'interfaces.

1.3 SPÉCIFICATION D'INTERFACES CONCEPTUELLES : UNE APPROCHE

Nous proposons de fait une approche qui se situe à la frontière de la conception d'applications utilisateur et de la spécification d'interfaces ergonomiques. Il s'agit là de la prise en compte de la tâche du point de vue de l'utilisateur dans le but de concevoir des applications ergonomiques. Pour ce faire, il est primordial de disposer de représentations adéquates (en termes d'objets) permettant d'analyser et de manipuler des descriptions de tâches complètes et cohérentes. Cet aspect constitue le premier volet de l'approche.

Le second volet concerne la définition des mécanismes qui permettent d'établir à partir des représentations formelles de la tâche, décrite à l'aide du formalisme MAD (Méthode Analytique de Description de tâches), des spécifications conceptuelles de l'interface et/ou de l'application d'un point de vue de l'utilisation. Ces mécanismes[2] permettent par exemple d'identifier les objets et les actions en rapport avec les objectifs de la tâche utilisateur, i.e. les connaissances d'utilisation. De ce fait, cette approche se distingue de celles qui s'inspirent essentiellement des connaissances de conception — les méthodes, les techniques et les outils de conception (§3) — et/ou de fonctionnement.

En effet, les connaissances de fonctionnement (e.g. [Richard83]) correspondent au noyau fonctionnel de l'application (e.g. accès à une base de données, calcul) alors que les connaissances d'utilisation reflètent les objets et actions de l'application, d'un point de vue de la sémantique des objectifs utilisateur, c'est-à-dire la définition de l'interface selon sa dynamique d'utilisation (e.g. les fonctionnalités, le dialogue, etc). A ce niveau de spécification, la frontière entre l'interface et l'application [Pffaf85] semble floue, à savoir que l'interface et l'application concernent les mêmes concepts mais selon des points de vue différents.

De par cette modélisation, nous espérons ainsi fournir des techniques permettant de concevoir des interfaces qui correspondent bien à la logique de travail de l'utilisateur et ce à tous les niveaux d'abstraction (e.g. présentation, dialogue). En particulier, nous nous intéressons aux aspects sémantiques et conceptuels sachant qu'ils concernent l'interface en terme d'adéquation des objectifs des deux partenaires de l'interaction : l'utilisateur et l'application ; et de fait déterminants lors de la spécification des aspects syntaxique et lexical. En outre, cette perspective concerne, à l'heure actuelle, les aspects les moins bien maîtrisés de la spécification d'interface car simplement les connaissances (e.g. cognitives) qu'elle implique ne sont pas toujours prises en

---

[2] Pour ce faire, il nous est apparu nécessaire de définir d'abord la notion de modèle conceptuel de l'interface et les concepts qui y participent (§5), afin d'expliciter et de justifier le passage du modèle de la tâche à un modèle approprié de l'interface et ce d'une façon incrémentale et cohérente.

considération en ce sens qu'elles sont souvent méconnues et/ou informelles. Par conséquent, la seule approche logicielle a montré ses limites de ce point de vue du fait que l'utilisateur demeure au centre de l'activité de collaboration homme-machine. De ce fait, ces différents aspects doivent concerner, de manière explicite, toute approche de spécification d'interfaces.

1.4 Objet et Organisation du Rapport

Dans un premier temps, nous donnons et situons les définitions des principaux concepts et notions au centre des diverses études et discussions de ce rapport ; en particulier, les concepts en rapport avec la spécification d'interfaces utilisateur. Nous présentons ensuite une étude critique de la situation actuelle dans le domaine de la conception d'interfaces homme-machine, en mettant l'accent sur les aspects ergonomiques. Cette étude concerne : les méthodes du génie logiciel, les modèles cognitifs de l'interaction, les principes et connaissances ergonomiques ainsi que les approches de l'ingénierie tels que les environnements et outils de développement d'interfaces.

Nous discutons, en effet, les apports et insuffisances de chacune de ces approches par rapport à la conception d'interfaces orientées-tâche (le niveau sémantique). D'autre part, nous avons pris, dans le cadre de cette revue, le parti d'une approche orientée utilisateur ; en particulier, par rapport au dilemme "système-utilisateur" à savoir : plus on automatise moins l'utilisateur contrôle ou inversement. Nous estimons que, pour ce faire, il devient indispensable de réfléchir à des modèles *consensuels*, c'est-à-dire à la limite de la coopération entre le système et l'utilisateur permettant à chacun de compléter les insuffisances de l'autre. Ceci nécessite sans doute des approches pluridisciplinaires, i.e. intégrant d'autres perspectives (e.g. ergonomique, linguistique, psychologique) d'analyse et de description et par conséquent de représentation et de modélisation.

L'évaluation des modèles de tâches utilisateur par rapport aux besoins et objectifs de spécification des aspects conceptuels de l'interface, et de la perspective de la modélisation orientée tâche en intelligence artificielle dans le cadre de la conception de systèmes à base de connaissances [e.g. Tong90], fera l'objet de la troisième partie. En particulier, on discutera l'approche qui consiste à définir des mécanismes de déduction récursifs afin d'inférer les aspects de l'interface à des niveaux élevés d'une structure arborescente de tâches [Pierret et al.89].

La dernière partie sera consacrée à l'approche de spécification d'interfaces orientées tâches. Elle concerne de fait la définition de l'interface conceptuelle, la modélisation objet de la tâche, la définition et modélisation des concepts de l'interface ainsi que les mécanismes et heuristiques de spécification. En guise de conclusion, nous discutons les

étapes que nous pensons affiner et approfondir dans l'objectif de concevoir une méthode formelle de spécification d'interfaces ergonomiques.

2 CONCEPTS ET DEFINITIONS

Nous avons jugé utile de définir, avant d'entamer l'étude sur les théories, les modèles et les architectures concernant la conception d'interfaces, les principaux concepts et notions utilisés dans ce rapport afin de mieux situer le cadre de nos objectifs sachant que les définitions concernant les concepts de l'interaction utilisateur-ordinateur varient, souvent, d'un auteur à l'autre.

2.1 L'ERGONOMIE DES INTERFACES

La psychologie ergonomique s'intéresse aux situations où l'homme et l'ordinateur doivent coopérer pour réaliser une tâche précise. Le problème général que l'ergonomie cherche à résoudre est celui de l'optimisation de cette coopération [Bisseret86]. Cela consiste à réduire l'écart (distance) dans tous ses aspects, en particulier sémantique, entre le monde réel (de l'utilisateur) et le monde abstrait (l'abstraction du monde réel par la machine) de l'application qui n'ont pas nécessairement, a priori, la même logique de fonctionnement. De fait, cette coopération suppose une compatibilité à deux niveaux : la représentation et le traitement des connaissances. A ce niveau, distinct du niveau lexical (e.g. caractéristiques des affichages, etc), on parle d'interfaces ergonomiques. C'est dans ce contexte précis que se situe ce rapport.

2.2 LE CONCEPT D'INTERFACE UTILISATEUR

En ergonomie, l'interface homme-machine englobe tous les aspects des systèmes informatiques qui influencent la participation de l'utilisateur à des tâches informatisées [Scapin86]. Dans ce rapport, on ne s'intéresse pas aux aspects annexes tels que les manuscrits d'aide, les documents papiers, etc. En effet, cette définition concerne en particulier les aspects de l'interface en rapport avec la tâche utilisateur. De ce point de vue, l'interface est considérée comme un dispositif qui assiste l'opérateur lors de l'utilisation des fonctionnalités d'un outil dans le cadre de sa tâche. Elle a pour rôle d'établir une communication entre l'outil et l'opérateur. Cette communication se traduit par des actions de l'opérateur sur l'outil et un changement de contexte, i.e. d'état, de l'outil en réponse à ces actions.

De fait, l'interface utilisateur constitue le moyen direct permettant à l'utilisateur de réaliser sa tâche. Elle devrait par conséquent concerner la tâche dans tous les aspects de l'interaction, qui sont d'ailleurs dépendants : d'une part, l'aspect présentation perceptible par l'utilisateur, et d'autre part celui de l'adéquation (cognitive) entre les représentations

de l'utilisateur et le contexte de l'application. A ce niveau de description, la frontière entre l'interface et l'application semble floue, c'est-à-dire que l'interface et l'application portent sur les mêmes concepts mais selon des points de vue différents.

2.3 LE CONCEPT DE LA TÂCHE

Avant de revenir sur les modèles et les aspects de la tâche à prendre en compte de façon explicite pour la conception d'interfaces (§4), on préfère dans un premier temps donner les perspectives des deux principaux courants utilisant cette notion, à savoir l'ergonomie et l'intelligence artificielle.

En ergonomie, le concept de tâche est utilisé comme un support de description de l'activité des opérateurs [e.g. Wilson et al.88]. En effet, l'activité est une notion proche du monde réel, i.e. où la symbiose des connaissances de l'opérateur (e.g. l'expérience) et de la tâche (e.g. le contexte) est forte. Il en découle que l'activité est une réalisation de la tâche. Ainsi, l'objectif dans une description orientée-tâche est d'établir de manière générique les diverses connaissances mises en oeuvre par les opérateurs (utilisateurs) dans leur activité. Il s'agit là d'une projection du monde réel sur un *espace-connaissance* permettant d'instancier et/ou de vérifier les différentes réalisations de la tâche. De fait, on parle de modèle de tâche utilisateur.

Une tâche est en général décrite par un but (l'objectif de la tâche), les conditions d'exécution de la tâche, la liste des sous-tâches et leurs contraintes fonctionnelles et/ou temporelles. Les conditions regroupent les pré- et post-conditions souvent décrites sous forme d'expressions booléennes. Par rapport à la spécification d'interfaces, les pré-conditions concernent les valeurs admissibles des objets nécessaires à l'exécution de la tâche. Les post-conditions, quant à elles, évaluent les variables de la tâche après l'exécution et effectuent un changement de contexte en mettant à jour d'autres variables d'interaction et/ou de l'application ; permettant par exemple d'enchaîner sur d'autres sous-tâches ou bien redonner le contrôle à l'utilisateur, etc.

D'autre part, le concept de tâche est utilisé en intelligence artificielle, en particulier en acquisition et modélisation des connaissances [e.g. Boy92]. Un des objectifs de cette perspective est d'expliciter les connaissances de contrôle, c'est-à-dire modéliser le raisonnement en termes de fonctions ou de rôles [Wielinga et al.86]. Cela, afin de distinguer les connaissances de surface (le contrôle) des connaissances profondes (le domaine). Ainsi, par exemple, le générateur de systèmes experts orienté objets, Smeci [Smeci91], fonctionne à partir d'une pile gérant les connaissances de contrôle en paquets de règles appelés "tâches" par l'intermédiaire d'une stratégie définie par le programmeur ou par défaut.

Les modèles de tâche en ergonomie ont tendance à rassembler dans une même unité-tâche deux types de connaissances : les connaissances déclaratives (les objets du domaine) et les connaissances procédurales (leurs mises en oeuvre en termes d'actions) alors que l'idée dans l'approche IA consiste à séparer les connaissances du domaine des connaissances de contrôle. Toutefois, ce sont les connaissances de contrôle qui mettent en oeuvre les connaissances du domaine au cours d'un raisonnement. Cela, par le biais de mécanismes procéduraux. Il semble ainsi que, de ce point de vue, la frontière entre les deux perspectives du concept de tâche soit floue.

2.4 LES CONCEPTS DE L'INTERFACE

La notion de "concepts de l'interface"[3] a aussi été utilisée dans le cadre des outils de l'ingénierie des interfaces, et concerne divers niveaux d'abstraction. Tels qu'ils sont définis dans les guides de style (e.g. Toolbox du Macintosh), ces concepts regroupent essentiellement les objets et les services des interfaces ainsi que les règles d'organisation. En effet, les services des boîtes à outils concernent les fenêtres de dialogue, les boutons, alors que ceux des environnements de génération d'interfaces ont pour vocation entre autres l'établissement des liaisons entre l'interface et l'application.

En revanche, la définition que nous avons adopté et attribué à cette notion est d'un plus haut niveau d'abstraction, celui de la tâche, en ce sens qu'elle est indépendante des outils de l'ingénierie. Nous parlons essentiellement du concept d'état associée à une tâche, de zone de dialogue ou de message, de fonctionnalité de l'interface, de la structure de dialogue, etc. Nous reviendrons sur ces concepts, lors de la description de l'interface conceptuelle, dans le dernier chapitre.

2.5 CONCLUSION

Ces différents concepts et définitions sont à la base de l'approche de spécification d'interfaces orientées tâche. En effet, à partir d'une représentation formelle de la connaissance utilisateur (la tâche), nous espérons mettre en évidence, par l'intermédiaire de recommandations ergonomiques, les concepts de l'interface et leurs relations de manière à bien rendre compte des objectifs et besoins de l'utilisateur dans une application, tant sur le plan utilisation (l'interface) que sur la plan fonctionnement (l'application).

Ces recommandations seront traduites sous forme d'heuristiques et de règles de production, formalisant, d'une part, les données liées à la tâche (e.g. structure des

[3] On trouvera dans [Normand92] une classification des concepts et services de l'interface par rapport aux composants et outils de construction et de génération d'interfaces homme-machine.

différents objets associés aux préconditions d'une tâche) et les spécifications d'interface (e.g. vérifier les contraintes (valeurs) des objets, informer l'utilisateur de l'état de ces objets), et de fait établissant, d'autre part, un lien sémantique entre la tâche et l'interface.

3 LA CONCEPTION D'INTERFACES : SITUATION ACTUELLE ET PERSPECTIVES

Dans la première partie de ce chapitre, nous allons étudier la perspective des méthodes du génie logiciel par rapport à la conception d'interfaces ergonomiques, c'est-à-dire intégrant les points de vue de l'ergonomie au niveau de la conception. Nous présentons ensuite les modèles à la base des méthodes intégrant le point de vue de l'utilisateur, les approches générant l'interface à partir des spécifications fonctionnelles de l'application, ainsi que les démarches utilisées aujourd'hui au sein des projets industriels pour la conception de produits ergonomiques.

La deuxième partie concerne les résultats des travaux en ergonomie des logiciels (e.g. les recommandations) ainsi que les théories et modèles de l'interaction homme-machine. La troisième partie sera consacrée à la présentation des concepts et approches des outils et environnements de développement d'interfaces. La plupart de ces derniers partent de l'idée, à savoir les travaux de Seeheim [Pfaff85], qui consiste à séparer le côté interface de celui du noyau fonctionnel de l'application. Enfin, nous terminons ce chapitre en discutant les aspects pertinents que nous pensons retenir à la base de notre modèle dont l'objectif ultime est la conception d'un outil de description de tâches pour la spécification de la structure de l'interface tels que les objets conceptuels de l'interaction (e.g. le concept d'état), leurs dynamiques et leurs liens avec les concepts de l'application ; aspects que nous décrivons dans le dernier chapitre.

3.1 LES MÉTHODES INFORMATIQUES ET LA SPÉCIFICATION D'INTERFACES

Les méthodes du génie logiciel [Théron88], telle que Merise [Tardieu et al.86], ont du mal à tirer profit des possibilités qu'offrent les nouveaux environnements matériels et logiciels facilitant, par conséquent, la conception d'interfaces interactives au sens d'interfaces graphiques et/ou animées (e.g. les systèmes de gestion de fenêtres), etc. Dans le même ordre de réflexion, [Barthet86 ; Seaton et al.92] exposent clairement le fait que l'utilisateur ne soit pas pris en compte, du point de vue de ses caractéristiques propres et du point de vue de sa tâche, dans les méthodes classiques de conception informatiques.

En effet, la méthode Merise, la plus répandue des méthodes de conception, beaucoup n'étant en fait que des variantes, étudie la décomposition des procédures fonctionnelles

en tâches au niveau organisationnel en terme de poste de travail sans donner aucune règle ou mode opératoire précis pour la prise en compte de la manière dont les utilisateurs se représentent leurs tâches, c'est-à-dire leur façon de travailler. En conséquence, il en résulte que la spécification de l'interface est repoussée au moment du développement du logiciel. D'autre part, les spécifications qui concernent le dialogue (l'interaction) homme-ordinateur sont déduites de la logique du découpage du système d'information et non de la logique d'utilisation du poste de travail qui constitue le point de vue de l'utilisateur. Par ces méthodes seules, les concepteurs n'appréhendent à aucun moment la conception du point de vue de l'utilisateur, en procédant par exemple par une analyse et une description de l'activité permettant d'aboutir à une modélisation des tâches qui soit un support à la spécification de l'interface du point de vue de l'utilisation. C'est cette approche orientée-tâche qui permet entre autres de mieux définir et structurer les fonctions, les objets et les opérations nécessaires à l'utilisateur pour accomplir avec le plus de facilité ses tâches. Cette perspective influence en effet la conception de l'application au niveau fonctionnel.

De fait, nous estimons que pour une amélioration de l'interaction homme-ordinateur, il est nécessaire d'appréhender au même niveau conceptuel l'interface et les fonctionnalités de l'application car cette amélioration passe par une structuration de l'interface mais aussi par une transformation de la structure de l'application au niveau conceptuel, celui de la tâche. Par ailleurs, ce constat se justifie de plus en plus pour les outils de développement d'interfaces utilisateur [Gomes et al.90]. Dans cette perspective, [Barthet88] propose une méthode, Diane, dont la caractéristique principale est d'être basée sur la méthode Merise profitant par conséquent de l'acquis de cette méthode, ce que nous allons décrire ci après.

3.1.1 La Méthode Diane

La méthode Diane se situe au niveau organisationnel de la conception d'une application. Elle permet de définir les rôles respectifs du contrôleur et de l'utilisateur. En effet, la méthode reprend le modèle conceptuel et organisationnel des traitements de Merise en le complétant par la notion d'événement utilisateur permettant ainsi de définir la répartition du contrôle homme-machine au niveau conceptuel (e.g. opération facultative ou obligatoire, déclenchement automatique ou optionnel).

3.1.1.1 Le Modèle

De fait, Diane permet à l'utilisateur de disposer d'une certaine marge de manoeuvre dans le choix des opérations et de leurs enchaînements afin de pouvoir s'adapter aux aléas de son activité. Cette caractéristique définit la "latitude décisionnelle" de l'utilisateur. La logique d'utilisation, qui représente le fonctionnement de l'application

du point de vue de l'utilisateur, est modélisée par l'intermédiaire de trois types de procédures. *La procédure prescrite* concerne la tâche officielle ; elle s'obtient par l'intermédiaire d'entretiens, de questionnaires, etc. *La procédure effective* constitue une variante de la procédure prescrite (e.g. type de l'utilisateur) en ce sens qu'une procédure prescrite peut avoir plusieurs procédures effectives. L'objectif étant de les mémoriser pour les adapter à des utilisateurs différents.

Cette perspective s'apparente avec celle de l'espionnage des générateurs de systèmes à base de connaissances, ou de l'observateur [Petoud90], permettant de découvrir des métaconnaissances [Pitrat90], et/ou d'identifier les raisonnements réguliers dans le but de fournir de l'aide à d'autres utilisateurs occasionnels. Enfin, *la procédure minimale* regroupe l'ensemble des opérations et enchaînements minimaux nécessaires au bon déroulement de la tâche. Ainsi, la méthode Diane enrichit la sémantique de la méthode Merise, où les enchaînements sont définis par rapport à un sous système (domaine) du système d'information, en définissant des enchaînements par rapport aux objectifs des utilisateurs. Dans le même ordre d'idées, une autre méthode, greffant les Réseaux de Pétri à chaque tâche définie à l'aide de la méthode Sadt [IGL89], permet de définir de façon formelle la répartition du contrôle homme-machine [Abed et al.91]. L'outil formel Petri permettant de prendre en compte les notions de parallélisme et d'interruptibilité, notions non traitées dans les grammaires et les réseaux de transition [Hartson et al.89].

3.1.1.2 Le Point de Vue Ergonomique

Le modèle à la base de la sémantique de Diane distingue trois niveaux de représentation. *La représentation conceptuelle* correspond au point de vue du concepteur et à la représentation qu'il a du nouveau système d'information. *La représentation externe* correspond à la représentation de l'application d'un point de vue de l'utilisateur, c'est-à-dire la logique d'utilisation du nouveau logiciel ; à ce propos, on peut s'interroger sur la variabilité de la notion de tâche effective selon qu'il s'agit des données d'une situation existante vs d'utilisation du logiciel. Enfin, *la représentation interne* correspond à la mise en oeuvre des représentations conceptuelles et externes. Elle correspond à la représentation qu'a le programmeur de l'application.

L'aspect ergonomique de la méthode intervient au niveau de la représentation externe. Celle ci étant dépendante de la représentation conceptuelle, de l'existant, de critères ergonomiques généraux et des possibilités techniques (logiciels et matériels). La méthode intègre ainsi deux classes de critères ergonomiques. La première concerne les critères à caractère général, la seconde intègre les critères propres à l'application provenant de l'existant. L'objectif est de donner une représentation externe aux *paramètres constants*, indépendants de l'application, tels que l'aide à l'utilisation (e.g.

interruption) et le guidage (liste des opérations possibles), et aux *paramètres variables* définis dans la représentation conceptuelle. Ces paramètres propres à l'application se divisent en paramètres provenant des procédures qui se traduisent en commandes activables par l'utilisateur (e.g. hiérarchie des menus), et en paramètres provenant des opérations qui se traduisent en données (e.g. zone de dialogue).

3.1.2 Les Spécifications Fonctionnelles et la Génération du Dialogue

Dans ce paragraphe, nous présentons le point de vue qui consiste à générer l'interface à partir des spécifications fonctionnelles de l'application. Il s'agit dans ce cas de construire des interfaces en s'appuyant sur des concepts tels que le graphe d'états, le tableau de bord, ou des règles ergonomiques de présentation.

3.1.2.1 L'Outil MacIDA

Dans la même perspective, les travaux de [Petoud90] centrés sur la répartition du contrôle utilisateur dans les applications de bases de données interactives, ont pour objectifs la génération automatique de l'interface et du dialogue utilisateur à partir des spécifications fonctionnelles décrites à l'aide du modèle Entité-Association, i.e. le modèle des données, dans le langage de spécification IDA. Les données et les fonctions de l'application sont ainsi traduites respectivement en messages et services de l'interface, appelés aussi objets interactifs. L'architecture du système reflète les trois niveaux du modèle de Seeheim. La composante principale du système est le moteur. Il s'occupe de la gestion du dialogue entre l'utilisateur et l'application. Sa tâche consiste à marquer un graphe d'états représentant l'enchaînement des fonctions auxquelles sont associées les messages d'entrées et/ou de sortie qui constituent leurs pré- et post-conditions à l'exécution. Le marquage permet ainsi d'indiquer à l'utilisateur où il se situe dans le graphe de dialogue et les (services) opérations possibles.

Un service est activable lorsque toutes les pré-conditions sont possibles, c'est-à-dire que tous les messages d'entrée sont prêts. Les post-conditions suivent l'exécution de la fonction associée au service et consistent en la génération de messages, la poursuite du marquage du graphe de dialogue ou la mise à jour du tableau de bord selon que celle-ci s'est bien déroulée ou pas. Le tableau de bord est un service permanent et évolutif contenant la liste de tous les problèmes rencontrés, l'état de déroulement des fonctions et suggérant des scénarios possibles. Cette modélisation se rapproche de celle adoptée dans l'environnement UIDE (§3.4).

3.1.2.2 Le Système Ergo-Conceptor

Le système Ergo-Conceptor [Moussa et al.92], part de la même idée, à savoir la génération de synoptiques industriels à partir de la description des besoins informationnels (tâches prescrites) des opérateurs ainsi que des objectifs de contrôle et de commande (analyse du système) issus de l'étape d'analyse du système opérateur-machine. Toutefois, ce système, relativement au précédent, ne génère que des vues statiques, en ce sens que ces vues ne gèrent pas le dialogue opérateur-machine mais plutôt l'impact des variables de contrôle et de commande du système par rapport à l'interface. En outre, la génération des vues est établie par une base de connaissances ergonomiques [Kolski et al.91]. Celle ci est formée de recommandations de présentation graphiques et de structuration de vues, et par conséquent elles ne s'appliquent qu'à ce contexte particulier de spécification de synoptiques. L'objectif de tels systèmes concerne particulièrement l'aide à la résolution de problèmes complexes (e.g. contrôle de procédés industriels) et considère l'interface comme une source d'information parmi d'autres (e.g. [Casale et al.92]).

3.1.3 Discussion

Dans le but de diminuer le fossé entre la logique de fonctionnement et la logique d'utilisation, [Barthet88, op. cit.] introduit la notion de variabilité des représentations qui se traduit par des logiques d'utilisation différentes, i.e. les procédures effectives. Cependant, ces distinctions semblent limitées quant à leurs utilisations pour la spécification de l'interface. En effet, aucun indice ou moyen (e.g. des critères ou mécanismes de catégorisation) n'a été défini pour la représentation et l'utilisation des procédures effectives, lesquelles déterminent les connaissances contextuelles de la tâche. En outre, la méthode Diane est à la base des travaux de [Tarby et al.91] ayant pour objectif la génération automatique de contrôleur de dialogue. Cette génération devrait permettre par conséquent un prototypage plus rapide et donc une prise en compte du dialogue depuis la conception.

A l'issue de l'étude des différents travaux menés dans le cadre de la prise en compte de l'utilisateur dans les modèles généraux et/ou spécifiques de conception de systèmes, on s'aperçoit que les réflexions sont plutôt orientées vers la modélisation d'applications interactives au sens de l'augmentation de la latitude décisionnelle, i.e. de la répartition de pilotage donnant l'initiative à l'utilisateur et le contrôle de cohérence à l'application. D'autre part, ces approches intègrent les principes généraux et/ou de surface de l'ergonomie (tels que la disposition des menus, les boutons) sans cependant aborder la modélisation ergonomique de l'interface au niveau sémantique. L'ergonomie dans ces cas intervient alors comme un plan de projection et non comme une source de modélisation.

Les efforts actuels (e.g. [Dzida et al.90 ; Walsh et al.88]) se concentrent sur les moyens de mise en correspondance de l'organisation de la tâche utilisateur et/ou les facteurs humains avec les concepts et caractéristiques de l'interface et éventuellement leur intégration dans des méthodes de développement à cycle de vie[4]. En effet, le modèle ATOM (§4.1) tente ainsi d'intégrer les spécifications qui découlent de l'analyse de la tâche en termes d'objets (les entités du domaine, les acteurs) et d'actions (les événements) à la méthode globale de développement de systèmes, JSD — Jackson System Development (résumée dans [Cameron86]). En l'état actuel des faits, et compte tenu des difficultés d'utilisation des différents résultats de l'ergonomie et des théories cognitives que nous discutons dans le paragraphe (§3.3), les approches de spécification à base de contrôle utilisateur constituent la majeure partie des méthodes et/ou outils utilisés actuellement dans le cadre de la spécification et de la génération d'interfaces.

3.2 LES DÉMARCHES ERGONOMIQUES

Toutefois, dans le cadre des applications industrielles, on trouve des démarches pragmatiques qui intégrent une double approche à la fois ergonomique et informatique. Cela, afin de concevoir et réaliser des logiciels ergonomiques.

3.2.1 L'Approche

En effet, l'ergonomie commence, dans le cadre de ces démarches, à sortir de son rôle d'évaluateur de produits finis pour intervenir au stade de la conception. Cette collaboration passe par une approche reprenant de fait les étapes classiques de conception d'applications informatiques et intégrant l'ergonomie dans les projets de conception [Benett87 ; Belloti88 ; Brisson et al.90 ; Barthe92])[5]. L'analyse de la demande, l'étude du terrain, la définition des besoins des utilisateurs et la conception de la maquette constituent les principales étapes permettant d'intégrer l'ergonomie au niveau de la conception. Les deux premières étapes permettent d'une part d'étudier la manière dont la démarche ergonomique s'intègre dans le processus de conception, et d'autre part d'effectuer l'analyse de l'existant et l'étude de l'activité de façon plus étendue que le domaine traité par l'application. L'analyse des informations recueillies permet de définir les besoins des utilisateurs, à partir desquels sont prédéfinies les premières spécifications de l'interface en y associant les techniques de dialogue appropriées.

---

[4] On trouvera dans [Benyon92; Diaper et al.92] deux réflexions controversées concernant le rôle de la tâche dans la conception d'applications utilisateur. La première situe le rôle au niveau de l'utilisabilité, la seconde à un plus haut niveau, celui de la spécification.

[5] En outre, le lecteur trouvera dans [Bruder et al.91] une réflexion sur une approche générale intégrant les facteurs humains lors de la conception et de l'introduction des systèmes à base de connaissances au niveau des entreprises.

La maquette est au centre de cette collaboration. Elle décrit la structure générale du dialogue et quelques fonctionnalités particulières simulant les traitements et la dynamique du dialogue. De fait, la maquette est conçue à partir des spécifications de base. Or, actuellement, on ne sait pas écrire les spécifications qui rendent compte correctement des aspects interactifs et navigationnels du dialogue. Pour ce faire, les ergonomes procèdent de manière incrémentale et itérative en effectuant les phases suivantes : définition des objets du dialogue ; définition des procédures de dialogue ; évaluation de la complexité, de l'homogénéité et de la cohérence de l'interface ; définition des objets et attributs de présentation (e.g. icônes, les couleurs), etc.

3.2.2 Discussion

De telles démarches s'inscrivent impérativement dans les projets conçus selon l'aspect d'indépendance entre l'interface utilisateur et les traitements informatiques de l'application. Ainsi, elles ne sont plus soumises aux contraintes de l'organisation interne de l'application et permet par conséquent de décharger les informaticiens d'un travail qu'ils appréhendent souvent, le dialogue utilisateur. Cependant, l'organisation de l'interface doit être compatible même si celle-ci est différente de l'organisation des données et des traitements ; et de fait ne peut concerner les aspects sémantiques. En outre, les méthodes et les règles, qui permettent le passage de la phase de définition des besoins des utilisateurs à la phase de conception de la maquette, demeurent implicites, floues et surtout dépendantes des connaissances et de l'expérience des ergonomes ; ce qui explique qu'à l'heure actuelle, le dialogue ergonome-informaticien passe par la maquette. Et par conséquent, l'absence d'approches (techniques) formelles intégrant les aspects utilisateur à la conception.

3.3 Les Théories et Modèles Cognitifs

Comme nous l'avons souligné en introduction, pour pouvoir intégrer explicitement l'utilisateur dans la conception d'interfaces, il est nécessaire de connaître les caractéristiques de sa tâche : son organisation, ses utilisations, etc [Shephered89].

La modélisation cognitive a pour objectif l'étude des représentations mentales des utilisateurs lors de la réalisation de leurs tâches manuelles et/ou automatisées, dans le but de concevoir et/ou d'adapter les applications utilisateurs selon plusieurs critères tels que l'adaptabilité, l'assistance, la flexibilité, etc. Nous discutons d'abord les perspectives de l'ergonomie en rapport avec la conception d'interfaces, puis les modèles cognitifs simulant l'interaction entre l'utilisateur et le système.

3.3.1 Les Perspectives de l'Ergonomie Cognitive et la Conception d'Interfaces

Depuis quelques années, apparaît de plus en plus le besoin d'intégrer les facteurs humains (aspects utilisateurs) au niveau de la conception d'interfaces homme-machine. Actuellement, cette intégration passe par des collaborations avec les ergonomes (Cf §3.2), ou l'utilisation des manuels (e.g. les guides de conception) sur les facteurs humains [Smith et al.86]. L'objectif aujourd'hui est de fournir des techniques voire des approches formelles permettant d'intégrer ces différentes connaissances dans un processus de conception. Toutefois, plusieurs études ont été menés en ergonomie et/ou en psychologie cognitive. Ces recherches ont abouti à plusieurs contributions. Il s'agit des principes de conception et des recommandations en ergonomie, et des théories et modèles cognitifs de l'interaction en psychologie cognitive.

3.3.1.1 Principes et Approches de Conception d'Applications Utilisateur

Il existe dans [Gould et al. 83 , Hewett et al.86 ; Belloti88 ; Scapin et al.88] un certain nombre de notions et de principes pour la conception d'applications et/ou d'interfaces utilisateur. La plupart de ces principes concernent les concepts (e.g. tâche, utilisateur) et étapes (e.g. mesures, itérations) nécessaires à la conception d'applications interactives. Dans [Johnson et al.89a ; Lim et al.92], on trouve en effet des approches faisant intervenir de manière explicite et/ou implicite les points d'entrées des considérations ergonomiques (human factors) tels que les outils d'analyse de tâches par rapport aux méthodes de conception. Ces études ont permis d'identifier et d'organiser les pré requis de tels outils. Et de fait examiner quant et comment les informations concernant la tâche utilisateur contribueraient à la définition de systèmes utiles et utilisables.

En outre, d'autres travaux ayant pour base la représentation cognitive et procédurale de la tâche utilisateur tels que CLG [Moran81, Browne et al.86], HTA [Shephered89], TAKD [Diaper90] ont respectivement abordé le problème de l'interface selon divers aspects : la modélisation des niveaux de conception d'un système interactif ; la représentation graphique ou tabulaire de la décomposition des buts en tâches, opérations et plans ; l'identification et la représentation des objets et actions ; etc. Les modèles de tâches sont aussi utilisés en évaluation. On distingue dans ce cas les modèles de compétence (competence model) et les modèles de performance (performance model) [Green et al.88]. Les premiers (e.g. TAG [Payne et al.89]) évaluent la complexité des représentations utilisateur d'une interface. Les seconds (e.g. GOMS [Card et al.83]) prédisent les performances des utilisateurs d'une interface donnée.

Cependant, il n'y a pas à l'heure actuelle de techniques et/ou outils formels qui permettent d'intégrer les spécifications conceptuelles de l'interface au cycle de vie de

conception d'applications utilisateur. Par exemple, des approches qui soient pertinentes et utilisables dans le cadre de l'ingénierie[6]. Dans cette perspective, on trouve dans [Johnson91 ; Johnson et al.92], des propositions ayant pour objectif d'intégrer les connaissances de la tâche, en l'occurrence le modèle TKS (Cf §4), avec celles des modèles d'architecture (e.g. [Hill et al.90]) afin d'aboutir à une formulation de la structure et des caractéristiques de l'interface en termes d'objectifs utilisateur. Cette mise en correspondance est envisagée par le biais des objets définis dans chacun des deux modèles.

Quant à la prise en compte explicite des différentes catégories d'opérateurs et de leurs caractéristiques, on trouve dans la méthodologie MOST [Carter91], une approche combinant plusieurs catégories de connaissances (utilisateurs, tâches, données et outils) sous forme de réseaux (Set of Records) interdépendants dans l'objectif de fournir des systèmes adaptatifs. Cette modélisation s'appuie sur le concept de base de connaissances "contextuelles"[7] permettant de faciliter et contrôler l'adaptation des applications et de leurs interfaces en se basant sur les caractéristiques des utilisateurs lors de l'interaction. Celle ci doit en outre être évolutive en ce sens qu'elle doit s'adapter à l'évolution des caractéristiques de ces mêmes utilisateurs.

3.3.1.2 Les Recommandations Ergonomiques

On trouve dans la littérature diverses interprétations de la notion de recommandations concernant la conception et/ou l'évaluation d'interfaces, qu'on peut répartir en deux catégories. La première regroupe les recommandations issues d'une analyse de l'activité et qui se rapportent à une application spécifique (§3.2), en ce sens qu'elles constituent les pré requis (requirements) à prendre en compte lors de la conception de l'interface pour l'application en question. La seconde catégorie regroupe les recommandations qui constituent un recueil de connaissances en conception et/ou en évaluation d'interfaces utilisateur [e.g. Smith86 ; Mosier and Smith.86]. Et donc plus intéressantes du point de vue de la formalisation. En effet, une partie d'entre elles, a été à la base des travaux sur les environnements de construction d'interfaces (e.g. le guide de style Motif [OSF90]), et connaît depuis quelques années une extension continuelle (§3.4). Néanmoins, elles demeurent insuffisantes en ce sens qu'elles ne font pas référence aux connaissances de la tâche. Les recommandations orientées tâches sont à l'heure actuelle très générales et

---

[6] Nous donnons dans le paragraphe (§3.4.3), une architecture générique dont l'objectif est d'établir des liens entre la spécification et la génération d'interfaces. En effet, ces liens sont envisagés par le biais de concepts abstraits (e.g. état) ; tels que ceux définis dans le modèle Arch [Arch92], lesquels toutefois sont utilisés pour des objectifs différents (e.g. l'indépendance vis à vis des boîtes à outils).

[7] Cette base de connaissances lie de manière dynamique les connaissances concernant les utilisateurs avec celles de la tâche par le biais des objets, de l'application et de l'interface, utilisés : le contexte d'utilisation [Schweighardt90].

nécessitent plusieurs raffinements avant d'être opérationnelles. De fait, les concepteurs se voient souvent contraints de les utiliser de manière ad hoc.

Les perspectives actuelles [e.g. Lim et al.92 ; Vanderdonct et al.93] concernent en partie ces diverses questions liés étroitement aux problèmes de formalisation et d'utilisation de ces recommandations dans un processus de conception et/ou d'évaluation d'interfaces. Pour ce faire, nous préconisons l'idée qui consiste à caractériser et formaliser ces recommandations en fonction des attributs de la tâche afin d'aboutir à des règles et heuristiques de spécification d'interfaces ergonomiques (§5). Par rapport à la classification de Mosier and Smith, nous nous situons plutôt dans la catégorie des règles de conception qui portent à la différence des premières sur des concepts abstraits (e.g. structure des actions, procédures d'interaction), ceux de la tâche.

3.3.2 Les Modèles de l'Interaction Homme-Ordinateur

Les théories et modèles de l'interaction s'intéressent à l'étude et à l'organisation des connaissances et mécanismes cognitifs permettant d'expliquer le phénomène de l'interaction homme-ordinateur [e.g. Green et al.88 ; Hoppe91], en particulier du point de vue de l'utilisateur. En effet, ces modèles servent de support à la définition, entre autres, d'approches quantitatives (e.g. [Bovair et al.90]) et/ou qualitatives (e.g. [Mc Donald et al.86]) pour la simulation et l'évaluation des interfaces utilisateur. Nous décrivons et discutons ici les concepts à la base de ces théories et modèles en indiquant leurs apports et leurs limites d'un point de vue ergonomique.

3.3.2.1 Le Modèle du Processeur Humain

Un des modèles représentatifs de cette catégorie est le modèle du Processeur Humain [Card et al.83]. L'objectif de ce modèle est de représenter l'individu comme un système de traitement de l'information, de définir les règles qui régissent ce système et de formaliser les procédures de son interaction avec l'ordinateur. Plusieurs études, à l'instar de [Kieras et al. 85 ; Tanaka et al.90] s'y sont en effet inspirées afin de prédire et quantifier la complexité d'apprentissage, valider la cohérence cognitive des affichages d'une interface utilisateur, etc. Cependant, ces paramètres permettent uniquement d'évaluer des phénomènes et des aspects de bas niveau. De plus, selon la terminologie de [Payne et al.89], ce modèle et ses dérivés (e.g. GOMS[8]) s'inscrivent dans l'espace

[8] GOMS (Goals, Operators, Methods, Selection Rules) modélise la connaissance que doit avoir l'utilisateur pour exécuter sa tâche sur un système. Il inclut la description des méthodes nécessaires à l'accomplissement de chaque objectif spécifique. Les méthodes consistent en une suite d'opérations (e.g. action motrice, cognitive) utilisateur. Lorsque plusieurs méthodes sont possibles pour réaliser un but, GOMS utilise des règles de sélection pour choisir la méthode appropriée au contexte.

cognitif des tâches dépendant des systèmes cibles (device-dependent tasks), et donc utilisables seulement en évaluation et/ou en prédiction.

Toutefois, dans [John et al.90], GOMS a été étendu à l'étude d'un domaine de tâches hautement interactives. Deux classes de GOMS : function-level operators et keystroke-level operators ont été implémentées selon l'architecture SOAR[9] afin de pouvoir effectuer des prédictions en temps réel. Ces prédictions concernent l'apprentissage, la détection et correction des erreurs, etc. Cette étude a permis d'effectuer des prédictions de performance pour des tâches complexes telles que les jeux vidéos. Cependant, elle ne s'intéresse qu'à l'évaluation des aspects quantitatifs de l'interaction.

3.3.2.2 Le Modèle de l'Action

Les travaux de [Norman86] sur la modélisation de l'action se rapprochent du modèle général de résolution de problèmes [Newell et al.72] ; lequel traduit la résolution par la réduction des différences entre les états, initial et final. En effet, Norman indique que pour réaliser sa tâche, l'utilisateur doit parcourir sept états chacun correspondant à une activité. La réduction des différences dans ce cas concerne les différentes interprétations nécessaires pour passer d'un état à un autre.

Cette théorie offre un cadre de pensée utile à la mise en évidence des problèmes clés de l'interaction homme-machine, et par conséquent constitue une aide à la conceptualisation des problèmes de la conception d'interfaces (e.g. la manipulation directe). Malheureusement, elle ne donne pas lieu à une méthode applicable, du fait de l'absence d'un support formel. Il serait en outre intéressant d'étudier la correspondance éventuelle entre les variables psychologiques (liées à la tâche) et physiques (liées au système) par l'intermédiaire de règles mettant en rapport les deux catégories de variables, i.e. l'état perçu (par rapport à la tâche) et l'état effectif (par rapport à l'interface). Ce point de vue est proche de la perspective de traduction de la tâche en spécifications d'interfaces.

*3.3.2.3 Le Modèle ACT**

Le modèle ACT* [Anderson83], s'inscrit dans la cadre de l'étude et la simulation du raisonnement. Il est à l'origine de programmes d'apprentissage lors de la résolution de problèmes. Le modèle procède par la construction d'un plan de résolution puis par son exécution. Il permet ainsi de résoudre des problèmes à un niveau général par une

---

[9] SOAR est une tentative pour fournir une architecture cognitive. Les tâches sont formulées en termes d'espaces de problèmes dans lesquels les opérateurs sont sélectionnés et appliqués aux états courants pour atteindre les états désirés [Laird et al. 87].

décomposition hiérarchique en sous problèmes, et ce jusqu'aux actions spécifiques. Ce concept constitue un support formel pour la résolution de problèmes complexes (e.g. la démonstration de théorèmes) [Nilsson82].

Cette modélisation ne s'attache pas particulièrement à la conception d'interfaces. Néanmoins, on retrouve la notion d'attachement procédural au plus haut niveau d'une structure hiérarchique dans [Pierret et al.89] dans le but de spécifier des procédures abstraites sous-jacentes aux actions élémentaires de la tâche (§4.2). En outre, dans le domaine de l'évaluation d'interfaces, on retrouve dans [Poitrenaud et al.90] le concept de la procéduralisation des actions relatives aux tâches, utilisé cette fois pour évaluer le rôle de la dynamique et de la sémantique des procédures d'un dispositif d'interaction par rapport aux caractéristiques des utilisateurs (e.g. connaissances du domaine) et leurs objectifs, et catégoriser les actions mises en oeuvre par rapport aux objets sur lesquels elles portent.

3.3.3 Discussion

Les différents résultats, modèles et théories concernant l'interaction homme-machine s'accordent tous sur le fait qu'il est nécessaire, pour une meilleure spécification de l'interface, de comprendre et d'analyser les caractéristiques des utilisateurs lors de l'accomplissement de leurs tâches. Cependant, ils ne fournissent pas de méthode et/ou de techniques explicites (e.g. les heuristiques) permettant la conception de l'interface selon ces divers pré requis. Toutefois, le concept de structuration hiérarchique de la tâche est à la base de la plupart des modèles. En effet, pour des tâches simples et structurées (e.g. traitement de texte), les modèles procéduraux ou syntaxiques (e.g. [Beringer et al.91]) permettent de prédire et d'évaluer la complexité cognitive de la tâche, etc.

Cependant, le problème de l'interaction reste entièrement posé au niveau sémantique, i.e. fonctionnel, en particulier selon la perspective de conception. A ce niveau, les connaissances nécessaires à l'interface sont analytiques, c'est-à-dire vont au delà des outils utilisés (device-independent knowledge), et par conséquent d'un niveau conceptuel. De fait, la prise en compte de la tâche, de ce point de vue, a pour objectif d'établir des spécifications d'une part *sémantiques* et d'autre part *procédurales*. C'est dans cette perspective que s'inscrit l'approche de spécification de l'interface conceptuelle que nous décrivons dans le dernier chapitre.

Par ailleurs, les outils de construction d'interfaces, comme nous le verrons dans le paragraphe suivant, ne prennent en compte que l'aspect composantes de l'interface (e.g. les objets d'interaction, leurs relations), l'aspect de l'interaction (transaction) utilisateur (e.g. la structure et les objectifs de l'interaction) ne peut être établi qu'à partir des

connaissances impliquant l'utilisateur, en particulier sa tâche. Par conséquent, et en accord avec [Johnson et al.90b], la conception d'interfaces utilisateur nécessite l'intégration de deux points de vue, i.e. de deux classes de modèles : ceux des architectures de construction d'interfaces (§3.4) et les modèles de tâches (§4) utilisateur. Les premiers traitent l'aspect support de développement, les seconds celui de méthode. Cette perspective sera sans doute à la base des nouvelles approches de l'interaction homme-machine.

3.4 Les Environnements et Outils de Construction d'Interfaces

L'objectif dans ce paragraphe n'est pas de faire un état de l'art complet du domaine des outils et environnements de construction des interfaces. Pour cela, on trouve dans [Hartson et al.89], [Coutaz90], [Myers89], [Nanard90], [El-Mrabet91] et [Beaudoin-Lafon91] des études discutant la situation actuelle et les perspectives des outils de l'interaction homme-machine. Néanmoins, nous présentons ici les concepts à la base de ces outils de façon à voir leurs apports et limites quant à leur utilisation pour la spécification d'interfaces d'un point de vue de la tâche utilisateur.
Trois principales catégories permettent de classer et d'étudier l'évolution des différents outils et environnements de l'ingénierie des interfaces : les boîtes à outils, les squelettes d'application et les générateurs d'interfaces.

3.4.1 Les Boîtes à Outils

Les boîtes à outils consistent en une collection de composants logiciels réutilisables mise à la disposition du programmeur des interfaces. Les fonctionnalités des boîtes à outils correspondent à deux niveaux d'abstraction distincts : les primitives de gestion de la station (e.g. fenêtrage, événements) et les primitives de gestion de dialogues (e.g. menus, boutons, boîte de dialogue). Selon les boîtes à outils, ces primitives peuvent être regroupées en une bibliothèque (e.g. la boite à outils MacApp [Schmucker86]) ou bien réparties dans des bibliothèques séparées (e.g. la Xlib : niveau gestion du poste de travail, et la Xt : niveau gestion du dialogue dans X-windows [Scheifler et al.86]).

Ces aspects ont un effet sur la portabilité des boîtes à outils. En effet, les boîtes à outils visent de plus en plus l'indépendance vis à vis du matériel (e.g. l'évolution des versions de X-windows). De par leurs définitions des primitives de bas niveaux, les boîtes à outils resteront le support de base pour le développement des interfaces. Elles présentent l'avantage d'extensibilité telle que la possibilité d'intégrer les attributs ergonomiques dans les paramètres des primitives par le programmeur. Cette possibilité présente toutefois deux inconvénients : d'une part, elle ne permet d'intégrer que les attributs ergonomiques de bas niveaux (e.g. position des champs), d'autre part, elle risque d'être fatale sur le plan ergonomique de l'interface puisque le programmeur

procède en utilisant son bon sens qui n'est pas forcément conforme aux recommandations ergonomiques.

3.4.2 Les Squelettes d'Application

Les squelettes d'application sont venus suite à la réutilisation d'une bonne partie du code, des boîtes à outils, pour plusieurs applications. Cette réutilisation concerne particulièrement la structure de contrôle (e.g. la gestion des événements). En effet, un squelette d'application abstrait la structure de contrôle réutilisable par les applications. Contrairement aux boîtes à outils, les squelettes d'application imposent une architecture prédéfinie partageable par plusieurs applications. Cette architecture regroupe un module de contrôle : le noyau d'exécution, et un module de présentation : les services et techniques d'interaction.

C'est à partir de cette architecture que le programmeur raffine quelques parties et/ou définit d'autres pour une application particulière. Cela est en effet facilité par les mécanismes de surcharge et d'extension de la programmation objet. D'autre part, les squelettes d'application introduisent la notion d'échange sémantique entre l'interface et l'application proprement dite, notion absente au niveau des boites à outils. Cet échange permet d'une part, de bien séparer le côté présentation de celui du dialogue, et d'autre part, le choix du niveau d'indépendance (protocole de communication) entre les abstractions et les représentations. APEX [Coutaz90], MacApp [Schmucker86] et DIOR [Nanard90] sont des exemples de squelettes d'application. Le premier, organisé selon le modèle PAC, ainsi que le deuxième ont été développés au dessus de la boîte à outils du Macintosh. Le troisième a été développé afin de permettre le développement d'applications graphiques sur micro-ordinateurs ne disposant à l'époque que d'un système d'exploitation primitif.

Quant à la prise en compte des facteurs humains, les squelettes d'application accusent un manque d'intégration des services ergonomiques tels que les aides, la gestion des erreurs, etc. En outre, malgré la structure prédéfinie qu'offrent les squelettes d'application permettant ainsi la réduction du temps de développement et la possibilité d'introduire des liens sémantiques entre l'interface et l'application, le problème de méthode de conception d'interfaces dont font défaut les boîtes à outils reste toujours posé. Nous n'entendons pas par méthode, l'aspect architectural de l'interface auquel répondent assez bien les squelettes d'application, mais plutôt les étapes et les contraintes que doit respecter le concepteur afin de modéliser une interface qui réponde aux objectifs des utilisateurs et à leurs caractéristiques.

3.4.3 Les Générateurs d'Interfaces

3.4.3.1 Situation Actuelle

Un générateur d'interfaces - UIMS pour User Interface Managment System - consiste en un outil ou un ensemble d'outils de haut niveau intégrés, extensibles, réutilisables, etc, permettant de concevoir, de prototyper, d'exécuter, d'évaluer et de maintenir les interfaces utilisateurs [Hartson et al.89]. La vocation des générateurs d'interfaces est qu'ils permettent de développer des systèmes interactifs en s'appuyant sur le principe de séparation de l'interface et du noyau fonctionnel de l'application. En effet, suite à la spécification de l'interface au moyen de formalismes tels que les grammaires, les langages spécifiques ou la manipulation directe [Myers89], le système génère les différents composants de l'interface, ainsi que leurs liens, qui sont souvent assimilés à ceux du modèle de Seeheim (Figure 1). Notons cependant que les langages de la plupart de ces outils[10] sont plutôt orientés vers des programmeurs que vers des concepteurs d'interfaces non spécialistes en informatique.

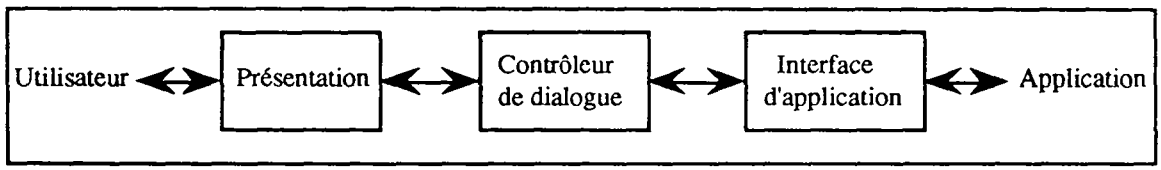

Figure 1 : Le Modèle de Seeheim.

La présentation constitue la partie visible par l'utilisateur lui permettant d'effectuer sa tâche. C'est à ce niveau que sont gérés l'affichage et les événements d'entrées-sorties. Le contrôleur de dialogue s'occupe de la gestion du dialogue, c'est-à-dire les liaisons qui existent entre les événements de l'utilisateur et leur traitement par l'application. L'interface d'application correspond au protocole de communication entre le noyau fonctionnel de l'application et le contrôleur de dialogue. Cette architecture est toujours rapprochée de la métaphore linguistique de l'interaction qui structure l'interface en quatre niveaux : lexical, syntaxique, sémantique et conceptuel [Foley et al.84]. Le niveau lexical décrit le comportement de bas niveaux des éléments ("tokens") de l'interface. Le niveau syntaxique correspond à la description des séquences d'entrées-

[10] Plusieurs produits existent déjà dans le marché ou sont en phase d'expérimentation. Masai [Aida/Masai92], Graffiti [Karsenty91], Peridot [Myers91], ITS [Wiecha et al.89] sont des exemples de générateurs d'interfaces basés sur un langage de spécification graphique. Ces générateurs sont composés d'un éditeur à manipulation directe et d'une bibliothèque de composants graphiques permettant ainsi le prototypage rapide. Toutefois, des outils tels que ITS (Interactive Transaction Systems) et JADE [Vander Zanden et al.90] introduisent en plus un niveau de séparation entre le dialogue (de l'application) et le style de dialogue (les règles de styles) de façon à constituer une base de règles réutilisable pour plusieurs applications. Notons cependant que ces règles ne concernent que les aspects de plus bas niveau tels que les fontes, les styles de lignes, etc.

sorties. Le niveau sémantique définit les liens entre les séquences d'entrées-sorties du niveau syntaxique avec les entités de l'application. Quant au niveau conceptuel, il correspond aux objets et opérations du noyau fonctionnel de l'application.

Du point de vue des facteurs humains [Ten Hagen91], par exemple tel qu'il se manifeste dans les guides de conception et d'évaluation, la partie présentation concerne l'ensemble des symboles, des objets et concepts de l'application, leurs moyens d'accès ainsi que les effets de leurs interactions. Le contrôle de dialogue concerne les commandes et la structure de dialogue par rapport au modèle qu'a l'utilisateur de l'application. L'interface d'application a pour vocation d'établir une correspondance sémantique entre la tâche que l'utilisateur veut exécuter et les opérations de l'application. Toutefois, ce point de vue semble plus descriptif que prescriptif, et de fait plutôt approprié à une approche d'évaluation, facilitant ainsi l'évaluation d'une application pour une re-conception éventuelle. En outre, on voit bien à travers ces différentes approches deux perspectives différentes de modélisation de l'interaction. La première (e.g. modèles d'architecture) se focalise sur la formalisation des échanges d'informations au sein d'un même modèle (cohérence interne), i.e. entre les différents modules de l'architecture. La seconde, par contre, insiste sur l'échange d'informations entre deux modèles distincts qui sont l'utilisateur et l'application (cohérence externe), i.e. les rôles et effets de chaque composant de Seeheim par rapport aux aspects utilisateur.

3.4.3.2 Perspectives

L'architecture linéaire de Seeheim, en ce sens que l'échange entre la présentation et l'interface d'application passe par le contrôleur de dialogue, constitue un des majeurs défauts de ce modèle et a montré ses limites : le feed-back sémantique, le dialogue multifils ; par exemple pour les applications à manipulation directe où l'interface devrait prendre en charge une part de la sémantique et réagir d'une façon très interactive aux événements des périphériques d'entrée [Hudson87] (e.g. déplacement des icônes dans la corbeille du Macintosh).

D'autre part, la description en couches (composants) d'une application interactive semble plutôt fonctionnelle que structurelle. En réalité la "séparation" n'est pas aussi évidente que cela semble paraître à travers l'architecture vu l'interdépendance intra-couches et inter-couches, liée au contexte de l'application, qu'il y a entre les différents éléments et concepts de chacune de ces couches lors de la conception. En effet, les paramètres (e.g. dénomination, position, fonctionnalités) d'un objet ou groupe d'objets en entrée ou en sortie sont déterminants aussi bien aux niveaux de la présentation et du dialogue qu'au niveau de la sémantique de la tâche, i.e. atteinte d'un but ou sous-but.

En outre, [Hartson89 ; Shevlin et al.91] étudient en profondeur la notion de séparation au niveau de la génération actuelle des UIMS et constatent que le goulot d'étranglement de cette séparation réside au niveau sémantique [Hurley et al.89], i.e. au niveau de la conjonction du code de l'interface avec le code source de l'application, facilitant les tests fonctionnels de l'application. En effet, les deux premiers composants du modèle de Seeheim semblent actuellement bien définis et maîtrisés (e.g. les automates) dans les UIMS et que le composant (principal) de cette séparation "l'interface d'application" fait encore appel à un effort de programmation du fait du nombre très limité de suggestions (e.g. les protocoles de communication) pour la spécification de ce composant.

Shevlin et al. soulignent de fait qu'il n'est pas nécessaire d'implémenter à un niveau de séparation physique (e.g. les gestionnaires de fenêtres) et préconisent deux niveaux de séparation : *le niveau logique* qui correspond au niveau spécification, et *le niveau virtuel* permettant à l'UIMS de générer un système intégré au niveau physique à partir des spécifications. Pour ce faire, ils proposent une architecture avec en plus un service graphique permettant au développeur de l'interface de résoudre interactivement les questions relatives à l'interface d'application, i.e. à la sémantique de l'application. Il s'agit en effet d'une intégration de la connaissance (habilité) du développeur au pouvoir syntaxique des UIMS.

L'architecture proposée par [Coutaz91], la méthode élaborée par [De Baar et al.92], et l'approche de [Cournarie91] s'inscrivent dans le même cadre de réflexion par l'introduction de concepts tels que la délégation récursive de la sémantique, les règles mettant en rapport le modèle de données et les éléments de contrôle et de présentation d'un style d'interface, et les contraintes et prototypes définissant de manière déclarative la sémantique des relations entre les objets de présentation et ceux de l'application.

3.4.3.3 Spécification Ergonomique et Génération d'Interfaces : Une Approche

Dans cette perspective, nous schématisons dans la figure 2 notre approche concernant la mise en correspondance entre la spécification (le niveau sémantique) et la génération d'interfaces (le niveau syntaxique). Cela, à partir d'une base de tâches structurées et d'heuristiques ergonomiques. En effet, ces liens (de passage) sont envisagés par le biais de concepts abstraits tels que objets conceptuels d'interaction (e.g. état, fonctionnalité). En outre, la figure explicite les deux pré requis, à prendre en compte de façon formelle pour la spécification d'interfaces orientées tâches, qui sont : les objectifs de la tâche et les connaissances ergonomiques d'un niveau conceptuel ; lesquels constituent, à l'heure actuelle, les limites des modèles et/ou outils de conception d'interfaces utilisateur, discutés dans ce chapitre. De plus, notre proposition se distingue des propositions décrites plus haut en ce sens que ces dernières abordent le problème d'un point de vue architectural, alors que notre approche insiste sur des aspects méthodologiques.

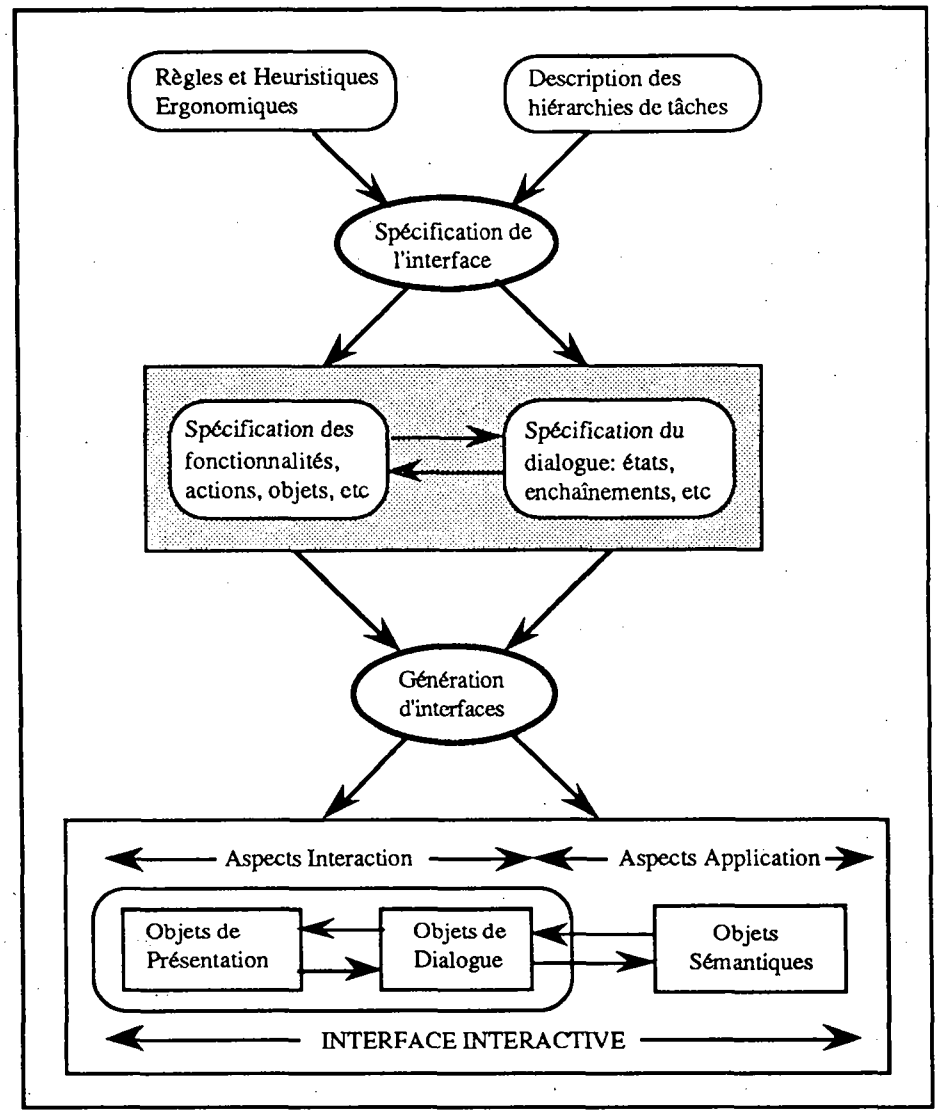

Figure 2 : Spécification et génération d'une interface ergonomique[11].

Nous estimons en effet que cela constitue l'un des besoins les plus importants actuellement où, comme nous l'avons souligné précédemment, malgré la bonne avancée enregistrée au niveau des outils de spécification et de génération d'interfaces, le problème de méthode de spécification d'application utilisateur reste toujours ouvert. Cette perspective demeure néanmoins le fossé à combler si l'on veut concevoir des applications utiles et utilisables. Elle est souvent abordée de manières implicite et/ou partielle ; cela est dû au fait qu'elle nécessite la mise en oeuvre de plusieurs catégories de connaissances, souvent indépendantes, telles que les connaissances cognitives,

11 La figure illustre en effet les deux prérequis nécessaires à la conception d'interfaces ergonomiques. Il s'agit ici d'intégrer les aspects sémantiques au pouvoir syntaxique des générateur d'interfaces. La sémantique ici provient la tâche utilisateur et concerne de fait les objets des différents composants de Seeheim. En particulier, le dialogue et la sémantique de l'application. En outre, on parle aussi de sémantique, mais selon la perspective d'implémentation (cohérence interne vs cohérence externe), décrite plus haut, lorsqu'il s'agit d'établir des liens entre l'interface et l'application (e.g. modélisation de l'interface d'application [Hurley et al.89]). Ces deux points de vue semblent néanmoins complémentaires.

ergonomiques, informatiques, etc. Chacune de ces approches aborde la question d'un angle plus ou moins formel apportant un savoir nécessaire mais non suffisant (Cf §3.6). Ceci conforte l'idée, que nous soutenons dans ce rapport, d'une approche "mixte", c'est-à-dire qui s'appuie sur les acquis et avantages des différents modèles et outils existants, qui consiste à fournir une *structure intermédiaire* permettant d'intégrer les différentes catégories de connaissances intervenant dans le processus de spécification et de conception d'interfaces ergonomiques.

L'objectif ultime, comme le montre la figure, est de greffer un outil support d'une méthode ayant pour entrées : une description de la tâche utilisateur, et pour sorties : les premières spécifications de l'interface en termes de fonctionnalités, d'états, de structure de dialogue, etc ; et ce par le biais de plusieurs catégories d'heuristiques ergonomiques structurées et utilisées selon plusieurs étapes de spécification. Ces spécifications concernent les aspects sémantiques (e.g. les états de l'interface) et les aspects procéduraux (e.g. les actions) de l'interaction. Ce niveau est représenté par la surface grisée de la figure.

La figure offre un cadre schématique de l'objectif de générer une interface telle qu'une maquette permettant de simuler le dialogue d'un point de vue de la tâche utilisateur. En outre, l'idée est de permettre l'utilisation d'un outil quelconque de génération à partir des spécifications. Cet aspect se rapproche des boîtes à outils abstraites (virtuelles) [Arch92]. En effet, encadrés en bas de la figure, sont représentés les objets établis suite à la phase de génération. Nous distinguons trois classes d'objets : les objets de présentation, les objets de dialogue et les objets sémantiques. Cela, afin de nous situer par rapport au cadre logique des modèles d'architecture. Toutefois, d'un point de vue utilisateur, la distinction entre les objets de présentation et les objets sémantiques est floue. Ce sont les objets sémantiques, i.e. de l'application, que l'utilisateur manipule pour réaliser sa tâche, et ce par le biais d'objets de présentation qui constituent l'interface physique entre l'utilisateur et l'application. On voit bien qu'à ce niveau de spécification la séparation entre présentation et sémantique n'est non seulement pas évidente, mais est aussi surannée d'un point de vue de l'utilisation.

De fait, nous adoptons une notion "virtuelle" de la séparation, c'est-à-dire en termes de *référent* (objets graphiques) et de *symbole* (objets de la tâche). Par exemple, une spécification peut concerner le fait qu'une action utilisateur puisse agir sur un objet de présentation sans qu'elle enclenche les mécanismes de mise à jour des données sémantiques associées, ou inversement. Toutefois, ces considérations plutôt logicielles [Hartson89] sont d'un intérêt utilisateur si elles facilitent la réalisation de sa tâche (e.g. temps de réponse). En outre, le cas qui nous intéresse ici consiste en la définition des aspects conceptuels (symboles) et sémantiques (liens) de l'interface en fonction des objectifs de la tâche. Le dialogue, le niveau intermédiaire, reflète la dynamique des buts

et sous buts c'est-à-dire les enchaînements, les états, la gestion de l'aide, des erreurs, etc. Nous avons ainsi modélisé ces divers aspects par des concepts abstraits dépendants de la sémantique de la tâche (Cf §5), tels que le concept d'objet fonctionnel, d'état de l'interface, d'interacteur, etc.

3.4.4 Discussion

A l'heure actuelle, les générateurs d'interfaces génèrent le code exécutable de l'aspect présentation de l'interface ainsi que le dialogue spécifique à l'application qui se charge de la gestion des échanges entre la présentation et l'application. Lorsque l'interface est construite par l'intermédiaire d'un interpréteur accompagné d'un éditeur graphique (e.g. SOS Interface [Hullot86]), le concepteur a ainsi la possibilité d'évaluer les effets des spécifications au fur et à mesure de la construction de l'interface. On parle alors d'outils interactifs de spécification ; ce qui s'apprête bien aux approches de maquettage et/ou de prototypage d'interfaces. Cependant, la plupart de ces outils ne gèrent que les deux premiers composants (e.g. les automates) de Seeheim en ce sens que les modèles d'architecture favorisent la spécification syntaxique des interfaces plutôt que la spécification sémantique.

La spécification syntaxique répond en effet à la question "comment représenter telle opération" alors que la spécification sémantique répond plutôt à la question "pourquoi telle opération et quelles sont ses effets avant et après son utilisation". Il s'agit dans la spécification syntaxique d'établir un lien entre la présentation et l'application en termes d'indirections et de transferts des tokens entre les objets des différents composants (cohérence interne) sans pour autant intégrer explicitement les aspects de la tâche à la conception. En effet, la définition des concepts, leurs rôles et leurs relations ne provient pas d'une analyse de la tâche mais de la représentation qu'a le concepteur de l'utilisation. En revanche, la spécification sémantique d'un point de vue de la tâche utilisateur consiste à établir et à expliciter les objets, les actions ainsi que les procédures d'utilisation (cohérence externe).

En somme, le progrès des outils de construction d'interfaces a permis de franchir un grand pas sur le plan de la technologie des interfaces, i.e. quant à la réalisation de l'interface, du fait de la modularité, la réutilisabilité et de l'extensibilité de l'interface générée. En outre, concernant la tâche de spécification de l'interface, du point de vue de l'ingénierie, les tendances actuelles consistent à produire de manière automatique l'interface à partir de la description des structures de données de l'application (§3.5). Il s'agit en général d'une association entre les structures de données de l'application et des objets de présentation de l'interface. Cette perspective de spécification est pour les générateurs d'interfaces ce qu'est le modèle conceptuel des données pour les systèmes de gestion de bases de données. De fait, il s'agit bien là de spécification d'interfaces d'un

point de vue *implémentation* qui se distingue des spécifications du point de vue *cognitif* [e.g. Tauber90], aspects que nous explicitons plus haut en termes de cohérence interne vs cohérence externe.

3.5 LES OUTILS DE SPÉCIFICATION D'INTERFACES

En effet, la perspective logicielle qui concernait la difficulté de réalisation de l'interface, due entre autres au fait qu'elle concerne plus de 60% du code de l'application auquel elle est fortement intégrée (contrôle interne vs contrôle externe), a constitué le centre d'intérêts de la plupart des travaux de l'ingénierie des interfaces. C'est à cette perspective que répondent, à l'heure actuelle, assez bien les outils de construction d'interfaces. L'autre perspective, celle de la prise en compte de l'utilisateur, et par conséquent des facteurs humains, absente dans les méthodes classiques de conception, demeure toujours mal ou partiellement abordée. En outre, des outils comme MacIDA [Petoud90], UIDE [Foley et al.88 ; 91], SIROCO [Normand92] permettent la génération automatique des interfaces à partir de spécifications sémantiques de l'application, i.e. incluant le modèle de données. Il s'agit des spécifications fonctionnelles du modèle Entité-Relation dans MacIDA (§3.2.1), des concepts du domaine dans SIROCO, et de bases de connaissances dans UIDE.

3.5.1 Le Modèle SIROCO

SIROCO est un langage de spécification conceptuelle d'une interface doublé d'un générateur de code réalisant l'interface. Le langage de spécification repose sur un modèle de représentation conceptuelle de l'interface orientée objet distinguant la dimension de fonctionnement et la dimension d'utilisation, c'est-à-dire l'application et l'interface. En effet, la dimension de fonctionnement regroupe l'ensemble des objets et opérations *(concepts du domaine)* de l'application. La dimension d'utilisation qui correspond à la notion de vue *(perspective)* et reflète l'organisation de l'interface *(l'espace de travail)* par rapport aux caractéristiques de la tâche, i.e. l'enchaînement des tâches. Elle consiste en l'organisation des opérations et des données en modules utilitaires selon le contexte d'utilisation.

La structure d'une spécification SIROCO comporte ainsi trois parties : la première concerne les concepts du domaine, la deuxième celle de la perspective et la troisième la notion d'espace de travail. La notion de perspective définit une vue sur un concept du domaine, c'est-à-dire de l'application, permettant d'adapter l'interface selon le contexte de la tâche. Cette notion se rapproche de la composante "interface d'application" du modèle de Seeheim en ce sens qu'elle établit un protocole sémantique entre l'application et l'interface, incluant de fait des liens sémantiques avec la présentation [Green85]. La perspective qui formalise cet aspect, par l'introduction de la notion de vue, est

modélisée par les concepts "type et type-objet"[12]. Elle permet de définir le protocole d'accès à un objet depuis l'environnement extérieur, et permet ainsi de définir par exemple des restrictions d'utilisation sur les concepts du domaine.

3.5.2 l'Environnement UIDE

UIDE, pour User Interface Development Environment, permet de générer de manière automatique et incrémentale une interface à base de menus à partir de la description des fonctions attendues par l'application. L'architecture UIDE, centrée sur le concept de base de connaissances, exploite les notions de pré- et post-conditions associées aux actions afin d'assister le concepteur dans les stratégies (alternatives) de conception, d'évaluation et d'implémentation de l'interface finale.

Pour ce faire, la description conceptuelle inclut une hiérarchie de classes d'objets du domaine de l'application, les propriétés des objets, les actions associées aux objets, les unités d'informations requises par les actions ainsi que les pré- et post-conditions des actions. Ces différentes catégories, organisées sous forme de hiérarchies de classes d'objets à héritage simple, sont représentées comme des instances de sept schémas (frames) ART[13] qui sont : *Action Schema, Precondition Schema, Postcondition Schema, Parameter Schema, Object Schema, Attribute Schema et Attribute type Schema.* Ces différents schémas, décrits d'abord dans un métalangage de spécification : IDL, Interface Definition Language [Gibbs et al.86], répartissent l'interface en deux modèles : le modèle de données et le modèle de contrôle.

Le modèle de contrôle concerne les actions, les paramètres et les séquences utilisateur. Le modèle de données concerne les types, les propriétés et les relations entre les objets spécifiques à l'application tels que le mot, la phrase et le paragraphe dans une application de traitement de texte. Cependant, la distinction entre ces deux modèles n'est pas explicite au niveau des hiérarchies de classes, en particulier afin de permettre de modifier facilement la sémantique de l'application indépendamment de leurs styles d'interaction (e.g. post-fixé). En effet, les attributs caractérisant ces divers aspects sont répartis dans les différentes catégories lesquelles sont interdépendantes ; en particulier Object schema et Action schema. La première catégorie concerne les objets de l'application, la seconde ceux de l'interface.

La base de connaissances constituée de schémas a entre autres pour objectifs de vérifier la cohérence et la complétude des descriptions, de permettre la modification des objets

---

[12] Ces notions, proches de celle du polymorphisme dans les langages à objets, sont définies dans l'environnement objet Guide — support de développement de SIROCO.

[13] ART, pour Automated Reasoning Tool (cité dans Foley et al.91), est un modèle de représentation et de manipulation de connaissances à base de frames.

de présentation (les fonctionnalités restant équivalentes, via des algorithmes de transformations) et d'établir un lien entre les spécifications et le SIUMS, Simple UIMS qui implémente l'interface de l'utilisateur final, lequel sert de support au processus de conception. En effet, le concepteur peut créer pour une application une variété de fonctionnalités restant équivalentes à plusieurs interfaces utilisateur. Le SUIMS se charge alors de produire l'interface correspondante dès qu'une transformation est appliquée par le concepteur.

Les algorithmes de transformation [Foley et al.87] sont implémentés sous forme de stratégies génériques à base de règles. Celles ci permettent de modifier, de supprimer ou d'instancier les sept types de schémas. Les transformations concernent entre autres la factorisation qui consiste à regrouper les objets en fonction de leurs occurrences dans les actions ; et à spécialiser ou généraliser les commandes par rapport à la hiérarchie de l'ensemble des objets sélectionnés. Les pré- et post-conditions sont générées automatiquement, par les règles de transformation, et évaluées à l'exécution pour chaque action. Les pré-conditions, lorsqu'elles sont à vrai, déterminent les actions possibles, alors que les post-conditions effectuent la mise à jour des variables et le changement de contexte. Elles concernent ainsi la spécification de la sémantique de l'application permettant d'inférer la sensibilité (context sensitivity) dans la présentation des menus et de l'aide par rapport au contexte défini dans les post-conditions.

3.5.3 Discussion

De telles approches consistent à apporter une couche sémantique aux générateurs d'interfaces qui focalisent l'interaction sur la répartition du contrôle entre l'application et l'utilisateur. Elles représentent ainsi l'interface en deux modèles : le modèle de contrôle et le modèle de données. Le premier concerne la gestion du dialogue, le second gère la connaissance : l'association entre les objets de l'application et ceux de l'interface tels que les types d'informations nécessaires à une commande, les types des commandes sur les objets, les conditions de disponibilité de certaines commandes, etc. En effet, de tels outils ont pour objectif la générer de manière automatique l'interface à partir des descriptions sémantiques de l'application. Ils utilisent en entrée une description des fonctions de l'application et un ensemble de règles.

Celles ci s'inspirent des guides de style, celui d'OSF/Motif [OSF90], dans SIROCO et portent sur les éléments d'une "vue" (espace de travail) ; notion que l'on retrouve dans ITS [Wiecha et al.89] permettant de séparer le dialogue (les actions) du style de dialogue (vues). Cela permet de fait de définir de manière déclarative des règles de styles normalisées associant les fonctions de l'application et celles de la présentation (e.g. les boîtes à outils) et ainsi permettre la génération automatique de l'interface à partir des spécifications de l'application. Dans UIDE, il s'agit d'heuristiques de

transformations instanciant des règles d'interaction spécifiques aux contextes de l'application (e.g. spécialisation, post-conditions). Cependant, ces expertises sont d'une part souvent implémentées de manière procédurale et donc tout changement (e.g. style d'interaction) nécessite la mise à jour de toute la base de règles, et d'autre part spécifiques aux applications traitées.

En outre, une évaluation de l'interface, de type keystroke ou apprentissage s'avère nécessaire [Foley et al.91]. En effet, même au niveau de ces approches, le point de vue ergonomique n'intervient, quand c'est le cas, qu'a posteriori et de fait à des niveaux syntaxique et lexical. Cela du fait que ce sont les spécifications fonctionnelles de l'application qui déterminent les concepts du dialogue ; les aspects liés à l'utilisation demeurent cependant implicites en ce sens que ces considérations restent à la charge du concepteur. Ces approches abordent le problème de méthode selon la perspective des modèles d'architecture ; ce qui ne correspond pas forcément à la définition de l'interaction du point de vue utilisateur.

De fait, cette perspective est tout de même sujette à la question de la prise en compte des aspects cognitifs de l'interaction, i.e. la définition du dialogue par rapport à la structure de la tâche utilisateur en tenant compte des connaissances ergonomiques (§3.3) en rapport avec la spécification d'interfaces. Nous illustrons dans la figure 2 une réflexion dans ce sens en explicitant les entrées possibles (les données et/ou connaissances) permettant de définir des techniques voire des outils de spécification d'interfaces d'un point de vue de l'utilisateur, sachant que les données et modèles utilisées par les concepteurs d'applications sont régis par des méthodes et/ou outils formels tels que Merise, Sadt.

3.6 CONCLUSIONS ET PROPOSITIONS

A la revue des diverses approches, menées souvent les unes indépendamment des autres, on constate que la plupart des modèles cognitifs et/ou informatiques qui existent et les connaissances ergonomiques utilisées actuellement, expliquent, évaluent, de façon formelle ou informelle le résultat d'une conception d'interfaces. Cependant, il n'existe pas encore une démarche formelle à suivre pour la conception d'interfaces ergonomiques. Il s'agit en fait d'une re-conception, lorsque cela est possible, après une évaluation ergonomique a posteriori.

Les outils de construction et/ou de spécification d'interfaces, quant à eux, appréhendent les problèmes de conception d'un point de vue plutôt architectural et/ou logiciel (e.g. les modèles d'entrées-sorties) structurant l'interface en plusieurs couches (niveaux) allant du périphérique physique d'entrées-sorties (e.g. le clavier, la souris) aux concepts

abstraits de l'interaction (e.g. la métaphore du Desk-top), et/ou de spécification (e.g. le niveau sémantique dans UIDE).

En effet, l'interaction est en général abordée par des modèles formels (e.g. [Carey et al.87]) tels que les grammaires, les réseaux de transition, ou les événements [Green87] permettant la modélisation du dialogue. Cependant, ils présentent néanmoins l'inconvénient qui faisait défaut aux méthodes classiques, à savoir la non prise en compte des aspects utilisateur dans le processus de conception. La plupart de ces outils recouvrent bien le contrôle d'homogénéité syntaxique et sémantique entre les objets propres à l'interface et ceux de l'application [e.g. Hill et al.90 ; Lee90]. Malheureusement, ces aspects ne correspondent pas forcément à ceux (e.g. l'organisation, l'homogénéité) qu'a l'utilisateur de sa tâche pour laquelle souvent une analyse de l'activité s'avère nécessaire (e.g. l'activité de contrôle aérien) avant toute spécification de l'interface. La première perspective concerne en effet les aspects implémentation (e.g. les protocoles de conversion), le seconde en revanche concerne les aspects en rapport avec l'utilisation, i.e. les aspects cognitif.

Notre contribution intervient à ce niveau et nous espérons ainsi concevoir, en utilisant les connaissances ergonomiques et la structure de la tâche utilisateur, une approche formelle de spécification d'interfaces compatibles avec les objectifs des utilisateurs. Une première étape concernant l'organisation des données (e.g. les configurations de la tâche, les recommandations) s'impose avant tous processus de formalisation et de modélisation. En effet, afin de définir *l'utilisation* de l'application en fonction des objectifs utilisateur, nous avons établi une structure conceptuelle de la tâche de manière à appliquer des heuristiques mettant en rapport les attributs de la tâche et les spécifications d'interface ; lesquelles consistent à définir les concepts de l'interface et leurs comportements vis à vis de l'utilisateur. Ces différents aspects constituent le modèle sous-jacent à l'approche de spécification d'interfaces ergonomiques.

L'approche consiste en l'exploitation d'une "base de connaissances orientées tâches" de manière à identifier les objets pertinents à l'interaction utilisateur, les opérations qui leurs sont associées, ainsi que leurs relations avec les concepts de l'application (§5). Cela permet non seulement de définir une répartition du contrôle entre l'utilisateur et l'application, i.e. interactive, mais aussi d'établir des spécifications d'interface adaptées aux objectifs de la tâche utilisateur, i.e. ergonomique. Il s'agit là d'une approche, reprenant à la fois les avantages des modèles d'architecture comme support de développement, et les connaissances inhérentes à la tâche utilisateur, utiles à la conception d'interfaces, comme méthode de spécification. Par conséquent, l'approche tâche utilisateur apparaît comme un "intermédiaire" à la conception d'interfaces, c'est-à-dire à la frontière des deux perspectives de conception d'applications interactives : les méthodes à cycle de vie et les modèles de l'ingénierie ; ce que nous allons discuter dans

les chapitres suivants. Cette perspective constituera sans doute la nouvelle génération d'outils de conception et de développement d'interfaces utilisateur.

4 LES FORMALISMES DE TACHES ET L'INTERFACE UTILISATEUR

Dans ce chapitre, l'objectif n'est pas de faire une étude de la littérature sur les méthodes de recueil des tâches utilisateur, i.e. ayant pour but la modélisation de l'activité des opérateurs. Pour cela, le lecteur pourra se reporter à [Green et al.88], [Wilson et al.88], [Diaper et al.89], [Senach91] et [Benyon92]. Néanmoins, nous présentons ici les concepts à la base des travaux sur les formalismes de tâches en vue de la conception et/ou l'évaluation d'interfaces utilisateur de façon à voir leurs apports et limites d'un point de vue ergonomique aux plus hauts niveaux d'abstraction. Nous discutons d'abord les principales orientations actuelles mettant en rapport des modèles de tâches et la spécification d'interfaces, et ce du point de vue des concepts et connaissances utilisés, en particulier les connaissances ergonomiques. La deuxième partie concerne les perspectives de modélisation orientée tâche des points de vue de l'intelligence artificielle et de l'ergonomie. En particulier, nous discutons la modélisation procédurale (les connaissances dynamiques) par rapport aux objectifs de spécification d'interfaces.

4.1 MODÈLES DE TÂCHES ET SPÉCIFICATION D'INTERFACES UTILISATEUR

4.1.1 Le Modèle TKS (Task Knowledge Structure)

Ce paragraphe composé de deux sections résume les recherches de Johnson et al. qui portent sur les aspects tâches et interfaces — ce modèle est représentatif du domaine en ce sens que beaucoup d'autres approches entrent dans le même cadre de pensée à des variantes près (e.g. TAKD [Diaper90]). Il s'agit essentiellement de deux orientations : le recueil et la modélisation des tâches utilisateur afin de les intégrer dans les méthodes de conception et/ou de définir des modèles intermédiaires en vue de la spécification d'interfaces au sens des modèles d'architecture.

4.1.1.1 La Théorie TKS et le Recueil de la Connaissance

Le modèle TKS [Johnson et al.88 ; 90a ; 91 ; Johnson91] est une approche théorique et méthodologique pour la représentation des tâches utilisateur. La tâche est modélisée sous forme d'une mémoire particulière décrite par la structure TKS. Elle consiste en une représentation des différents types de connaissances acquises et utilisées lors de l'accomplissement d'une tâche. Une structure TKS est reliée à d'autres structures TKSs par le nombre de relations possibles en termes de liens fonctionnels ou de points de vue. On distingue deux types de relations : les Relations Intra-Rôles (Within Role), et les

Relations Inter-Rôles (Between Role). Les premières définissent un lien entre les TKSs en termes de leurs associations à un rôle donné ; par exemple une personne peut avoir plusieurs rôles : auteur, enseignant, etc. Les secondes concernent les liens de similarité entre les tâches à travers différents rôles ; ceci se présente lorsque plusieurs personnes participent à une même tâche. Ainsi les TKSs associées à chaque sous-tâche d'une tâche devraient avoir un lien fonctionnel avec les autres TKSs de la même tâche mais exécutée sous un autre rôle[14].

En effet, Les auteurs soutiennent l'idée que le déroulement d'une tâche ne peut s'effectuer indépendamment des autres, et que les structures des connaissances de la tâche sont fonctionnellement équivalentes aux structures des connaissances traitées et utilisées par les opérateurs lors de l'exécution de la tâche. Une méthode[15] comportant plusieurs techniques d'analyse de recueil des données (e.g. observation directe, construction des listes d'actions, d'objets, de procédures) a été développée afin d'identifier les différentes connaissances, tels que les rôles et les buts, formant les structures TKS simples, et construire les propriétés génériques, c'est-à-dire des structures TKS génériques, communes à plusieurs tâches simples, donnant lieu à un modèle de tâches abstrait indépendant du contexte d'utilisation (e.g. les outils utilisés).

Enfin, le modèle de TKS complet produit à partir de l'analyse comprend les structures de connaissances associées à une collection de tâches reliées par des rôles. Le plus haut niveau concerne les relations tâches/rôles vues précédemment. Le deuxième niveau de représentation concerne le contenu des TKS d'une tâche donnée. Ces représentations incluent des liens de contrôle sur les éléments des niveaux inférieurs : le plan, les stratégies, les procédures, ainsi que les structures d'actions et d'objets associées. Le plus bas niveau du modèle représente les catégories objets et leurs actions associées. On trouvera dans [Johnson et al.88] un exemple détaillé d'une structure TKS. Il s'agit là d'une modélisation à structure de contrôle hiérarchique. Cependant, il n'est pas clair si cette structure est procédurale et/ou fonctionnelle (les relations) ; ce qui n'est pas sans effet par rapport à la spécification d'interfaces. La première a pour vocation de modéliser les représentations mentales, la seconde tente plutôt d'intégrer les aspects utilisateur (e.g. le contrôle) dans les méthodes de conception (Cf §3).

---

[14] Ces deux types de relations sont représentées par deux attributs dans chaque TKS, laquelle renferment les connaissances suivantes : la *structure de buts* et sous-buts ; le *plan* qui représente l'exécution complète d'un but ; la *structure de procédures* laquelle contient les procédures alternatives pour exécuter un sous-but particulier ; les conditions d'exécution d'un but ou d'un groupe de procédures, i.e. les *stratégies*. Enfin, les *objets et actions*, représentés selon une structure taxonomique, associés aux procédures.

[15] Cette méthodologie est connue sous le nom de KAT pour Knowledge Analysis of Tasks [Johnson et al.89b]. Un prototype de simulation est décrit dans [Johnson et al.90a].

En outre, malgré les objectifs de complétude et de détail dans la description en vue de définir entre autres la répartition tâches/rôles, cette approche semble de fait orientée vers la modélisation des connaissances expertes (user knowledge model), c'est-à-dire la description des tâches en termes de points de vue utilisateur particuliers (les rôles), et par conséquent constitue des connaissances génériques qui s'instancient en fonction d'autres paramètres tels que le contexte, le type d'opérateurs, etc. A partir de ces descriptions, un modèle intermédiaire est construit ; celui ci servant de support à la conception d'un nouvel environnement utilisateur. Il s'agit d'identifier à partir des divers niveaux du modèle TKS les informations utiles à la spécification des fonctionnalités : le plan, les procédures, les actions nécessaires à l'exécution d'une tâche donnée, etc. Ce que nous allons décrire ci après.

4.1.1.2 Du Modèle de la Tâche à un Modèle de Conception

Ce modèle intermédiaire consiste en une approche de décomposition itérative, basée sur la représentation en frames, du modèle de la tâche en un modèle de conception. Cette décomposition comprend trois niveaux. Le premier niveau concerne la transformation du modèle TKS en un modèle général de tâches (GTM) qui représente en une série de frames les rôles, les buts, les actions, les stratégies, ainsi que les définitions d'objets. Le second niveau transforme le GTM en un modèle spécifique de tâches qui le complète par des informations concernant la répartition des tâches entre l'utilisateur et le système, cela en termes d'actions interface et des objets sur lesquels elles portent. La dernière étape permet le passage du STM à un modèle spécifique de l'interface (SIM)[16] qui décrit la structure et le contenu de l'interface utilisateur aux niveaux conceptuel et sémantique (e.g. [Johnson et al.92]). A ce niveau de description, il est clair que l'interface est indépendante des caractéristiques de la machine hôte. Ainsi, d'autres niveaux de décomposition dépendant cette fois de l'environnement utilisé sont alors nécessaires pour une situation spécifique.

En termes de frames, un GTM est décrit par les structures hiérarchiques suivantes : les GGF (General Goal Frame) décrivent la structure en buts de la tâche, les IGF (Instrumental Goal Frame) décrivent comment, à partir des procédures TKS, chacun des buts est atteint en termes de Macro-Actions, lesquelles sont décomposées en Micro-*Actions*. Lorsque la décomposition des Macro-Actions ne peut être déterminée a priori, l'approche utilise des *Stratégies* décrites sous forme de règles de production. Les Actions à tous les niveaux sont reliées par des conditions de causalité : nécessaire et/ou

---

[16] (GTM) pour General Task Model. (STM) pour Specific Task Model ; par exemple, une action peut être soit entièrement supportée par le système, soit interactive, i.e. s'effectue par une collaboration entre le système et l'utilisateur, ou soit entièrement supportée par l'utilisateur (e.g. décision). (SIM) pour Specific Interface Model.

suffisante. Les *Objets* représentent les structures d'objets associées aux structures TKS. Cependant, il semble que ces différentes étapes fournissent plutôt un support d'aide à la spécification des recommandations de l'interface en terme d'utilisation (e.g. procédure d'accès) qu'une méthode explicitant les connaissances ou règles permettant de traduire des caractéristiques de la tâche en spécifications d'interfaces en ce sens que ces connaissances demeurent implicites. En outre, de récents travaux sur le modèle TKS ont tendance à définir des structures intermédiaires [Johnson91 ; Johnson et al.92] entre les connaissances de la tâche (e.g. les objets, les actions) et celles définies dans les modèles d'architecture tels que PAC ou TUBE. Cette perspective semble en effet prometteuse ; la proposition que nous esquissons dans la figure 2 et détaillons dans le dernier chapitre entre dans le même cadre de réflexion, à l'approche près.

4.1.2 D'autres Modèles et Approches

Plusieurs modèles de tâches relatifs à l'étude de l'interaction homme-machine, existent dans la littérature dont les principaux ont été décrits dans le chapitre 2. La plupart de ces modèles ne s'appliquent qu'à des problèmes spécifiques tels que : le temps nécessaire à l'interaction, la complexité d'utilisation, les difficultés d'apprentissage, etc. De plus, ces modèles ont malheureusement pour vocation l'évaluation d'interfaces et non la conception d'interfaces a priori, i.e. la modélisation des données tâches comme un processus de conception. Toutefois, des modèles tels que TKS, MOST recueillent et modélisent la connaissance à un haut niveau d'abstraction avec la perspective de fournir des architectures d'acquisition et de modélisation des structures de la tâche et de leurs attributs (e.g. la fréquence des procédures, les types de relations), et l'objectif de fournir des outils d'aide à l'identification des exigences de la tâche à prendre en compte lors de la conception et/ou de la génération des premières spécifications de l'interface. Ce que nous allons décrire dans les paragraphes suivants.

4.1.2.1 Graphes de Tâches et Spécification d'Interfaces

On trouve dans [McGrew91] une approche singulière de l'analyse des tâches en ce sens qu'elle utilise des techniques mathématiques formelles. Il s'agit d'une méthode basée sur la théorie des graphes et l'algèbre matricielle permettant de décrire et d'analyser les tâches utilisateur en termes de leurs relations avec l'objectif d'un transfert cohérent des connaissances concernant les tâches depuis l'analyse jusqu'à la conception de l'interface entre les différentes équipes d'un projet. En premier, les tâches sont représentées sous forme de graphes puis de matrices d'adjacence. L'application des techniques et méthodes de transformations matricielles (e.g. points connectés, chemins courts) permet de simplifier les relations entre les tâches afin de générer par la suite des structures de données (e.g. sous forme d'arbres) et dériver le contenu de l'interface en termes d'écrans

et de menus. Les graphes utilisés dans la méthode sont tels que seuls les noeuds représentant les tâches sont étiquetés et que les liens peuvent être unidirectionnels ou bidirectionnels. La matrice carrée reproduit les liens entre les tâches en mettant à 1 la case correspondant à deux tâches connectées. La matrice possède en plus une ligne et une colonne calculant la somme des liens en horizontal et en vertical.

Cette matrice sera ensuite transformée en termes de *cliques* et de *points de coupure*[17]. Le premier concept permet de regrouper les tâches en sous ensembles afin de constituer et structurer par la suite les menus, les commandes, etc. Le second permet de déterminer les liens fonctionnels et logiques entre les cliques. Ainsi, la matrice résultante donne lieu à un nouveau graphe simplifié qui sera transformé en d'autres formats selon les besoins de chacune des phases de la conception, et utilisé pour la génération des structures de haut niveau de l'interface (e.g. les fenêtres, les icônes, les boîtes). Pour ce faire, le graphe simplifié sera converti en une structure arborescente, laquelle sera étendue par l'introduction des tâches du graphe original à chacun des niveaux. Cette arborescence étendue permettra de générer les menus, leurs contenus ainsi que leurs interrelations. Cette approche semble intéressante du fait qu'elle soit simple et formelle. Toutefois, elle ne précise pas le contenu da la tâche ni comment celle ci s'acquiert ; et de fait ne peut s'appliquer à des tâches complexes et structurées. Elle considère en effet les tâches comme des processus fonctionnellement dépendants.

4.1.2.2 Le Modèle ATOM

Le modèle ATOM, Analysis for Task Object Modelling [Walsh89], entre dans le cadre de l'intégration de l'analyse de la tâche dans les méthodes structurées de conception de systèmes [Walsh et al.88 ; 89] avec l'objectif d'aboutir à des spécifications d'interfaces compatibles avec celles de ces méthodes en termes de description et d'utilité. Pour ce faire, la méthode utilise un langage de description intermédiaire permettant de décrire les objets, les actions et les relations. Les objets, principal concept du modèle, regroupent les objets concrets *désignables* et les objets abstraits *non désignables* en ce sens que les premiers sont physiques, i.e. associés à des noms propres alors que les seconds ne le sont pas. Parmi les objets désignables, on distingue ceux qui sont impliqués lors de l'interaction avec l'utilisateur : les acteurs. Les actions telles qu'elles sont utilisées dans la description des tâches sont associées aux objets préalablement établis. Les relations permettent de décrire les liens fonctionnels entre les objets. Ce point de vue semble en effet proche de la conception orientée objet ; les modèles

[17] Une clique est un sous graphe où tous les noeuds sont connectés. Si une clique a trois noeuds, le graphe sera vu comme un triangle. Si elle en a quatre, elle sera vu comme un rectangle. Un ou des points de coupure sont des noeuds qui lorsqu'ils sont supprimés découpent le graphe en plusieurs sous graphes séparés.

d'analyse de tâches, par contre, raisonnent d'abord en termes de buts (objectifs) utilisateur, lesquels impliquent, par la suite, les objets et informations utilisés.

Ce langage sert de support pour décrire le modèle conceptuel de la tâche suite à l'étape de définition des pré requis utilisateur, i.e. à l'identification des catégories de fonctions ainsi que leurs recommandations. Pour ce faire, ATOM offre des guides pour traduire ces dernières par rapport aux différents concepts du langage. A cette étape, un modèle statique reflétant une base d'objets hypertexte est constitué. Pour illustrer la dynamique de ce modèle en termes d'événements utilisateur, ATOM a recours à une représentation structurelle de la tâche : TSD (Task Structure Diagram), décrivant les enchaînements des actions (e.g. séquence, sélection) de chacune des entités tâches identifiées [Lim et al.90]. La troisième étape de la méthode consiste à traduire le modèle conceptuel en spécifications de l'interface via un ensemble d'heuristiques (e.g. conception des menus, des affichages). Par exemple, l'heuristique de conception des menus suggère que les entités soient représentés par des menus et que la sélection d'un menu donne l'accès aux actions qui lui sont associées. Cependant, cette approche concerne particulièrement la méthode JSD ; et de fait l'intégration des facteurs humains est très spécifique à la méthode en ce sens qu'elle est influencée par ses notations ainsi que par ses niveaux de conception.

4.1.2.3 Le Modèle MOST

Partant du constat de la non prise en compte explicite de l'utilisateur final que ce soit dans les méthodes de l'ingénierie ou dans les approches de l'interaction homme-machine, [Carter90a ; 90b ; 91] propose une approche s'appuyant sur la notion de contexte d'utilisation, c'est-à-dire ayant pour objectif d'intégrer explicitement les connaissances relatives aux trois modèles nécessaires à la conception de systèmes centrés utilisateur (adaptatifs) à savoir : le modèle de conception (e.g. les données), le modèle de l'interaction (l'interface) ainsi que le modèle de l'utilisateur [Totterdell et al. 87]. Les connaissances relatives à ces modèles sont organisées selon l'architecture CMS, Context Management System [Schweighardt90], qui permet d'adapter l'interface[18] à partir de l'état courant (current focus) des connaissances en rapport avec les différents modules du système adaptatif : l'utilisateur, l'interface et l'application.

En effet, la méthodologie MOST, Multi-Oriented Structured Task analysis, tente de combler les problèmes de cohérence et de complétude dont font défaut les approches de conception actuelles. Elle consiste en des techniques, couplées d'un prototype

[18] CMS gère en effet plusieurs bases de connaissances (e.g. l'utilisateur), ainsi que leurs liens, afin de fournir des alternatives d'adaptabilité en fonction du contexte courant d'utilisation, i.e. les chemins par lesquels le système peut adapter son interface tout en maintenant sa cohérence fonctionnelle.

développé en HyperCard qui implémente la méthodologie sous forme d'un hypertexte, d'acquisition des catégories connaissances concernant les différents modèles. Chaque catégorie représente ses connaissances sous forme d'un réseau sémantique (e.g. tasks focus) dont les noeuds correspondent à des structures de données prédéfinies[19]. Ces bases de connaissances qui sont au nombre de cinq concernent les utilisateurs, les tâches, les données, les outils (e.g. les UIMS) et les contraintes externes (e.g. la législation). Elles sont utilisées d'une part pour déterminer les stéréotypes d'utilisateurs en fonction des tâches et des données qu'ils traitent, et d'autre part pour sélectionner et/ou développer les interfaces en rapport avec les caractéristiques des utilisateurs (e.g. les types de fonctions), tout en tenant compte des contraintes externes (e.g. les droits d'accès). L'analyse consiste à remplir les différents sections des objets (records) relatifs à chacune des bases de connaissances. Ces objets sont structurés en fonction de leurs liens internes (l'héritage) et externes, i.e. avec les objets des autres bases. Les différentes sections des objets et leurs liens constituent, selon l'auteur, un support de conception en ce sens qu'elles permettent une description flexible et cohérente.

L'idée dans MOST est de permettre au concepteur d'alterner entre les phases d'analyse et de conception, et dans n'importe quel ordre (top-down, bottom-up), afin d'aboutir à des spécifications (informelles) tenant compte des caractéristiques évolutives (les connaissances d'utilisation) des utilisateurs dans un système interactif. Cela avec le souci de cohérence et de complétude du fait de la modélisation explicite des utilisateurs, des tâches et de leurs liens avec les composants internes qui sont les outils et les données. De fait, cette approche semble plutôt une perspective d'aide à l'acquisition et l'intégration (e.g. gestion dynamique de la cohérence) de plusieurs types connaissances, en vue d'intégrer dans un seul processus les divers aspects de la conception, qu'une approche de spécification d'interfaces. En effet, elle concerne plutôt la spécification de la structure des outils nécessaires à la conception d'un style de dialogue que la structure du contenu du dialogue. En outre, l'idée que les connaissances d'utilisation peuvent s'acquérir a priori paraît réductrice en ce sens que ces connaissances dépendent non seulement des tâches à effectuer mais aussi des possibilités qu'offre le nouveau système. Cela du fait qu'il n'est pas clair si les outils analysés sont les mêmes que ceux de l'environnement de l'utilisateur final.

---

[19] Toutes ces catégories utilisent une structure de donnée commune, de type Record, composée de plusieurs sections (e.g. le type, l'identification, les liens intra-types ou inter-types). Les liens et l'héritage entre les records sont à la base de la méthodologie. En effet, plusieurs types de liens (e.g. is-a ; is-an instance ; is-a part ; is-not-an instance) et de champs pour l'héritage (e.g. default ; exclude ; new) sont utilisés.

4.1.3 Discussion

De telles approches entrent, pour la plupart d'entre elles, dans le cadre des méthodes d'aide à la description des tâches en vue d'assister les concepteurs lors des processus d'analyse, en particulier des points de vue cohérence et complétude [Braudes91], et/ou de spécification de l'interface [De Haan et al.91]. Il est à noter que toutes ces méthodes manipulent plus ou moins les mêmes concepts (e.g. les objets, les rôles, les prédicats) et ont pour base l'exploitation des différents liens (e.g. liens fonctionnels) qui existent entre les tâches. D'autre part, des niveaux intermédiaires, en termes de plans (e.g. [Swell et al.90]), de procédures, d'actions, etc, entre les structures de tâches et les spécifications d'interfaces ont été définies comme un support à des alternatives de conception ; aspects que nous retenons dans notre approche de spécification (§5). Cependant, la prise en compte des connaissances ergonomiques (human factors) semble plus implicite qu'explicite en ce sens que soit ces aspects incombent au concepteur auquel cas il s'agit d'outils d'aide à l'analyse de la tâche, soit la traduction passe par des heuristiques empiriques et surtout spécifiques (e.g. la génération de menus, etc) aux cas traités. De ce point de vue, nous préconisons l'introduction de concepts génériques (e.g. la notion d'état) permettant d'établir une structure intermédiaire, et par conséquent des liens sémantiques, entre les concepts de la tâche et ceux de l'interface. Cela par le biais d'heuristiques ergonomiques en rapport avec la sémantique de la tâche.

A la revue des différentes recherches, il apparaît que l'un des centres d'intérêts actuels concerne l'intégration et/ou la traduction des connaissances concernant la tâche et/ou l'utilisateur dans les méthodes de conception ; ce qui permet par conséquent de profiter des acquis de ces méthodes. Cependant une telle perspective conduit souvent à focaliser les spécifications sur le contrôle de pilotage (la charge cognitive) entre l'utilisateur et le système, c'est-à-dire selon l'approche de séparation entre l'interface et l'interaction. Encore faut-il pouvoir définir l'interaction indépendamment du contenu de l'interface. Cet aspect ne concerne qu'un des principes ergonomiques à prendre en compte à la conception. En effet, ces différentes notions correspondent à plusieurs niveaux de l'interaction homme-machine [De Haan et al.91], lesquels sont interdépendants et surtout déterminants à la définition d'un modèle *cohérent* de l'interface utilisateur, c'est-à-dire en termes de représentation (l'interface) et d'utilisation (l'interaction).

4.2 Le Concept de Tâche et la Perspective de Modélisation

D'autres travaux (e.g. [Wielinga et al.86 ; Tong90 ; Pierret91 ; Boy92]) orientés acquisition de connaissances au niveau tâche, et qui relèvent plus des problèmes d'intelligence artificielle que de la spécification d'interfaces utilisateur (Cf §2.3), utilisent le concept de tâches structurées dans l'objectif d'établir entre autres un modèle conceptuel de la connaissance. Il s'agit dans ce cas d'abstraction des connaissances de

contrôle de manière à les distinguer des connaissances du domaine ; et de fait la définition et l'organisation de plusieurs niveaux de contrôle permettant l'acquisition des connaissances au niveau conceptuel. Cela afin de pallier aux insuffisances des systèmes experts de première génération où ces différents types de connaissances sont éparpillés dans le résolveur et/ou la base de connaissances. Toutefois, les définitions et utilisations du concept de la tâche, selon les points de vue de l'intelligence artificielle ou de l'ergonomie cognitive, ne semblent pas totalement indépendantes sachant qu'elles concernent dans les deux cas les problèmes de représentation et de manipulation des connaissances orientées tâches — e.g. le modèle TKS (§4.1.1).

Nous allons dans ce paragraphe discuter le concept de modélisation orientée tâches. En particulier, nous étudions l'approche de représentation objet des tâches [Pierret et al.89], décrites à l'aide du formalisme MAD [Scapin et al.89], à la frontière des deux perspectives soulignées plus haut. En effet, l'approche consiste en la formalisation du concept de tâches hiérarchisées à partir d'un modèle de représentation objet de type frames, et par conséquent appréhende des questions fondamentales sur les modèles à objets pour la représentation sémantique des connaissances structurelles, l'objet composite[20]. Le modèle objet utilisé est le modèle classique à base de frames, Shirka [Rechenmann et al.88], ce qui a conduit à la précision et à l'introduction de certains concepts au niveau méta tels que *l'instance générique* et *l'instance prototypique*[22]. Enfin, nous discutons les perspectives de telles approches par rapport aux objectifs de spécification d'interfaces.

4.2.1 La Modélisation Orientée Tâches

L'objectif dans la modélisation des tâches utilisateur est d'établir une représentation opérationnelle de la tâche de manière à en tenir compte lors de la spécification d'applications interactives, et ce aux niveaux conceptuel et sémantique. Une étape préalable concerne le recueil de données ; on trouve dans [El Farouki et al.91] une approche de recueil relative à l'étude d'une activité très complexe, celle des contrôleurs aériens. A l'issue de cette étape, la description produit une structure hiérarchique des tâches allant du plus haut au plus bas niveaux d'abstraction (e.g. les actions).

Pour modéliser au plus haut niveau d'abstraction le concept générique de tâche et d'objet composite, le modèle utilise le niveau "méta sh-tâche" défini comme une spécialisation d'un objet prédéfini, l'objet "schéma". Le concept d'objet composite (e.g.

---

[20] Un objet composite est formé par l'agrégation d'autres objets, appelés composants, qui décrivent chacun une de ses parties [Masini et al.89]. Une *instance générique* est un représentant abstrait des autres instances, en ce sens qu'elle est décrite par des variables universelles. Une *instance prototypique* est une instance générique permettant de construire une instance par analogie, elle décrit en effet les valeurs génériques de la classe.

[Hill et al.90]) a été introduit dans l'objectif de modéliser le niveau structure de la tâche, et d'enrichir la notion d'appel procédural des modèles classiques par l'apport d'un mécanisme d'exploitation dynamique qui permettrait l'attachement de méthodes de haut niveau. La décomposition structurelle de la tâche est de fait considérée comme une propriété de la classe et non de l'instance.

L'idée est de permettre des appels de procédures spécialisées des niveaux inférieurs, c'est-à-dire des appels récursifs d'une suite de tâches structurées afin d'inférer la valeur d'un attribut lorsqu'elle est nécessaire, ou calculer, en terme d'aide à la conception d'interfaces, les étapes (chemins) nécessaires à la réalisation d'une tâche quelque soit son niveau dans l'arbre. Cependant, cette approche semble plutôt orientée vers la modélisation de connaissances dynamiques (e.g. le raisonnement) que la modélisation des connaissances à des fins d'analyse (e.g. vérification de la complétude [Braudes91]) et/ou de spécification (e.g. application d'heuristiques de spécification en fonction des patterns, tels que la structure et les objets d'une tâche, préalablement identifiés). De ce point de vue, les connaissances concernant la tâche utilisateur ne sont pas considérées comme des connaissances expertes, dont l'objectif est de modéliser le raisonnement, mais plutôt comme des connaissances de l'activité, en terme d'analyse de l'existant, afin de pouvoir les exploiter comme une source d'informations utiles à la spécification des contraintes de l'application d'un point de vue de l'utilisateur.

4.2.2 La Représentation Objet des Tâches

Comme souligné plus haut, des modifications ont été nécessaires pour appréhender la question de la représentation des connaissances structurelles[21]. Une tâche est modélisée, dans un premier temps, en fonction de sa structure et ses objets en entrées et en sorties selon le modèle à trois niveaux (Figure 3). Le *niveau méta* représente la tâche au plus haut niveau d'abstraction par le méta-schéma sh-tâche qui fournit une description générique de la tâche indépendamment de la nature et du type des objets. Le *niveau classe* concerne une tâche spécifique, laquelle est décrite par le schéma classe, instance du méta-schéma sh-tâche, décrivant le profil propre à cette tâche, c'est-à-dire la liste de ses entrées, la liste de ses sorties et la liste de ses sous-tâches, ainsi que les types (classes) des différents objets manipulés. Le *niveau instance* correspond à une réalisation particulière de la classe tâche. Elle constitue ainsi une mémoire d'exécution de la tâche. Ceci en instanciant les objets décrivant les différents attributs de la tâche.

---

[21] Les principaux descripteurs de la tâche selon MAD sont : *Nom*, identifie la tâche. *Etat initial et Etat final*, regroupent les objets en entrée et en sortie de la tâche. *Préconditions et Postconditions*, prédicats exprimant des conditions sur les objets de l'état initial et de l'état final. *Structure*, décrit la liste des sous tâches reliées par un constructeur de décomposition (e.g. séquentiel, parallèle).

Une tâche est donc représentée par une classe ; chaque instance de cette classe modélise une réalisation particulière de la tâche. Une tâche spécifique, créée par l'utilisateur, est représentée par une sous-classe (spécialisation) d'une classe prédéfinie. En effet, le niveau classe correspond à une hiérarchie d'objets génériques instances du niveau méta représentant les tâches prédéfinies et hiérarchisées selon la nature de leur structure et le type des objets en entrée et en sortie.

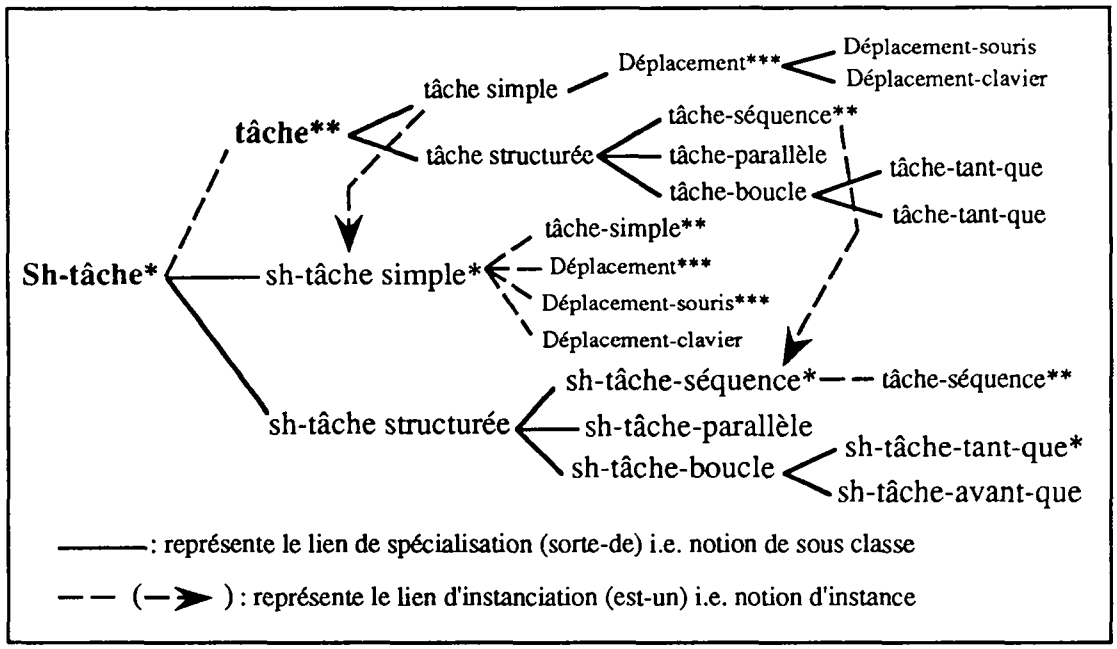

Figure 3 : Le Modèle Objet des Tâches[22].

En termes d'objets, une tâche utilisateur est représentée au niveau classe. Le couple (classe, instance) permet de décrire une tâche quelconque où le niveau classe désigne la tâche générique et le niveau instance une tâche effective. Les méthodes (actions) correspondant aux tâches (e.g. Déplacement-souris/clavier) diffèrent de par leurs exécutions, mais se retrouvent dans une super-classe décrivant la structure générique commune à toutes les classes effectuant un déplacement. De plus, les classes définies à ce niveau ont soit un corps vide, soit un corps contenant des méthodes ne répondant à aucun message, c'est-à-dire des méthodes abstraites.

4.2.3 Discussion

Cette modélisation appréhende en effet les aspects liées à la représentation structurelle des tâches et des conditions (e.g. pré-conditions) en termes de classe et instance, avec la perspective générale d'aide à la conception d'interfaces. Cependant, la perspective de

[22] Sont représentées les hiérarchies des sh-tâches (niveau méta*) — e.g. sh-tâche-simple, des tâches (classes prédéfinies**) — e.g. tâche-simple, et des tâches utilisateur *** (e.g. Déplacement clavier).

modélisation, ou d'aide à la conception, de l'interface par des mécanismes d'inférence tels que les calculs et les attachements procéduraux semble complexe et surtout indépendante de la logique ergonomique ; il n'est en effet pas défini de liens explicites (modélisation) ou implicites (aide à la conception) avec les aspects de l'interface (e.g. mécanismes d'identification des caractéristiques particulières des tâches telles que la fréquence des tâches, les structures d'objets, les structures d'actions, etc).

L'objectif de telles approches consiste plutôt à expliciter les connaissances superficielles (le contrôle) des connaissances profondes (le domaine). Cela afin de modéliser les connaissances selon leurs fonctions et/ou rôles (aspect statique) et leurs utilisations (aspect dynamique) ; et par conséquent mieux expliciter et expliquer le raisonnement par rapport aux entités du domaine. De ce point de vue, la modélisation en termes de tâches (e.g. [Tong90 ; Pierret91]) concerne l'acquisition et/ou l'extraction des connaissances expertes du domaine (e.g. le diagnostic), et a pour but la simulation du raisonnement tel qu'il est accompli par les experts. En outre, les tâches telles qu'elles sont décrites dans le formalisme MAD sont *inertes* en ce sens qu'il ne s'agit pas de reproduire le raisonnement de l'utilisateur mais de pouvoir décrire, de façons cohérente et complète, la tâche utilisateur sous formes *structurelle et fonctionnelle* afin de l'exploiter pour la spécification conceptuelle de l'interface.

D'autre part, dans la perspective de modélisation des tâches et des règles en vue de la spécification d'interfaces, en plus des difficultés de description [Delsol92], des extensions (respectivement au niveau description) du formalisme MAD s'avèrent nécessaires. Une catégorie de modèles à base d'objets semble propice à ce type de modélisation, i.e. la représentation des connaissances à base d'objets et de règles. Il s'agit des modèles hybrides [Masini et al.89 ; Ferber89] tels que Kool [Bull91], Smeci [Smeci91]. Ce qui importe dans une telle modélisation c'est d'aboutir à une représentation conceptuelle de l'interface compatible avec les objectifs des utilisateurs. Cela, bien évidemment, par l'intermédiaire d'heuristiques déduites des recommandations ergonomiques. Ce que nous allons décrire dans le chapitre suivant.

5 VERS UNE METHODOLOGIE DE SPECIFICATION D'INTERFACES ERGONOMIQUES : UNE APPROCHE

Nous allons dans ce chapitre décrire les principes et les étapes de l'approche concernant la spécification d'interfaces à un haut niveau d'abstraction. En effet, nous donnerons les définitions des concepts que nous visons par rapport à l'interface, leurs relations et donc le modèle conceptuel de l'interface. Mais avant cela, nous discutons le modèle MAD par rapport aux nouveaux besoins de formalisation. Il s'agit de préciser les descripteurs (attributs) existants et/ou d'argumenter l'ajout, si nécessaire, d'autres descripteurs par

rapport aux objectifs décrits dans les chapitres précédents, à savoir la définition de la sémantique de l'interface. Nous nous intéresserons en particulier à la définition de descripteurs nécessaires à la définition de la répartition du contrôle entre l'interface et l'utilisateur afin de pouvoir décrire le dialogue utilisateur de manière explicite. Nous présentons enfin le modèle en termes d'objets ainsi que la perspective entreprise dans le cadre de nos travaux quant à cette modélisation. Il sera question de décrire, en fonction des points soulignés ci dessus, les règles sous jacentes à la méthode de spécification.

5.1 La Spécification d'Interfaces Utilisateur

L'objectif dans ce paragraphe est de donner notre point de vue, en termes de définition et de concepts, concernant la spécification d'interfaces utilisateur. Cela consiste en une perspective permettant d'aborder les aspects interface d'un point de vue autre que ceux des outils de spécification et/ou des méthodes de conception. En effet, nous abordons l'interface du point de vue de la tâche utilisateur[23], et donc à un plus haut niveau d'abstraction : le niveau cognitif. De plus, notre approche se veut analytique puisqu'il s'agit de fournir des spécifications, a priori, en se basant sur une base de connaissances ergonomiques (Figure 4).

5.1.1 Le Formalisme MAD : Rappels et Perspective

Le formalisme MAD permet de décrire les tâches allant plus haut niveau (e.g. tâche générique, au sens de Johnson et al. et de Diaper (§4), c'est-à-dire que les objets décrits à ce niveau d'abstraction s'instancient aux niveaux des tâches subalternes) au plus bas niveau d'abstraction (Cf §2.3) — e.g. une action élémentaire telle que l'activation d'une commande simple (créer document) dans un traitement de texte. Les objets mis en oeuvre à ce niveau d'abstraction concerne généralement le niveau instance en ce sens qu'ils sont soit *instanciés* (coller "texte") par les objets des niveaux supérieurs, soit *instanciables* (e.g. demande d'une valeur d'un paramètre, les paramètres d'impression). Bien sûr, nous considérons, dans ces exemples, le cas des objets *interactifs*, i.e. les objets impliqués lors de l'interaction utilisateur. Cela du fait que ce n'est pas la seule catégorie d'objets intervenant à ce niveau de définition de l'interface (§5.3).

---

[23] Cette perspective se situe à un niveau logique, i.e. indépendant des outils utilisés. Il s'agit là d'exprimer les besoins utilisateur des points de vue fonctionnel et procédural. Nous estimons que les choix techniques concernent les niveaux de conception et d'implémentation. En effet, ce niveau va au delà des modèles de définition de l'interface à partir des données de l'application (e.g. [Carey et al.87 ; De Baar et al.92]) e/ou les langages de description d'interfaces (e.g. [Foley et al.91 ; Olsen89]). De plus, l'utilisation des connaissances ergonomiques permet, entre autres, l'indépendance entre la tâche et les langages de description des interfaces. On peut, en effet, envisager à l'issue de la spécification des concepts et relations de l'interface, l'utilisation, par exemple, de notations formelles telles que ETAG [Tauber90].

Pour décrire les tâches utilisateur, le formalisme utilise le paradigme de la planification hiérarchique [Sacerdoti77], en l'élargissant à la notion de hiérarchies d'items et en y incluant certains aspects de synchronisation [Scapin et al.89]. Chaque item est caractérisé par un ensemble de descripteurs distinguant l'aspect déclaratif de l'aspect procédural. Le premier décrit les objets et valeurs des éléments (e.g. les conditions) intervenant lors de réalisation de la tâche, le second concerne soit la décomposition structurelle (pour des tâches structurées), ou soit la définition de l'action (pour des tâches élémentaires).

Nous avons décrit dans le chapitre précédent les descripteurs relatifs à l'aspect déclaratif de la tâche (Cf Note 23). La décomposition structurelle des tâches s'opère à l'aide d'opérateurs de synchronisation exclusifs dont les plus importants sont : Séquentiel, pour décrire les tâches en séquence ; Parallèle, pour décrire les tâches parallèles et ce faisant abstraction du facteur temps, i.e. à l'heure actuelle, en effet, nous considérons que cela peut concerner, en terme de temps, des tâches simultanées, imbriquées, etc. D'autant plus que ces considérations ne sont pas prépondérantes du point de vue de la spécification de l'interface. Enfin, les constructeurs Boucle, Facultatif et Alternatif concernent respectivement le fait que la tâche peut boucler sur une condition (e.g. post-condition), être optionnelle en ce sens qu'elle n'est pas obligatoire, et que le choix des tâches subordonnées est exclusif à savoir qu'une seule est effectuée à la fois.

5.1.2 Interface Conceptuelle : Une Définition

Comme nous l'avons souligné dans les chapitres précédents, la spécification d'interfaces a été abordée selon divers perspectives, et a donc été l'objet de plusieurs définitions. Dans ce paragraphe nous donnons notre propre définition ; cela en essayant de la distinguer de par les concepts utilisés et les objectifs visés. De fait, on introduit ici la notion *d'interface conceptuelle* ; on décrira plus loin les mécanismes de spécification de l'interface. En effet, cette nécessité de précision de la notion d'interface tient à la diversité des définitions existantes (e.g. interface au sens de l'ingénierie [Hartson et al.89] ; interface en ergonomie [Scapin86]). Dans la perspective de spécification de l'interface à un haut niveau d'abstraction, on considérera la définition suivante de l'interface conceptuelle.

Définition : L'interface conceptuelle est à la frontière de l'application et de l'interface au sens de l'ingénierie informatique. Elle est plus restreinte que l'interface dans sa globalité (telle que définie en ergonomie). L'interface conceptuelle concerne tous les aspects de haut niveau d'abstraction de l'interface en ce qu'ils sont liés aux tâches utilisateurs. L'interface conceptuelle peut être définie par ses éléments constitutifs : Etat, Schéma de Procédures, Procédure, Fonctionnalité, Action, Objet Fonctionnel, Interacteur, Ecran et Evénement.

L'idée est de définir ces concepts abstraits (aspect statique) ainsi que leurs relations (aspect dynamique) en adéquation avec la structure de la tâche. Cette modélisation est le résultat d'une mise en correspondance des caractéristiques sémantiques de la tâche avec des "heuristiques" (recommandations) ergonomiques de haut niveau (Figure 4). Par exemple, le fait que les préconditions d'une tâche consistent en la validation simultanée de ses arguments entraîne par le biais d'une ou plusieurs règles que ces arguments (Objets Fonctionnels) doivent être accessibles (Actions) à l'utilisateur et cela de façon à faciliter leurs validations (e.g. selon le critère d'affichage, d'accès, etc).

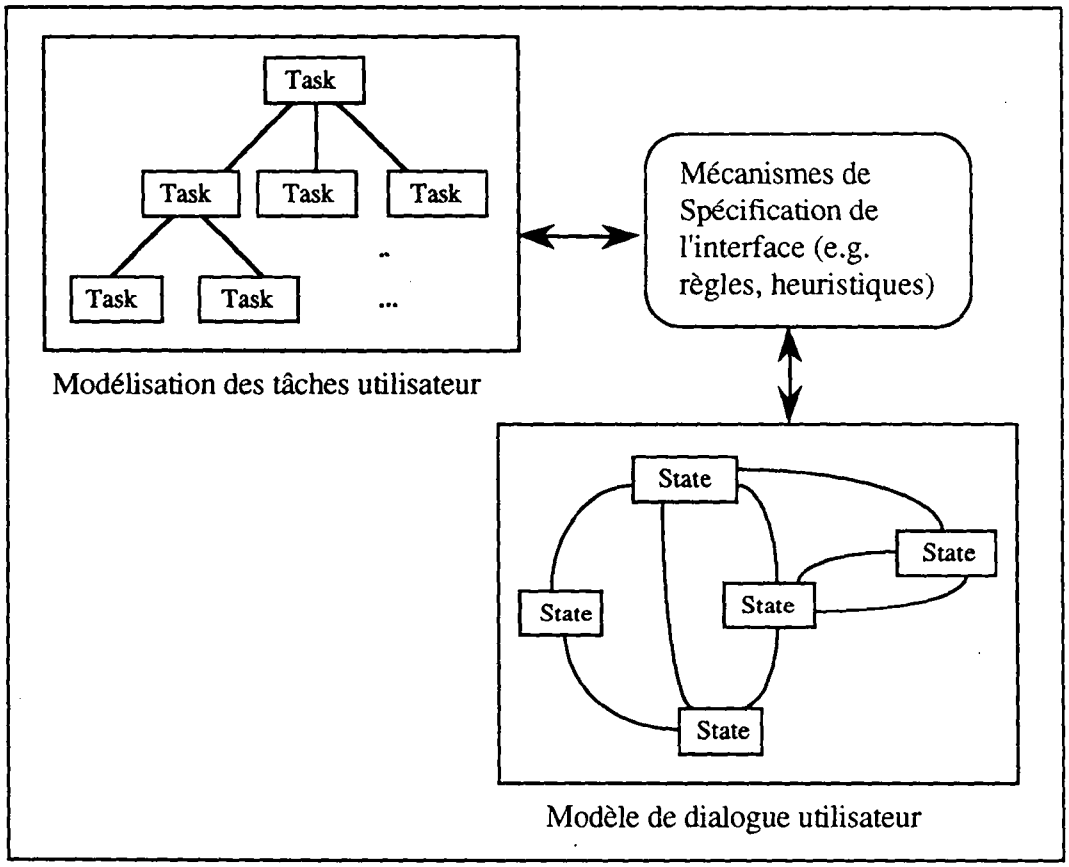

Figure 4 : Approche de spécification de l'interface conceptuelle

5.1.3 Les Concepts Intermédiaires

Suite à la définition du paragraphe précédent, nous donnons ici les définitions des concepts *intermédiaires* qui constituent le support des objectifs visés, à savoir la définition de la structure de l'interface, i.e. le dialogue utilisateur, à partir de la tâche. On parle aussi dans ce cas d'*interface orientée-tâche* (e.g. [Seaton et al.92]).Ces définitions caractérisent les objets conceptuels de l'interface qui nous permettront par la suite de contrôler par exemple la génération d'interfaces. Ces concepts sont à la base de

la structure (formalisme) intermédiaire à la tâche et à l'interface[24] que nous décrivons plus loin dans le paragraphe (§5.3).

Etat

Un état correspond à l'ensemble des objets de l'interface conceptuelle tels qu'ils sont transformés par un traitement de l'application ou un événement utilisateur. On peut dire qu'un état représente une *fonction d'interaction* appliquée aux objets existants de l'interface conceptuelle ; la fonction correspondant à un événement de l'application (en sortie) ou de l'utilisateur (en entrée). Un état se caractérise par tout changement d'un ou plusieurs objets désignables (e.g. interacteur) ou non (e.g. titre) ayant une traduction perceptible par l'utilisateur. Il constitue le concept central de la méthode en ce sens qu'il regroupe tous les éléments de la spécification[25]. Un état est donc un élément composite.

Schéma de Procédures

Un Schéma de Procédures (ou stratégie) gère l'ensemble des procédures. Il correspond à la synchronisation des différentes procédures, i.e. un *objectif-tâche*, et éventuellement, à la gestion des interruptions entre procédures ; ce qui conduit à la définition de la notion *contexte global*. L'instanciation du contexte global sera déterminé par les événements (e.g. interruption, application) pouvant affecter les différents procédures (contextes locaux).

Procédure

Une procédure (d'interaction) gère une liste d'actions synchronisées permettant d'enclencher un dialogue avec l'utilisateur, et atteindre ainsi un *objectif-dialogue*. Elle correspond à la spécification de l'interaction concernant une sous activité de l'utilisateur (e.g. une configuration de tâches) ; ce qui conduit à la définition de la notion de *contexte local* ; laquelle sera utilisée pour la gestion des interruptions entre actions.

---

[24] D'autres attributs concernant les concepts ci-dessus sont susceptibles d'être définis au fur et à mesure de la définition de la structure de l'interface. Néanmoins, les aspects synchronisation des éléments de l'interface (e.g. actions, dialogue) seront principalement déduits des constructeurs logiques de la tâche et inclus lors de la définition des schémas de procédures (stratégies) et des procédures.
D'autre part, la notion de "contexte de l'interface" (e.g. Carter et al.87 ; Foley et al.91) concerne, selon notre point de vue, la structuration sémantique de l'interface, ainsi que sa dynamique. Cela par le biais des différents concepts introduits dans ce paragraphe. Il s'agit, ici, d'une notion globale permettant d'effectuer des choix et des décisions quant à la réaction de l'interface, et donc de l'application, lorsque des ambiguïtés ou des incohérences s'imposent (e.g. le fait d'appuyer sur la touche "coller" du traitement de texte Word copie le dernier mot ou texte se trouvant dans "le presse-papier").

[25] Remarquons que selon qu'il s'agisse de spécifier l'interface pour les environnements nouveaux (e.g. les systèmes de gestion multi-fenêtres) ou classiques, les choix de conception diffèrent. En effet, avec les récents environnements, l'utilisateur peut effectuer plusieurs dialogues dans plusieurs fenêtres représentées dans un même état. Ce qui n'est pas le cas pour les systèmes classiques. Ainsi la notion d'état et ses paramètres varient selon le système cible de la conception. Toutefois, la spécification telle que nous l'abordons ici se situe à un niveau plus abstrait et par conséquent indépendant.

Fonctionnalité

Une fonctionnalité consiste en une fonction (un ordre) exécutée par le système, en particulier à la demande de l'utilisateur (e.g. afficher, rechercher). Le cas des fonctions non explicitement demandées par l'utilisateur, aurait pour fonctionnalité d'informer l'utilisateur, si cela est utile d'un point de vue de sa tâche (e.g. vérification de la cohérence des entrées). Une fonctionnalité peut éventuellement déclencher d'autres fonctionnalités subalternes. Son utilisation se rapproche de celle de la procédure. Cependant, on différencie une fonctionnalité d'une procédure par le fait que les actions associées à une fonctionnalité portent obligatoirement sur la même classe d'objets ou une de ses classes spécialisées.

Action

Une action est une fonctionnalité élémentaire (atomique) en ce sens qu'elle satisfait un *objectif-utilisateur*. Une action correspond ainsi à une transaction élémentaire, i.e. un événement utilisateur, une réponse système, sur un objet conceptuel de l'application. De fait, on s'intéresse particulièrement ici aux actions qui impliquent les concepts de l'application lors de l'interaction utilisateur.

Objets Fonctionnels

Ce sont les objets explicites, ou implicites, qui interviennent dans les différents états de l'interface et qui sont nécessaires à la réalisation des objectifs de la tâche. Ces objets représentent les différentes instanciations des fonctionnalités de l'application et de leurs arguments (e.g. couper (texte), afficher (message), etc).

Interacteur

Un interacteur représente tout objet (élément) de l'interface (e.g. [De Baar et al.92]) désignable ou non qui permet une opération de l'utilisateur (e.g. confirmation) et/ou de l'application (e.g. message d'erreur) et qui se traduit par un retour (feed-back) quelconque suite à l'opération (la transaction). Un interacteur peut être considéré comme une représentation graphique de l'objet fonctionnel.

Ecran

Un écran est un interacteur d'affichage particulier au sens *d'objets de présentation* (Cf Figure 2) ; et constitue de fait la classe principale dont les sous classes concernent les différents interacteurs (e.g. fenêtre, bouton). Un écran représente une caractéristique physique de l'interface. Dans les nouveaux environnements de gestion d'interfaces, une fenêtre constitue un interacteur qui peut être un des éléments de l'écran alors qu'elle s'apparente à l'écran dans les systèmes classiques. Ainsi, de manière abstraite, nous assimilons le concept d'écran à une surface d'affichage (un display) — e.g. au sens de Display Object [Hix et al.87].

Evénement

Un événement constitue l'élément permettant de déclencher une interaction entre l'utilisateur et l'application, i.e. de spécifier le contrôle utilisateur-application. On distingue trois types d'événements : les *événements utilisateur*, les *événements de l'application* et les *événements extérieurs*. Les premiers décrivent le dialogue dirigé par l'utilisateur, les seconds émanent de l'application afin d'informer (avertir) l'utilisateur (e.g. arrivée d'un strip, horloge) de l'évolution de l'activité, ou pour une demande d'informations (e.g. acquisition de données ou de paramètres). Les derniers permettent la prise en compte des interruptions entre les tâches (e.g. la priorité, le contexte) et éventuellement entre les utilisateurs ; ils peuvent de fait provenir de l'utilisateur et/ou de l'application. On distingue cette catégorie des deux premières uniquement dans l'objectif de modéliser explicitement la gestion des interruptions par le biais des contextes définis plus haut. En outre, ce concept sera directement utilisé par les pré- et post-conditions pour la spécification de la dynamique de l'interface : le dialogue.

Les heuristiques et/ou procédés qui nous permettront par la suite d'inférer à partir de la structure de la tâche, ou d'établir seulement par appariement (mapping), les concepts définis ci dessus, ainsi que leurs dépendances (Cf Figure 6), seront décrits dans les paragraphes suivants.

5.2 La Modélisation Conceptuelle des Tâches

Dans l'objectif d'identifier et d'utiliser les caractéristiques pertinentes de la tâche à prendre en compte au niveau de l'interface (e.g. la structuration des actions), nous avons établi une modélisation des tâches de manière à l'exploiter par les mécanismes de spécification que nous décrivons plus loin. En effet, nous estimons que la perspective de modélisation orientée tâche est influencée par les objectifs visés (Cf §4.2), à savoir ici par l'approche de spécification de l'interface conceptuelle. De fait, par l'intermédiaire d'une modélisation objet des descripteurs de la tâche, voire la définition d'autres attributs (e.g. le type, les contraintes d'exécution de la tâche), nous procédons dans un premier temps par calculer et organiser les structures (e.g. les hiérarchies d'objets, les fréquences d'actions) de la tâche utiles à la spécification de l'interface.

5.2.1 Le Modèle : Les Objets et leurs Relations

Nous adoptons ainsi une modélisation orientée *objets-structurés* avec le souci de modularité, de réutilisation et surtout de son exploitation par les règles de spécification. Cette représentation tente par conséquent de modéliser de manière standard les éléments qui composent la tâche ainsi que leurs liens *structurels*. Les liens *fonctionnels* (sémantiques) sont implicites dans une telle représentation en ce sens qu'ils sont véhiculés par les *pré-* et *post-conditions* ainsi que par les *contraintes d'exécution*

(Figure 5). L'un des intérêts de cette approche consiste à permettre de reconnaître facilement les hiérarchies des objets en entrées et en sorties utilisées par les différentes tâches afin, par exemple, de vérifier la *cohérence* et la *complétude* de la structuration des objets par rapport à leur utilisation dans les tâches (e.g. [Carter91]), ou recenser les différentes actions en fonction des objets sur lesquels elles s'appliquent, etc.

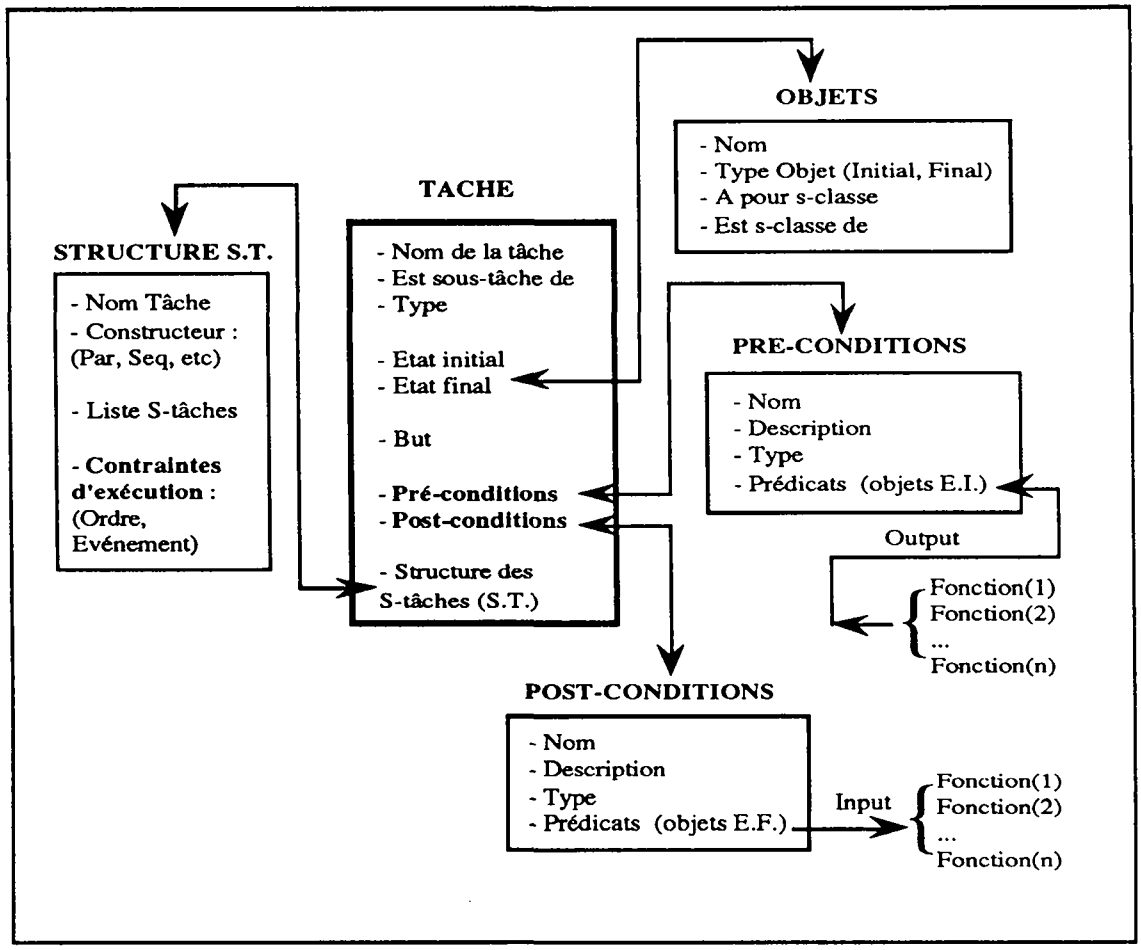

Figure 5: La Modélisation Objet des composants de la Tâche et de leurs Relations.

La figure schématise la sémantique que nous associons aux différents composants de l'entité Tâche, d'une part, avec l'objectif d'acquisition et de représentation des tâches utilisateur par le concepteur en vue de faciliter la vérification de la cohérence et de la complétude des tâches des points de vue de leurs attributs et des *contraintes* du concepteur [Braudes91]. Et d'autre part, afin de pouvoir identifier les relations et les concepts pertinents pour la spécification de l'interface. Par exemple, les liens définis entre les fonctions et les pré- et post-conditions peuvent être utilisées à la spécification de ce qu'on appelle le *feed-back sémantique* d'une application interactive. Nous consacrons ce paragraphe à la description des objets, leurs attributs, leurs relations, et par conséquent, leurs *rôles* par rapport à l'analyse de la tâche et la spécification d'interfaces.

Est représentée en gras l'entité *Tâche*, objet central du modèle en ce sens que les autres entités représentent les composants associés aux attributs de l'entité tâche. Une tâche (ou sous-tâche) est identifiée par l'attribut *Nom*. Les attributs *Etat-Initial* et *Etat-Final* regroupent les objets intervenant en entrée et en sortie de la tâche ainsi que leurs types. Ces attributs sont liés à l'entité *Objets*, et ce à tous les niveaux de l'arbre des tâches.

L'attribut *Type* indique le type de la tâche ; nous retenons en effet trois types : mentale (e.g. prise de décision), manuelle (e.g. en utilisant des outils) ou interactive (éventuellement automatique : l'horloge). En outre, on peut inférer à partir des constructeurs adoptés la *modalité* de la tâche, i.e. optionnelle ou obligatoire — cette inférence s'établit par le test du constructeur "Facultatif". De plus, il serait intéressant d'introduire des constructeurs permettant de décrire le partage des tâches entre utilisateurs et/ou les tâches coopératives ainsi que les moyens de coopération[26].

Les attributs *Pré-* et *Post-conditions* décrivent la dynamique des tâches du point de vue des valeurs actives des objets qu'elles mettent en oeuvre et qui conditionnent leurs exécutions. L'attribut *But* décrit la fonction que la tâche est censée assumée. Enfin, le champ *Structure* concerne la décomposition structurelle de la tâche quand celle ci n'est pas une action élémentaire, c'est-à-dire une fonction de l'application ou de l'utilisateur.

Sont ensuite représentées les classes d'objets associées aux attributs de la tâche décrits en haut. Ces différentes classes seront utilisées par l'approche pour spécifier les différents concepts décrits dans le paragraphe précédent, et éventuellement utilisables pour d'autres besoins telles que l'interrogation, la classification, etc. Les liens sont bidirectionnels, cela afin de permettre plusieurs points d'entrée facilitant des *analyses ascendantes ou descendantes* (e.g. remonter la hiérarchie des préconditions, et de leurs objets, pour évaluer la cohérence d'une procédure donnée).

La classe *Objets* est associée aux attributs (e.g. Etat-initial) décrivant les éléments intervenant au niveau de la tâche. L'attribut *Type* indique si l'objet intervient en entrée, en sortie ou les deux en même temps. Nous conservons aussi la hiérarchie des objets par les deux derniers attributs de la classe pour des besoins de cohérence et d'accessibilité par les règles et/ou les heuristiques de spécification.

Les classes *Pré-conditions* et *Post-conditions* permettent, telles qu'elles sont organisées, d'établir la hiérarchie des "contraintes" que doit respecter le déroulement d'une tâche donnée ainsi que les objets (et les valeurs) qu'elles impliquent. Il s'agit en fait d'une encapsulation des différents objets intervenant lors de l'exécution d'une tâche donnée

---

[26] Le lecteur pourra se référer à des études, orientées résolution de problèmes en situation de coopération, menées selon la perspective utilisateur telles que ([Hamalainen et al.91], [Mahling et al.90] et [Luh et al.90]).

par des opérations (ou groupe d'opérations) à vérifier respectivement avant et après l'exécution de chacune des sous-tâches soit par l'utilisateur ou soit par l'application elle-même. Cela est schématisé sur la figure par des liens en entrée et en sortie par rapport à l'application donnant ainsi lieu à des spécifications des fonctions permettant d'effectuer les tests fonctionnels de l'application interactive, et de fait des points d'entrés entre l'interface et l'application (au sens du noyau fonctionnel).

Enfin la classe *Structure* concerne le composant qui nous indique comment d'un point de vue *ordre et synchronisation* tous les enchaînements seront associées aux objets, aux préconditions et par conséquent aux fonctions de l'application. En effet, elle nous renseigne directement sur les tâches du niveau immédiatement inférieur ainsi que sur les *contraintes d'exécution* explicitant les contraintes en termes de temps (e.g. simultanéité d'accès vs simultanéité d'exécution) et/ou de données (e.g. données partageables par les différentes sous-tâches d'un même niveau).

5.2.2 Discussion

Il apparaît à travers la description des différents objets du modèle générique de la tâche, et des rôles qu'on leurs associe dans le processus de spécification, que l'objectif est d'aboutir à un modèle conceptuel (de l'interaction) le plus proche possible du modèle cognitif (la tâche). Celui ci, implicitement, a pour vocation de représenter entre autres deux aspects importants qui sont : les objets impliqués dans la description des tâches et leurs diverses utilisations. Ainsi, cela permettra d'expliciter, par exemple, les objets impliqués dans la réalisation d'une action (un but), ou procédure, et/ou ceux relatifs aux conditions d'utilisation de l'action ainsi que les objets mis à jour, ou à mettre à jour par l'utilisateur, suite à son exécution.

Ce que nous avons appelé ici "utilisations" ressemble de manière très étroite à ce qui est communément nommé *le modèle des traitements* dans les méthodes du génie logiciel à la différence sensible que l'organisation des données (objets) sont ici dépendantes d'un point de vue de leur utilisation alors qu'elles sont indépendantes selon ce point de vue dans ces méthodes (Cf §3). Par ailleurs, une implémentation, à l'aide d'objets Smeci [Smeci91], de la modélisation orientée tâche (Figure 5) décrite plus haut, est en cours. Celle ci a pour objectifs, d'une part, de permettre, via une interface graphique, la description des tâches utilisateur de manière incrémentale et interactive ; cela en utilisant les composants de la boîte à outils Aida [Aida/Masai92]. Et d'autre part, d'établir des hiérarchies d'objets structurés de manière à vérifier la cohérence et la complétude de celles ci, et à appliquer les différentes techniques (e.g. calcul des différents objets, et leurs conditions, associés à une action ; définition des procédures d'interaction, etc), permettant de fournir les premières spécifications de l'interface conceptuelle. Ce que nous détaillons dans la section suivante.

5.3 DU MODÈLE DE TÂCHES AU MODÈLE INTERMÉDIAIRE DE L'INTERFACE

Avant de décrire de manière plus formelle et explicite les étapes et moyens de mise en correspondance ou d'inférence du modèle conceptuel de l'interface, nous procédons d'abord à la description de la "structure" des entités — i.e. les concepts intermédiaires (§5.1) — de ce modèle et leurs relations.

5.3.1 Le Modèle Conceptuel de l'Interface

La figure 6 schématise le modèle générique de l'interface tel qu'il sera établi à partir du modèle de la tâche. C'est à partir d'une telle structure que l'on pourra ensuite utiliser, par exemple, les langages (notations) formels (e.g. [Siochi et al.91]) de description et/ou de spécification du contrôle et de la présentation de l'interface finale.

5.3.1.1 Le Modèle

En effet, cette représentation (modélisation) permettra, par le biais de l'approche de spécification, de valider ou de contraindre les choix à prendre au niveau de l'interface de manière à préserver la structure de l'information et de son contrôle : le dialogue utilisateur (e.g. [Hix et al.87]) établis depuis le modèle des tâches. On parle dans ce cas d'un *procédé d'instanciation de l'interface* comme cela est utilisé dans les travaux de [Foley et al.91, op. cit.], même si les niveaux et objectifs de chacune des deux approches sont différents.

Nous définissons dans la figure 6 la structure générale des concepts de l'interface ainsi que leurs liens et rôles respectifs. Nous représentons en gras le concept central du modèle à savoir *l'Etat*. En effet, c'est à partir de ce concept que l'on décrira les éléments pertinents de l'interface permettant à l'utilisateur de réaliser le ou les objectifs sous-jacents à sa tâche (Cf Figure 4). Ainsi, nous définissons des *liens sémantiques* entre le concept d'Etat (§5.1) et celui de Tâche (§5.2). Ces liens seront explicités lors de la description de l'approche. Nous insistons ici sur les fonctions de chacune des entités ainsi que sur leurs liens, et ce par rapport à l'interface conceptuelle.

La figure représente en encadré les *deux principaux modules* du modèle. Le premier concerne les spécifications des procédures, des actions et des objets sur lesquels elles portent (la sémantique). Le second concerne les objets intermédiaires, en terme de l'interaction, qu'on associe à ces spécifications ; lesquels seront utilisés pour instancier un exemple d'interface graphique (la présentation), cela par le biais des composants d'une boîte à outils (Cf §3.4.3.3). En haut, sont représentés les concepts de *Schéma de Procédures et d'Evénement* ; ces derniers jouent un rôle prépondérant dans la

spécification en ce sens qu'ils contrôlent la dynamique de l'interface ainsi que la gestion des interruptions, par le biais du contexte global, quelles qu'elles soient leurs origines.

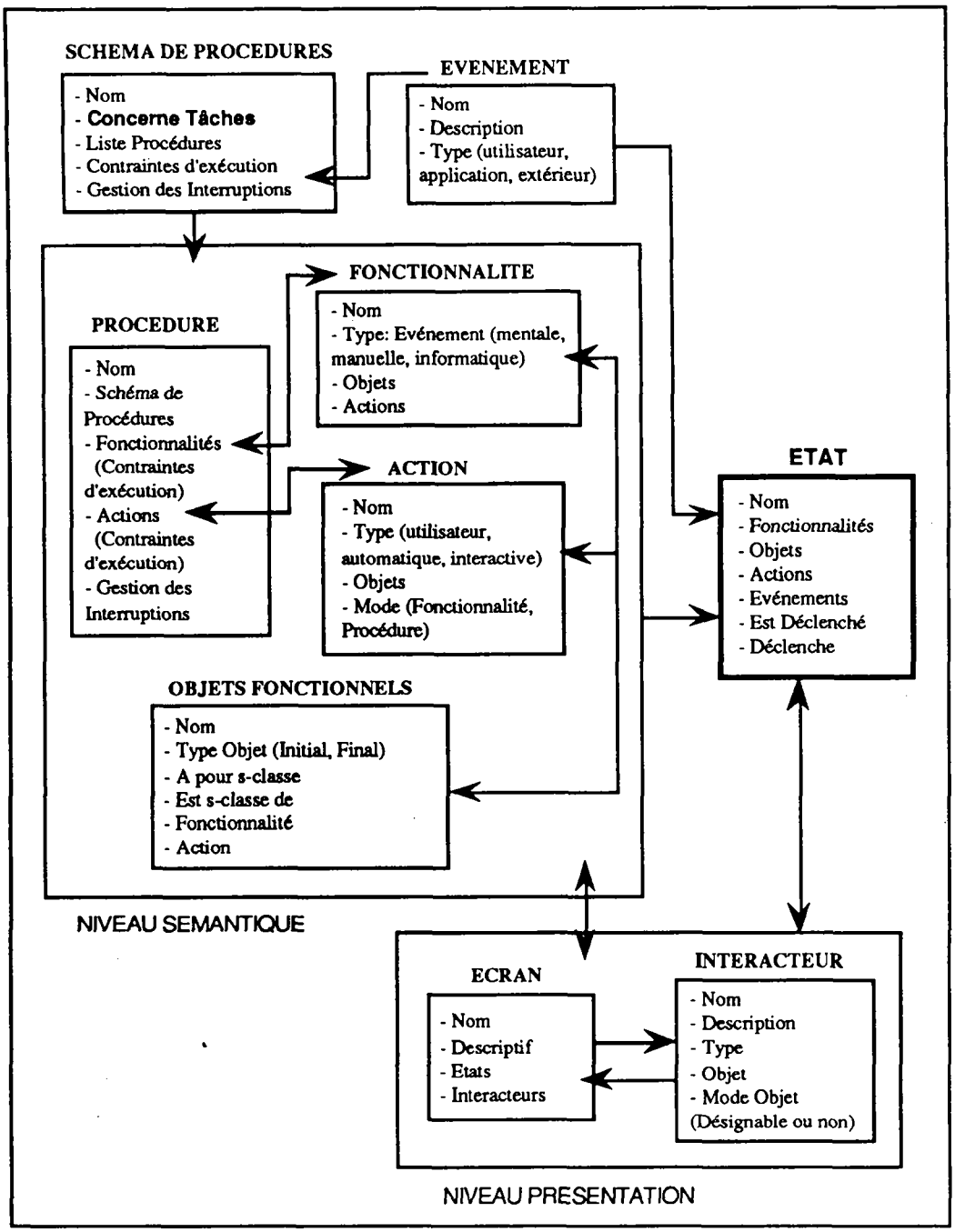

Figure 6: La Structure Conceptuelle de l'interface orientée-tâche.

Ainsi, l'entité *Evénement* modélise les différentes catégories d'événements intervenant lors de l'interaction de l'application avec l'utilisateur dans le cadre de la réalisation de sa tâche. L'interface réagit en effet selon le type des événements ; si celui ci est, par exemple, de type extérieur, l'interface devra informer l'utilisateur du type et du contenu de l'événement lui donnant le contrôle quant à son traitement.

Un Schéma de Procédures (ou stratégie) est associée à une tâche (ou un sous ensemble de tâches) au sens d'objectif-tâche (Cf §5.1.3) par l'attribut *Concerne Tâches*. Un schéma de procédures renferme les différents procédures permettant de réaliser la tâche. Par rapport à la modélisation des tâches (Figure 5), les procédures sont établies à partir des attributs "structure de la tâche et/ou pré- et post-conditions" ; ce qui correspond aux mécanismes de liaisons sémantiques entre la structure de la tâche et celle de l'interface conceptuelle. Ces derniers (les pré- et post-conditions) seront, entre autres, utilisés pour la spécification des fonctions de l'application permettant de confirmer ou d'infirmer les conditions *d'accès* et/ou *de mise oeuvre* de la tâche. En effet, les deux derniers attributs du schéma de procédures participent à la gestion de l'ordonnancement des procédures ainsi que d'éventuelles interruptions, i.e. le contexte global.

On dira qu'une fonction est *accessible* lorsque le contexte de la tâche le permet sauf que son utilisation nécessite la vérification et/ou la validation des conditions locales à la fonction, que ce soit par l'utilisateur ou par l'application. En outre, une fonction peut être *mise oeuvre* (ou utilisable) si toutes les conditions préalables, les contextes global et local (Cf Figure 7), sont établies. Par exemple, supposons qu'un utilisateur d'un Editeur soit en train d'effectuer une tâche donnée : soient la tâche "d'édition et impression d'un document". Dans ce cas, une fois le document ouvert, l'utilisateur peut à tout moment effectuer une impression car la fonctionnalité devient *utilisable* par défaut, à l'introduction (entrée) de quelques paramètres près. En parallèle, un message informe l'utilisateur de l'arrivée d'un "mail", il s'agit ici de ce que nous avons appelé *le contexte global* (e.g. gestion des événements), et qu'il veut le traiter immédiatement. A ce moment et par le biais de la gestion de contexte - rendant possible la prise en compte d'une telle situation, la fonction devient *accessible* mais pas utilisable car l'utilisateur doit changer de tâche, i.e. d'*objectif-tâche*, en effectuant des actions (e.g. post-conditions) préalables, ou que l'application doit effectuer (e.g. sauvegarde du document en cours, etc). Une fois l'*état-interface* (Cf §5.3.2.1) correspondant à l'objectif-tâche "gestion du mail" actif, la fonctionnalité traitement (e.g. lecture, et/ou rangement) du mail devient utilisable.

Une procédure renferme, à un niveau plus bas, les fonctionnalités et/ou actions relatives à un objectif-dialogue. Ces dernières concernent la réalisation d'objectifs plus élémentaires (objectif-utilisateur). De fait, la gestion des contextes locaux est effectué à ce niveau, cela par l'intermédiaire des attributs *Contraintes d'exécution* et *Gestion des interruptions*. En outre, ce sont les actions et les fonctionnalités qui portent directement sur les objets de l'application, appelés ici *Objets Fonctionnels*, de par les liens fonctionnels définis entre ces objets et les entités *Action* et *Fonctionnalité*. La nuance qu'on attribue à chacun de ces deux concepts a un effet sur le niveau d'évolution des

différents états de l'interface. En effet, une fonctionnalité regroupe des actions particulières en ce sens qu'elles portent sur la même classe d'objets. On pourrait l'assimiler à une Macro-action selon la terminologie de [Johnson et al.88, op. cit.], quoique les interprétations de chacune des deux notions par rapport à la future interface sont différentes. On remarquera par ailleurs l'attribut *Mode* dans l'entité Action indiquant si celle ci est attachée à une fonctionnalité ou une à procédure. Dans le premier cas un test sur la catégorie des objets impliqués est effectué.

Quant au *deuxième module* représenté en bas de la figure, il concerne un niveau plus proche de l'interface en ce sens qu'il définit deux concepts abstraits *Ecran* et *Interacteur* interprétables dans la génération d'interfaces. En outre, ce module associe aux différents concepts décrits plus haut, par le biais de l'entité Etat, en quelque sorte la façade apparente (interactive et/ou perceptible ; selon qu'on se place du point de vue l'ingénierie ou de l'ergonomie) des éléments qui interviendront à chaque état de l'interface : la présentation — au sens des modèles d'architecture (Cf §3.4).

Enfin, on dira qu'il s'agit bien ici d'une définition de l'interface aux niveaux conceptuel et sémantique. En effet, nous estimons que cette architecture a pour vocation de rendre compte de la "transparence" à mettre en oeuvre entre deux modes de fonctionnement qui sont l'application et l'utilisateur de manière à ce que les effets de chacun, i.e. de l'interaction, soient perceptibles en termes de l'utilisateur, le cas le plus sensible et souvent le moins pris en compte[27].

---

[27] Par ailleurs, nous donnons ici une esquisse d'une autre approche de formalisation concernant les définitions et le modèle décrits précédemment, et ce en vue de définir un formalisme logique, que nous appelons intermédiaire, décrivant de manière formelle et structurée - au sens de la spécification informatique - la sémantique de l'interface utilisateur. Nous énonçons ainsi une première version selon ce point de vue que nous affinerons éventuellement lors de son utilisation par la méthode. Notons tout de même, comme souligné dans le paragraphe précédent, qu'il y a deux niveaux de spécification: le premier en termes de l'application (sémantique), le second plus proche de l'interface (fonctionnel), i.e. la perceptibilité. L'association entre ces deux niveaux s'établit par le biais de liens entre les objets de l'application et ceux de l'interface (e.g. les boîtes à outils). La composition, au sens des langages à objets, des entités (e.g. Procédure) est implicite dans une telle représentation en ce sens qu'elle est incluse dans les attributs des entités en question.

<Task-Objective> —> <Procedure Schema> / <Procedure Schema> and-or <Procedure Schema>
<Procedure Schema> —> <Procedure> / <Procedure Schema> // <Display>
<Procedure> —> <Action> / <Functionality> / <Procedure> // <Dispaly>
<Functionality> —> <Action> (*) / <Functionality> // <State>
<Action> —> Call-function <Object, Event> // <State>
<Object> —> object / <Object> // <State>
<Event> —> event / <Event> // <State>
<Display> —> <State> <Interactor> / <Display>
<Interactor> —> Interface-Object (e.g. Zone de message, de données, etc) // <State>
<State> —> <Object> <Functionality> <Action> <Event>

(*): Les actions doivent dans ce cas porter sur la même catégorie d'objets.
(//): Signifie que la spécification concerne les entités State ou Display.
Task-Objective (ou Objectif-tâche) est lié, comme souligné plus haut, au modèle des tâches par les mécanismes définissant le schéma des procédures ; voir les mécanismes de spécification (§5.3.2).

5.3.1.2 Discussion

On remarque à travers cette modélisation *deux niveaux de structuration* de l'interface. Cette structuration coïncide en effet avec l'approche de spécification, que nous décrivons plus loin, laquelle aussi comporte deux principales étapes. Les étapes de l'approche ainsi que les mécanismes de mise en oeuvre des objets conceptuels de l'interface tels qu'ils sont représentés dans le premier module de la figure 6, concernent essentiellement la définition des différents objets (aspect statique) et de leurs relations sémantiques (aspect dynamique) lesquels nous utiliserons par la suite pour "instancier" un exemple d'interface en termes *d'écrans* relativement aux interactions procédurales, et en termes *d'interacteurs*, puis d'objets physiques (e.g. les menus), relativement aux interactions fonctionnelles.

Il faut noter en effet que lors d'une interaction fonctionnelle, c'est-à-dire de l'utilisation des fonctionnalités d'un état donné sans pour autant changer de schéma de procédures, i.e. d'objectif-tâche, cela revient à parcourir une ou plusieurs branches dans un même sous arbre concernant une ou plusieurs catégorie d'objets. Il ressort de là deux niveaux de structuration imbriqués (Figure 7).

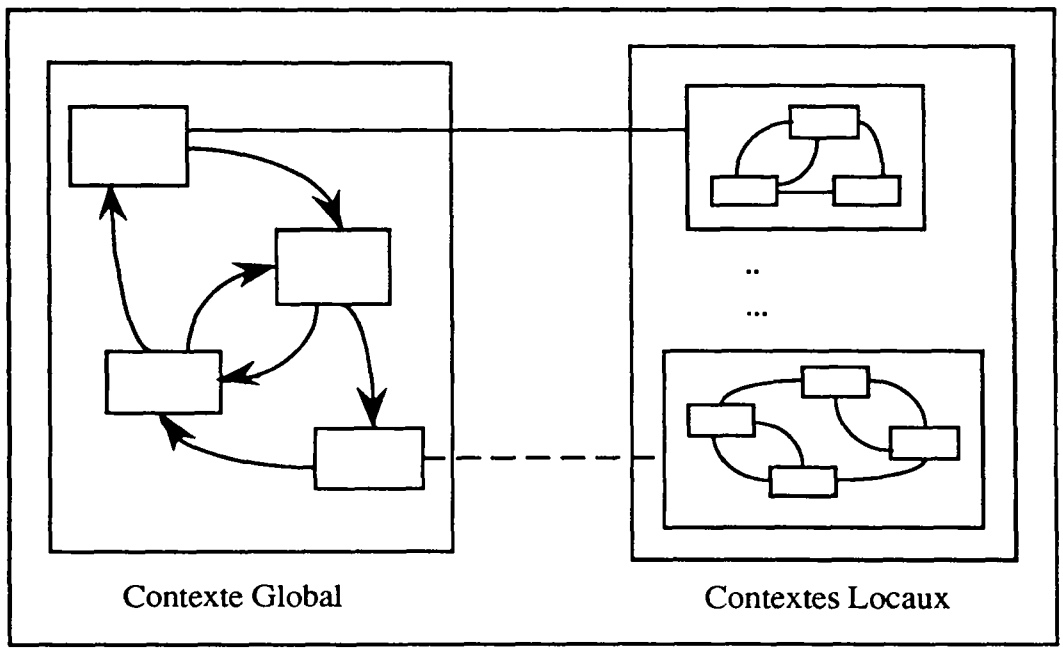

Figure 7: Spécification de l'interface conceptuelle par le biais des contextes.

Le premier concerne les *aspects procéduraux*, le second porte sur les *aspects fonctionnels*. On dira, par conséquent, que l'aspect procédural correspond à une tâche (un objectif-tâche) donnée alors que l'aspect fonctionnel correspond aux contenu et déroulement de cette tâche. De plus, le premier aspect sera utilisé pour la gestion de ce que nous avons appelé le "contexte global", à savoir les objectif-tâches, leurs relations

et de fait les événements intervenant à ce niveau (voir exemple plus haut). Le second, en outre, sera utilisé pour la gestion des "contextes locaux", à savoir les actions et/ou fonctionnalités telles que la définition de diagrammes d'enchaînements des actions et des fonctionnalités relatifs à une procédure ; la définition, par exemple, d'une mémoire locale pour les objets associés, et/ou communs, aux actions fréquemment utilisées ; la possibilité d'exécuter de manière *simultanée* deux actions lorsque celles ci mettent en oeuvre des objets différents ; etc.

5.3.2 Les Mécanismes de Spécification d'Interfaces : Règles Génériques et Heuristiques

Ce paragraphe introduit les mécanismes de mise en correspondance entre le modèle de la tâche et la structure conceptuelle de l'interface : les règles génériques et/ou les heuristiques de spécification issues d'une formalisation et/ou interprétation (traduction) des recommandations ergonomiques de haut niveau, i.e. ayant trait à la tâche utilisateur. Ces mécanismes nous permettront entre autres d'établir un lien entre le modèle de la tâche et le modèle de l'interface.

Les règles ont pour vocation d'établir un lien entre ces deux modèles en termes de prémisses et de conclusions. Les prémisses porteront sur les données de la tâche, les conclusions quant à elles permettront d'établir (ou générer) des spécifications (Cf exemple §5.3.2.3) ou simplement des contraintes textuelles de conception, cela quand les règles (ou le contexte) de spécification ne s'y appliquent pas.

Les heuristiques quant à elles se répartissent en deux catégories. La première consiste à identifier quelques concepts du modèle de l'interface, par le biais essentiellement de procédés (techniques) d'appariement (mapping), telles que les objets fonctionnels, leur structure, etc. La seconde catégorie concerne l'utilisation des règles, cela afin de compléter le structure conceptuelle de l'interface et particulièrement les aspects sémantiques. L'utilisation croisée de ces différents procédés est tout à fait envisageable sachant que le processus de conception d'interfaces est un processus itératif et non rigide. Ainsi, par exemple, elles seront utilisées pour l'identification des *objectifs-tâches* et leur dépendance ou indépendance (synchronisation), la structuration des objets en fonction des actions (ou inversement), etc.

5.3.2.1 Les Règles Génériques

L'utilisation des règles passe par une classification de celles ci en fonction des objectifs visés, i.e. la définition de la structure et de la sémantique de l'interface conceptuelle. En effet, nous estimons que cette approche modulaire favorise l'acquisition de nouvelles règles, cela en attribuant à chacune des classes de règles une fonction bien déterminée

dans le processus de spécification. Nous avons jusqu'à présent identifié trois classes de règles chacune ayant un rôle spécifique. Il s'agit des règles orientées-tâches, des règles de structure et des règles d'interaction.

Les règles orientées-tâches : Ces règles concernent l'identification des objectifs-tâches, c'est-à-dire les tâches (ou sous ensemble de tâches) utilisateur qui seront considérées comme des "tâches-interface" (ou états-interface) à part entière[28].

Les règles de structure : Ces règles permettront de définir à partir d'un objectif-tâche les différentes procédures concernées. Elles permettront par conséquent la définition de la structure de l'interface en termes d'états et/ou d'écrans.

Les règles d'interaction : Ces règles concernent l'exploitation des différentes relations fonctionnelles entre les actions et les objets afin de spécifier en terme d'interaction leurs effets par rapport aux différents états et/ou écrans.

5.3.2.2 Les Heuristiques

Comme souligné précédemment, des heuristiques d'appariement (de mapping) entre les modèles de la tâche et de l'interface seront utilisées pour définir et structurer des concepts tels que les objets et leurs pré- et post-conditions d'utilisation, et par voie de conséquence déduire et organiser les actions utilisateur (ou fonctions de l'application) attachées aux pré- et post-conditions.

D'autre part, d'autres heuristiques permettant la spécification des *points d'interruptions*, i.e. priorité entre les tâches, ainsi que la répartition des "rôles" entre l'utilisateur et l'interface seront définies en fonction des différents types d'événements rencontrés. Elles concernent particulièrement les informations (les attributs) concernant la structure de la tâche, son type (e.g. interactive) ainsi que son but (sa fonction).

5.3.2.3 Exemples de Règles et d'Heuristiques de Spécification

Nous donnons ici quelques exemples illustrant le format général de l'application des règles et des heuristiques. Ces exemples utilisent des instances de tâches de l'activité du contrôle aérien [El Farouki et al.91]. Par ailleurs, un exemple détaillé concernant l'activité d'une application concrète nous permettra dans un court terme d'expliciter et de valider l'utilisation des différents mécanismes décrits précédemment.

---

[28] Les heuristiques qui permettront d'inférer la notion d'"objectif-tâche" consistent à établir une relation bijective avec la notion d'indépendance entre les tâches. Cette notion d'indépendance peut s'obtenir soit par le calcul de la fréquence des tâches, soit en fonction du nombre d'actions et d'objets impliqués dans une tâche, soit enfin par le biais du constructeur "Parallèle" concernant les tâches situées aux plus hauts niveaux de l'arbre.

a) Format général des règles

SI (Préconditions = [Prédicat] ensemble d'objets)

ALORS (en fonction du type des opérations impliquées, soit mettre à la disposition de l'utilisateur les opérations associées, soit calculer (par l'application) les nouvelles valeurs des objets et les rendre accessibles à l'utilisateur). Cette spécification ne prend pas en compte le prédicat lequel est optionnel.

ALORS (en fonction du type d'événement associé au Prédicat, soit mettre à la disposition de l'utilisateur l'action ou la fonctionnalité en question, soit l'effectuer par l'application et informer l'utilisateur, si le résultat conditionne le bon déroulement de sa tâche).

Un exemple d'utilisation de cette règle:
Précond^tâche i = strips, instructions secteurs ...
ALORS (les objets et opérations impliqués (e.g. strips, liste strips éveils, etc) : doivent être accessibles et utilisables) ...

b) Format général des heuristiques

Soit la tâche "Planifier le trafic" (T0) parallèle à quatre autres tâches (T1, T2, T3, T4).

Exemple d'Heuristique
Les tâches (T0, T1, T2, T3, T4) seront définies chacune comme des "objectif-tâche".

Soit la tâche T01 (prendre en compte un avion) une sous-tâche de T0 qui se décompose en cinq tâches séquentielles.

Exemple d'Heuristique
Déclenchement des différentes procédures selon le schéma de procédures et les procédures établies ou à établir (selon le contexte), ainsi que les états/écrans (e.g. Zone d'affichage indiquant l'arrivée des strips au contrôleur) correspondants, avec comme événement déclencheur de *type application* : arrivée d'un strip.

5.3.3 Les Différentes Etapes de Spécification : Perspective

A travers la chronologie de description des différents modèles et mécanismes de spécification orientée-tâche se dessinent implicitement les différentes étapes de la méthode. Toutefois, nous les récapitulons ici de manière succincte et explicite et ce en ayant supposé effectuées les phases de description et de modélisation des tâches (§5.2). La première étape consiste à inférer au niveau d'un objectif-tâche donné la liste des procédures, les actions et/ou fonctionnalités impliquées ainsi que la structure des objets

concernés (aspect statique). Pour ce faire, les règles orientée-tâche et les règles de structure seront appliquées. Il sera ensuite question dans une deuxième étape de déterminer les procédures d'interaction (aspect dynamique) en termes d'utilisation. A ce niveau de spécification interviennent les règles d'interaction. Enfin, les règles et les heuristiques concernant les pré- et post-conditions seront utilisées pour attacher les opérations à mettre en oeuvre avant et après l'utilisation ou l'exécution des fonctionnalités (ou actions) de l'utilisateur et/ou de l'application.

Ces spécifications constitueront l'ensemble des fonctions et objets utilisés ou à utiliser par l'application (suite à des actions utilisateur, ou à un traitement interne) pour réaliser un objectif (e.g. objectif-dialogue) donné à un instant donné. Il arrive parfois qu'on ait des spécifications générales aux niveaux les plus hauts de l'arbre des tâches, celles ci s'instancient néanmoins (e.g. édition, impression, message) au fur et à mesure que l'on spécifie les fonctionnalités et/ou actions des tâches subalternes.

6 CONCLUSION ET PERSPECTIVES

Nous avons étudié et proposé, dans le cadre de ce rapport, une approche de mise en relation des modèles de description des tâches utilisateur, en l'occurrence le formalisme MAD, avec la spécification d'interfaces à un haut niveau d'abstraction. En effet, l'objectif de cette approche est d'établir, par le biais de mécanismes d'inférence (et d'appariements), les spécifications de l'interface de manière à permettre, par exemple, de générer le dialogue utilisateur en terme de maquette, constituant ainsi une aide à la conception d'interfaces qui va au delà des aides existantes telles que les guides de styles normalisant la structuration des composants de l'interface (e.g. les widgets Motif).

Pour ce faire, nous avons examiné les différents travaux et résultats en rapport avec la conception d'interfaces et/ou la modélisation orientée-tâche. Plusieurs domaines sont notamment concernés par ces travaux et impliquent de fait plusieurs perspectives d'études. Il s'agit des méthodes du génie logiciel dont l'objectif est de prendre en compte les aspects utilisateur au plutôt de la conception, et ce de manière à compléter les méthodes existantes ; des théories et modèles cognitifs dont la vocation est d'établir les représentations mentales des utilisateurs afin de comprendre et d'analyser leurs caractéristiques, et par conséquent d'expliquer les étapes intervenant lors de l'interaction homme-machine et/ou de prédire les performances et complexités d'utilisation ; des approches de l'ingénierie qui consistent à fournir des outils de construction d'interfaces en se basant sur le principe de séparation de la présentation du noyau fonctionnel ; des travaux en rapport avec les modèles et formalismes de description des tâches utilisateur en vue de fournir des outils d'analyse et/ou d'aide à la conception d'interfaces ; et enfin de la perspective de modélisation de la notion de tâche d'un point de vue de

l'intelligence artificielle, et donc des questions fondamentales liées à la représentation des connaissances à un niveau sémantique, et son intersection avec les modèles de description des tâches utilisateur.

Nous avons souligné dans le cadre de cette étude les apports et insuffisances de chacune de ces perspectives par rapport à la conception d'une application interactive en termes des objectifs (et besoins) des utilisateurs. En particulier, nous avons insisté sur les procédés et techniques utilisés de manière à voir les points communs ainsi que les relations entre les différentes approches et perspectives. De fait, nous avons étudié l'idée de mise en relation d'une approche de spécification ergonomique avec les outils de construction d'interfaces tels qu'ils sont définis dans les récents travaux de l'ingénierie.

Cela nous a conduit à introduire et définir la notion d'interface conceptuelle, en ce sens qu'elle est décrite par le biais de concepts abstraits, d'une part de manière à situer l'interface à un haut niveau d'abstraction (à la frontière de l'application et de l'interface au sens de l'ingénierie), et d'autre part afin d'établir des liaisons sémantiques entre la tâche utilisateur et l'interface conceptuelle en question, et ce par le biais de l'approche de spécification d'interfaces ergonomiques.

Ces liaisons (mécanismes) concernent les heuristiques et les règles ergonomiques de haut niveau, c'est-à-dire en rapport avec les caractéristiques de la tâche. Dans cette optique, nous avons établi une modélisation objet des descripteurs de la tâche permettant d'analyser et de manipuler des configurations de tâches complètes et cohérentes, et par conséquent de calculer et d'organiser les structures (e.g. les hiérarchies d'actions et d'objets) de la tâche utiles à la spécification. Ce modèle intermédiaire, entre les phases d'analyse et de conception, permet d'identifier la structure et la sémantique de l'interface conceptuelle des points de vue de *l'utilisation* et de *l'interaction* ; ce qui n'est pas incompatible, mais complémentaire, avec les spécifications de l'application (au sens informatique) puisqu'il s'agit bien ici de l'étape des objectifs et donc des besoins informationnels entre autres.

En outre, une implémentation, à l'aide d'objets structurés, de la modélisation conceptuelle des tâches est en cours. Celle ci a pour objectifs, d'une part, de permettre, via une interface graphique, la description des tâches utilisateur de manière incrémentale et interactive ; cela en utilisant les composants d'une boîte à outils. Et d'autre part, d'établir des hiérarchies d'objets structurés de manière à vérifier la cohérence et la complétude de celles ci, puis à appliquer les différentes techniques (e.g. calcul des conditions associés aux actions), permettant d'établir les premières spécifications de l'interface conceptuelle.

Concernant les perspectives futures à entreprendre en vue d'aboutir à une approche formelle de spécification de l'interface à partir de la tâche, des efforts restent à faire, notamment afin de formaliser les représentations et manipulations des mécanismes de spécification ; de préciser et détailler la grammaire décrivant la structure de l'interface conceptuelle, d'expliciter et affiner les différentes étapes de l'approche, et enfin de définir une technique rigoureuse de mise en correspondance entre les concepts abstraits et les concepts concrets (les composants) tels qu'ils sont utilisés dans les environnements de construction d'interfaces.

Toutefois, nous espérons, dans un court terme, travailler sur un exemple concret d'application avec l'objectif d'analyser la structure de tâche, et surtout d'établir et de valider la structure conceptuelle de l'interface. Ceci consiste à définir les spécifications en termes de procédures, d'actions, d'objets, etc. Enfin, la dernière phase concerne la définition de la répartition, par le biais de la gestion des événements, du contrôle (la dynamique) entre l'utilisateur et l'application. En outre, cet exemple fera sans doute l'objet de l'implémentation d'un prototype de système expert dont le but principal est de valider la faisabilité de l'approche de spécification d'interfaces orientées-tâches.

BIBLIOGRAPHIE

Abed, M., Bernard, J.-M., & Angué, J.-C. (1991). Méthodologie d'Analyse et de Modélisation de l'Interaction Homme-Machine avec des Outils de Spécification. *Journées CENA*, 3-4 Décembre, Paris : CENA/92-065.

Aida/Masai (1992). *An Automatic Generator of Graphical Interfces*. Reference Manual, Version 1.2. Ilog S.A., Paris.

Anderson, J. R. (1983). *The Architecture of Cognition*. Cambridge, MA : Harward University Press.

Arch (1992). The UIMS Workshop Developers : A Metamodel for Runtime Architecture of an Interactive System. *SIGCHI Bulletin, 24, 1*, 32-37 (January 1992).

Barthe, M. (1992). Ergo-Meth : Principes d'une Méthodologie d'Informatisation visant à intégrer naturellement les Apports de l'Ergonomie dans la Démarche de Conception de Logiciels Interactifs de Gestion. *Actes du Colloque ERGO-IA'92*, 334-353, Biarritz.

Barthet, M.-F. (1986). *Conception d'Applications Conversationnelles Adaptées à l'Utilisateur*. Thèse d'Etat, Institut National Polytechnique de Toulouse, France.

Barthet, M.-F. (1988). *Logiciels Interactifs et Ergonomie, Modèles et Méthodes d'Application*. Paris : Dunod.

Beaudoin-Lafon, M. (1991). Interfaces Homme-Machine : Vue d'Ensemble et Perspectives. *Actes du congrès Génie logiciel & Systèmes experts, Interfaces homme-machine, Maquettage & Prototypage, 24*, 4-16, France.

Bellotti, V. M. E. (1988). Implications of the Current Design Practice for the Use in HCI Techniques. In D.M. Jones and R. Winder (Eds.), *People and Computers*, (pp. 13-34), London : Cambridge University Press.

Benett, J. L. (1987). Collaborations of UIMS Designers and Human Factors Specialists. *Proceedings of ACM/SIGGRAPH'87*. In Computers Graphics *21, 2* (April 1987), 102-105.

Benyon, D. (1992). The Role of Task Analysis in System Design. *Interacting with Computers, 4 (1)*, 102-123.

Beringer, J., & Wandmacher, J. (1991). Object-Based Action Planning. In D. Ackermann and M. J. Tauber (Eds.), *Mental Models and Human-Computer Interaction 2*, (pp. 135-155), North-Holland : Elsevier Science Publishers.

Bisseret, A. (1986). *Psychologie pour la Conception Ergonomique de l'Assistance Informatique*. Cours et Séminaires INRIA : Les nouveaux outils du spécialiste de l'information, France.

Bovair, S., Kieras, D. E., & Polson, P. G. (1990). The Acquisition and Performance of Text-Editing Skill : A Cognitive Complexity Analysis. *Human Computer Interaction, 5*, 1-48.

Boy, G. (1992). Méthodologies et Outils pour l'Interaction Cognitive Homme-Machine. Thèse d'Habilitation à diriger des Recherches. Université Pierre et Marie Curie, Paris VI, France.

Braudes, (1991). Conceptual Modelling : A look at System-Level User Interface Issues. In J. Karat (Ed.), *Taking Software Design Seriously* (pp.195-207), San Diago : Academic Press Inc.

Brisson, G., Drouin, A., & Faveaux, L. (1990). Le Maquettage dans une Démarche Ergonomique de Conception d'Interfaces Utilisateur. *Actes du Congrès Ergo-IA'90*, France.

Browne, D. P., Sharrat B. D., & Norman, M. A. (1986). The Formal Specification of Adaptative User Interfaces using CLG. In M. Mantei, & P. Orbeton (Eds.), *Human Factors in Computing Systems, CHI'86 Conference Proceedings* (pp. 256-261), NY : ACM.

Bruder, R., & Rohmert, W. (1991). Knowledge-Based Systems : Object and Method of Ergonomic Research. In Y., Quinnec, & F. Daniellou, *Designing for Evryone, Proceedings of the 11th Congress of IEA* (pp. 610-613), France.

Cameron, J. R. (1986). An Overview of JSD. *IEEE Transaction on Software Engineering, Vol. SE-12, 2, February, 222-240.*

Card, S. K., Moran, T. P., & Newell, A. (1983). *The Psychology of Human-Computer Interaction*. London : Lawrence Erlbaum.

Carey, T. T., & Graham, C. H. (1987). Modular Specification for User Interface. In H.-J. Bullinger and B. Shackel (Eds.), *HCI-Interact'87* (pp. 403-408), North-Holland : Elsevier Science Publishers.

Carter, J. A. (1990a). The Dimensions and Degrees of Adaptations: A Synergistic Analysis. *Proceedings of the 34th Annual Meeting Human Factors Society, 1*, 336-340, Santa Monica : The Human Factors Society.

Carter, J. A. (1990b). Juggling for Completeness and Consistency with concerns for Flexibility and Adaptability using MOST. *Proceedings of the 34th Annual Meeting Human Factors Society, 1*, 341-345, Santa Monica : The Human Factors Society.

Carter, J. A. (1991). Combining Task Analysis with Software Engineering in a Methodology for Designing Interactive Systems. In J. Karat (Ed.), *Taking Software Design Seriously* (pp. 209-235), San Diago : Academic Press Inc.

Carter, J. A., & Schweighardt, M. (1987). The Basis for User-oriented, Context Sensitive Functions. In H.-J. Bullinger and B. Shackel (Eds.), *HCI-Interact'87* (pp. 1027-1032), North-Holland : Elsevier Science Publishers.

Casale, A., & Czerwinski, C. (1992). Formentor : Interface Générique et Ergonomie. *Actes du Colloque ERGO-IA'92*, 354-366, Biarritz.

Cournarie, E. (1991). Contraintes et Prototypes dans les Architectures IHM. *3ème Journées sur l'ngénierie des Interfaces Homme-Machine*, Dourdan, France.

Coutaz, J. (1990). *Interfaces Homme-Ordinateur, conception et réalisation.* Paris : Dunod.

Coutaz, J. (1991). Architectural Design for User Interface. *Proceedings ESEC'91, 3rd European Software Engineering, 7-22*, Milan, Italy.

De Baar, D. J., Foley, J. D., & Mullet, K. E. (1992). Coupling Application Design and User Interface Design. In P. Bauersfeld, J. Bennett, & G. Lynch (Eds.), *CHI'92 Conference Proceedings* (pp. 259-266). Reading, MA : Addison-Wesley.

De Haan, G., Van der Veer, G. C., & Van Vliet, J. C. (1991). Formal Modelling Techniques in Human-Computer Interaction. *Acta Psychologica 78*, 27-67, North-Holland.

Delsol, P. (1992). Description des Tâches du Contrôle Aérien. *Rapport de contrat intermédiaire INRIA-CENA* 91/C.007, France.

Diaper, D et al. (1989). *Task Analysis for Human Computer Interaction.* England : Ellis Harwood Limited.

Diaper, D. (1990). Analysing Focused Interview Data with Task Analysis for Knowledge Descriptions (TAKD). In Diaper et al. (Eds.), *HCI-Interact'90* (pp. 277-282), North-Holland : Elsevier Science Publishers.

Diaper, D., & Addison, M. (1992). Task Analysis and Systems Analysis for Software Development. *Interacting with Computers, 4 (1)*, 124-139.

Draper, S. W., & Norman, D. A. (1986). *User Centred System Design.* Hillsdale, NJ : Lawrence Erlbaum Associates.

Dzida, W., Freitag, R., Hoffman, C., & Valder, W. (1990). Bridging the Gap between Task Design and Interface Design. In Diaper et al. (Eds.), *HCI-Interact'90* (pp. 239-245), North-Holland : Elsevier Science Publishers.

El Farouki, L., Scapin, D. L., & Sebillotte, S. (1991). Prise en compte des Tâches du Contrôle Aérien pour l'Ergonomie des Interfaces. *Rapport de fin de contrat INRIA-CENA*, France.

El Mrabet, H. (1991). *Outils de Génération d'Interfaces: Etat de l'Art et Classification.* Rapport Technique 126, France : INRIA.

Ferber, J. (1989). *Objets et Agents : Une Etude des Structures de Représentation et de Communications en Intelligence Artificielle.* Thèse d'Etat, Université de Paris VI, Laboratoire Formes et Intelligence Artificielle, France.

Foley, J. D., Gibbs, C., Kim W., & Kovalevic, S. (1988). A Knowledge-based User Interface Management System. In E. Soloway, D. Frye, & S. B. Sheppard (Eds.), *CHI'88 Conference Proceedings* (pp. 67-72). Reading, MA : Addison-Wesley.

Foley, J. D., & Van Dam, A. (1984). *Fundamentals of Interactive Computer Graphics.* Addison-Wesley.

Foley, J. D., Kim, W. C., & Gibbs, C. A. (1987). Algorithms to Transform the Formal Spécification of a User-Computer Interface. In H.-J. Bullinger and B. Shackel (Eds.), *HCI-Interact'87*, 1001-1006.

Foley, J. D., Kim, W. C., Kovacevic, S., & Murray, K. (1991). UIDE : An Intelligent User Interface Design. In J. Sullivan, & S. Tyler (Eds.), *Architectures for intelligent Interfaces : Elements and Prototypes*, (pp. 339-385), Reading, MA : Addison-Wesley.

Gibbs, C., Kim, W. C., & Foley, J. D. (1986). Case Studies in the Use of IDL : Interface Definition Language. *Report GWU-IIST-86-30, Dep. of EE&CS, G. Washington University*, Washington, DC 20052.

Gomes, M. et al. (1990). Toolkits, Environments and the Object Oriented Paradigm. In D. A. Duce et al. (Eds.), *Proceedings of the workshop on User Management Systems and Environments*, 7, Berlin : Springer-Verlag.

Gould, J. D., & Lewis, C. (1983). Designing for Usability : Key Principles and What Designers think. *Proceedings of the CHI'83 Conference on Human Factors in Computing Systems* (pp. 50-53), NY : ACM.

Green, M. (1985). Report on Dialogue Specification Tools. In G.E. Pfaff (Ed.), *User Interface Management Systems*, Amsterdam : Springer-Verlag.

Green, M. (1987). A Survey of three Dialogue Models. *ACM transactions on Graphics, 5 (3)*, 244-275.

Green, T. R. G., Schiele, F., & Payne, S. J. (1988). Formalisable Models of User Knowledge in Human-Computer Interaction. In G. C. Van Der Veer et al. (Eds.), *Working with Computers : Theory versus Outcome*, (pp. 3-46), London : Academic Press.

Hamalainen, M., & Holsapple, C. W. (1991). User Interface Design Principles for Team Support Systems. *Proceedings of the 24th Annual Hawaï International, IEEE Conference on System Sciences*, 461-470.

Harrison, M. D., & Abwod, G. D. (1991). Formal Methods in Human Computer Interaction. *Tutorial presented at CHI'91 : Conference on Human Factors in Computing Systems*, New Orleans, Louisiana, 28 Avril-2 Mai.

Hartson, H. R., & Hix, D. (1989). Human-Computer Interface Development : Concepts and Systems for its Management. *ACM Computing Surveys, 21 (1)*, 1-92.

Hartson, R. (1989). User-Interface Management Control and Communication. *IEEE Software, January*, 62-70.

Hix, D., & Hartson, H. R. (1987). A Structural Model for Hierarchically Describing Human-Computer Dialogue. In H.-J. Bullinger and B. Shackel (Eds.), *HCI-Interact'87*, 695-700, North-Holland : Elsevier Science Publishers.

Hill, R. D., & Hermann, M. (1990). The Composite Object User Interface Architecture. In D. A. Duce et al. (Eds.), *Proceedings of the workshop on User Management Systems and Environments, 24*, Berlin : Springer-Verlag.

Hoppe, H. U. (1991). A Grammar Approach to Unifying Task-Oriented and System-Oriented Interface Descriptions. In D. Ackermann and M. J. Tauber (Eds.), *Mental Models and Human-Computer Interaction 2*, (pp. 354-373), North-Holland : Elsevier Science Publishers.

Hudson, S.E. (1987). UIMS Support for Direct Manipulation Interfaces. *Proceedings of ACM/SIGGRAPH'87.* In Computers Graphics *21, 2* (April 1987), 120-124.

Hullot, J. M. (1986). SOS interface : Un Générateur d'Interfaces Homme-Machine. *Actes des Journées Afcet-Informatique, les Langages orientés objet*, Bigre+Globule, 69-78, France.

Hurley, W. H., & Sibert, J. L. (1989). Modelling User Interface-Application Interactions. *IEEE Software, January*, 71-77.

IGL (1989). *SADT : Un Langage pour Communiquer.* I.G.L. Technology, Paris : Eyrolles.

John, B. E., Vera, A. H., & Newell, A. (1990). Towards Real-Time GOMS. *Report CMU-CS-90-195, School of Computer Science*, Canergie Mellon University Pittsbourg.

Johnson, P., Johnson, H., Waddington, R., & Shouls, A. (1988). Task related Knowledge Structures : Analysis, Modelling and Application. In D.M. Jones and R. Winder (Eds.), *People and Computers*, (pp. 137-155), London : Cambridge University Press.

Johnson, H., & Johnson, P. (1989a). Integrating Task Analysis into System Design : Surveying Designers'needs. *Ergonomics Special Issue.*

Johnson, P., & Johnson, H. (1989b). Knowledge Analysis of Tasks : Task Analysis and Specification for Human-Computer Systems, In A. Downton (Ed.), *Engineering the Human-Computer Interface*, London : Mc Graw Hill.

Johnson, P., & Nicolsi, E. (1990a). Task-Based User Interface Development Tools. In Diaper et al. (Eds.), *HCI-Interact'90* (pp. 383-387), Holland : Elsevier Science Publishers.

Johnson, P., Drake, K., & Wilson, S. (1990b). A Framework for integrating UIMS and User Task Models in the Design of User Interface. In D. A. Duce et al. (Eds.), *Proceedings of the workshop on User Management Systems and Environments, 20*, Berlin : Springer-Verlag.

Johnson, H., & Johnson, P. (1991). Task Knowledge Structures : Psychological Basis and Integrated into System Design. *Acta Psychologica 78*, 3-26, North-Holland.

Johnson, P. (1991). User Interaction : A Framework to relate Tasks, Users, and Design. In H.-J. Bullinger (Ed.), *HCI'91*, Stuttgart : Elsevier Science Publishers.

Johnson, P., Markopoulos, P., & Johnson, H. (1992). Task Knowledge Structures : A Specification of User Task Models and Interaction Dialogues. *Proceedings of Task Analysis in Human Computer-Interaction, 11th Interdisciplinary Workshop on "Informatics and Psychology"*, June 9-11, Schaerding, Austria.

Karsenty, S. (1991). La Construction d'Interfaces Utilisateurs. *Actes du Congrés Génie logiciel & Systèmes experts, Interfaces homme-machine, Maquettage & Prototypage, 24*, 18-27, France.

Kieras, D., & Polson, P. (1985). An Approach to the Formal Analysis of User Complexity. *International Journal of Man-Machine Studies, 22*, 365-394.

Kolski, C., & Millot, P. (1991). A Rule-based Approach to Ergonomic "Static" Evaluation of Man-Machine Graphic Interface in Industrial Processess. *Int. Journal of Man-Machine Studies, 35*, 657-674.

Kool (1990). Introduction à Kool. Version Kool V2, Bull S.A., Cedoc-Dialog, BP 110 Val de Reuil cedex, France.

Laird, J. E., Newell, A., & Rosenblom, P. S. (1987). SOAR : An Architecture for General Intelligence. *Artificial Intelligence, 33*, 1-64.

Lee, J. (1990). Intelligent Interface and UIMS. In D. A. Duce et al. (Eds.), *Proceedings of the workshop on User Management Systems and Environments, 14*, Berlin : Springer-Verlag.

Lim, K. Y., Long, J. B., & Silcock, N. (1990). Integrating Human Factors with Structured Analysis and Design Methods : An Enhanced Conception of the Extended Jackson System Development Method. In Diaper et al. (Eds.), *HCI-Interact'90* (pp. 225-230), North-Holland : Elsevier Science Publishers.

Lim, K. Y., & Long, J. B. (1992). A Method for (Recruiting) Methods Facilitating Human Factors Input to System Design. *CHI'92 Conference Proceedings* (pp. 549-556), Reading, MA : Addison-Wesley.

Luh, P. B., Pattipati, K. R., & Kleinman, D. L. (1990). Task and Resource Coordination in Human Teams. *Proceedings of the Fifth IEEE International Symposium on Intelligent Control*, (pp. 113-119), Sept. 1990, Philadelphia.

Mahling, D. E., Coury, B. G., & Croft, W. B. (1990). User Models in Cooperative Task-Oriented Environments. *Proceedings of the 23th Annua Hawaï International, IEEE Conference on System Sciences*, 94-100.

Masini, G., Napoli, A., Colnet, D., Léonard, D., & Tombre, K. (1989). *Les Langages à Objets*. Edition. Paris : IIA.

Mc Grew, J. F. (1991). Tools for Tasks Analysis : Graphs and Matrices. In J. Karat, (Ed.), *Taking Software Design Seriously* (pp. 287-314), San Diago : Academic Press Inc.

McDonald, J., Dearholt, D., Peap, K., & Schvaneveldt, R. (1986). A Formal Interface Design Methodology Based on User Knowledge. In M. Mantei, & P. Orbeton (Eds.), *Human Factors in Computing Systems, CHI'86 Conference Proceedings* (pp. 285-290), NY : ACM.

Moran, T. P. (1981). The Command Language Grammar : A Representation for the User Interface of Interactive Computer Systems. *International Journal of Man-Machine Studies, 15*, 3-50.

Mosier, J. N., & Smith, S. L. (1986). Application of Guidelines for Designing Interface Software. *Behaviour & information technology, 5 (1)*, 39-46.

Moussa, F., & Kolski, C. (1992). Vers une Formalisation d'une Démarche de Conception de Synoptiques Industriels : Application au Système Ergo-Conceptor. *Actes du Colloque ERGO-IA'92*, 209-221, Biarritz.

Myers, B. (1989). User Interface Tools : Introduction and Survey. *IEEE Software, January*, 15-23.

Myers, B. (1991). Using AI Techniques to Create User Interfaces by Example. In Sullivan, J. and Tyler, S. (Eds.), *Architectures for intelligent Interfaces : Elements and Prototypes* (pp. 386-401), Reading, MA : Addison-Wesley.

Nanard, J. (1990). *La Manipulation Directe en Interface Homme-Machine*. Thèse d'Etat, Université Montpellier II, Sciences et Techniques de Languedoc, France.

Newell, A., & Simon, H. A. (1972). *Human Problem Solving*. Englewood Cliffs, NJ : Prentice Hall.

Nilsson, N. J. (1982). *Principles of Artificial Intelligence*. NY : Springer-Verlag.

Norman, D. A. (1986). Cognitive Engineering. In Draper, C.W. and Norman, D.A. (Eds.), *User Centered System Design*, Hillsdale, NJ : Lawrence Erlbaum.

Normand, V. (1992). *Le Modèle SIROCO : De la Spécification Conceptuelle des Interfaces Utilisateur à leur Réalisation.* Thèse de Doctorat 3ème cycle de l'Université Joseph Fourier, Grenoble I, France.

Olsen, D. (1989). A Programming Language Basis for User Interface Management. In K. Bice & C. Lewis (Eds.), *CHI'89 Proceedings of the Conference on human factors in computing systems* (pp. 171-176), Reading, MA : Addison-Wesley.

OSF (1990). Osf/Motif Style Guide. *Programmer's Reference Manual. Open Software Foundation,* Eleven Cambridge Centre, MA : Cambridge.

Payne, S. I., & Green, T. R. G. (1986). Task-Action Grammar : A Model of Mental Representations of Task Languages. In Diaper, D. (Ed.), *Task Analysis for Human Computer Interaction* (pp. 75-105), London : Ellis Harwood Limited.

Petoud, I. (1990). *Génération Automatique de l'Interface Homme-Machine d'une Application de Gestion Hautement Interactive.* Thèse de Docteur en Sciences Economiques, Université de Lausanne, Suisse.

Pfaff, G. (1985). *User Interface Management Systems.* Proceedings of the Workshop on UIMS held in Seeheim, November 1-3, Edited by G. E. Pfaff.

Pierret, C. (1991). *Acquisition des Connaissances orientée Modèles de Tâches.* Rapport de Recherche LRI 694, Orsay, France.

Pierret-Golbreich, C., Delouis, I., & Scapin, D. L. (1989). *Un Outil d'Acquisition et de Représentation des Tâches Orienté-Objet.* Rapport de Recherche 1063, France : INRIA.

Pitrat, J. (1990). *Métaconnaissances : Futur de l'intelligence artificielle.* Paris : Hermès.

Poitrenaud, S., Richard, J. F., & Tijus, C. A. (1990). An Object-oriented description for Evaluation of Interfaces. *Proceedings of the Fifth European Conference on Cognitive Ergonomics* (pp. 431-440), Urbino, Italy.

Rechenmann, F. (1988). *Shirka : Système de Gestion de Bases de Connaissances Centrées Objets, Manuel d'utilisation.* France : INRIA.

Reisner, P. (1981). Formal Grammar and Human Factors Design of Interactive Graphic System. *IEEE Transactions on Software Engineering,* SE-7/2, 229-240.

Richard, J.-F. (1983). *Logique d'Utilisation et Logique de Fonctionnement.* Rapport de Recherche 202, France : INRIA.

Sacerdoti, E. D. (1977). *A Struture for Plans and Behavior.* NY : Elsevier Computer Library.

Scapin, D. L. (1986). *Guide Ergonomique de Conception d'Interfaces Homme-Machine.* Rapport de Recherche 77, France : INRIA.

Scapin, D. L., & Pierret-Golbreich, C. (1989). MAD : Une Méthode Analytique de Description de Tâches. *Actes du Colloque sur l'ingénierie des Interfaces Homme-Machine,* Sophia-Antipolis, France.

Scapin, D. L., Reynard, P., & Pollier, A. (1988). *La Conception Ergonomique d'Interfaces: Problèmes de Méthodes.* Rapport de Recherche 952, INRIA, France.

Scheifler, R. W., & Gettys, J. (1986). The Xwindow System. *ACM transactions on Graphics, 5 (1),* 79-109.

Schmucker, K. J. (1986). MacApp : An Application Framework. *Byte 11(8),* 189-193.

Schweighardt, M. F. (1990). Using the Context of Interactions to Adapt to Users. *Proceedings of the 34th Annual Meeting Human Factors Society, 1,* 346-350, Santa Monica : The Human Factors Society.

Seaton, P., & Stewart, T. (1992). Evolving Task oriented Systems. In P. Bauersfeld, J. Bennett, & G. Lynch (Eds.), *CHI'92 Conference Proceedings* (pp. 463-469). Reading, MA : Addison-Wesley.

Sebillotte, S. (1991). *Décrire des Tâches selon les Objectifs des Opérateurs : De l'Interview à la Formalisation.* Rapport Technique 125, France : INRIA.

Senach, B. (1991). *Evaluation Ergonomique des Interfaces Homme-Machine : Une Revue de la Littérature.* Rapport de Recherche 1180, Sophia Antipolis, France : INRIA.

Shephered, A. (1989). Analysis and Training in Information Technology Tasks. In Diaper, D. (Ed.), *Task Analysis for Human Computer Interaction* (pp. 15-54), London : Ellis Harwood Limited.

Shevlin, F., & Neelamkavil, F. (1990). Designing the Next Generation of UIMSs. In D. A. Duce et al. (Eds.), *Proceedings of the workshop on User Management Systems and Environments, 13,* Berlin : Springer-Verlag.

Shneiderman, B. (1987). *Designing the User Interface : Strategies for Effective Human-Computer Interaction.* Reading, MA : Addison-Wesley.

Simon, T. (1988). Analysing the Scope of Cognitive Models in Human-Computer Interaction : A Trade-off Approach. In D.M. Jones and R. Winder (Eds.), *People and Computers*, (pp. 79-93), London : Cambridge University Press.

Siochi, A. C., Hix, D., & Hartson, H. R. (1991). The UAN : A Notation to Support User-Centered Design of Direct Manipulation Interfaces. In J. Karat (Ed.), *Taking Software Design Seriously* (pp.157-194), San Diago : Academic Press Inc.

Smeci (1991). *Générateur de Systèmes Experts*. Manuel de Référence, Version 1.5, Ilog S.A., Paris.

Smith S. L. (1986). Standards versus Guidelines for Designing User Interfaces Software. *Behaviour & information technology, 5 (1)*, 47-62.

Smith, S. L., & Mosier, J. N. (1986). *Guidelines for Designing User Interface Software* (Rapport ESD-TR-86-278). Bedford, Massachusetts : Mitre.

Swell, D. R., & Gedds, N. D. (1990). A Plan & Goal Based Method for Computer-Human System Design. In Diaper et al. (Eds.), *HCI-Interact'90* (pp. 283-288), North-Holland : Elsevier Science Publishers.

Tanaka, T., Eberts, R. E., & Salvendy, G. (1990). Derivation and Validation of a Quantitative Method for the Analysis of Consistency for Interface Design. *Proceedings of the 34th Annual Meeting Human Factors Society, 1*, 329-333, Santa Monica : The Human Factors Society.

Tarby, J. C., & Barthet, M.-F. (1991). Production d'Interfaces : Vers la Génération Automatique du Contrôleur de Dialogue. *3ème Journées sur l'ngénierie des Interfaces Homme-Machine*, Dourdan, France.

Tardieu, H., Rochfeld, A., Colletti, R. (1986). *La Méthode Merise : Principes et Outils (Tome1)*. Paris : Les Editions d'Organisation.

Tauber, M. (1990). ETAG : Extended Task Action Grammar - A Language for the Description of the User's Task Language. In Diaper et al. (Eds.), *HCI-Interact'90* (pp. 163-168), North-Holland : Elsevier Science Publishers.

Ten Hagen, P. J. W. (1990). Critique of the Seeheim Model. In D. A. Duce et al. (Eds.), *Proceedings of the workshop on User Management Systems and Environments, 1*, Berlin : Springer-Verlag.

Tong, X. (1990). *Carmen : Plateforme de Construction de Systèmes Experts de Seconde Génération*. Thèse de Doctorat 3ème cycle de l'Université de Paris Sud, Orsay, France.

Totterdell, P. A., Norman, M. A., & Browne, D. P. (1987). Levels of Adaptativity in User Interface Design. In H.-J. Bullinger and B. Shackel (Eds.), *HCI-Interact'87*, 715-722, North-Holland : Elsevier Science Publishers.

Vanderdonct, J. M., & Bodart, F. (1993). Encapsulating Knowledge For Intelligent Automatic Interaction Object Selection. *INTERCHI'93 Conference Proceedings* (pp. 424-429). Amsterdam: ACM.

Vander Zanden, B., & Myers, B. A. (1990). Automatic Look-and-Feel Independent Dialog Creation for Graphical User Interfaces. In J. Carrasco, & J. Whiteside (Eds.), *CHI'90 Conference Proceedings* (pp. 27-34). Reading, MA : Addison-Wesley.

Walsh, P. (1989). Analysis for Task Object Modelling : Towards a Method of Integrating Analysis with Jackson System Development for User Interface Software Design. In *Task Analysis for Human Computer interaction* (pp. 186-209), England : Ellis Harwood Limited.

Walsh, P. A., Lim, K. Y., Long, J. B., & Carver, M. (1988). Integrating Human Factors with System Development. In N. Heaton and M. Sinclair (Eds.), *Designing end-User Interfaces*, Oxford : Pergamon Info Tech.

Walsh, P. A., Lim, K. Y., Long, J. B., & Caver, M. K. (1989). JSD and the Design of User Interface Software. *Ergonomics, 32, 11*, 1483-1498.

Wiecha, C., Benett, W., Bores, S., & Gould, J. (1989). Generating Highly Interactive User Interface. In K. Bice & C. Lewis (Eds.), *CHI'89 Proceedings of the Conference on human factors in computing systems* (pp. 277-282), Reading, MA : Addison-Wesley.

Wilson, M. D., Bernard, P. J., Green, T. R. G., & Maclean, A. (1988). Knowledge-Based task Analysis for Human-Computer Interaction. In G. C. Van Der Veer et al. (Eds.), *Working with Computers : Theory versus Outcome*, (pp. 47-88), London : Academic Press.

Unité de Recherche INRIA Rocquencourt
Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)

Unité de Recherche INRIA Lorraine Technopôle de Nancy-Brabois - Campus Scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 VILLERS LES NANCY Cedex (France)
Unité de Recherche INRIA Rennes IRISA, Campus Universitaire de Beaulieu 35042 RENNES Cedex (France)
Unité de Recherche INRIA Rhône-Alpes 46, avenue Félix Viallet - 38031 GRENOBLE Cedex (France)
Unité de Recherche INRIA Sophia Antipolis 2004, route des Lucioles - B.P. 93 - 06902 SOPHIA ANTIPOLIS Cedex (France)

EDITEUR
INRIA - Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)

ISSN 0249 - 6399