Thèse No 7470

METHODES DE PRESENTATION ET DE RECHERCHE D'INFORMATION : L'INTERFACE UTILISATEUR DE CALIBAN

THESE

présentée à l'Ecole Polytechnique Fédérale de Zurich
pour l'obtention
du grade de Docteur ès sciences techniques

par

Jean-Frédéric JAUSLIN

licencié ès sciences de l'Université de Neuchâtel
né le 31 juillet 1954
Le Locle (Neuchâtel)

Acceptée sur proposition de
Dr. H.-P. FREI, rapporteur
Professeur Dr. J. NIEVERGELT, corapporteur

1984

Préface.

L'organisation de ce travail, que l'on peut suivre sur le schéma ci-dessous, est définie de la manière suivante. Le premier chapitre donne une ***vue générale*** du travail et des différents points qui vont être examinés et auxquels il va être tenté d'apporter une solution. Ensuite le travail se divise en deux parties qui vont se dérouler parallèlement et selon un même schéma. La première partie traitera du domaine de la *Recherche* d'*Information*, de sa situation à l'état actuel, des divers systèmes existants, des nouveautés, des travaux actuellement en cours et des problèmes qui se posent pour le moment dans ce domaine. La seconde partie analysera les mêmes points mais en ce qui concerne les ***systèmes informatiques interactifs*** et plus particulièrement le domaine des *interfaces utilisateurs*. C'est un domaine qui évolue rapidement, en fonction notamment des progrès de la technologie et des découvertes incessantes au niveau du matériel. Ce chapitre étudiera la situation actuelle pour permettre de définir des méthodes qui tendront à améliorer les rapports que l'homme aura avec une machine ces prochaines années. Le parallélisme se poursuit lors des deux chapitres suivants pour expliquer dans quelles conditions ce travail a pu être réalisé. Ceci recouvre non seulement une présentation et une analyse des outils informatiques, qui ont été à notre disposition pour la mise en oeuvre, mais également quelles options ont été choisies au niveau du traitement de l'information. La suite du travail continuera alors en séquence afin de définir et d'analyser des *principes fondamentaux* que l'on a tenté d'établir lors de ce travail. Cette analyse se basera sur les deux domaines que l'on a traités auparavant pour tenter de trouver une synthèse judicieuse pour l'avenir des systèmes de traitement de Recherche d'Information. Suivra alors une présentation du système *Caliban* qui donne un exemple d'outil qui a été défini selon ces principes. Quelques points plus techniques achèveront la présentation du système, suivis des conclusions que l'on peut en tirer et des vues et développement futurs qui peuvent être envisagés.

J'aimerais tout d'abord remercier sincèrement H.-P. Frei qui m'a donné la possibilité d'effectuer et de mener à bon terme ce travail. Je lui sais gré de ses conseils judicieux et de son soutien constant qui m'a permis d'achever ce projet. Je le remercie enfin d'avoir accepter de diriger et de rapporter cette thèse.

J'aimerais également remercier M. Bärtschi et P. Lamb pour les fructueuses discussions que nous avons eues et le travail actif qu'ils ont fourni dans la réalisation du projet.

Mes remerciements vont au Professeur J. Nievergelt et à tous les membres de l'Institut d'Informatique de l'Ecole Polytechnique Fédérale de Zürich pour leur collaboration.

Enfin je voudrais remercier Carol de sa lecture et des corrections qu'elle a effectuées dans ce texte et surtout de la patience dont elle a fait preuve lors de la phase terminale du projet.

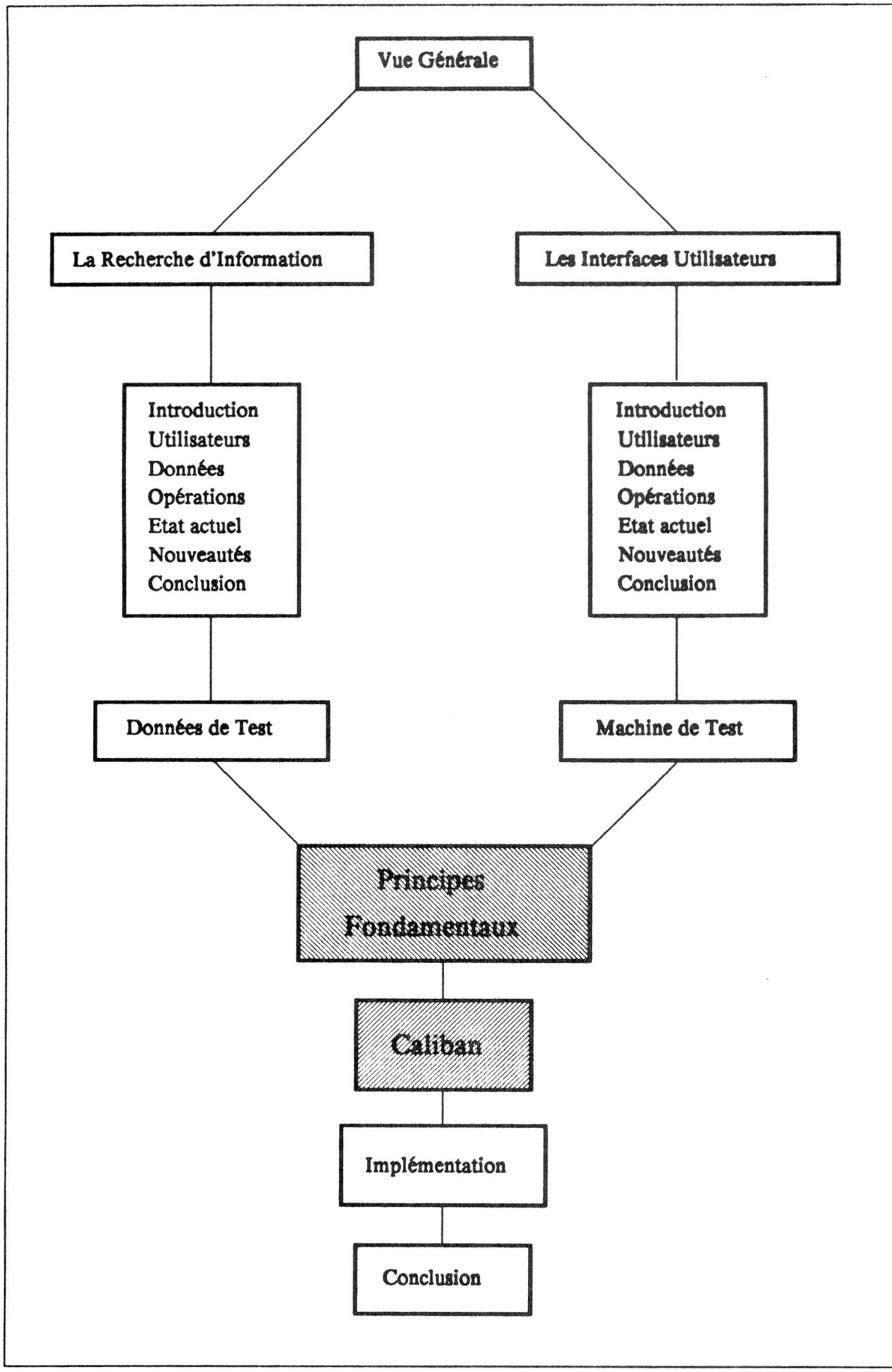

Vue Générale
La Recherche d'Information
Les Interfaces Utilisateurs
Introduction
Utilisateurs
Données
Opérations
Etat actuel
Nouveautés
Conclusion
Introduction
Utilisateurs
Données
Opérations
Etat actuel
Nouveautés
Conclusion
Données de Test
Machine de Test
Principes Fondamentaux
Caliban
Implémentation
Conclusion

Table des Matières

Résumé

Ce travail présente les concepts et la réalisation d'un système de recherche d'information. L'aspect de l'interface entre l'utilisateur et la machine est plus particulièrement traité. Le but est de montrer qu'il est possible d'amener l'utilisateur final à rechercher lui-même les renseignements qu'il désire obtenir sans avoir à passer par un intermédiaire. Les méthodes qui ont été développées doivent satisfaire aussi bien l'utilisateur occasionnel qu'une personne habituée au système. Ces principes se basent principalement sur une représentation graphique de l'information et des commandes disponibles.

Un schéma général dégage deux manières d'effectuer des recherches qui correspondent à la manière non automatisée avec laquelle une personne cherche des données bibliographiques. Tout d'abord le survol, qui permet d'obtenir des renseignements sur un sujet précis. Ensuite la formulation de requête qui, tout en conservant les avantages du survol, permet de trouver des documents incluant plusieurs thèmes qui se recoupent.

Caliban est un système expérimental qui a été développé sur l'ordinateur personnel Lilith et écrit en langage de haut niveau (Modula-2). Son interface utilisateur supporte la manière associative de pensée de l'être humain et même suggère des associations d'idées. L'information et les commandes sont affichées dans des fenêtres apparaissant sur l'écran. Une représentation graphique a été prévue pour des structures en forme de listes, de hiérarchies et de réseaux. Toutefois la quantité d'information que l'on peut présenter sur l'écran est limitée, c'est pourquoi des moyens de déplacement au sein de ces structures sont mis à disposition.

Kurzfassung

Diese Arbeit präsentiert Konzepte und Realisierung eines Information Retrieval Systems. Der Schwerpunkt der Arbeit liegt auf dem Entwurf der Benutzerschnittstelle. Ziel der Arbeit ist es, Möglichkeiten aufzuzeigen, wie der Dialog mit dem System ohne Hilfe eines Spezialisten geführt werden kann. Die entwickelten Methoden sollen sowohl den gelegentlichen wie auch den regelmässigen Benutzer zufriedenstellen. Diese Konzepte basieren zur Hauptsache auf einer graphischen Darstellung der gespeicherte Information und der verfügbaren Retrieval-Befehle.

Ein generelles Schema bietet zwei Retrieval-Möglichkeiten, die dem Vorgehen eines Benutzers einer Handbibliothek entsprechen : zuerst ein Überblick, der ihm erlaubt, Dokumente zu jeweils genau einem Sachgebiet durchzusehen (Browse); dann die Formulierung einer Abfrage, die -unter Wahrung der Vorteile des "Browsing"- die Zusammenstellung verschiedener, sich überschneidender Interessensgebiete erlaubt.

Caliban ist ein experimentelles System, das auf dem Arbeitsplatzrechner Lilith in der höheren Programmiersprache Modula-2 geschrieben wurde. Seine Benutzerschnittstelle unterstützt das assoziative Denken des Menschen und legt ihm sogar Gedankenverbindungen nahe. Die Information und die Befehle werden in verschiedenen Fenstern auf dem Bildschirm präsentiert. Graphische Darstellungen werden für Listen, hierarchische und netzwerkartige Strukturen verwendet. Die auf dem Bildschirm gezeigte Informationsmenge ist jedoch beschränkt, weshalb Befehle zur Verfügung gestellt werden, die eine Bewegung innerhalb der Strukturen erlauben.

Abstract

This work presents concepts and realization of an Information Retrieval System. The aspect of the user interface is particularly treated. The aim is to show how it is possible to allow the end user to search by himself without needing the help of an intermediary. The methods which have been developed, should satisfy the casual as well as the regular user. These principles are based mainly on a graphical presentation of the information and the possible commands.

A general scheme shows two ways of searching, corresponding to the way a patron uses a small library to retrieve bibliographic data : first browsing, which allows to retrieve documents on a precise subject; then query formulation, which allows to find items on different overlapping topics.

Caliban is an experimental system developed on the personal computer Lilith and written in a high-level system programming language (Modula-2). The user interface supports the associative way of human thinking and even suggests associations of thoughts. The information and the commmands are displayed on a graphics screen in separate windows. A graphical representation has been provided for lists, hierarchical structures and networks. However the amount of information which is possible to present on a screen is limited, and therefore commands to move inside the structures are available.

"An Information Retrieval System will tend not to be used whenever it is more painful and troublesome for a customer to have information than for him not to have it"

"Mooers' Law"

1. Vue générale

On définit la notion de *données* comme un ensemble de faits tirés d'observations ou de mesures effectuées dans notre univers. L'Information est une interprétation judicieuse et une corrélation de ces données qui permettent de prendre des décisions, ainsi que le soulignent Tsichritzis et Lochovsky [Tsic77]. Ces données existent donc dans le monde qui nous entoure et il s'agit d'en tirer une information essentielle pour parvenir à augmenter nos connaissances et affiner notre perception des choses. Si, dans les siècles passés, il était très difficile d'accéder à ces données par manque de moyens de communications, de nos jours il est plutôt difficile de faire un tri dans l'énorme masse qui a souvent tendance à nous submerger. Cet afflux est dû en grande partie à la fabuleuse évolution des mass média mais également aux possibilités grandissantes de stockage par les ordinateurs. Nous avons à disposition, par exemple, des bases de données bibliographiques formées de documents tels que livres, journaux, articles ou références à ces documents, qui fournissent un ensemble de données incontestablement gigantesque. Il est possible de trouver de la documentation sur n'importe quel sujet et de n'importe quelle période. La quantité de documents que chaque centre de traitement et de stockage de données contient est absolument ahurissante. Le système DIALOG de Lookheed [DIAL82], qui est un des systèmes les plus utilisés, ne propose pas moins de 55 millions de références, réparties sur plus de 170 bases de données. Ces références sont tirées de quelque 60.000 publications écrites en une quarantaine de langues différentes. Il faut noter que cette évolution est relativement récente. Heaps [Heap78] indique par exemple que le nombre de journaux scientifiques a passé de 100 en 1800 à 1000 en 1850, 10.000 en 1900 et 100.000 en 1966.

Pour aider une personne à trouver l'information qu'elle désire, des techniques de recherches ont dû être mises au point. C'est actuellement le domaine de prédilection des chercheurs en traitement de l'information, qui développent des méthodes qui permettent de chercher des objects dans un ensemble de données sans pour cela se perdre ou être débordé par la quantité de documents qu'il s'agit de compulser. La quantité de données ainsi que la complexité des méthodes de recherche tendent toutefois à augmenter la difficulté d'accès. C'est pourquoi certains spécialistes de recherche, appelés intermédiaires, sont apparus pour aider et conseiller les utilisateurs dans leur recherche. Ils connaissent les méthodes et les systèmes de recherche et peuvent ainsi considérablement aider un utilisateur à trouver ce qui l'intéresse. Malheureusement sa connaissance du centre d'intérêt de l'utilisateur est très faible et une perte d'information se produit à ce niveau.

Aux environs de 1945, bien avant l'arrivée des ordinateurs, V. Bush [Bush45] a imaginé une machine mécanique personnelle nommée "memex" qui permettait à un utilisateur de stocker, traiter et rechercher des quantités quasi infinies de données. Il s'agissait d'un pupitre muni d'un clavier, d'un ensemble de boutons et de leviers, et d'un écran sur lequel il était possible de projeter des données. Bush voyait cette machine imaginaire comme une partie intégrante d'un bureau où l'on travaillait. Elle

devait permettre de travailler selon ses aptitudes car, comme Bush le soulignait, l'esprit humain agit par associations, il suit les chemins de sa pensée et s'intéresse à une autre information qui est suggérée par une association d'idée. Malgré les progrès faits dans le domaine de la technologie et en particulier celui des ordinateurs, il n'existe encore aucune réalisation des rêves de Bush. Les problèmes techniques de stockage de l'information ont certes pu être résolus mais la plupart des systèmes interactifs ne sont pas suffisamment flexibles pour supporter la manière de penser associative utilisée par l'être humain.

Les ordinateurs et les systèmes d'exploitation ont tendance, à l'heure actuelle, à se rapprocher des idées de Bush. Un courant favorable aux ordinateurs personnels s'est fait sentir très nettement ces derniers temps; la plupart des constructeurs proposent actuellement sur le marché des outils qui permettent de travailler seul sur la machine avec des puissances de traitement plus qu'honnêtes. Ces petits ordinateurs ont généralement des possibilités au niveau du dialogue avec l'utilisateur tout à fait intéressantes. Un gros effort est investi actuellement dans les écrans et dans les outils permettant d'entrer de l'information et de communiquer avec la machine. Ces efforts permettent maintenant à des utilisateurs qui n'ont pas de connaissances particulières en informatique de s'initier à des systèmes basés sur un traitement électronique des données. En particulier, toutes les personnes qui travaillaient avec des machines à écrire, vont se trouver confrontées à des systèmes de traitement de textes qui vont leur offrir certainement plus de possibilités mais aussi une complexité nettement supérieure à celle qu'elles avaient connue jusqu'à présent. Les auteurs de ces systèmes doivent donc s'efforcer de concevoir des systèmes simples, susceptibles de convenir à un très large éventail d'utilisateurs et surtout capables de satisfaire au mieux chaque type d'utilisateurs. Les exigences de ces groupes risquent d'être très diverses et il faudra trouver une solution pour parvenir à un consensus.

C'est dans cette optique que ce situe le travail actuel. Tout d'abord un besoin commence à se faire sentir au niveau de la recherche d'information. Le stockage de données va devenir un des principaux buts des ordinateurs individuels. Chaque ordinateur risque dans un proche avenir de remplacer entre autres l'agenda de bureau, les classeurs, les archives sur papier, le courrier interne d'une industrie, les systèmes d'emprunts d'une bibliothèque. Tout cela signifie un grand nombre de données qu'il s'agit de stocker et de conserver, et là, les moyens matériels commencent à remplir des exigences qu'il n'aurait pas été permis d'imaginer quelques années auparavant. Il faudra toutefois aussi penser à retrouver cette information. C'est là que vont se situer les plus grands problèmes et, jusqu'à présent, ce domaine a été particulièrement laissé de côté. Des moyens conventionnels basés sur une organisation simple ne suffiront plus. Des méthodes de recherches et des systèmes permettant d'appliquer ces méthodes devront être apportés à un public qui n'aura pas toujours des connaissances méthodologiques et informatiques de haut niveau.

Le but de ce travail est d'essayer de faire un pas dans cette direction pour permettre

aux utilisateurs futurs de ne pas se perdre dans les méandres de la recherche d'information. Il s'agit également de montrer aux créateurs de systèmes qu'il est possible d'imaginer des outils simples et efficaces basés sur des schémas et des modèles déjà définis dans la vie courante. Il faut garder à l'esprit qu'un système ne doit pas contraindre l'utilisateur à de nouvelles méthodes mais suggérer une amélioration de celles qu'il connaît. Les points suivants, qui représentent quelques thèmes d'une ligne de conduite à suivre lors de la conception de système de recherche, seront examinés tout au long de ce travail et mis en évidence lors de la présentation du système qui a été développé selon ces principes :

Suppression des intermédiaires lors de la recherche. Rapprochement de l'utilisateur final et des systèmes de recherches.

Adaptation et emplois de nouveaux moyens informatiques dans le cadre des systèmes de recherche d'information.

Développement de systèmes susceptibles de s'adapter aux méthodes familières à l'utilisateur, en particulier support et suggestion d'associations d'idées.

Amélioration de la présentation des informations et de ces structures.

Uniformisation des méthodes de recherches par rapport à des structures d'accès différentes.

Simplification du système à utiliser sans pour cela perdre en efficacité, en rapidité et en puissance.

Nouvelles méthodes d'organisation et de recherches dans le cadre des ordinateurs personnels au sein d'un bureau.

Pour illustrer les différents points qui ont été mentionnés, un schéma représentatif des systèmes d'information et en particulier des systèmes de recherche de données va être mis au point et développé tout au long du travail. Chaque chapitre a pour but de définir ou de préciser un point particulier de ce schéma ou de développer un nouvel aspect qui ne ferait pas encore partie intégrante des schémas analysés jusque-là.

Considérons tout d'abord le schéma général d'un système d'information donné par B. Croft [Crof82].

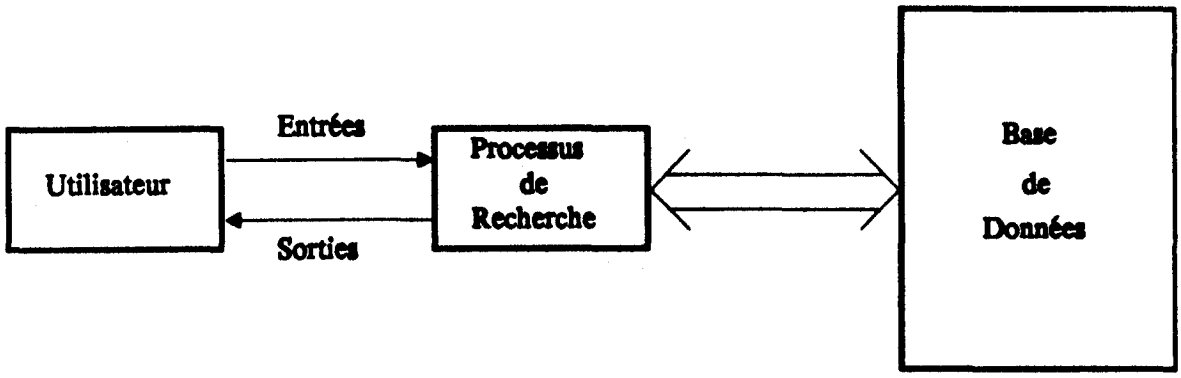


Fig 1.1 Schéma de B. Croft

Ce diagramme paraît à prime abord susceptible de satisfaire n'importe quel système de recherche. Ces différentes parties correspondent aux points importants que l'on peut trouver dans tout système d'information, à savoir:

- Un utilisateur.
- Un module de recherche.
- Des données

L'*Utilisateur* fournit des requêtes au *processus de recherche* qui s'occupera de trouver les *données* qui sont en rapport avec la question et les soumettre à l'utilisateur qui pourra les étudier. Croft suggère que les différents systèmes de recherches ont en commun le fait que le processus de recherche comprend un *dialogue* entre l'utilisateur et le système. Ce dialogue, qui à mes yeux représente une phase très importante, si ce n'est la plus importante, n'apparaît malheureusement pas explicitement dans ce schéma. Si l'on considère une session de recherche effectuée par un utilisateur, elle se compose en fait d'une phase de préparation de requête -la formulation-, du temps nécessaires au système pour trouver les objets correspondants à la question -la recherche- et de la présentation des résultats -la visualisation-. Le temps utilisé par le système pour effectivement répondre à une question de l'utilisateur ne va représenter qu'une très petite partie de la session. Il faut donc se rendre compte que l'amélioration des temps de réponses d'un système de recherche ne représente qu'un faible avantage pour l'utilisateur. Un effort tout particulier doit donc être fait premièrement dans l'amélioration de la phase de préparation de la question et secondement dans la visualisation des résultats, puisque ces deux phases représentent la majeure partie de la session de recherche. On peut définir un modèle général de système de recherche de la manière suivante. Celui-ci ne contredit pas la schéma de Croft mais le précise quelque peu en mettant surtout l'accent sur la partie *interface utilisateur* qui est réellement le premier contrat que devrait remplir un système d'information et en particulier un système de recherche de données:

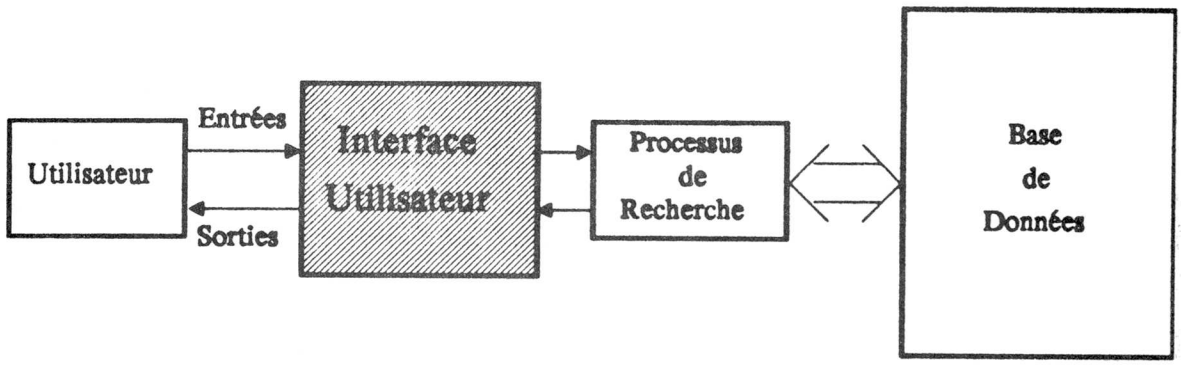


Fig 1.2 Schéma Interface Utilisateur

Cet interface va jouer le rôle d'intermédiaire entre l'utilisateur et le processus de recherche et un dialogue va s'établir. Il faut remarquer que ce dialogue comprend deux parties:

La première va permettre à l'utilisateur de formuler les questions qu'il va poser. Des outils doivent être mis à sa disposition pour préparer ses questions. Il doit comprendre facilement comment les formuler et comment les soumettre au système. Il s'agit principalement de définir des sujets ou des centres d'intérêt à propos desquels on désire obtenir des données. L'utilisateur veut alors connaître les objets concernés par tous les sujets qu'il a choisis. Ces sujets n'ont peut-être pas pour lui la même importance mais ils interviennent tous à des degrés divers dans le processus de recherche des documents concernés.

La deuxième partie de l'interface utilisateur comprend un système qui permet de visualiser, d'étudier, de sauver ou de modifier les résultats obtenus suite à la requête proposée. Chaque question peut fournir un nombre variable d'objets, à savoir de zéro à l'ensemble des données contenues dans le système. Il faut fournir à l'utilisateur un moyen puissant de recherche et de visualisation de ces données. Si le nombre en est trop élevé, l'utilisateur sera tenté de préciser sa question afin de diminuer ou d'améliorer ses résultats. Lorsqu'il parvient à un nombre qu'il juge convenable, le système doit lui permettre d'étudier ces résultats de manière interactive.

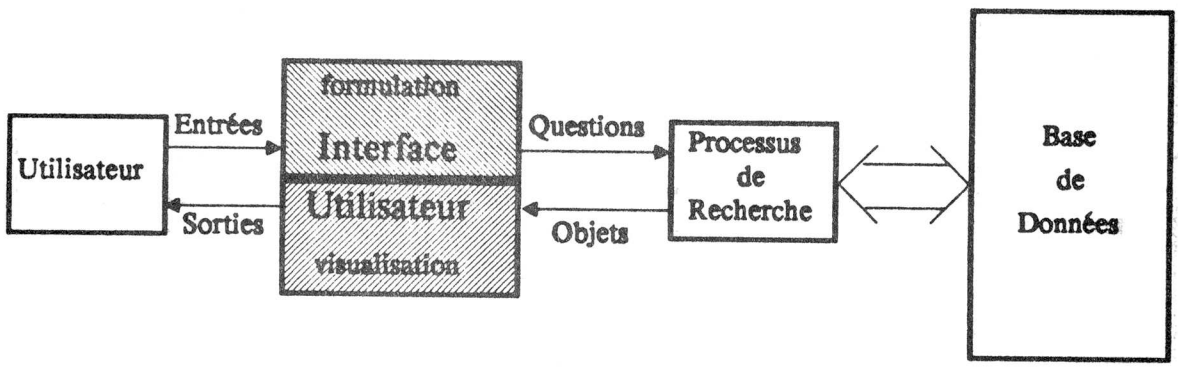


Fig 1.3 Schéma Interface Utilisateur divisé

Nous verrons plus particulièrement dans les chapitres 6 et 7 comment ces deux phases peuvent être réalisées et comment on peut définir un modèle qui reflète le schéma de pensée et la manière d'agir d'un utilisateur. Nous verrons également comment il est possible de séparer l'interface utilisateur du système de recherche et sous quelles formes il est judicieux de définir la forme des questions par rapport à celle des objets que l'on désire retrouver.

Mais voyons tout d'abord ce que signifie le concept "système d'information" puis quelles sont les tendances au niveau des "interfaces utilisateurs".

2. La recherche d'information

2.1 Introduction

Le terme "Système d'Information" a été utilisé pour un très grand nombre d'applications allant du système classique de comptabilité aux systèmes les plus sophistiqués qui peuvent quasiment simuler un dialogue en langage naturel sur un thème précis. Dans tous les cas, un système d'information contient un certain nombre de *données*, des *opérations* sur ces données peuvent être effectuées, et un *utilisateur* commande ces opérations. On peut distinguer trois types de systèmes d'informations: Les systèmes de gestion de base de données (SGBD), les systèmes de recherche documentaire (RD) et les systèmes experts (SE) [Crof82], [Salt83]. Il ne s'agit pas là d'une classification car tous les systèmes ne peuvent se répartir dans ces catégories mais ils en couvrent la plus grande partie et la majorité des personnes travaillant dans ce domaine se situent dans ces trois catégories. Examinons un peu plus en détail les caractéristiques de ces trois types de systèmes.

Les systèmes de gestion de base de données (SGBD)

Les objets que l'on trouve dans ces bases de données se distinguent par leurs structures. Les enregistrements sont formés de différents champs contenant les données. Chaque champ contient une donnée d'un même type dont les valeurs peuvent être soit des chaînes de caractères, soit de type numérique. Les entités de la base de données sont des combinaisons de ces champs. Par exemple, une entité "Employé" peut avoir les champs "numéro d'employé", "âge", "fonction" et "salaire". Une autre entité "Poste de travail" contiendra les champs "nombre d'employés", "emplacement", "matériel", etc. Il faut alors définir des relations logiques entre ces enregistrements de manière à définir clairement quels sont les rapports entre les entités existant dans le système et afin de minimiser la place occupée; en effet, chaque champ ne sera conservé si possible qu'une seule fois pour, d'une part diminuer la quantité de données stockées, et d'autre part réduire les problèmes de consistence lors de modifications.

Les types d'utilisateurs d'un SGBD sont très variables, allant du directeur désirant obtenir des renseignements sur le personnel, en passant par le chef de rayon qui désire contrôler son stock, à l'administrateur de la base de donnée qui définit la structure de la base et contrôle son état et sa consistance. Les connaissances de ces utilisateurs varient fortement et ils ont chacun des besoins très différents.

Les opérations effectuées sur les données sont généralement la recherche, l'insertion, la modification et l'élimination d'enregistrements. Afin de déterminer quels sont les enregistrements contenus dans la base de données, l'utilisateur devra préparer une question en donnant les champs et le domaine de valeurs que ces champs doivent posséder. On parle alors de clés primaires ou secondaires. Une question type pourrait être "Donner la liste des noms des employés qui gagnent plus de Frs 40.000 par an et

qui se trouvent aux postes de travail X ou Y". On voit que ces questions peuvent se formuler au moyen de combinaisons booléennes des champs qui intéressent l'utilisateur. La caractéristique primordiale se situe au niveau des résultats qui sont toujours définis de manière unique. A chaque question correspond un nombre précis d'enregistrements et chaque élément des résultats correspondra de manière exacte à la question posée.

Ceci ne donne qu'un bref aperçu d'un système de gestion de base de données. Une très nombreuse littérature existe sur le domaine et l'on peut se référer par exemple au livre de Date [Date77] ou à l'ouvrage en français de Delobel et Adiba [Delo82].

Les systèmes de recherches documentaires

Les données stockées dans de tels systèmes sont des documents tels que livres, articles, rapports ou n'importe quel autre texte écrit en langage naturel. Cependant, pour des raisons de place et d'organisation, on ne conserve souvent que des références à ces documents; c'est-à-dire un ensemble de mots-clés et de termes représentant au mieux le contenu du document. A cela s'ajoute quelquefois un ensemble de champs, tels qu'ils sont définis dans les SGBD, qui donnent une information précise sur l'ouvrage (date de parution, nombre de pages, maison d'édition, langues, etc). Enfin d'autres champs, tels que Titre ou Résumé, dont les valeurs ne couvrent pas un domaine strict mais sont sous forme de texte, qui fournissent une très grande information sur le contenu de l'ouvrage. Il faut remarquer que les termes ou mots-clés d'une référence peuvent s'apparenter au champ que l'on trouve dans un SGBD. Toutefois, la relation qu'il existe entre la valeur du champ et l'enregistrement est nettement moins spécifique. Si l'on décrit un ouvrage en lui assignant un mot-clé, ou une série de mots-clés, ceux-ci ne vont pas définir de manière exacte et précise l'ouvrage considéré. Cela dépendra essentiellement de la spécificité des termes employés.

Les utilisateurs finaux d'un tel système sont en majeure partie des utilisateurs occasionnels. Ils n'ont, en général, que peu d'intérêt à la structure du système et ne souhaitent qu'une liste précise de références couvrant au mieux le domaine qui les intéresse. Les recherches documentaires ne sont pas souvent des opérations quotidiennes pour un utilisateur et l'apprentissage du système sera très restreint et très vite oublié. Certes, une catégorie d'experts en recherche existe, les intermédiaires, mais ils ne représentent que très mal la majorité d'utilisateurs finaux qui sont réellement intéressés à trouver ces documents.

En ce qui concerne les opérations, il s'agit pratiquement uniquement de recherche de références et, comme déjà signalé auparavant, ces questions ne vont pas fournir un ensemble précis et bien défini de références. Les résultats dépendront de la précision de la question et de la valeur de l'indexation des documents, c'est-à-dire de la spécificité des termes index par rapport aux documents eux-mêmes. Cette phase d'indexation a certes une importance capitale pour la qualité du système. Toutefois

l'utilisateur final, celui qui nous intéresse, n'intervient pas dans cette phase, c'est pourquoi nous laisserons ici de côté cette opération.

Les systèmes experts

Ces systèmes, comme leurs noms l'indiquent, jouent le rôle d'experts dans un domaine précis. Leur but est donc de permettre d'obtenir des informations déduites de faits donnés dans un contexte prédéterminé. Eliza [Weiz76] est un des premiers systèmes du genre qui simulait un psychanalyste et entretenait un dialogue avec un patient. Les données contenues dans le système sont un très grand ensemble de faits en relation avec la situation donnée. D'autre part des règles de déductions, nommées *productions*, permettent de tirer certaines conclusions de faits déterminés. La "connaissance" du système est fonction de la quantité des faits stockés, de la qualité et du détail des relations entre ces faits. Ces relations sont généralement représentées sous forme de graphes appelés *réseaux sémantiques*. La quantité de données nécessaires à de tels systèmes restreint actuellement leur domaine d'application et il n'est guère possible d'en envisager un emploi général. Les utilisateurs sont donc des personnes fortement intéressées au sujet traité par le système et il est difficile de leur donner certaines caractéristiques communes. Il peut s'agir aussi bien de patients dans le cas d'Eliza que de médecins pour le système MYCIN [Rand77] qui les aide à déterminer un traitement dans le cas d'infection du sang. Les questions posées par l'utilisateur ainsi que les réponses fournies par le système sont généralement en langage naturel. Cela signifie qu'une analyse lexicale et syntaxique doit permettre de déterminer les faits prépondérants de la question. Vient ensuite une étude des productions possibles qui déterminera les réponses à formuler. Elles ne seront plus formées de références, comme dans les systèmes de recherches documentaires, mais de faits existants dans la base de données ou déduites des éléments qui la constituent. Certains faits, n'étant pas explicitement contenus dans la base, peuvent être déduits des règles de productions et des faits spécifiques; c'est là une des grandes différences avec les deux types de systèmes vus auparavant. En fait, les techniques utilisées se basent très fortement sur les principes de l'intelligence artificielle. Pour une étude plus détaillée, on peut se reporter à Nilsson [Nils80], Winston [Wins77] ou Raphael [Raph76].

Le tableau suivant (Fig 2.1) résume les différences entre les trois systèmes présentés. Il ne tient compte que des trois aspects mentionnés, à savoir le type des données, les utilisateurs et les opérations que l'on peut effectuer.

On voit donc qu'un point commun à ces systèmes est la possibilité d'effectuer des recherches pour déterminer quelles sont les données contenues dans la base. Que peut-on dire sur la nature et la forme de ces recherches ? Dans le cas d'un SGBD, les questions sont posées de manière très précise. L'algorithme de recherche devra déterminer quels sont les enregistrements qui correspondent exactement à la question posée. La liste des résultats est donc complètement déterminée à priori et ne requiert aucune évaluation ou interprétation de la part du système de recherche.

	Données	Utilisateurs	Opérations
SGBD	.format fixe .structurées .schémas	.connaissances très variées	insertion, modification, élimination, recherche d'enregistrements
RD	.peu structurées (texte) .références aux documents	.variés mais suivant un même but .occasionnel	.recherche de références
SE	.réseaux sémantiques .productions	.variés .fonction du domaine d'application du système	.recherche de faits et d'implication d'événements

Fig 2.1 Schéma récapitulatif des systèmes d'information

Dans le cas de la recherche de documents, les questions peuvent s'apparenter à celle des SGBD. Elles sont également formulées de manière précise en donnant un ensemble de termes ou de champs qui intéressent l'utilisateur. Les résultats, par contre, vont être déterminés par la qualité de l'indexation des documents, par la prise en compte de termes voisins ou synonymes ou encore par l'importance que l'on attribue à chaque champ de la question. Une même question peut donc fournir un résultat variable suivant l'algorithme de recherche.

Les systèmes experts, quant à eux, permettent de poser des questions relativement vagues, formulées généralement en langage naturel. Le système répondra en fonction de son degré de connaissance et de la qualité de ses règles de productions. Les réponses fournies peuvent donc varier très fortement.

En résumé, voici comment se présente le système de recherche des 3 cas examinés:

SGBD:
- Questions précises
- Réponses précises

RD:

- Questions précises
- Réponses variables

SE:

- Questions vagues
- Réponses variables

Van Rijsbergen [Rijs83] donne des exemples de question type pour chaque cas:

SGBD:

- Est-ce que New York est la plus grande ville des Etats-Unis ?

RI:

- Donnez-moi les documents qui concernent la pollution de la ville de New York.

SE:

- Pourquoi est-ce que New York est la plus grande ville des Etats-Unis ?

Ceci nous permet de mieux situer les différents genres de systèmes d'information et surtout de déterminer précisément le domaine dans lequel se situe ce travail. Nous n'envisagerons que la situation de l'utilisateur d'un système de recherche de documents, parce que d'une part il s'agit d'un utilisateur général et que d'autre part, les besoins de recherche d'information vont se situer à l'avenir dans ce domaine-là. Tout le matériel provenant des machines qui vont aider à former le bureau du futur, en particulier les systèmes de traitement de textes, sera du type de ceux que l'on trouvent dans le domaine de la recherche de documents. Chaque rapport, chaque mémo, chaque lettre sont des documents semblables à ceux que l'on trouve dans les systèmes bibliographiques et les techniques utilisées devront être les mêmes pour tous les genres de documents.

Examinons maintenant un peu plus en détail les différents composants de ces systèmes de recherche de documents.

2.2 L'utilisateur.

Les utilisateurs d'un système de recherche de documents ont tous un point en commun : ils désirent trouver des références relevantes aussi rapidement que possible. La majorité d'entre eux ne sont pas intéressés par les principes du système de recherche mais par la rapidité avec laquelle ils vont obtenir des documents et par la qualité de ces documents. Les critères de qualité peuvent varier : d'aucuns préfèreront obtenir peu de documents très spécifiques, d'autres, le plus possible de documents sur le sujet. Nous reviendrons sur ce point lors de l'explication des opérations que l'on effectue dans un système de recherche (Précision-Rendement).

Une recherche est un processus relativement ponctuel; on ne recherche pas des références bibliographiques tous les jours. Cela signifie que l'apprentissage du système est un point très important. Les interfaces utilisateurs actuels sont relativement complexes, comme nous le verrons plus loin. C'est la raison pour laquelle il existe des intermédiaires spécialisés dans l'emploi de ces systèmes qui aident les utilisateurs finaux à préciser leurs requêtes. On compte, pour la formation de ces intermédiaires, un temps d'apprentissage de l'ordre de six mois [Blic83]. Ceci est absolument impossible à supporter pour un utilisateur final. D'autre part, dans le cas d'une aide par l'intermédiaire, il se produit une perte d'information dans la communication entre les deux personnes. L'utilisateur n'arrive pas à formuler ses requêtes telles que l'intermédiaire les attend et l'intermédiaire ne peut avoir des connaissances sérieuses dans le domaine d'intérêt de l'utilisateur final. Les domaines couverts par les systèmes de recherche bibliographique sont beaucoup trop étendus pour le permettre. Cette perte d'information représente un problème sérieux dans le cas des systèmes traditionnels et conduit à un phénomène de non-satisfaction de l'emploi de tels procédés.

Si l'on essaie de supprimer ces intermédiaires, on se heurte au problème de l'apprentissage et de l'utilisation. La complexité interdit d'envisager actuellement une utilisation directe du système par l'utilisateur final sans un effort important de sa part. Le temps d'apprentissage n'est pas le seul facteur qui empêche ce passage. L'éventail des personnes susceptibles d'utiliser un tel service va en s'élargissant. Leur capacité de compréhension risque de diminuer et l'on aura, comme utilisateur, des personnes n'ayant aucune connaissance de domaines qui paraissent évidents à des chercheurs ayant une formation académique.

Prenons un exemple : la plupart des systèmes se basent sur des principes de spécification de combinaisons booléennes de termes. La logique booléenne généralement employée dans les systèmes de Recherches d'Information n'est pas forcément naturelle à un utilisateur occasionnel. La notion de logique booléenne est devenue tellement familière aux informaticiens et aux concepteurs de système qu'elle n'est même plus remise en cause lors de la spécification de système. G. Salton, dans son dernier livre [Salt83] , donne l'exemple d'une recherche faite sur le concept "INFORMATION RETRIEVAL". Chaque mot qui forme ce concept fait partie de

l'index du système alors que le concept lui-même, formé de deux mots, ne fait pas partie du dictionnaire. Pour rechercher les occurences correspondantes à ce concept, l'utilisateur doit donner comme requête :

INFORMATION AND RETRIEVAL

ou éventuellement

INFORMATION OR RETRIEVAL

Nous avons tous, en tant que spécialistes du traitement de l'information, l'habitude de ce genre de combinaison; elle nous paraît tout à fait naturelle et évidente. Je ne suis pas sûr toutefois que l'utilisateur occasionnel, lorsqu'il pense à un concept formé de deux mots, voit immédiatement que ces deux mots sont en fait liés par une relation booléenne et que l'opérateur entre ces deux mots est AND, OR ou encore AND NOT.

Cet exemple peut sembler quelque peu artificiel, mais où je suis sûr de l'incompétence de l'utilisateur néophyte, c'est au niveau de la priorité de ces opérateurs lors de combinaisons plus complexes. Si l'on imagine une expression booléenne telle que

INFORMATION OR DATA OR DOCUMENT AND RETRIEVAL OR SEARCHING NOT ORGANIZATION

il est relativement difficile, même pour un habitué à ce genre d'expression, de savoir immédiatement et surtout intuitivement quels sont exactement les termes touchés par une telle recherche. Il faut tout d'abord discerner les opérateurs puis définir quelles sont les règles de priorité entre ces opérateurs; c'est-à-dire, de quelle manière s'effectue l'évaluation de l'expression.
Prenons l'exemple suivant:

A OR B AND C.

Si l'opérateur OR possède une priorité plus grande que AND, alors l'évaluation de cette expression consistera à choisir un élément parmi A et B (évaluation du OR) puis d'y ajouter obligatoirement C. Les solutions sont dans ce cas "A suivi de C" ainsi que "B suivi de C".

Si la priorité est inversée et que l'opérateur AND a une priorité plus grande que OR, le résultat de cette expression sera "A" comme première solution et "B suivi de C" comme seconde. Ce qui diffère sensiblement de la solution précédente.

Pour récapituler et tenter de connaître un peu mieux les différents utilisateurs, on peut définir certaines catégories. Ingwersen [Ingw83] les sépare en quatre classes :

- l'*élite* ayant des connaissances conceptuelles aussi bien dans un domaine spécialisé que dans la recherche d'information.

- les *intermédiaires* connaissant la recherche d'information mais n'ayant pas de connaissances conceptuelles approfondies.

- l'*utilisateur final* ayant des connaissances conceptuelles spécialisées mais aucune dans la recherche d'information.

- le *citoyen* n'ayant aucune connaissance spéciale.

L'élite, bien que ne représentant qu'un faible pourcentage (~15%), effectue environ 40% des recherches. Ce sont des personnes susceptibles de s'adapter aux systèmes et qui en utilisent toutes les possibilités. Ils ont un temps de préparation et de dialogue très faible par rapport à l'emploi des commandes, aux termes recherchés et à la qualité des résultats obtenus.

Les intermédiaires peuvent également utiliser les possibilités du système, leur manque de connaissances appronfondies dans un domaine précis les empêche toutefois d'employer leur savoir à bon escient.

Restent les utilisateurs finaux et les citoyens qui représentent les utilisateurs de demain et auxquels il faut fournir des systèmes suffisamment simples et flexibles pour qu'ils puissent rechercher eux-mêmes l'information qui leur manque.

Si cette phase de simplification ne se produit pas, la recherche d'information restera un outil réservé à une élite et ne pourra jamais s'établir sur un marché de masse et ainsi satisfaire la majorité des utilisateurs potentiels

2.3 L'information.

Ce chapitre est destiné à préciser clairement les concepts de données, de références, de structures d'accès, et de la relation qu'il existe entre eux. Nous avons vu que, dans le cadre des systèmes de recherche de documents, les objets que l'on traite sont généralement des références à des documents. Il s'agit en fait de listes d'attributs qui sont censés représenter au mieux le contenu du document original. Si l'on désigne par A l'ensemble des attributs possibles et par a_i, le ième élément de cet ensemble, on peut dire qu'une référence peut se définir de la manière suivante :

$$R = \{ a_1, a_2, ..., a_n \}$$

Si l'on envisage une référence du livre de Van Rijsbergen "Information Retrieval" [Rijs79], on peut définir cette référence comme

R = { Information Retrieval, Automatic Text Analysis, Automatic Classification, File Structures...}

Toutefois, les valeurs, ou termes, que l'on trouve dans l'ensemble des attributs, peuvent se situer dans des domaines très différents. On définit des catégories d'attributs et par conséquent des ensembles A_i différents. On aura donc comme référence :

$$R = \{ a_{11}, a_{12}, .., a_{21}, a_{22}, ..., a_{m1}, a_{mn} \}$$

On peut prendre comme exemple d'attribut un thesaurus, une classification décimale, l'ensemble des auteurs considérés ou même la date de parution. Dans ce dernier cas, nous n'aurons qu'un seul terme dans la référence, à savoir la date elle-même. L'ensemble de ces références forme donc les données dans lesquelles l'utilisateur va chercher de l'information. On appelle l'ensemble de ces objets un fichier. Chaque référence peut évidemment avoir une longueur variable. Un résumé, par exemple, fait partie généralement d'une référence. La longueur de ce champs varie beaucoup d'une référence à l'autre et il n'est pas possible de définir une longueur maximale à priori. C'est un problème dont il faut tenir compte lors de l'organisation des données. La manière la plus simple de stocker ces données, de manière informatique ou non, est une juxtaposition de toutes les références, un fichier séquentiel. C'est l'organisation la plus primitive et la plus simple. Dans certains cas, un attribut est choisi pour servir de clé de tri et le fichier est ordonné selon cette clé. Si les références sont simplement ajoutées à la fin du fichier, la clé est, dans ce cas, la date de parution. Les avantages de ce type d'organisation sont les suivants :

- la facilité d'implémentation
- l'économie de place
- un accès relativement rapide à l'élément suivant dans l'ordre lexicographique.

Les deux principaux désavantages sont :

- la difficulté de mise à jour
- la lenteur de l'accès direct

Ces deux points, et tout spécialement le dernier, nous amènent à envisager d'autres types d'organisation, basés sur des structures auxiliaires qui vont permettre un accès plus rapide aux différents éléments du fichier.

Les structures d'accès.

Lorsque la quantité de données que l'on a pu stocker a été suffisamment grande pour ne plus pouvoir rechercher directement dans l'ensemble des objets, il a fallu trouver des nouveaux moyens d'accès, en particulier des systèmes de pointeurs ou de fichiers séquentiels indexés. L'unique raison de cette indexation, vient des problèmes de recherche eux-mêmes. Un simple fichier séquentiel n'étant plus possible, les objets ont dû être ordonnés et posséder une structure.

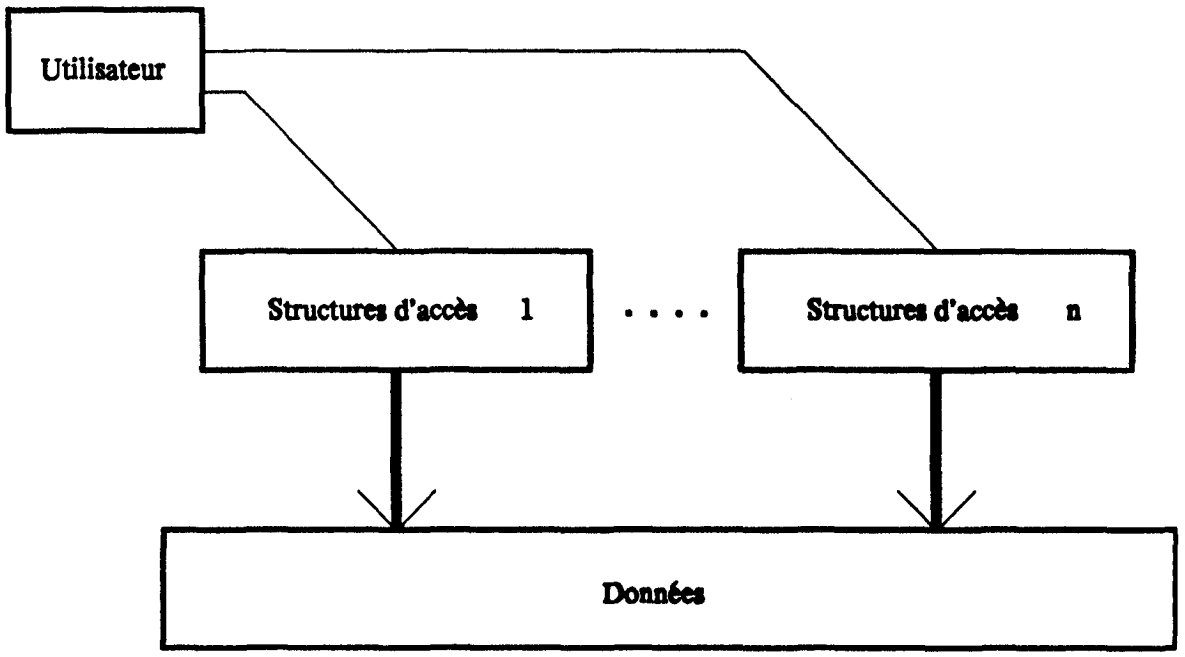


Fig 2.2 Les Structures multiples d'accès

Quelles sont ces structures et que peut-on dire sur leur forme ou leur définition? L'exemple le plus connu est certainement la structure de fichier inversé.

Le fichier inversé

On appelle fichier inversé une structure auxiliaire contenant les valeurs des attributs notés a_{ij} plus haut. On associe à chaque terme les adresses des références dans le fichier de base qui contiennent le terme a_{ij}. Cette structure auxiliaire sera généralement organisée de manière alphabétique si les termes sont des mots-clés. Il sera donc aisé de trouver quelles sont les références qui contiennent le mot-clé a_{ij} en parcourant la liste des adresses que l'on trouve dans la structure d'accès A_i. On peut avoir pour un même ensemble de données plusieurs types de structures différentes, comme par exemple une liste alphabétique de mots-clés et une liste alphabétique des noms d'auteurs.

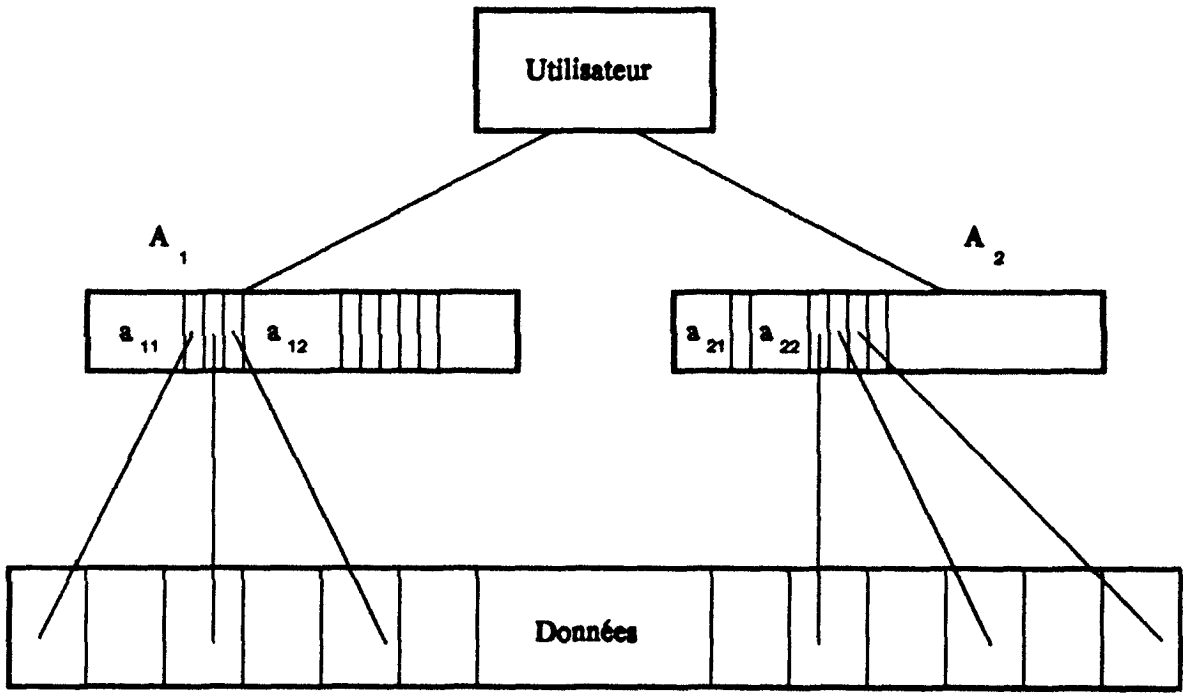


Fig 2.3 Fichiers inversés multiples

Les structures d'accès hiérarchisées

Il est évident qu'au sein d'un même attribut on trouve certaines similitudes entre les termes. Chaque terme a_{ij} n'est pas complètement indépendant du terme a_{ik}. On est alors tenté d'organiser ces termes en classes et sous-classes pour permettre à nouveau un accès plus rapide aux références contenues dans les données. Cette nouvelle organisation nous amène au concept de structure d'accès hiérarchique, telle qu'elle se présente dans le cas d'une classification décimale.

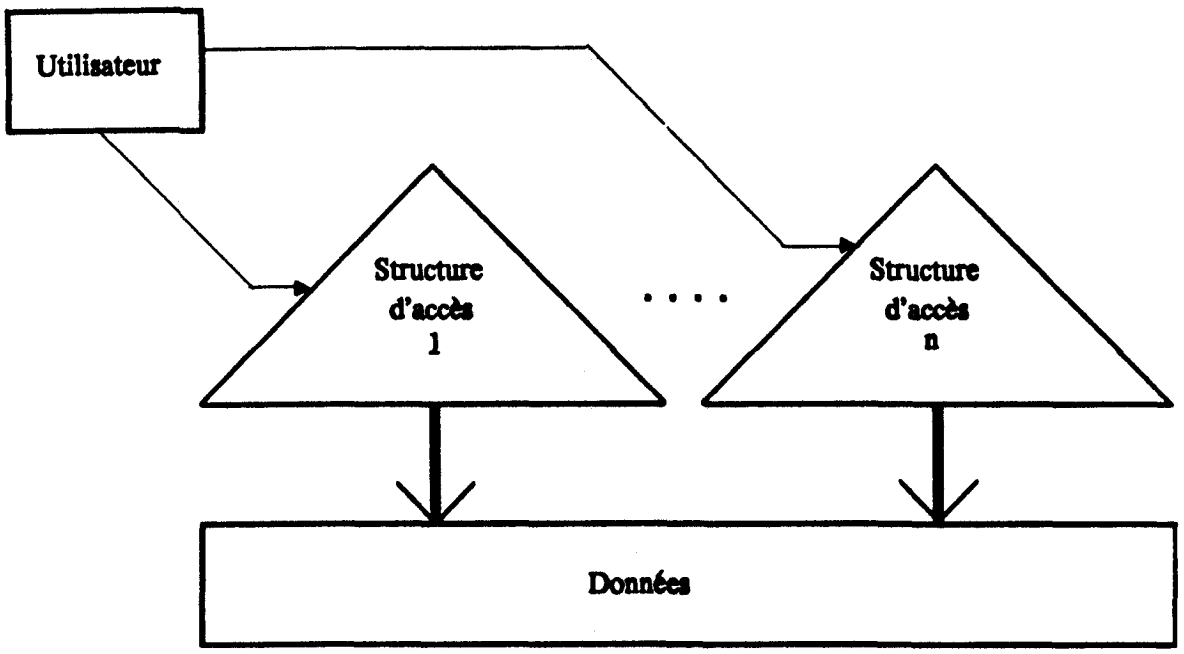


Fig 2.4 Structures hiérarchisées d'accès

Chaque terme n'est plus sur le même plan que les autres mais peut être un sous-concept ou un concept plus général par rapport aux autres termes. Aussi le terme "Information Science" peut être un concept moins précis, et se trouver par là plus haut dans la hiérarchie, que les termes "Information Storage" ou "Information Retrieval".

Les structures d'accès en réseaux

Ceci nous amène à envisager une structure plus complexe qu'une simple hiérarchie. En effet, un terme peut avoir plusieurs concepts généraux différents; "Computer networks" peut être un sous-concept de "Communication networks" et de "Digital systems". De plus, d'autres types de relations plus faibles peuvent intervenir. On arrive à une organisation en réseaux des termes qui forment la structure d'accès aux références.

Ces structures ont très rapidement pris de très grosses proportions jusqu'à dépasser en volume les données elles-mêmes. J'en tiens pour preuve les différentes structures que l'on peut trouver dans le commerce (Classification décimale ou Thesaurus). Celles-ci représentent sans aucun doute une aide à la recherche d'objets car elles permettent à l'utilisateur de rechercher plus systématiquement et de réduire la quantité d'objets à considérer. L'utilisateur peut ainsi définir des sous-ensembles de la base de données et travailler plus aisément avec ces sous-ensembles, à savoir les combiner pour en former d'autres plus spécifiques, jusqu'à obtenir une quantité d'information raisonnable à visualiser et étudier. Un domaine qui à été, à mon sens, quelque peu laissé de côté est le traitement des structures elles-même. D'une part en ce qui concerne la génération et l'entretien, mais encore, et c'est le domaine qui nous intéresse, l'accès, la visualisation et la navigation dans ces structures d'accès.

2.4 Les opérations

La première opération principale que l'on effectue avec un système de recherche consiste à déterminer quelles sont les objets dans le système susceptibles de satisfaire certains critères. L'utilisateur doit donc réussir à formuler au mieux ces critères afin d'obtenir des résultats correspondant à ces désirs. Dans le cas de systèmes basés sur des fichiers inversés (la plupart actuellement), il va s'agir de trouver quels sont les termes disponibles dans le système qui satisfont au mieux les besoins de l'utilisateur. Il peut sembler, à priori, que c'est un problème simple à résoudre, étant donné qu'il existe une liste, dans l'ordre lexicographique, de tous ces termes. Malheureusement, la quantité de ces termes stockée dans le système est généralement si grande qu'il est très difficile de faire des recherches conventionelles. La plupart du temps, on agit par tâtonnement en donnant un terme et en espérant qu'il soit contenu dans le système. Il faut rappeler que ce processus de recherche dépend évidemment de la manière dont sont indexés les documents. Les termes employés par l'indexeur ne correspondent par forcément à ceux utilisés par la personne qui fait des recherches. De plus, si la personne qui fait des recherches est un intermédiaire, elle n'aura pas la flexibilité lui permettant d'imaginer un terme semblable ou un synonyme ou encore de reconnaître un tel synonyme. Nous verrons, dans le chapitre suivant quels sont les outils qui permettent, dans les systèmes actuels courants, de spécifier sa question.

La seconde activité principale, du point de vue de l'utilisateur, consiste à évaluer les résultats fournis par le processus de recherche; c'est-à-dire, estimer la valeur et la qualité des documents ou des références que l'on reçoit suite à une question. A nouveau, il semble aisé de déterminer si ces documents sont relevants. Toutefois, si l'on imagine que le nombre de documents peut être relativement grand (rappelons que Lookheed possède environ 55 millions de références !), on s'aperçoit que cette évaluation pose certains problèmes. De plus, les résultats fournis sont rarement ordonnés selon leur qualité. Il devient alors absolument impossible de faire cette évaluation sans une étude complète des résultats. Les critères qui déterminent l'arrêt des recherches se résument au nombre de références trouvées. L'utilisateur stoppe sa recherche lorsqu'il estime qu'il a obtenu un nombre raisonnable de références et qu'il peut s'astreindre à une étude plus approfondie de ces documents pour en déterminer la qualité.

Pour arriver à resteindre la quantité de données fournie par le système, l'utilisateur entre dans un processus itératif (formulation d'une question - étude des résultats) jusqu'à ce qu'il obtienne ce nombre jugé satisfaisant. Pour ce faire, il faut trouver des critères de restriction pour éliminer certains documents. Ces critères peuvent être du genre - date de parution, si l'on ne s'intéresse qu'à une certaine période -, ou - langage utilisé dans le document, puisqu'il ne sera de toute façon pas possible d'étudier les documents écrits dans une langue inconnue -. Ces critères vont généralement permettre de réduire de manière drastique le nombre de documents mais ils ont le désavantage de s'appliquer à l'aveugle, c'est à dire d'éliminer des objets qui pourraient intéresser l'utilisateur. Si, par exemple, on spécifie une date

pour éliminer les documents parus auparavant, il est très possible qu'un document particulièrement intéressant, mais sortit juste avant cette date, ne disparaisse des résultats.

Nous avons vu que, dans le cas d'un système de recherche documentaire, les résultats fournis étaient déterminés par des critères subjectifs (indexation, évaluation de la question) et correspondaient au mieux (best match) à la question posée et non pas exactement (exact match) comme c'est le cas dans les systèmes de gestions de base de données. Lorsqu'un utilisateur reçoit, en réponse à sa question, M documents et qu'il estime que parmi ceux-là, P documents sont relevants, on définit un coefficient appelé *précision*, égal au rapport P/M. Ce coefficient est donc égal à 1 si tous les documents dans les résultats sont jugés relevants par l'utilisateur. En admettant que l'utilisateur puisse déterminer ou connaisse tous les documents qui sont relevants dans le système et qu'ils soient au nombre de R, on définit un autre coefficient, le *rendement*, donné par P/R, c'est à dire le nombre de documents relevants qui ont été trouvés par rapport au nombre de documents relevants dans le système.

On voit que ces deux critères se contredisent puisque, pour obtenir un rendement satisfaisant, on aura tendance à diminuer les critères de sélection et par conséquent à augmenter le nombre de documents dans les résultats, alors que pour obtenir des documents très précis il faudra, au contraire, resteindre ces critères pour diminuer le nombre de documents non relevants.

Ces critères ne sont toutefois pas contrôlables par l'utilisateur et servent plutôt à déterminer la qualité d'un système sur une collection de données déterminées (p. ex. la collection Cranfield).

2.5 L'état actuel et les systèmes existants.

Exemple de dialogue conventionnel.

Dans un dépliant publicitaire pour le système DIALOG de Lookheed [Dial82], on peut lire l'exemple suivant qui illustre assez bien l'état actuel des interfaces utilisateurs et du trou qu'il existe entre la manière naturelle que l'on conçoit dans un dialogue et le jargon informatique auquel doit se soumettre un utilisateur. Ceci commence tout d'abord par une explication en clair de ce que désire une certaine personne et des réponses que le système est sensé lui fournir. Ensuite vient le dialogue effectif qu'il s'est produit entre l'utilisateur et la machine.

- Ce que vous dites à DIALOG est écrit en lettres normales

• Ce que DIALOG répond est écrit en italique

- J'aimerais chercher dans le fichier 15 s'il-vous-plait.

• Que voulez-vous que je vous trouve ?

- Avez-vous certains articles qui possèdent le mot **stress** ou le mot **tension** ?

• Oui, j'en ai 1564 qui se réfèrent à **stress** *et 395 qui comprennent une référence à* **tension**, *avec un total de 1840 documents qui mentionnent soit un ou les deux termes.*

- Combien d'articles avez-vous qui mentionnent **executives** ou **managers** ou **administrators** ?

• J'ai les références suivantes : executives - 7'072; managers - 14'435; administrators - 997; pour un total de 20'707.

- Combien de ces articles ou documents contiennent les termes **stress** ou **tension** ainsi que les termes **executives** ou **managers** ou **administrators**

• 460

- Je m'intéresse uniquement aux articles récents. Parmi les 460 combien ont été publiés en 1981 ?

• 80

- J'aimerais connaître les informations suivantes sur le premier de ces 80 documents : numéro d'enregistrement, titre, titre de la parution, date, nombre de pages et résumé de l'article.

• Le titre de cet article est "Executive Stress: Pressure in a grey Flannel Suit" écrit par "Roger H Lourie" et paru dans le "Direct Marketing Magazine" volume 44, numéro 8 aux pages 46 à 49 du numéro de décembre 1981. (• suivit du résumé de l'article •)

- Pour les 9 articles suivants, donnez moi seulement les informations de base sans

résumés.

• (• affichage des données demandées •)

- Merci j'ai fini, déconnectez-moi et donnez-moi le prix de cette recherche.

• Cette recherche a été faite le 19 février 1982 et terminée à 11h54. Le numéro d'identification de l'utilisateur est 3468. Le prix du temps ordinateur est de $5.91. Le temps utilisé pour cette recherche est de 0.081 heures. La recherche a été faite sur le fichier 15. Six termes ont été utilisés. Les frais de communications (TELENET) sont de $0.49 et le coût total approximatif est de $6.40.

Ce genre de dialogue paraît tout à fait raisonnable à première vue. Les demandes de l'utilisateurs sont claires et précises et l'on pourrait imaginer qu'il serait facile de déterminer un langage de commande qui, bien que n'étant pas du langage naturel, se rapprocherait de ce que montre ce dialogue. Voici pourtant le jargon auquel cet utilisateur a dû se plier pour recevoir les informations qu'il désirait.

(• le point d'interrogation en début de ligne signifie que le système attend une commande •)

? BEGIN 15
File 15:ABI/INFORM-71-82/JAN
(Copr. Data Courier Inc.)
Set Items Description

?SELECT STRESS OR TENSION
1564 STRESS
395 TENSION
1 1840 STRESS OR TENSION

?SELECT EXECUTIVES OR MANAGERS OR ADMINISTRATORS
7072 EXECUTIVES
14435 MANAGERS
997 ADMINISTRATORS
2 20707 EXECUTIVES OR MANAGERS OR ADMINISTRATORS

?COMBINE 1 AND 2
3 460 1 AND 2

?SELECT S3 AND PY=1981
26623 PY=1981
4 80 S3 AND PY=1981

?TYPE 4/7/1
4/7/1
82001738
Executive Stress: Pressure in a grey Flannel Suit
Lourie, Roger H.

Direct Marketing V44N8 46-49 Dec 1981
Chronic stress in business executives is usually found more in middle management than in top management.....

?TYPE 4/3/2-10

4/3/2
82001706
Preventing Environmental Stress in the Open Office
Rader, Martha; Gilsdorf, Jeanette
Jrnl of Systems Mgmt V32N12 25-27 Dec 1981

4/3/3
82000967
Managing Stress for Increased Productivity
Huber, Vandra L.
Supervisory Mgmt V26N12 2-12 Dec 1981

****

4/3/10
Does Your Head Hurt?
Cohen, Irving J.
Inc. V3N12 117-119 Dec 1981

?LOGOFF

19feb82 11:54:10 User3468
$5.91 0.081 Hrs File 15 6 Descriptors
$0.49 Telenet
$6.40 Estimated Total Cost.

Cet exemple sert à illustrer l'état actuel des systèmes de recherche documentaire traditionnels. Les systèmes utilisés de nos jours présentent tous plus ou moins le même interface utilisateur et oblige la personne à se plier à un jargon relativement complexe et long à apprendre.

Les systèmes actuels

Quels sont ces systèmes et quelles sont leurs caractéristiques ? Tous les systèmes que je citerai sont basés sur un principe de fichiers inversés.

En premier lieu, il convient de citer le système DIALOG produit par Lookheed Information Systems à Palo Alto [Bour79]. C'est sans aucun doute actuellement le système le plus utilisé, dû au fait de sa relative facilité d'emploi. L'idée de base consiste à créer des ensembles successifs de références au moyen de la commande SELECT suivie d'une expression booléenne de termes qui vont définir le contenu de l'ensemble. Chaque ensemble est numéroté chronologiquement et l'utilisateur a la possibilité de former d'autres ensembles en combinant (commande COMBINE) ceux déjà obtenus avec des opérateurs booléens. On peut, lors de la spécification, ne

donner que la racine des termes que l'on désire, comme par exemple INFO suivi d'un point d'interrogation, pour préciser que l'on entend par là tous les termes qui commencent par INFO (INFO?). Il est possible de donner dans la recherche plusieurs types d'attributs différents tels que auteur (AU), date de publication (PY), langage (LA) entre autres possibilités.

STAIRS est un autre système connu, développé par IBM Corporation [Stai76]. Il se compose de deux parties

- Un système de gestion de base de données
- Un système de recherche interactif nommé AQUARIUS

STAIRS possède deux modes de recherche différents; le premier (commande SEARCH) pour traiter des opérations au niveau du texte et le second (commande SELECT) pour rechercher des données structurées dans le sens d'une gestion de base de données. Un dictionnaire contenant tous les mots apparaissant dans le système est défini de manière à permettre une utilisation soit en tant que fichier inversé (vocabulaire contrôlé), soit pour servir une recherche au niveau du texte directement. Ceci permet, entre autre, d'avoir des exigences sur la position des termes l'un par rapport à l'autre (2 mots adjacents (ADJ), 2 mots intervenants dans la même phrase (WITH), dans le même paragraphe (SAME) ou encore de déclarer que certains mots sont considérés comme synonymes (SYN)). Il semble toutefois que, selon certaines statistiques faites dans la pratique [Blic83], ces opérateurs soient assez peu employés. Le mode SEARCH possède les opérateurs suivants (donnés dans leur ordre de priorité)

- ADJ
- SYN
- WITH
- AND, NOT
- OR, XOR

Le mode SELECT permet en plus de préciser, de manière exacte, certaines valeurs de champs (p.ex. AGE EQ 30). Les opérateurs relationnels de ce mode sont EQ (égal), NE (différent), NG (plus petit ou égal), NL (plus grand ou égal), GT (plus grand), LT (plus petit), WL (dans les limites) et OL (en dehors des limites).
Enfin STAIRS offre la possibilité d'ordonner les documents trouvés (commande RANK) en attribuant certains poids aux différents termes de la question. Ce poids est fonction de:

- La fréquence du terme dans le document
- La fréquence du terme dans l'ensemble des documents trouvés
- Le nombre de documents de l'ensemble dans lesquels le terme apparaît.

L'utilisateur ne peut pas indiquer l'importance que ces termes ont pour lui; il ne peut que choisir un des algorithmes proposés qui combinent les trois valeurs mentionnées

ci-dessus de manière différente.

STAIRS est un système qui a beacoup de possibilités; il a toutefois l'inconvénient d'être relativement complexe à utiliser et de nécessiter un ordinateur très puissant pour le faire tourner.

Un autre système connu et largement utilisé est celui développé par la bibliothèque nationale de médecine, qui permet de trouver de l'information dans le domaine biomédical. Ce système, MEDLARS [Medl79], a été développé en parallèle avec ORBIT [Orbi75] qui fut créé par System Development Corporation. La structure de MEDLARS est basée sur trois fichiers inversés, INDEX, POSTINGS et DATA qui donnent respectivement les termes, le nombre de documents contenant ces termes et les documents eux-mêmes. Le vocabulaire d'accès est fixe et le système ne permet pas de rechercher au niveau du texte en lisant chaque document dans la base de donnée. Il est possible, par contre, de faire une telle recherche sur un ensemble de documents déterminé par une question antérieure. Ce procédé réduit sensiblement le coût en temps d'une telle recherche.

Beaucoup d'autres systèmes sont couramment utilisés. On peut citer par exemple:

- LEXIS contenant des informations juridiques [Lexi76]
- RECON développé par Lookheed pour la NASA et donc apparenté à DIALOG
- BRS de l'université de New York et basé sur STAIRS
- Information Bank contenant les articles publiés par le New York Times [Roth69]

Ils se basent tous sur le même principe de fichiers inversés et ne proposent que des variations mineures pour l'utilisateur par rapport aux systèmes mentionnés. Plusieurs personnes se sont intéressées à comparer ces différents systèmes et à définir les opérations de base nécessaires à la recherche documentaire. On peut citer entre autre, Lancaster et Fayen [Lanc73] Minker [Mink77], Salton [Salt83] et un travail de semestre fait à l'Ecole Polytechnique Fédérale de Zurich par M. Bärtschi qui compare les interfaces utilisateurs des systèmes STAIRS, ORBIT et DIALOG [Bärt78].

2.6 L'évolution.

Il existe en fait trois tendances dans l'évolution des systèmes de recherche.

- L'amélioration des systèmes traditionnels
- La mise au point de nouvelles bases théoriques
- L'intégration des systèmes dans un contexte nouveau

Ces trois points ne sont pas incompatibles et dans les quelques exemples qui vont suivre, on peut se rendre compte que certains systèmes contribuent à plus d'un titre à cette évolution. Chaque point peut être vu sous deux angles.

- La contribution à l'évolution du fondement théorique ou de l'organisation du système
- Les modifications apportées à l'interface utilisateur et donc relativement importantes pour l'image que l'on se fait du système

L'amélioration des systèmes traditionnels.

Plusieurs commandes ont été ajoutées aux systèmes traditionnels pour essayer d'offrir des possibilités plus vastes et donc un système plus performant. Comme exemple, on peut citer une récente présentation faite par Ingwersen [Ingw83] d'une nouvelle commande ajoutée au langage de recherche ESA-QUEST (le système de l'European Space Agency). ZOOM - c'est le nom de la commande - permet, sur un ensemble de documents, de rechercher des termes selon leur fréquence. Chaque texte est analysé et tous les mots-clés sont ordonnés selon leur fréquence d'apparition. On peut ainsi dégager une nouvelle série de mots qui apparaissent fréquemment et qui auraient échappé à l'utilisateur.

Ce genre de développement permet certainement de combler des lacunes mais nous avons vu que la puissance d'un système est généralement proportionelle à sa complexité et qu'en fait elle ne sert qu'un petit groupe d'utilisateurs.

La mise au point de nouvelles bases théoriques.

Le système SMART [Salt71] est un des premiers à s'être éloignés du système de fichiers inversés. L'idée de Salton repose sur la définition d'un espace vectoriel où chaque terme représente une dimension de l'espace. Un document est donc caractérisé par sa position dans l'espace, c'est-à-dire par le vecteur de termes qui l'identifie. A chaque terme du vecteur peut être assigné un poids pour déterminer le rôle que joue ce terme dans le texte. Les documents semblables vont alors se trouver à proximité l'un de l'autre dans cet espace vectoriel. On peut ainsi définir un sous-espace (cluster) contenant les documents semblables. Il est caractérisé par un document, nommé centroïde, qui joue le rôle de représentant de cette classe d'objets. Si l'organisation du fichier où l'on stocke ces documents respecte cette répartition en classes, il est possible d'accélérer le processus de recherche étant donné qu'une

question n'est plus comparée qu'avec le centroïde des classes pour définir l'ensemble des documents formant les résultats. SMART permet également d'abandonner la logique booléenne puisqu'il s'agit de préciser l'importance que l'on accorde à chaque terme de la question et non plus la relation qu'il existe entre eux. Le système THOMAS [Oddy77] abandonne aussi la logique booléenne en s'appuyant sur le principe d'amélioration de la question en présentant à l'utilisateur des documents tirés d'un sujet choisi. Il est possible d'orienter sa recherche en se basant sur les objets directement et non plus sur des structures d'accès. Ce principe, nommé post-recherche, s'appuie sur les décisions quant à la relevance des documents pour modifier une question antérieure. Le principe contraire, la pré-recherche, oblige une investigation du vocabulaire d'indexation avant de décider des termes qui doivent être inclus dans la question.

L'intégration des systèmes dans un contexte nouveau.

La tendance actuelle de l'informatique vers les ordinateurs personnels va créer certains besoins nouveaux au niveau de la gestion et de la recherche de documents. Sans toutefois atteindre un niveau de complexité d'indexation aussi grand que celui que l'on trouve dans les systèmes proposés au public, certains systèmes devront permettre d'indexer et de rechercher des documents qui sont produits en grande quantité dans les bureaux actuellement. Cette tendance se fait sentir par l'apparition de certains produits sur le marché. On peut citer, comme exemple, le système CONDOR développé chez SIEMENS [Fisc82] ou plus récemment le projet de Croft nommé Tess [Crof83] qui passe résolument de la bibliothèque ouverte à un système destiné à un bureau mais qui utilise certaines notions de stockage et de recherche tirées des grands systèmes de recherche documentaire. On retrouve l'idée de classe telle qu'on l'a vue chez Salton, mais cette fois-ci sous forme de réseaux de termes et de documents et non plus d'espace vectoriels de termes.

2.7 Conclusion

Pour conclure ce chapitre de présentation de la recherche d'information, j'aimerais citer Mahon, qui affirme que "de nouvelles techniques de recherche sont un must". Les utilisateurs potentiels et non qualifiés ne peuvent se plier au jargon défini par les personnes qui développent les systèmes de recherche actuels. Il s'agit de prouver qu'il est possible de créer des systèmes ayant des possibilités satisfaisantes mais qui restent simples d'emploi. Un large public peut et doit être atteint car la recherche documentaire ne doit pas rester un domaine réservé à une élite. La mise au point de réseaux d'information va atteindre la population par le biais de la télévision. D'énormes quantités de données vont submerger les gens. Il faut que les techniques de selection et de recherche soient capables de contenir un tel flot et surtout qu'elles soient claires à tous. Pour ce faire, il faut développer des nouvelles techniques d'interfaçage entre les utilisateurs et les machines, basées sur un matériel qui sera disponible ces prochaines années. Nous allons voir dans le chapitre suivant qu'elle est la situation actuelle et l'évolution possible des interfaces utilisateurs avant de définir clairement les principes qui ont été adoptés dans ce travail.

3. Les interfaces utilisateurs

3.1 Introduction

Ben Shneiderman [Shne80] donne une liste relativement exhaustive des principes qu'il faut suivre lors de la spécification d'un interface utilisateur. Le premier point qu'il cite est donné par Hansen : "connaître l'utilisateur". Il est en effet primordial de définir la classe d'utilisateurs potentiels du système pour savoir quelles sont ses connaissances à priori. Lors de la conception et de la réalisation du système, il faut, à tout moment, faire des choix parmi les solutions possibles. Il est très difficile pour un concepteur de système de faire ces choix non pas en fonction de ses connaissances mais en fonction de celles des futurs utilisateurs. Ceci est d'autant plus difficile lorsque la classe des utilisateurs ne correspond pas à celle du concepteur; lorsque celui-ci définit l'interface utilisateur d'un éditeur de texte, il prépare un produit destiné à la même catégorie de personnes que lui et il sera un utilisateur de son système. Dans le cas d'un système de recherche documentaire, la classe d'utilisateur est différente de celle des personnes ayant des connaissances informatiques et il faut se placer au niveau de celui qui possède le moins de connaissances pour lui permettre également d'utiliser facilement le système. Le piège à éviter est alors de ne pas créer un système qui ne s'adapte qu'à cet utilisateur et par conséquent qui ne convient plus aux utilisateurs plus avancés

Shneiderman cite les critères que Gebhardt et Stellmacher [Gebh78] ont mis au point pour les langages de recherches documentaires. Dans le cadre de leur travail, ils ont défini plus de 150 exigences qui devraient être satisfaites par un système de recherche idéal. Ils se basent sur trois principes fondamentaux.

- Un système de recherche est destiné à un utilisateur occasionnel car c'est lui qui a besoin de l'information. Ils suggèrent implicitement la suppression des intermédiaires puisque cela ne peut qu'engendrer des pertes de communications dans le transfert d'information entre l'utilisateur final et les données (et vice versa).

- Les langages naturels ne sont pas des méthodes de communication satisfaisantes entre le système et l'utilisateur car ils sont redondants, imprécis et ne sont, jusqu'à présent, implémentés que sous forme très formelle.

- Les objets traités dans le système doivent pouvoir être structurés.

Sur la base de ces principes, ils définissent des critères de valeurs qu'ils répartissent en 8 catégories :

1. Simplicité.
   - 1.1 Peu de mots-clés - peu de commandes.
   - 1.2 Simplicité de l'entrée - rapidité de l'entrée (en respect avec la structure du clavier);
     - . abréviations mnémotechniques;
     - . structure simple de l'entrée.
   - 1.3 Commandes courtes - mots-clés courts;
     - . peu de redondance;
     - . élimination d'entrée multiple;
     - . emploi d'options par défaut.
   - 1.4 Commandes simples - structure de commande simple;
     - . syntaxe simple de commandes; correspondance entre la syntaxe et la sémantique;
     - . structure simple du dialogue.

2. Clarté.
   - 2.1 Structure hiérarchique - structure hiérarchique du langage de commande (commandes et sous-commandes).
   - 2.2 Fonctionnalité - séparation fonctionnelle des commandes;
     - . éviter des commandes différentes pour la même fonction;
     - . pas de commandes avec des fonctions multiples;
     - . élaboration claire des cas spéciaux importants.
   - 2.3 Homogénéité - structure identique pour toutes les commandes;
     - . même signification des mots-clés pour toutes les commandes;
     - . mêmes possibilités dans des contextes identiques;
     - . interprétations uniformes des paramètres manquants.
   - 2.4 Orientation du problème - pas de restrictions ou d'exceptions techniques évitables (causées par la structure des données ou par des considérations de programmations);
     - . pas de séparation évitable entre les branche du dialogue;
     - . chaque commande est admissible à tout point du dialogue.

3 Unicité.
   - 3.1 Déterminisme - chaque commande est totalement déterminée par ses opérandes et ses options prédéfinies.
   - 3.2 Pas d'états non-définis - tous les états du système sont toujours parfaitement définis.

4 Confort du langage.
   - 4.1 Commandes puissantes qui effectuent plusieurs opérations.
   - 4.2 Flexibilité - formes longues et courtes des mots-clés;
     - . formes multilinguales;
     - . opérandes directs et indirects;
     - . ajustement du système aux connaissances et à l'expérience de l'utilisateur;

. commandes adaptées à l'utilisateur occasionnel, régulier et professionnel;
. contrôle par l'utilisateur des options du système.

4.3 Dialogue court - des commandes complètes (y compris sous-commandes) et même des séquences de commandes peuvent être entrées en une fois;
. de nouvelles commandes (ou partie de commandes) peuvent être définies par des macros (reformulation de chaînes de caractères).

4.4 Emploi complet des structures de données - toutes les structures de données peuvent être visualisées et utilisées pour la recherche.

5. Autre confort.

5.1 Confort de l'entrée - réutilisation d'une entrée ou d'une sortie (possibilités de corrections);
. technique de menu.

5.2 Interruption - le dialogue peut être interrompu en tout temps.

5.3 Langage de sortie - clair, concis, messages du système compréhensibles;
. sortie pouvant être discernée de l'entrée;
. réutilisation de la sortie en entrée (dans les cas appropriés).

5.4 confort supplémentaire - différentes améliorations au niveau du software ou du hardware telles que : touches-fonctions, signal accoustique en fin de sortie, soulignement ou mise en valeurs, arrangement clair de la sortie, édition et visualisation claire de table, techniques de survol en avant et en arrière; carnet de notes.

6. Evidence et réutilisation.

6.1 Evidence - évidence de l'état du système (attente d'entrée, entrée, attente de sortie, sortie);
. confirmation des commandes exécutées;
. messages périodiques lors d'attente;
. avertissements à propos de commandes difficiles.

6.2 Fonction Help - fonction help donnant l'état du système, la fonction utilisée actuellement, toutes les fonctions possibles, la structure et le contenu de la base de données, le dialogue effectué jusqu'à présent, les possibilités pour continuer;
. protocole du dialogue.

6.3 Réutilisation - les commandes précédentes et la sortie réutilisable en entrée;
. insertion de commandes précédentes dans la commande présente (en particulier, dans la formulation de questions);
. sauvetage de commandes pour des exécutions ultérieures.

7. Stabilité.

7.1 Traitement d'erreurs - messages claires lors de graves fautes en entrée;
. corrections lors d'erreurs légères;
. traitement uniforme des erreurs;

. pas de conséquence graves lors d'entrée courte.

7.2 Pas de situations contraignantes - pas d'obligation à continuer le dialogue de manière fixe;

. le dialogue peut être stoppé en tout temps.

8. Sécurité des données

. des mots de passe différents pour les structures de données et les données elles-mêmes;

. les mots de passe manquants peuvent être donnés ultérieurement;

. lors d'accès à des données secrètes, le système doit réagir comme si elles n'existaient pas;

. cela ne doit pas provoquer de discontinuités dans le dialogue;

. les besoins de sécurité d'une partie des données ne doivent pas empêcher l'accès aux données libres.

Il ressort de ces principes que l'utilisateur, quel qu'il soit, désire avant tout une simplicité et une clarté dans le dialogue doublé du confort que peuvent lui apporter les nouveaux outils en matière de software ou de hardware.

Bien que n'étant pas parti sur ces bases, nous verrons que nous sommes arrivés à des résultats très semblables à ceux de Gebhardt et Stellmacher. Les différences principales se situent au niveau des moyens techniques (aussi bien hardware que software) que l'on a à disposition à l'heure actuelle. Souvent leurs principes tentent d'éliminer des erreurs provoquées par du matériel ou des méthodes du début des années 70. Un principe, tel que celui cité au point 4.1, vient de l'utilisation de terminaux alphanumériques où l'utilisateur doit entrer au clavier de longues séquences de caractères pour exécuter des commandes. Maintenant que des outils permettent de pointer sur ces opérations ou d'utiliser des touches-fonctions, le nombre d'opérations à effectuer n'est plus primordial puisque l'utilisateur peut exécuter toute une série de commandes très rapidement et sans perdre du temps à taper sur le clavier.

Nous avons développé quelques points qui manquent à cette liste de critères; il s'agit tout d'abord des possibilités de navigation dans les structures de données ainsi que dans les données elles-mêmes. C'est un point qui paraît primordial dans un système de recherche étant donné qu'il va permettre de retrouver plus facilement et plus précisement l'information. Souvent l'utilisateur a tendance à se perdre dans des structures complexes ou à ne pas trouver exactement l'information qu'il recherche. C'est pourquoi la navigation est un point très important. Nous avons de plus défini clairement quels étaient les types de structures que l'on pouvait rencontrer et surtout souligné l'importance de la représentation de ces structures. Enfin, l'homogénéïté ne doit pas se situer qu'au niveau des commandes et de la manière dont on les exécute mais également entre les objets traités par le système, en particulier entre les questions et les réponses.

3.2 L'utilisateur

Notre société actuelle n'est pas encore habituée aux ordinateurs en tant qu'outils et beaucoup de gens voient la machine comme un "monstre" mystérieux et tout puissant. Cette représentation vient très certainement de ce qu'ils ne savent pas comment fonctionne exactement un ordinateur et qu'ils pensent que cette machine est capable d'effectuer des opérations qui les dépassent. De plus, le mythe de la machine toute puissante est entretenue par beaucoup de spécialistes en informatique qui désirent conserver l'image de ceux qui possèdent "la Science". Shneiderman [Shne80] cite également la tendance à personnaliser l'ordinateur dans un dialogue lorsque la machine répond "Hello, I am Betsy, what is your name ?". Ceci ne contribue qu'à troubler l'utilisateur occasionnel et à augmenter cette impression de mythe.

Le premier point essentiel est donc d'essayer de diminuer cette "crainte" de l'inconnu et de rendre l'utilisateur le plus confiant possible. Dans ce but, il faudra améliorer l'attrait que l'utilisateur trouvera à employer un programme. Nombre de jeux électroniques ont un immense succès grâce à leur faculté de distraire le joueur. Les premiers jeux qui sont apparus sur des ordinateurs étaient relativement "verbeux" et réservés à un certain nombre de connaisseurs qui étaient capables de passer par toutes les tracasseries d'un système d'exploitation. Et bien que l'attrait propre du jeu était assez grand, le plaisir de jouer diminuait très vite. Dès que les moyens technologiques l'ont permis, les efforts investis dans la visualisation de ces jeux ont permis d'arriver à l'engouement général que l'on connaît actuellement. Plusieurs jeux qui existaient sous forme textuelle ont été modifiés uniquement au niveau de la présentation sans en changer la stratégie; ils ont connu tout de suite un très grand succès. Il est donc très important, lors de la conception d'un système informatique destiné à des utilisateurs occasionnels ou non professionnels de l'informatique, d'améliorer le plus possible le côté attractif du système.

Un autre exemple de rapport que l'être humain doit établir avec un outil se trouve dans le domaine de l'horlogerie. L'évolution au niveau de l'intégration en électronique permet de produire des montres ayant un grand nombre de fonctions. Celles-ci toutefois n'étaient possibles que grâce à un affichage digital permettant plus de flexibilité au niveau de la présentation. Jusqu'à présent toutefois, l'utilisation des possibilités reste très complexe pour quelqu'un de non averti : la visualisation de la date par exemple n'est déjà pas triviale, quant à la remise à l'heure, celle-ci est du domaine de l'impossible sans l'étude appronfondie du manuel qui est livré avec la montre. Une tendance actuelle revient très nettement à une simplification des modèles, même s'il s'agit de perdre quelques fonctions, ou à une tentative d'amélioration de cette interface utilisateur qui reste encore à découvrir. Un point important du développement d'un produit consiste à s'adapter aux besoins et habitudes de l'utilisateur potentiel et non pas de lui imposer une nouvelle structure rendue possible grâce à une certaine évolution technologique. Le retour à l'affichage analogique dans l'horlogerie est un exemple frappant de ce phénomène.

3.3 L'information

L'échange d'information entre un utilisateur et une machine consistait, jusque récemment, à des chaînes de caractères. L'utilisateur entrait au clavier les caractères qui formaient le nom d'une opération et le système lui répondait en affichant les résultats de l'opération. Le processus s'effectuait séquentiellement puisque les terminaux ne permettaient que ce genre de traitement ligne par ligne. L'amélioration des moyens technologiques a permis dernièrement de concevoir des méthodes de transfert d'information différentes. Tout d'abord, les terminaux permettent maintenant un affichage graphique qui autorise la présentation de données de manière beaucoup plus attractive. Ceci a tout de suite débouché sur une utilisation plus judicieuse de l'écran, en divisant l'écran en plusieurs parties. La technique de fenêtres permet d'utiliser une grande partie de l'écran, en général la partie de droite, qui n'était que très rarement utilisée. D'autre part la présentation graphique de l'information permet souvent d'expliquer clairement et avec peu d'effort des concepts qu'il serait long et difficile d'expliquer en peu de mots. C'est une réelle application du dicton qui précise qu'un dessin peut valoir 1000 mots. Le côté graphique ne doit toutefois pas devenir excessif et l'on doit, de temps à autre, faire preuve de bon sens dans la conception de système. Ce qui paraît intéressant dans cette évolution graphique, c'est l'emploi bi-dimensionnel de l'écran qui est certainement plus naturel à l'utilisateur et qui permet de jouer sur la position de l'information en fonction de son importance. On sait que dans nos contrées occidentales, l'être humain analyse une image de gauche à droite et de haut en bas. L'information qu'il rencontrera en premier lieu se situera dans le coin en haut à gauche. Par contre, celle qui se situe en haut à droite le frappera beaucoup moins vu que le mouvement des yeux se fait le long d'une diagonale. On ne mettra donc pas de l'information très importante dans des endroits qui risquent d'échapper à l'utilisateur.

La nature, la quantité et la position sur l'écran de l'information jouent maintenant un rôle primordial dans les interfaces utilisateurs et il n'existe encore que très peu de méthodes qui définissent clairement les règles à appliquer.

3.4 Les opérations

Si l'on compare les possibilités qu'offre une machine à celles que possède un utilisateur, on remarque que l'ordinateur peut :

- stocker de grandes quantités d'information
- rechercher cette information rapidement et de manière sûre
- la traiter rapidement et précisement
- effectuer une suite d'instructions prédéfinies

L'homme quant à lui est capable de :

- reconnaître des formes
- formuler des buts
- identifier de nouvelles possibilités
- résoudre des ambiguïtés.

Il faut, lors de la conception d'un système, laisser à chaque partie le domaine qui lui sied le mieux et dans lequel il est supérieur à l'autre. Un exemple typique est l'étude actuelle de la reconnaissance automatique des langages naturels. C'est un problème très complexe à résoudre avec des ordinateurs alors que l'être humain n'a que peu de problèmes à formuler ses requêtes de manière un peu plus abstraite pour permettre une analyse simplifiée par la machine.

Ce dialogue entre la machine et l'être humain peut se présenter sous plusieurs formes. Stewart [Stew83] en cite huit avec leurs avantages et leurs inconvénients. Ces formes de dialogues sont les suivantes :

- questions - réponses
- remplissage de formulaires
- sélection dans un menu
- touches-fonctions
- langage de commande
- langage de recherche
- langage naturel
- techniques graphiques interactives

Stewart donne, pour chacune de ces techniques, leurs avantages et leurs inconvénients et, point intéressant, compare le temps d'apprentissage requis par un utilisateur au temps de réponse utilisé par la machine.

Chaque type de dialogue peut s'appliquer à presque toutes les opérations. Lors du choix d'une technique, il faut tenir compte d'une part des utilisateurs futurs du système et des opérations que l'on va proposer.

Les opérations effectuées sur des systèmes interactifs sont extrêmement variées, allant des possibilités offertes par un système de traitement de texte, telles que

l'insertion, la copie, l'effacement ou le remplacement des chaînes de caractères, en passant par les opérations de requêtes ou de manipulations sur une base de données jusqu'au activités les plus complexes des systèmes d'aide à la construction (CAD, CAM, VLSI-Design). On peut classer la plupart des opérations en trois catégories:

- La création et la modification de données. Par exemple dans les bases de données ou dans le traitement de texte (dans ce dernier cas, les données sont des chaînes de caractères)

- Le calcul. Les opérations arithmétiques sur de nombres ou les opérations sur des ensembles de données (union, intersection, exclusion...)

- La recherche et la navigation. L'étude des structures de données, la recherche d'information, l'étude des éléments d'une base de données

Chaque opération, qui peut atteindre un niveau de complexité extrême et des possibilités très attractives, se décompose en trois phases. Le temps de réponse de l'utilisateur, le temps de calcul et le temps d'affichage des résultats calculés par le système. Si l'on considère le rapport que l'homme a avec la machine comme un dialogue, alors on peut tenter de faire des parallèles entre le dialogue homme-homme et le dialogue homme-machine. En ce qui concerne le temps de réponse que l'on attend dans une conversation entre êtres humains, on considère que le dialogue est ininterrompu si le temps de réponse de chacun n'excède pas 2 à 4 secondes. Ceci peut donc servir de base au développement d'un système interactif. Le temps de réponse utilisé par le système est un facteur primordial qui détermine le succès et l'acceptation par ses utilisateurs d'un nouveau système. En 1968 déjà, R.B. Miller [Mill68] proposait une liste d'opérations avec des temps de réponses maximum autorisés. Ces temps vont de 0.1 seconde pour un contrôle d'activation (p. ex. l'écho d'un caractère entré au clavier) jusqu'à 60 secondes pour le chargement et le redémarrage complet d'un programme. Cette liste propose 17 différentes opérations avec leurs temps de réponses suggérés. Il est clair qu'à l'heure actuelle, les systèmes ont nettement amélioré leurs prestations et que le temps maximum suggéré par Miller pour certaines opérations semble extrêmement long. Il ne faut toutefois pas oublier que l'amélioration des prestations des ordinateurs a contribué, pour des raisons essentiellement économiques, à l'augmentation de la charge des machines et du nombre d'utilisateurs et non pas à l'amélioration des prestations, en particulier la diminution des temps de réponses.

L.H. Miller [Mill77] souligne que le plus grand facteur d'irritation provient non pas des temps de réponses élevés lors de certaines opérations mais de la variation de ces temps de réponses. Il semble que les utilisateurs préfèrent un système qui répond toujours dans un temps de 4 secondes à celui dont les temps oscillent entre 1 et 6 secondes. Ces variations proviennent de deux facteurs :

- La même opération est effectuée à deux moments différents lorsque la charge du système n'est plus la même ce qui entraîne un retard lors de la

réaction du système.

- Deux opérations, jugées de complexité identique par l'utilisateur, n'induisent pas le même nombre d'opérations effectuées par le système et donc produisent des temps de réponses différents.

Le premier point est sans aucun doute le plus irritant et c'est certainement une des raisons du succès des ordinateurs personnels qui répondent parfaitement au critère énoncé plus haut : des temps de réponses peut-être plus lents mais qui ne varient pas en fonction de phénomènes extérieurs.

3.5 L'état actuel

La situation actuelle sur le marché de l'informatique couvre un très large domaine. Il est clair que dans les milieux académiques, en particulier les universités, règne une situation qui ne reflète pas la position actuelle du marché. L'état actuel de l'informatique doit être examiné au niveau de l'industrie et du secteur des services. C'est dans les banques, dans les assurances et dans les grandes industries que se pratique l'informatique courante. Le niveau des interfaces utilisateurs dans ces milieux reste encore très bas. Il n'est pas rare de voir des utilisateurs se battre avec des messages ésotériques fournis par le système, ou compulser des manuels pour trouver le nom de l'opération qu'ils désirent effectuer. Plusieurs facteurs contribuent à cet état. En premier lieu, la situation économique actuelle retient les responsables de faire des investissements, aussi bien en hardware qu'en software, pour améliorer un matériel qui, à leurs yeux, leur donne entière satisfaction. Ces personnes ne voient pas la nécessité d'un changement de matériel coûteux qui ne contribuerait qu'à l'amélioration des conditions de travail de leurs employés. D'autre part, peu d'entreprises ont les moyens d'entretenir un secteur de développement informatique et ils sont totalement dépendants de l'entreprise qui leur fournit leur matériel informatique. Une adaptation permanente du matériel coûte très cher et ne peut être soutenue par la plupart des entreprises. En Suisse, le problème du monopole des télécommunications retient très certainement une amélioration des techniques destinées aux interfaces utilisateurs étant donné que les vitesses de transfert autorisées sur les lignes de téléphone sont très faibles et ne permettent pas un traitement satisfaisant de l'information tel qu'on le conçoit de nos jours.

3.6 L'évolution

L'évolution probable dans le domaine des interfaces utilisateurs sera dictée par l'arrivée en masse sur le marché des micro-ordinateurs. Cette tendance se confirme par l'arrivée sur le marché des systèmes très intéressants. Deux firmes ont présenté récemment des machines qui révolutionnent les principes d'interfaces utilisateurs. Il s'agit de Star conçu par la maison XEROX [Smit82] et de Lisa proposé par Apple [Will83]. Ces machines, basées sur le même principe, proposent des applications susceptibles de remplacer les tâches courantes dans un bureau. Lisa propose un jeu de six applications (application budgétaire, traitement de texte, application graphique, dessin de forme géométrique, gestion de projets et gestion de fichiers). L'idée consiste à simuler un bureau et les objets qui s'y trouvent. Dans les deux systèmes, il est fait un usage intensif de la représentation graphique d'objets (icons) qui permettent à l'utilisateur de reconnaître les objets qui lui sont familiers. Ces objets peuvent être de deux types. Ils représentent soit des données contenues dans la machine (des fichiers) soit des opérations qui peuvent être exécutées. L'exécution d'une opération s'effectue en "amenant" les données vers l'opération. Ce mouvement se fait en pointant sur les données et en les déplaçant, à l'aide d'un curseur, sur l'opération à effectuer. L'utilisateur n'a plus besoin de se souvenir des opérations disponibles et surtout ne doit plus taper sur le clavier des noms de commandes qui n'ont pas toujours un sens très clair. La représentation graphique devrait permettre de reconnaître les objets et les opérations facilement. Chaque donnée peut être visualisée dans une fenêtre séparée où de nouvelles commandes de positionnement et de modifications des données sont disponibles.

Bien que beaucoup de problèmes restent encore à résoudre (superposition des fenêtres qui diminuent la lisibilité de l'information, représentation claire et unique des objets, consistence dans le dialogue en particulier dans les méthodes d'exécution des opérations...), c'est sans aucun doute une approche juste des interfaces utilisateurs car elle est attractive et facile à apprendre pour un utilisateur débutant et reste rapide et agréable pour une personne plus expérimentée.

3.7 Conclusion

En conclusion, on peut citer les 5 points que Selander [Sela82] suggère pour améliorer les systèmes interactifs en général et les interfaces utilisateurs en particulier. Ils s'agit de :

- comprendre les buts des utilisateurs
- utiliser les méthodes courantes et habituelles des utilisateurs
- Préparer des aides à l'apprentissage du système
- présenter l'organisation du système

En ce qui concerne le dernier point, Selander voit une amélioration de l'intelligence de la machine. Nous pensons, pour notre part, que l'intelligence ne se trouve pas dans la machine mais chez l'utilisateur et que c'est la flexibilité et la rapidité qui sont les qualités de la machine.

4. Notre situation au niveau informatique

Nous avons réalisé un système expérimental à l'institut d'informatique de l'Ecole Polytechnique Fédérale de Zurich. Lilith [Wirt81], développé par le Professeur Wirth, a servi de base à la réalisation du système nommé Caliban. Cet ordinateur, qui correspond à la tendance actuelle des ordinateurs personnels, possède les caractéristiques suivantes. Il est basé sur un microprocesseur 16 bits bit-slice AM 2901. Sa capacité en mémoire centrale est de 256 kbyte dont la moitié est adressable directement. Une mémoire de 4 kbyte permet de stocker le micro-code.

Les périphériques reliés à cet ordinateur sont particulièrement intéressants pour le développement d'un système interactif. Il s'agit tout d'abord d'une unité de disque interchangeable de 10 MBytes. Certains modèles sont équipés d'un disque d'une capacité de 20 MBytes, dont 10 sont fixes et 10 sont interchangeables. Un clavier du type "qwerty" permet d'entrer des données alphanumériques et une souris munie de trois boutons permet de contrôler un curseur sur l'écran. Elle donne la possibilité de choisir des objets qui sont présentés à l'utilisateur. Son emploi en est particulièrement simple et permet une utilisation très rapide d'un système qui emploie cet outil comme moyen d'entrée. L'écran est basé sur une technique de "raster scan" avec une résolution de 594 lignes par 768 points. Les nouveaux modèles sont maintenant équipés d'un écran de format A4.

Le système d'exploitation, nommé Medos-2 [Knud83], a également été développé à l'institut d'informatique et permet l'utilisation de Modula-2 [Wirt82] qui est le seul langage à disposition sur la machine. Le système d'exploitation a lui-même été écrit dans ce langage. Un certain nombre d'utilitaires permettent l'élaboration de programmes d'applications. Un éditeur de texte, un compilateur pour le langage Modula-2 [Geis83], un "debugger" interactif, un système de gestion de fichiers sont des outils mis à disposition de l'utilisateur par le groupe du Professeur Wirth [Lili82].

5. Notre situation au niveau recherche d'information

Plutôt que de créer nos propres données pour tester notre système, nous avons décidé d'acheter des données bibliographiques sur le marché. Plusieurs raisons nous ont amenés à cette décision. En particulier, nous aurions dû soit trouver un sujet de données qui s'adaptait au problème, soit créer un jeu de données artificielles qui n'aurait certainement pas couvert la totalité des cas possibles. De plus, la qualité et la quantité des données que l'on serait parvenu à mettre au point ne justifiaient pas l'effort requis pour le faire. Nous avons donc acheté des données à la maison INSPEC qui couvre trois thèmes, la Physique, l'Electronique et le Contrôle. Cette décision s'est révélée très judicieuse car nous nous sommes heurtés à des problèmes que nous n'aurions jamais rencontrés si nous avions construit les données. Nous nous trouvions dans une situation qui correspondait à celle qui existe dans la pratique ce qui nous a permis de mieux saisir les problèmes réels. Les solutions que nous proposons ne sont donc pas basées sur des données de test qui pourraient se révéler artificielles mais correspondent exactement à la réalité.

Ces données contiennent les références bibliographiques stockées pendant un mois par la maison INSPEC. Chaque référence possède plusieurs champs tels que le titre de l'ouvrage, le ou les auteurs, la date de parution ou un résumé de l'article ou du livre original. D'autres champs qui décrivent le contenu de l'ouvrage sont également disponibles. Ce sont par exemple des mots-clés répertoriés dans un Thesaurus, ou plus simplement des concepts choisis de manière libre qui définissent le sujet couvert par l'ouvrage. Enfin des références à une classification décimale permettent d'organiser un peu plus systématiquement les données. Nous avons également acheté la classification décimale et le thesaurus défini par INSPEC étant donné qu'ils jouent un rôle primordial pour l'organisation et l'accès aux données. Ces deux structures sont définies de manière relativement stables et ne nécessitent que peu de modifications pour un nouveau jeu de données. Le premier travail consistait donc à transférer ces données et leurs structures sur Lilith. Cette organisation est décrite dans [Bärt82]. Par la suite, d'autres structures ont été implémentées pour accéder aux références bibliographiques. En particulier, la totalité des mots-clés qui forment un index libre, la liste des auteurs et la liste des maisons d'édition forment trois structures d'accès suplémentaires.

Nous sommes toutefois restés conscients de la dépendance que le système pouvait avoir par rapport à ces données. C'est pourquoi, nous avons tout de même essayé d'utiliser le système avec d'autres jeux de données, crées automatiquement à cet effet. Nous avons en particulier construit un ensemble de données formées de modules de définition et de modules d'implémentation écrits en Modula-2. Les structures d'accès contiennent, dans ce cas, les objets définis dans les modules tels que les procédures, les variables ou les types. De chaque point de la structure, il est possible d'accéder à la définition ou à l'implémentation de l'objet concerné dans le texte du programme. De plus, le langage définit une structure de réseau, puisque chaque module doit importer les objets qu'il utilise et qui sont définis dans d'autres

modules et il doit lui-même exporter les objets qu'il définit. Cette structure de réseau est également implémentée dans les structures de données.
Nous verrons dans les prochains chapitres comment se présentent plus précisément ces données.

6. Principes fondamentaux d'un système de recherche

6.1 L'information

Le rapport entre information et données

Dans tout système de recherche d'information, il existe une interaction entre l'utilisateur final et le système. Le système contient des données qui vont être sélectionnées puis présentées à l'utilisateur. Un processus de transfert d'information entre le système et l'utilisateur s'établit. Pour bénéficier de l'information qui circule, chaque personne se crée un modèle de son monde qui est organisé et structuré en catégories et concepts. Ce modèle est parfois appelé "Structure de Connaissance". Dès que l'environnement soumet ce modèle à des contraintes qui lui créent des ambiguités, la personne se sent obligée de réagir soit en réorganisant son modèle par la réflexion, soit en acquérant des connaissances supplémentaires d'un système extérieur qui lui permettent d'éliminer ces ambiguités. Cet apport de l'extérieur s'effectue en étudiant des *données* d'un autre système qui seront intégrées dans le modèle de la personne. Les données qui sont intégrées dans le modèle et qui contribue à une augmentation de la connaissance indivuduelle sont appelées *Information.*

Prenons un exemple tiré de Heaps [Heap78]. Supposons qu'une personne connaisse la valeur de Π avec 4 décimales (3.1415) et qu'elle désire connaître quelques décimales supplémentaires. Si cette personne trouve une donnée indiquant la valeur de Π égale à 3.1415, l'information qu'elle pourra en déduire est nulle. Par contre, si elle trouve que Π vaut 3.1415927, elle va pouvoir intégrer l'apport de données (927) dans sa structure de connaissance et donc ainsi l'augmenter. Heaps indique que le contenu de l'information est quantifiable et qu'il est égal à -log(p) où p est la probabilité à priori avec laquelle la personne peut prévoir les données. Ainsi, pour la cinquième décimale, cette probabilité est plus grande ou égale à 1/10 puisque le choix possible va de 0 à 9.

Dans le processus de perception, les données sont transformées en information par les structures de connaissance. Certaines données ne contiennent aucune information, comme c'est le cas pour les quatre premières décimales de Π. D'autres peuvent ne pas être reconnues, ou ne transforment pas la connaissance actuelle, par exemple si la personne ne s'intéresse qu'à la cinquième décimale et non aux deux suivantes.

Pour un système de recherche d'information, Ingwersen [Ingw83] distingue deux types de données:

- celles qui concernent l'emploi du système.
- celles qui sont contenues dans le système, et donc qui intéressent l'utilisateur.

L'utilisateur doit avoir des connaissances dans les deux domaines pour réussir à résoudre son problème et trouver l'information qu'il cherche. L'état de connaissance requis pour utiliser le système doit être minimum pour en permettre l'emploi à un très large éventail d'utilisateurs. La structure de connaissance de l'utilisateur peut être ramené à un domaine élémentaire si l'apport de données fourni par le système est suffisamment grand pour pratiquer un auto-apprentissage et donc employer le système très rapidement. Les connaissances requises à priori peuvent alors se résumer à un minimum tel que savoir lire, écrire ou taper sur clavier. En ce qui concerne le niveau de connaissance dans le domaine de recherche, il faut évidemment que celui du système soit plus grand que celui du chercheur pour satisfaire les besoins de la recherche. D'autre part, il faut que la structure de connaissance du système se rapproche de celle de l'utilisateur, ou du moins qu'elle soit compréhensible pour qu'il puisse l'interpréter. Voyons comment doivent se présenter les données et leurs structures pour que l'utilisateur puisse en retirer un maximum d'information en un minimum de temps.

Définition des objets du système

Un système de recherche d'information travaille avec des objets (items). Un objet est un mot abstrait qui peut couvrir plusieurs entitées réelles telles que un document (un livre, une lettre, un mémo, ...), une référence à un document (liste de termes construite manuellement à partir d'un document) ou tout simplement un terme qui est utilisé dans une référence. Par abus de langage, il arrive souvent que l'on appelle une référence à un document tout simplement un document. Ce raccourci n'engendre toutefois pas de confusion étant donné que les deux entités se réfère finalement au livre ou à la lettre considérée. Une référence à un document peut être représentée sous la forme d'une liste linéaire formée de termes. Ces termes sont classés et ordonnés selon différents attributs. Ces attributs précisent le type de l'information que contient le terme considéré. C'est un ensemble fini et ordonné dont les éléments peuvent être "auteur", "titre" ou "résumé" pour des références bibliographiques ou encore "en-tête", "adresse", "paragraphe" ou "salutations" pour une lettre.

On distingue deux types d'attributs dans une référence. Ceux qui contribuent à une description formelle du texte et ceux qui décrivent plutôt son contenu. Parmi les premiers, on peut citer la date de création ou de parution du texte, le ou les noms des auteurs, le lieu de parution ou encore une indication qui précise l'endroit où se trouve le document actuellement. Il ne s'agit là que de données formelles qui ne donnent aucun renseignement quant au contenu du document. Les autres termes contribuent à décrire ce contenu. Des mots-clés, un résumé, un en-tête de lettre font partie de cette catégorie de termes. Nous verrons plus loin quelles sont les différentes caractéristiques de ces attributs et quelles méthodes il faut appliquer pour rechercher les termes qu'ils contiennent. Ce que l'on peut déjà affirmer, c'est que les attributs d'une description formelle s'apparentent aux données que l'on trouve dans un système de gestion de base de données, alors que ceux de la description du contenu

sont les éléments typiques d'un système de recherche d'information.

Définition du niveau des objets

La nature et la structure de ces objets diffèrent suivant le niveau dans lequel on se trouve. On peut toutefois établir un rapport entre chacun de ces niveaux en partant du plus simple au plus complexe. Dans beaucoup de systèmes actuellement (traitement de texte, éditeur,...) l'objet de base que l'on considère est le caractère, à savoir un élément de l'alphabet, un chiffre ou encore quelques caractères spéciaux tels que les signes de ponctuation. Lorsque l'on considère les différents systèmes actuels de recherche d'information, on s'aperçoit qu'ils ont presque tous la possibilité de faire des recherches au niveau du texte et de la position dans le texte. Les opérateurs tels que ADJ, WITH, SAME de STAIRS [Stai76] ou (W) de DIALOG [Bour79] indiquent que les opérandes, à savoir des termes, doivent se trouver à une certaine place l'un par rapport à l'autre. L'utilisateur effectue des opérations au niveau des caractères et de leur position respective dans le texte. Le niveau de définition d'un objet de base ne doit pas être aussi primitif pour un système de recherche d'information. Il s'agit de rester à un niveau conceptuel plus haut où il est possible à un utilisateur de lier l'objet qu'il manipule avec un concept logique. Ce concept logique sera basé sur ce qu'on peut appeler la *sémantique* de l'objet. Chaque objet représente ainsi une *idée* pour l'utilisateur et non plus une suite de caractères telle qu'on pourrait l'avoir dans certains systèmes de recherches. Si l'on désire chercher des documents selon un certain sujet et que ce sujet est déterminé par un terme composé de plusieurs mots, l'utilisateur ne devrait pas avoir à se soucier de la position de ces mots l'un par rapport à l'autre. Si l'on prend par exemple le sujet "Information Retrieval", il n'est pas formé des deux termes "Information" suivi de "Retrieval" mais doit être considéré comme une seule entité. Trop souvent, le concept de terme index est associé à celui de mot. Le mot se situe au niveau de la syntaxe définie par le language considéré. Le terme index, lui, se situe à un niveau logique plus élevé, c'est-à-dire au niveau de la sémantique de la phrase. Il est donc tout à fait juste de considérer que le terme "Information Retrieval" est un tout et que l'utilisateur ne devrait pas devoir se préoccccuper, lors de sa recherche, de problèmes syntaxiques.
Prenons quelques exemples :

- Un utilisateur veut avoir des informations sur la "recherche". Dans un système conventionnel, il devra entrer sur un clavier les différentes lettres qui composent le mot "recherche". Il peut s'avérer que les informations qu'il désire obtenir ne sont pas indexées par ce terme mais plutôt par "Retrieval" ou encore "Information Retrieval".

- Un autre exemple célèbre est celui de la personne qui désire trouver les ouvrages écrits par "Tchebychev". Il y a plusieurs transcriptions du nom de cet auteur si l'on n'utilise pas l'alphabet cyrillique, comme par exemple "Czebyczew", "Tschebyschew", "Chebichev" etc.

- Un troisième exemple est celui de la manière de donner une date. Que l'on se trouve en Europe ou aux Etats-Unis, l'ordre de spécification du jour, du mois et de l'année diffère, de plus il n'est pas évident de savoir s'il faut écrire "1983", "June 1983", "6/83" ou " '83" pour désirer obtenir une publication sortie en juin 1983.

On voit par ces exemples que l'utilisateur qui se trouve au niveau des caractères, ou plus précisément au niveau lexical ou syntaxique pour spécifier des objets se heurte à ce genre de situation. Si par contre il travaille à un niveau plus élevé, ces inconvénients peuvent être éliminés. Voyons tout d'abord quels sont ces niveaux, nous montrerons ensuite comment on peut aider l'utilisateur en choisissant un niveau approprié.

La nature des objets peut être définie de la manière suivante :

NIVEAUX	OBJETS
Lexical	Caractères
Syntaxique	Mots (Symboles)
Sémantique	Termes
Structurel	Structures

Sur le niveau sémantique vient se greffer un niveau logique que l'on pourrait appeler "Organisationnel" ou "Structurel". Chaque entité sémantique peut faire partie d'une structure définie par le système. Si l'on prend les termes contenus dans un système bibliographique, ils apparaissent dans une ou plusieurs structures d'index et en tant qu'éléments d'un ou plusieurs documents. Le terme "computer" intervient par exemple dans un thesaurus, dans une classification décimale et dans tous les documents se rapportant aux ordinateurs. La sémantique liée à ce terme est toujours la même, la relation avec d'autres éléments et par conséquent son niveau structurel est différent.

La plupart des systèmes actuels travaillent au niveau lexical voire syntaxique. En effet, l'utilisateur doit s'occuper de chaînes de caractères s'il veut obtenir des informations sur un terme qu'il a choisi ou qu'il a trouvé. Certes, tous les systèmes ne travaillent pas que sur ce mode et lorsqu'il s'agit d'accéder à de l'information par des fichiers d'accès inversés, plutôt que de faire de la recherche au niveau du texte, l'utilisateur se situe entre les niveaux syntaxiques et sémantiques. Malheureusement, les outils, que les systèmes actuels lui fournissent pour travailler sur cet autre plan, restent les mêmes. Très peu d'améliorations ont été offertes dans ce domaine. Chaque système continue à fournir la possibilité de travailler avec des mots d'index dont la racine ou la terminaison sont coupées ou avec d'autres artifices du genre "Compute? ??" signifiant dans le système DIALOG "chercher tous les termes commençant par "Compute" et suivit d'au maximum deux lettres derrière cette

racine". Cette approche me paraît fausse car l'utilisateur s'intéresse alors à un *concept* et non plus une chaîne de caractères et il devrait avoir à sa disposition un moyen de rechercher des termes qui sont sémantiquement semblables ou encore mieux "structurellement" semblables. Il ne s'agit pas de pouvoir chercher les termes qui satisfassent "Retriev*", où l'étoile signifie n'importe quelle chaîne de caractères, mais où l'on peut dire "Tous les termes semblables à "Retrieval" ", ceci incluant "Retrieval", "Recherche", "Information Retrieval", "Information Retrieval Systems" et peut-être même "Query".

L'idée de concept permet aussi de traiter le problème soulevé par Bell et Jones [Bell80] et qu'ils appellent le syndrome de la "Red Hot Water Bottle". Dans ce terme, il n'est pas possible de savoir à priori si l'on parle de bouteille rouge d'eau chaude ou de bouillote rouge. En fait, le problème provient également de l'étude du concept au niveau syntaxique. Si un document traite de ce concept et qu'il est indexé par les termes "red" et "hot water bottle", ces termes étant présents dans la structure d'accès, le problème ne se pose plus car l'utilisateur y accèdera directement et de manière unique par la structure d'accès.

Définition des structures d'information

L'information peut être structurée sous trois formes: des listes (ordonnées ou non), des arbres ou des réseaux. Selon la nature de cette information et de sa structure inhérente, une de ces formes sera plus adaptée qu'une autre pour structurer l'information. Si on choisit une structure inappropriée, soit on pert de l'information concernant la structure soit on en rajoute de manière superflue.

La structure la plus simple est une liste linéaire telle que se présente une liste alphabétique de noms d'auteurs.

Très souvent les gens ont tendance à structurer l'information d'une manière hiérarchique. Des concepts généraux sont définis puis divisés en groupe plus spécifiques qui à nouveau seront divisés pour arriver finalement à des *atomes* d'information. Telle se présente une classification décimale. Les arbres généalogiques ou les tables des matières sont d'autres exemples typiques de cette représentation. Il semble naturel à l'être humain de diviser des concepts généraux en sous-concepts plus détaillés. De plus, les représentations hiérarchiques sont suffisamment utilisées pour permettre à quelqu'un qui n'a pas défini une structure, de comprendre aisément sa signification.

Malheureusement, la nature de l'information est parfois plus complexe qu'une stricte hiérarchie. Un réseau est souvent la structure appropriée. Un thesaurus est un exemple typique où une quantité de connections (p. ex. "related terms") ne satisfont plus à cette hiérarchie. Un système de recherche doit être capable de traiter non seulement les listes mais également les structures hiérarchiques et les réseaux.

Visualisation des structures d'information

Comme beaucoup de structures d'informations (classification, thesaurus) sont complexes et de taille relativement grande, il s'agit de savoir comment les présenter à l'utilisateur. La quantité d'information qu'un utilisateur peut percevoir est relativement limitée; de plus, les dimensions des écrans ou autres systèmes de sorties sont très restreintes. Le cas de la liste linéaire se résout naturellement étant donné qu'elle nécessite une visualisation séquentielle. La seule option à prendre doit définir si l'on veut la représenter horizontalement ou verticalement. Ceci est généralement une affaire de goût et ne peut être influencée de manière objective que par la quantité d'information que l'on peut présenter dans les deux options. Dans le cas des deux autres types de structures, elles sont de nature bi-dimensionnelles et il s'agit de les représenter sous cette forme en gardant une logique naturelle et en perdant le moins d'information possible. Il existe plusieurs possibilités de résoudre ce problème. On peut tout d'abord concevoir une représentation en "identation", c'est-à-dire où chaque ligne contient un terme et où les sous-concepts sont décalés sur la droite par rapport au concept plus général. Cette solution a l'inconvénient de distancer les termes de même niveau et elle ne permet pas une réalisation facile pour les réseaux. Une deuxième possibilité consiste à présenter cette structure graphiquement en deux dimensions. Dans ce cas, la quantité d'information que l'on peut donner est encore plus restreinte que dans le premier cas et les possibilités de déplacement sont dans plusieurs directions. Pour ce faire, Herot [Hero80] prévoit un effet de zooming sur l'information représentée spatialement en deux dimensions. Malheureusement, une grande partie de l'information est perdue lors de la navigation dans la structure car on ne voit plus les termes contenus dans les noeuds de la structure. On peut sinon déplacer la fenêtre qui contient cette structure représentée graphiquement (Fig 6.1). Cette solution a l'avantage de conserver la structure et de présenter constamment les concepts qui sont définis. De plus elle est naturelle à l'utilisateur. Si l'on choisit cette présentation graphique, beaucoup d'arêtes vont se connecter à des noeuds en dehors de l'écran. Ceci est valable pour les trois types de structures considérées. Toutefois, les arêtes d'une liste et d'une structure hiérarchique amènent invariablement à un noeud du voisinage immédiat dans la structure, à savoir soit un successeur ou un prédécesseur dans le cas d'une liste soit un père ou un fils dans le cas d'une structure hiérarchique. Les arêtes d'un réseau peuvent conduire eux à un noeud relativement distant de l'endroit où l'on se trouve, la distance étant ici considérée au niveau de la représentation graphique. On peut penser toutefois que ces arêtes sont susceptibles de troubler l'utilisateur et décider de ne pas les afficher.

En ce qui concerne les réseaux, il est toujours possible de définir au moins une structure hiérarchique qui en comprend tous les noeuds. Berge [Berg58] donne un théorème qui prouve qu'un graphe $G = (X,U)$, X étant l'ensemble des noeuds et U l'ensemble des arêtes, possède un graphe partiel qui est un arbre si et seulement si ce graphe est connexe et que le nombre de noeuds $|X|$ est plus grand ou égal à deux. L'exemple le plus simple est donné par l'arbre défini lorsque que l'on traverse tous les noeuds du réseau.

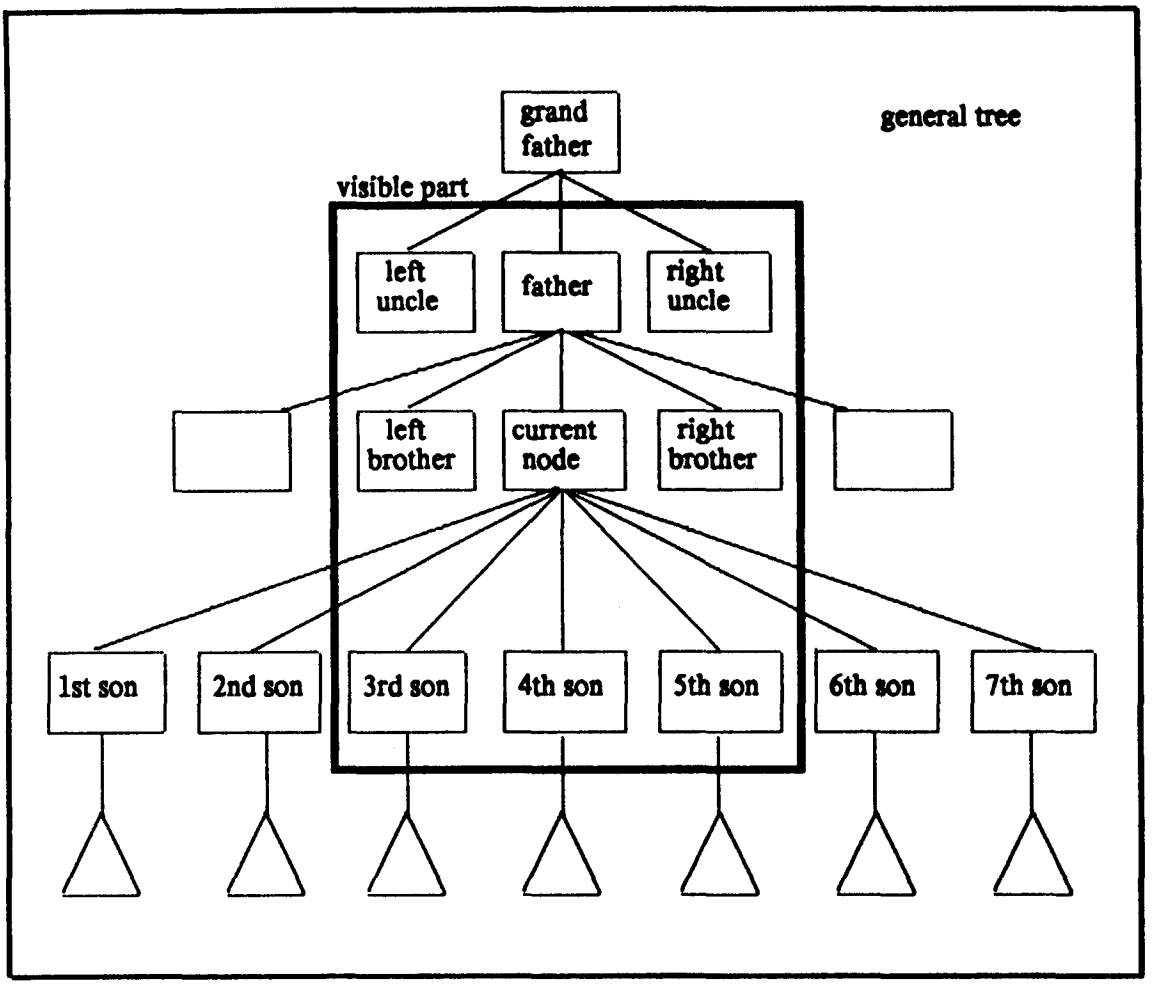


Fig 6.1 La partie visible d'un arbre

Dans le premier cas, on décide de traverser tous les noeuds du réseau. Ceci nous amène à perdre l'information que A et D sont connectés directement et qu'une arête de D mène à C. Dans le deuxième cas, on parcourt toutes les arêtes du graphe. On passe alors plusieurs fois par le même noeud et l'arbre contient des noeuds identiques. Cette solution a l'avantage de conserver toute l'information. Ces deux cas apparaissent à la figure 6.2.

Le cas d'un thesaurus est même plus complexe puisque les arêtes n'ont pas toutes le même type. La relation "Broader term", "Narrower term" forme un arbre tandis que la relation "Related term", qui vient se greffer par dessus, forme un réseau.

C'est une de ces hiérarchies que l'on peut présenter à l'utilisateur, en dépit du fait que, dans plusieurs cas, d'autres connections sont présentes. De cette manière, des arbres représentés graphiquement sont montrés à l'utilisateur plutôt que des réseaux. Toutefois, l'utilisateur doit avoir la possibilité de demander les autres relations qui forment le réseau. Nous verrons dans le chapitre suivant comment on peut présenter ces structures, en particulier les arbres et les réseaux. Mais nous aimerions d'abord définir les principes que l'on doit appliquer au sujet de la recherche d'information et des techniques d'interfaçage.

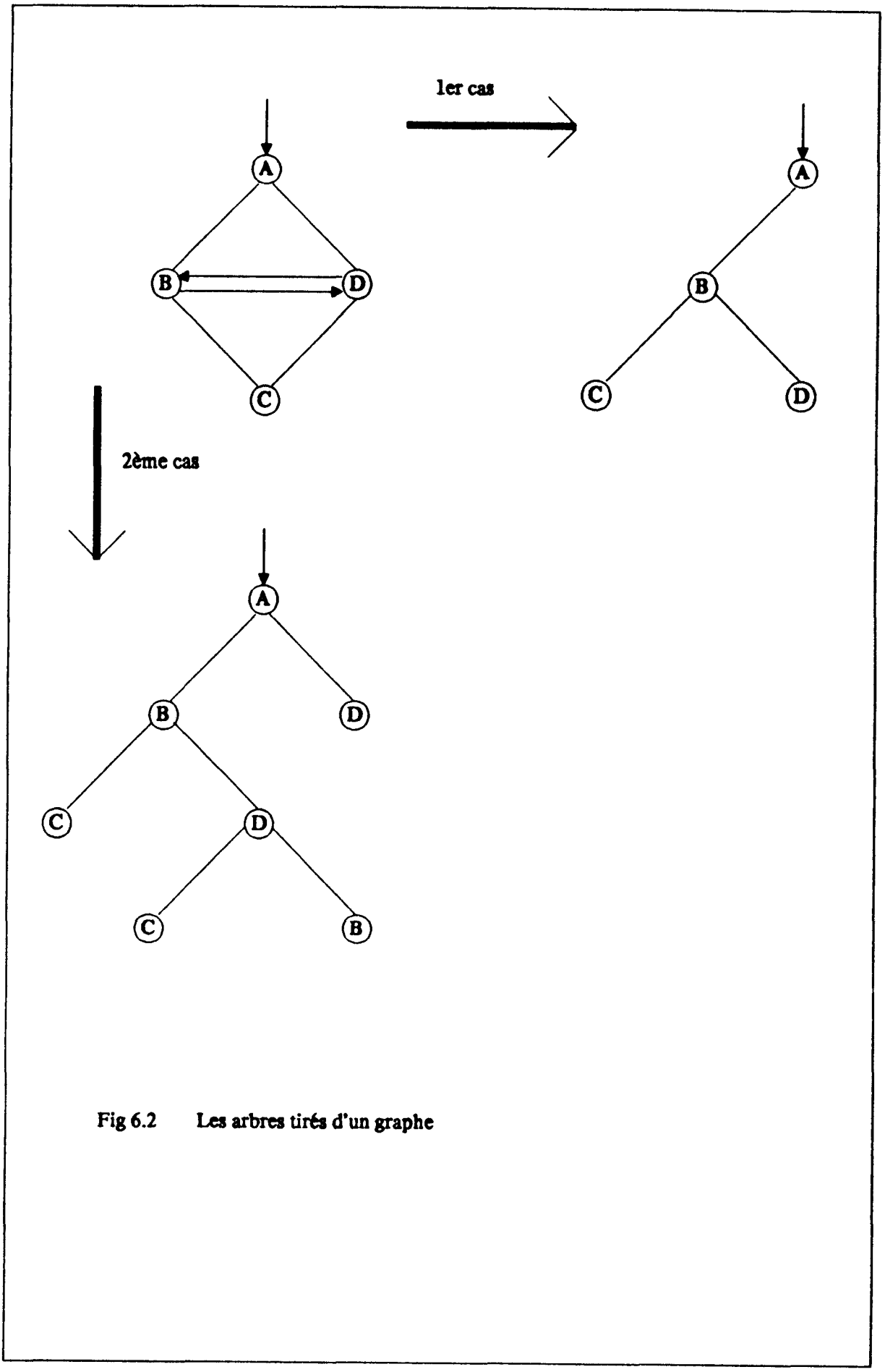


Fig 6.2 Les arbres tirés d'un graphe

6.2 Techniques de recherche

Introduction

Une des clés du succès d'un système consiste à étudier le comportement d'un utilisateur potentiel avant d'entreprendre la conception générale et en particulier la conception de l'interface utilisateur qui sera le représentant du système. Pour réaliser cette idée, il faut analyser le comportement d'une personne cherchant de l'information; nous prendrons des données bibliographiques basées sur une documentation technique. Plusieurs situations sont possibles, mais l'on peut tirer certains critères de l'attitude d'une personne dans une petite bibliothèque, soit personnelle, soit publique, dans laquelle l'utilisateur a un accès direct aux ouvrages qu'il désire consulter. Ce genre de bibliothèque est appelée ouverte. Ceci va nous permettre de définir comment travaille cette personne et quels sont les outils nécessaires pour améliorer ou accélérer ce type de recherche que j'appelerais **principe de recherche simple**. Une autre étude doit être faite pour l'utilisateur qui n'a pas un accès direct à l'information, comme c'est le cas dans des bibliothèques fermées ou dans les grands systèmes de recherches d'information. Il s'agit dans ce cas d'étudier tout d'abord les structures d'accès et de préparer des questions avant d'accéder à l'information. C'est ce que l'on peut appeler le **principe de recherche combinée**. Il faudra alors déterminer quels sont les rapports entre ces deux principes pour pouvoir dégager une méthode susceptible de satisfaire aux exigences des deux cas.

Principe de recherche simple

Une personne cherchant un document ou de l'information dans une petite bibliothèque agit de la manière suivante. Le premier point consiste à étudier l'organisation de la bibliothèque qui représente en fait la première et éventuellement l'unique structure d'accès. Savoir où et comment sont rangés les ouvrages et quels sont les moyens systématiques de retrouver un document. Ceci peut se faire en étudiant la classification de la bibliothèque ou son ordre de rangement. Puis la personne va se diriger vers les rayons pour étudier quelques livres et s'intéresser aux ouvrages qui sont présents. Ce faisant, il arrive très souvent de tomber sur un ouvrage qui paraît intéressant. Les facteurs qui dirigent cette découverte peuvent être très différents, allant du titre ou de l'auteur jusqu'à la couleur de la couverture. En tout les cas, la personne en question va se saisir du livre et commencer à l'étudier. Si celui-ci est hors du sujet, elle va très rapidement décider de passer plus loin. Si au contraire, il présente un certain intérêt, une étude plus approfondie aura lieu, éventuellement une lecture de certains passages ou d'un résumé. Il arrive très souvent que l'on s'intéresse aux références bibliographiques généralement citées à la fin de l'ouvrage. Lorsque la personne aura effectué cette collecte d'information, c'est-à-dire la mémorisation des données du livre et des références qu'il comporte, elle poursuivra son passage devant les rayons de la bibliothèque.

Ce processus est résumé à la figure 6.3.

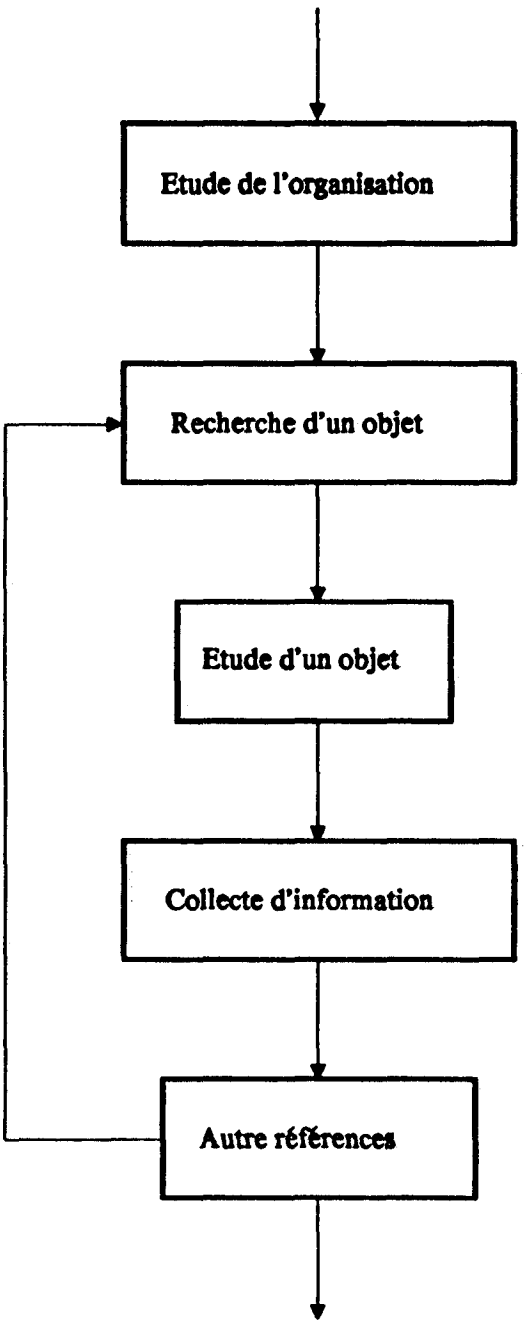


Fig 6.3 Recherche simple

Les deux activités intéressantes à noter dans ce cas sont d'une part le *survol* que fait l'utilisateur en passant devant les rayons et d'autre part la *collecte d'information* tout au long de ce survol. Dans un cas absolument non-automatisé, la personne est gênée car elle doit noter explicitement les renseignements qu'elle trouve. D'autre part, une solution automatisée permet d'améliorer le survol de l'information. Ce processus de survol et de collecte dinformation, lorsqu'il se fait manuellement, est très lent, tout d'abord parce que les données sont dispersées tout au large de la bibliothèque et ensuite parce que l'étude d'un document nécessite de prendre le livre, de l'ouvrir à différentes pages, de trouver l'information qu'on cherche, de la noter puis de reposer le livre. Une liste contenant les informations élémentaires d'un document est déjà nettement plus rapide à étudier. De plus, le processus qui consiste à suivre des références bibliographiques devient vite fastidieux. Si l'on trouve plusieurs références intéressantes, il faut les étudier séparément. Chacune peut apporter plusieurs nouvelles références intéressantes. Le processus devient complexe et il est très difficile de faire des recoupements pour trouver quelles références sont citées plusieurs fois. Ces quelques points peuvent être améliorés grandement dans un système automatisé.

Principe de recherche combinée

Passons maintenant à l'étude de ce que l'on a nommée la recherche combinée. Le cas se dissocie de celui de la recherche simple avant tout par le fait que la personne qui recherche de l'information n'a pas un accès direct à l'information. C'est-à-dire qu'il n'est pas possible de compulser certains documents pour en tirer de l'information ou des références utilisables ultérieurement. Par contre l'utilisateur a très souvent des *structures d'accès,* plus évoluées que dans une bibliothèque ouverte, qui lui permettent d'orienter sa recherche et de rassembler des *termes* contenus dans ces structures et donnant accès aux références ou aux ouvrages eux-mêmes. Il est important de remarquer que généralement l'accès aux données se fait depuis plusieurs points ou termes à la fois et non plus ponctuellement comme c'était le cas dans la recherche simple. Ceci explique le nom de recherche combinée, à savoir qu'il est possible de former une *question* qui est une *combinaison de termes index.* Examinons l'attitude d'une personne cherchant à formuler une question dans un système de recherche de données.

La partie "étude de l'organisation" du schéma précédent se trouve inclue maintenant dans le survol des structures d'accès que fait la personne pour trouver les premiers termes qui correspondent à ses souhaits. Intervient ensuite la phase de formulation ou de modification de la question qu'on va poser au système. La recherche faite par le système produira un ensemble de documents ou de références à des documents que l'utilisateur pourra compulser. Suite à cette étude des résultats fournis, il pourra soit modifier ou améliorer sa question initiale, soit stopper là sa recherche et demander l'impression de ces résultats ou commander directement les références des ouvrages qu'il a reçues. La figure 6.4 indique les différents stades de ce processus.

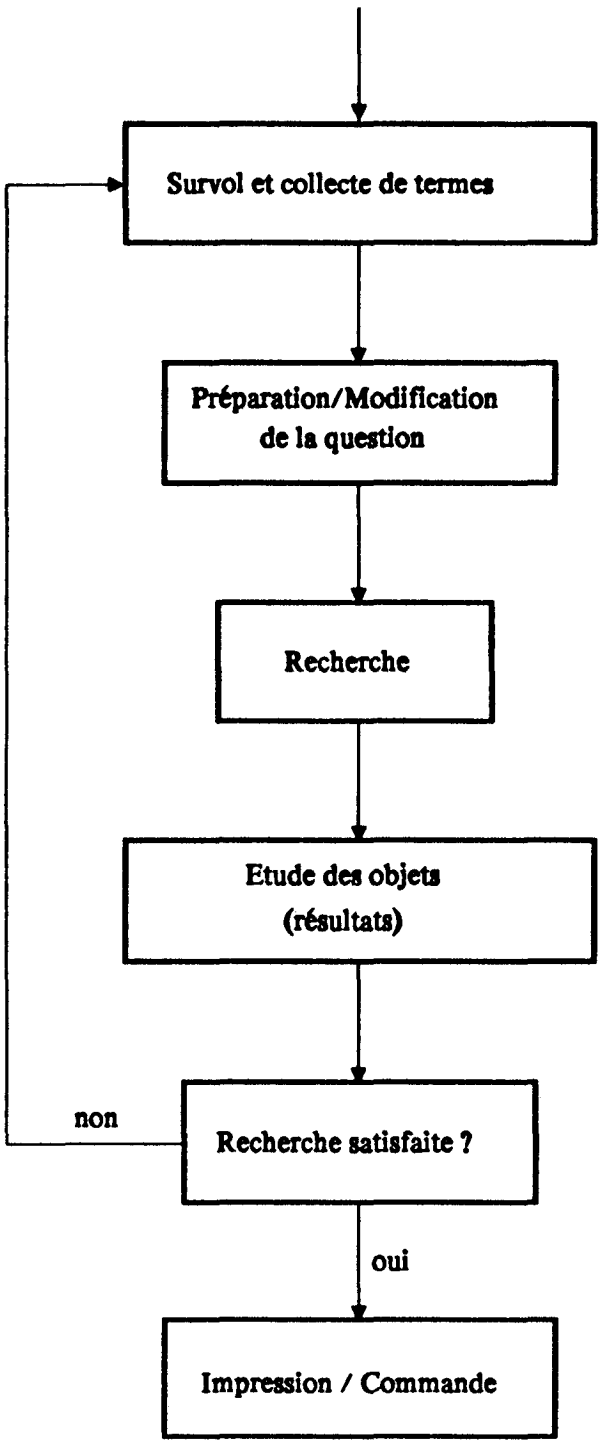


Fig 6.4 Recherche combinée

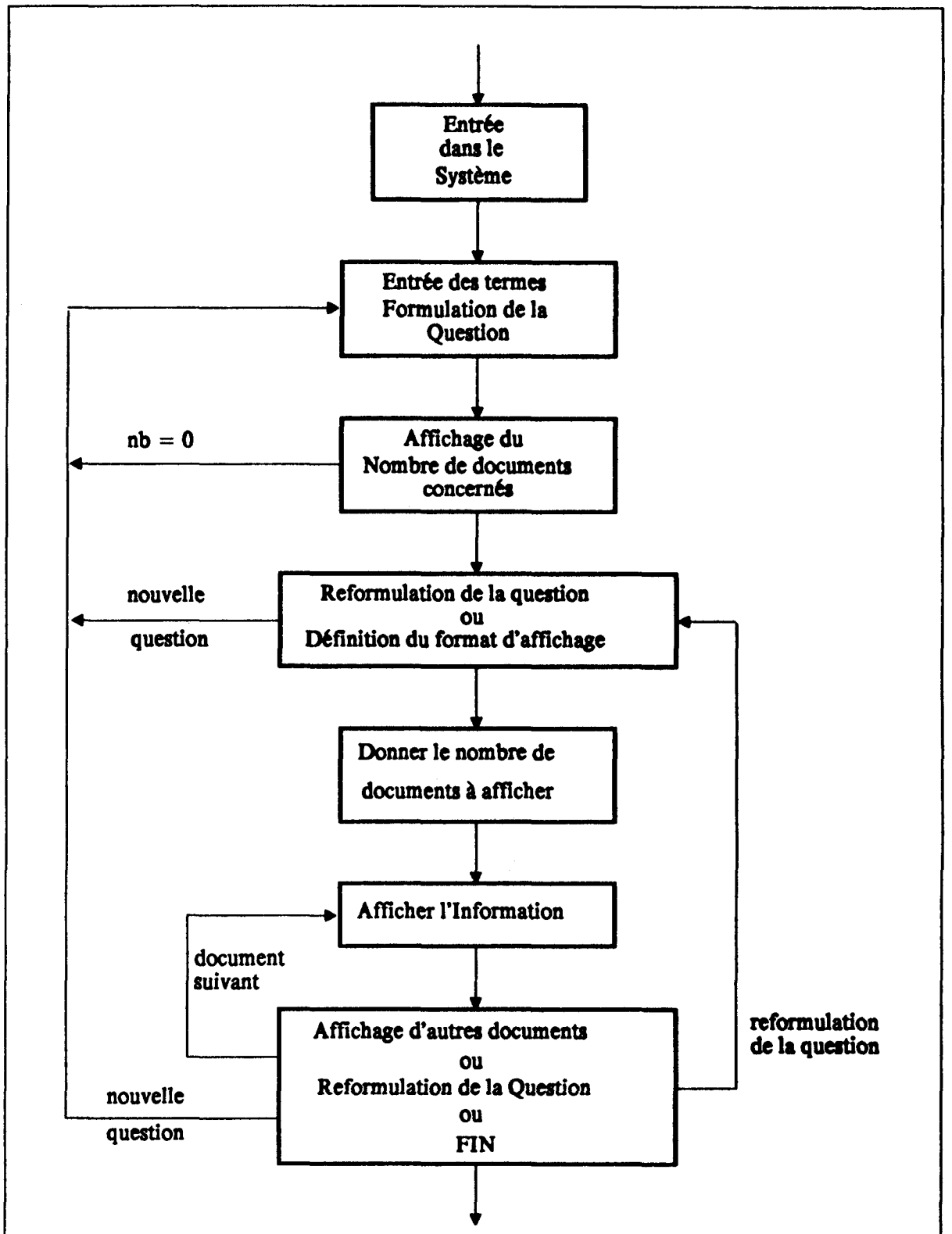


Fig 6.5 Schéma de Salton

Il est intéressant de remarquer que la recherche faite avec un système de RI s'apparente fortement à celle que l'on fait dans une bibliothèque fermée. La phase de recherche est remplacée par la commande des documents à la personne qui ira les chercher sur les rayons et fréquemment une étude rapide permet d'éliminer les documents hors du sujet. Cette même étude permet de retrouver d'autres documents qui sont cités comme références dans ceux que l'on étudie.

Principe général d'un système de recherche

Il s'agit maintenant de tirer une synthèse de ces différentes attitudes d'utilisateurs et de définir un schéma général susceptible de couvrir les besoins de l'utilisateur dans ces cas. Plusieurs personnes se sont déjà penchées sur la définition d'un tel schéma. Salton [Salt83] par exemple propose un système relativement intéressant qui définit clairement les besoins d'un utilisateur dans pareil cas. Il suggère que lors d'une session interactive en recherche, deux types d'opérations soient effectuées par l'utilisateur, d'une part la soumission de questions et la visualisation des réponses obtenues, d'autre part la formulation de ces questions, aidée parfois par quelques structures d'index telle qu'un Thesaurus. Il en déduit le schéma que l'on trouve à la figure 6.5.

La première objection que l'on peut faire à ce schéma est qu'il ne montre pas clairement quand se situe le processus de recherche. Il semble logique qu'il intervienne entre l'entrée des termes et l'affichage du nombre de documents concernés. D'autre part, la formulation et la reformulation d'une question apparaissent à deux endroits différents alors qu'il s'agit du même processus.

Heaps [Heap78] propose un schéma très simple mais qui apporte un point intéressant.

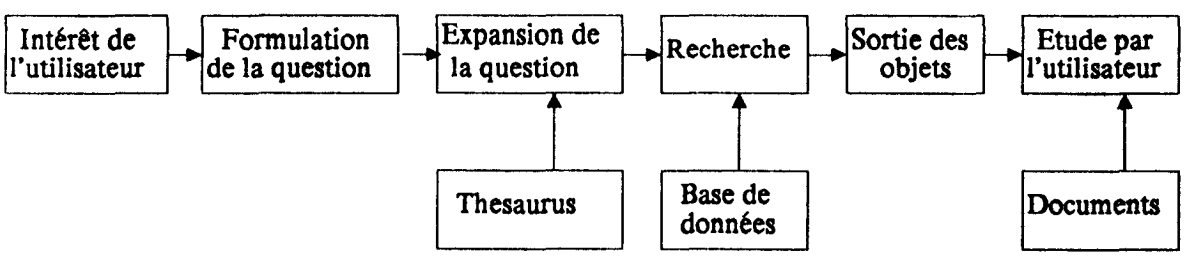


Fig 6.6 Schéma de Heaps

Il propose une phase de *formulation de la question* et une autre d'*expansion de cette question*. L'utilisateur a donc la possibilité de choisir un terme et de préciser qu'il désire inclure automatiquement dans sa question tous les termes du thesaurus qui sont en rapport avec celui qu'il a choisi. C'est un point très intéressant car il permet de réduire le nombre de termes que l'utilisateur aurait manqués lors de la préparation de la question. Il faut remarquer toutefois que l'intégration de *tous* les termes, liés d'une manière quelconque au concept que l'on a donné, peut conduire à

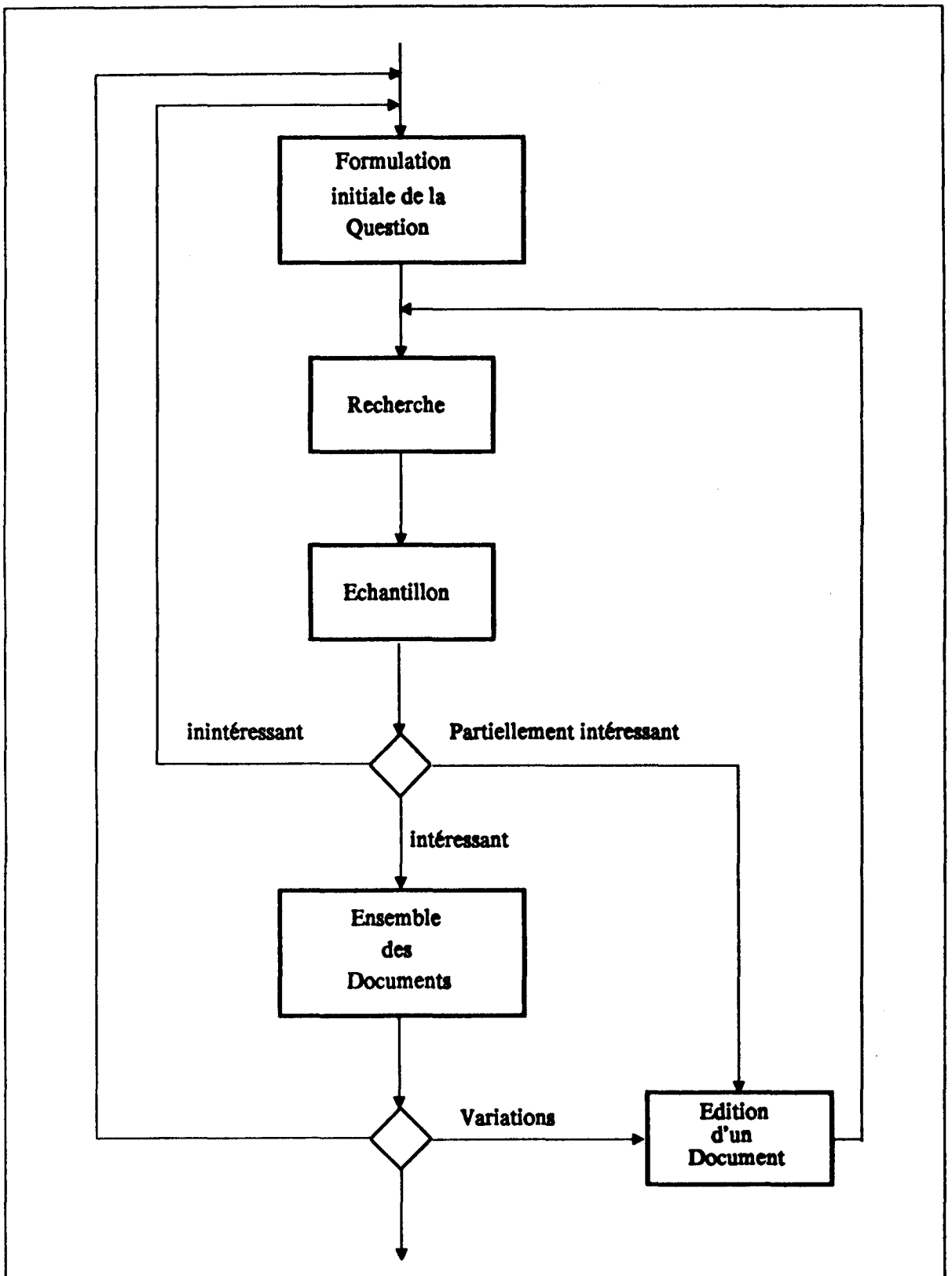


Fig 6.7 Schéma de Frei

une question très longue et qui requiert un très long processus de recherche. Il serait souhaitable de donner le choix à l'utilisateur de préciser individuellement quels termes il désire intégrer dans sa question.

Frei [Frei82] donne également un schéma (Fig 6.7) qui a l'avantage d'aller un peu plus loin dans l'intégration et l'étude de l'utilisateur. Il amène aussi l'idée qu'une question et une référence à un document sont des objets relativement semblables et qu'une telle référence peut être utilisée comme question dans le cadre d'une amélioration de la recherche.

Il introduit la notion d'échantillon qui est un document représentant l'ensemble des documents trouvés et susceptible d'être modifié puis utilisé à nouveau comme question. Ce phénomène active sensiblement la rapidité avec laquelle le chercheur trouvera les documents qui l'intéressent. La notion de "partiellement intéressant" semble être juste puisque dans la plupart des cas l'utilisateur ne sait pas à priori ce qu'il recherche et qu'il ne peut décider, sans une étude complète, si l'objet est très intéressant.

Ces trois schémas pêchent toutefois par leur excessive généralité à propos de la formulation de la question. Comme nous l'avons dit auparavant, c'est une des phases les plus importantes et les plus longues de la recherche car l'utilisateur ne sait pas à priori quels sont les termes disponibles. Nous allons tâcher de définir un schéma général qui intègre d'une part le principe de dualité de la recherche (simple et combinée), d'autre part qui fait ressortir les éléments intéressants des schémas que l'on a vus (principe itératif d'amélioration de question, expansion des termes, analogie question-document,...) et enfin qui essaye d'exprimer plus précisément la formulation et la reformulation de la question.

Schéma général

Après l'entrée dans le système, le schéma, intitulé schéma général (Fig 6.8), se divise en deux parties; soit l'utilisateur connaît déjà le système et ses structures et il a une *idée précise* de ce qu'il va faire et de la question qu'il va poser (sans obligatoirement connaître les termes de la requête), soit il ne sait pas comment on peut formuler une question et surtout quel est le vocabulaire et quelles sont les structures d'accès disponibles. Il suivra alors la branche nommée *idée imprécise.*

Plusieurs structures d'accès peuvent être disponibles, comme par exemple un thesaurus, une classification, une liste d'auteurs, etc. Cette séparation des structures n'existe pas obligatoirement et tous les termes d'un vocabulaire fixe peuvent être regroupés dans une même structure, auquel cas l'utilisateur n'aura pas besoin d'effectuer ce choix. Il me paraît toutefois important de faire cette différence qui permet d'accélérer la recherche des termes étant donné que les structures gardent des dimensions plus raisonnables.

Vient ensuite le *survol* de ces structures. L'utilisateur doit absolument, dans le cas

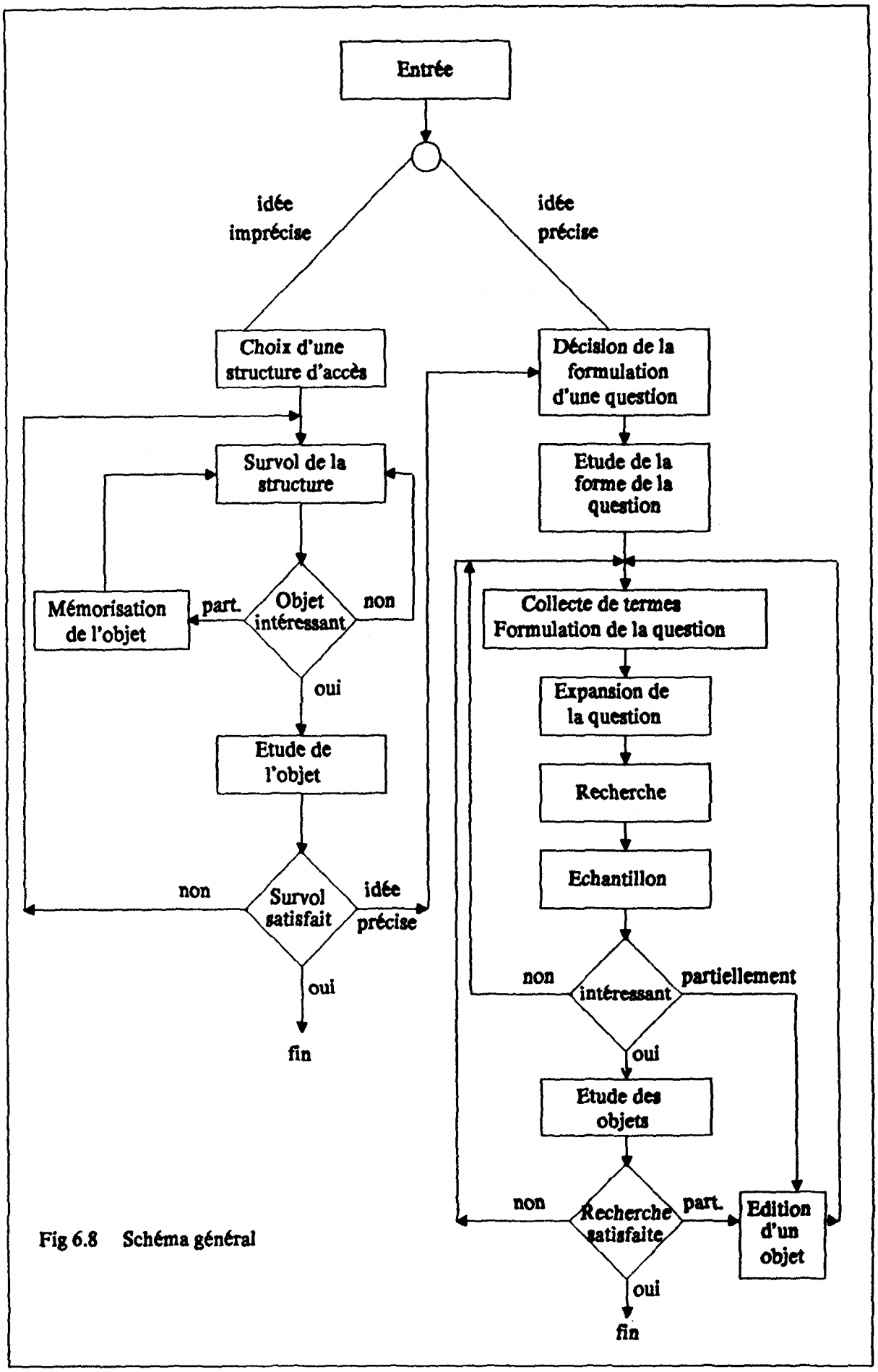


Fig 6.8 Schéma général

d'un vocabulaire fixe, pouvoir déterminer quels sont les termes disponibles. Les techniques de tâtonnement, qui consistent à entrer un terme pour ensuite constater qu'il n'existe pas dans le système, ne sont plus satisfaisantes. Il faut que le système *présente* les termes qu'il contient. Dans cette phase, la navigation dans les structures d'accès et parmi les objets référencés (en particulier, les documents indexés par un terme) doit être optimale. L'utilisateur doit avoir un moyen rapide de trouver les termes, d'accéder à d'autres termes en rapport avec ceux qu'il a déjà trouvés et d'obtenir directement les documents référencés. Il peut ainsi immédiatement décider si l'objet est intéressant ou non, procéder directement à une étude plus approfondie ou noter sa position pour y revenir par la suite après avoir terminé son survol.

A la fin de cette phase, l'utilisateur a probablement une idée plus précise sur la manière de préparer une question complexe, s'il n'a pas déjà satisfait ses besoins en information par ce survol. Il peut passer à la phase de formulation de la question puisqu'il connaît maintenant les structures d'accès et la forme d'un document. Il rejoint ainsi la situation de la personne qui débute sa session en connaissant déjà le système. La requête se présente sous la forme d'un document tels qu'ils sont représentés dans le système; l'utilisateur connaît cette forme puisque en principe à ce stade il a déjà vu un ou plusieurs documents. La phase de mise au point en collectant des termes ressemble grandement au survol effectué auparavant. Il s'agit tout simplement de pouvoir "piquer" les termes que l'on rencontre et de les intégrer automatiquement dans la question Cette collecte est suivie de la phase facultative d'expansion de la question. Le système doit proposer des termes auxquels l'utilisateur n'a pas pensé pour qu'il puisse décider s'il veut les intégrer ou non. Une fois ce pas franchi, il soumet sa requête au système. C'est à ce moment qu'intervient la mise au point de l'importance de chaque terme de la question. Si l'on décide d'abandonner la logique booléenne, parce que trop complexe et peu naturelle pour un utilisateur occasionnel, on peut tout de même lui proposer de donner une importance relative à chaque terme de sa question. Cette phase ne peut se passer plus tôt puisqu'il faut pouvoir comparer les termes de la question entre eux pour leur donner leur importance relative. Une fois la recherche effectuée, un échantillon de l'ensemble des données concernées, éventuellement sous la forme du document correspondant le mieux à la requête, est présenté à l'utilisateur, qui en conséquence décidera soit de modifier sa question, soit d'étudier l'ensemble des objets trouvés, soit encore de prendre un document comme nouvelle requête à fournir au système, établissant ainsi un processus itératif d'amélioration des résultats.

Méthodes de recherche

Nous avons vu qu'il existe en fait deux méthodes de recherches. La première consiste en un survol des informations où l'accès aux documents se fait de manière ponctuelle, comme par exemple lorsque l'utilisateur d'une petite bibliothèque ouvre un livre ou, dans un système de recherche d'information, lorsque l'on demande à connaître les documents attachés à un seul terme sans vraiment formuler une question. La deuxième méthode est basée sur le principe de la recherche combinée

où l'on formule une requête. Voyons comment se présente la situation de l'utilisateur face aux données définies dans le chapitre 2 (Fig 2.4).

Dans le premier cas, l'utilisateur accède aux données soit directement soit par l'intermédiaire des structures d'index mais, et c'est important, par un seul chemin. Les données qui sont montrées dans la figure ci-dessous peuvent représenter des livres sur l'étagère d'une bibliothèque ou des références dans une base de données bibliographiques.

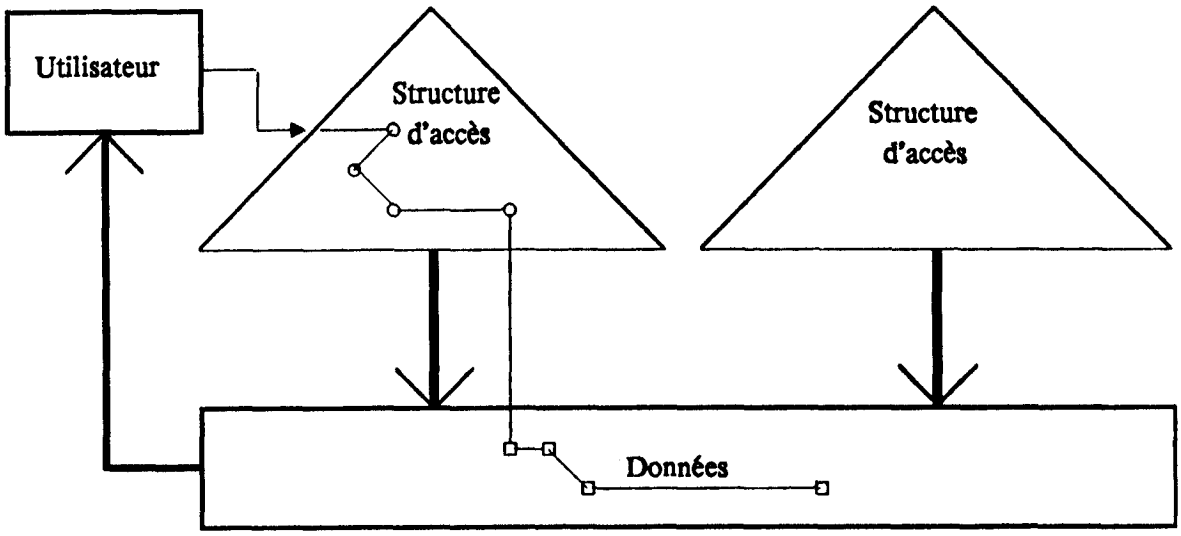


Fig 6.9 Survol

Les points dans les structures d'accès représentent les termes qui sont définis dans le vocabulaire. Dès qu'il a trouvé un point intéressant, l'utilisateur peut visualiser un ensemble d'objet caractérisé par ce terme. Chaque structure d'accès peut être étudiée individuellement, mais dans ce cas aucun accès ne se fait depuis deux ou plusieurs termes simultanément.

Dans le deuxième cas, basé sur la recherche combinée, la méthode peut également être éclaircie par une formalisation dans le cadre de notre schéma. L'accès cette fois-ci est multiple. C'est-à-dire que l'utilisateur va suivre simultanément plusieurs chemins pour formuler sa question.

Il a toujours les possibilités d'accès ponctuels aux données qui lui donnent plus d'information sur un terme qui ne serait pas suffisamment clair en lui-même. Les termes de plusieurs structures peuvent être intégrés dans la question. Cette question est soumise au système de recherche qui fournit les résultats de ces découvertes. Un échantillon peut ensuite être pris comme question ou des termes, tirés des résultats, peuvent être intégrés dans l'ancienne question. C'est cette phase qui permet d'améliorer sa requête.

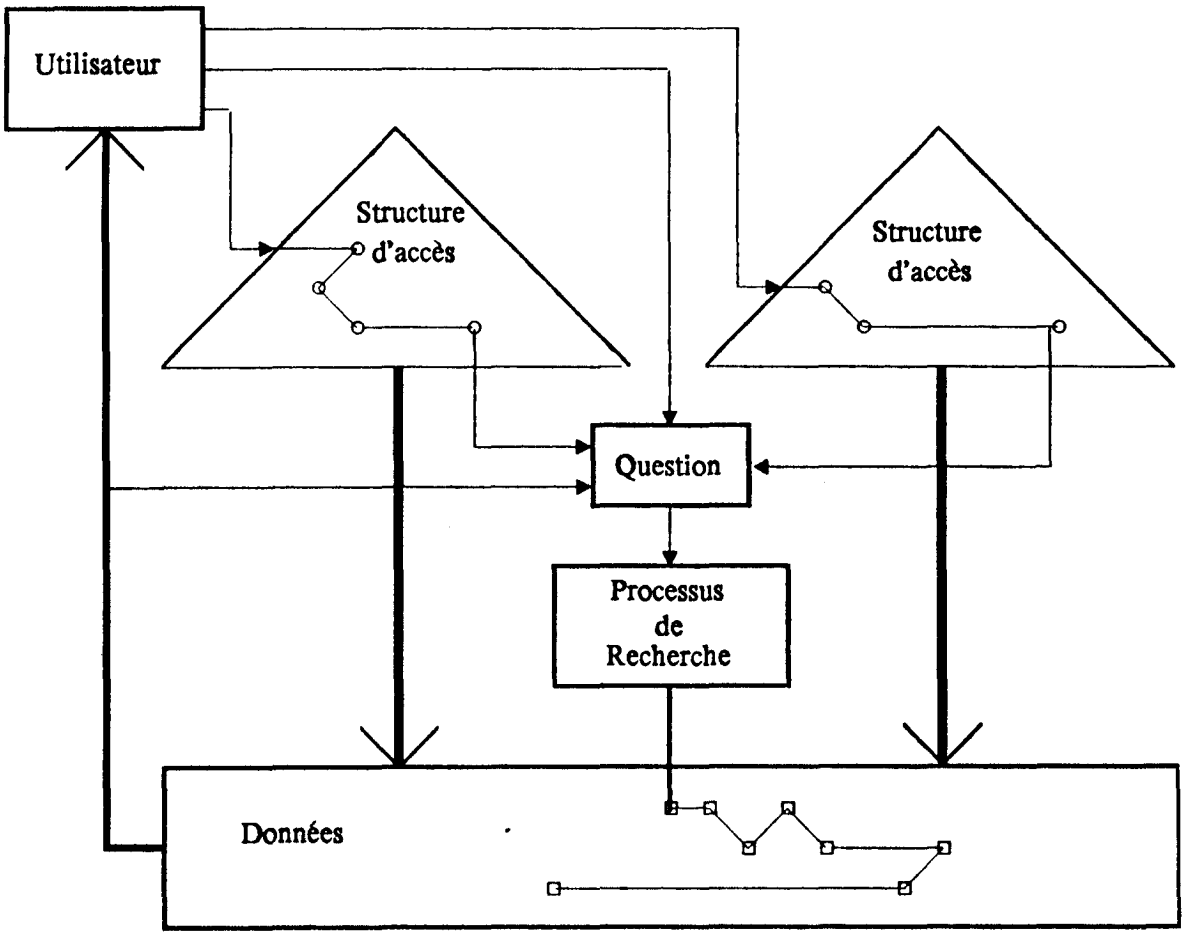


Fig 6.10 Préparation de requête

Conclusion

Pour résumer, voici une liste des points qui me semblent très importants dans un système de recherche d'information:

- Vocabulaire fixe.
- Dualité des principes de recherche.
- Plusieurs structures d'accès différentes (liste, arbre, réseau).
- Visualisation de ces structures.
- Possibilité de navigation dans les structures et parmi les objets.
- Formulation de questions.
- Similitude de forme entre la requête et les objets.
- expansion semi-automatique de la question.
- Abandon de la logique booléenne.
- Méthode itérative d'amélioration de la question.
- Visualisation des objets.

Nous montrerons dans le chapitre 7 comment un système peut être développé suivant ces principes, mais tout d'abord je tiens à préciser quelques points importants concernant les interfaces utilisateurs.

6.3 Les Techniques d'Interfaçage

Introduction

Les systèmes de recherche actuels ne sont pas équilibrés. Les efforts investis pour améliorer les algorithmes de recherche et pour augmenter la quantité d'information disponible sont nettement supérieurs à ceux fournis dans le domaine de l'intégration de l'utilisateur dans le système. Et ceci est justifié puisqu'il est nécessaire de pouvoir offrir de l'information et de donner des méthodes pour la retrouver avant même d'améliorer les outils qui permettent d'accéder à ces données. Pourtant ce déséquilibre est trop important et une mise à jour des interfaces utilisateur aux techniques actuelles dans le domaine devient nécessaire. Comme il a été décrit dans [Frei83], les langages de commande traditionnels doivent évoluer et devenir de véritables interfaces utilisateur qui s'adaptent au matériel et aux méthodes actuelles. En particulier une formalisation correcte du dialogue entre l'utilisateur et la machine va permettre d'éliminer les contradictions que l'on trouve dans les systèmes et d'intégrer les nouveautés qui arrivent sur le marché.

Des langages de commande aux interfaces utilisateur

Les langages de commande sont des éléments très importants dans les systèmes interactifs. Ce sont eux qui permettent d'accéder aux services disponibles dans le système. Souvent même les utilisateurs ont tendance à confondre système et langage de commande et la valeur du système est jugée au travers de ce langage. Si l'on considère les langages de commande actuels et l'utilisation des systèmes qu'ils contrôlent, on s'aperçoit que l'utilisateur "est souvent dans la position d'un maître absolu par rapport à un esclave puissant et respectueux qui parle une langue étrange et complexe" comme le souligne Miller et Thomas [Mill77].

Examinons tout d'abord les caractéristiques principales d'un langage de commande typique. Nous pourrons alors donner une liste des principaux arguments qui ont suscité la remarque de Miller et Thomas, puis présenter les alternatives que l'on peut proposer pour une nouvelle approche de l'interfaçage d'un utilisateur avec une machine.

Le langage de commande typique d'un système interactif conventionnel consiste généralement en une relativement longue liste de commandes. Un tel système est équivalent à une machine d'état fini qui ne possède qu'un seul état; dès qu'une commande a été exécutée, le système revient dans l'état dans lequel il se trouvait avant l'exécution. L'utilisateur voit cette machine de la manière suivante: chaque fois qu'il désire entrer un ordre, il effectue un *choix* parmi cette très longue liste de commandes. Elle lui est très souvent *invisible* et il a rarement un moyen de la visualiser. Les commandes sont généralement entrées au clavier en tapant une chaîne de caractères. Ceci peut se révéler être un tâche ennuyeuse et génératrice d'erreurs. Seuls les systèmes récemment développés possèdent des touches spéciales, des

boutons ou des touches-fonction pour exécuter les commandes les plus fréquentes.

En résumé, les noms des commandes, suivis éventuellement d'arguments doivent être entrés au clavier. Ces noms et leurs significations doivent être appris avant de pouvoir être utilisés. Si le nombre de commandes mémorisées et la rapidité d'accès à ces commandes augmentent, le système sera mieux employé par l'utilisateur. Souvent ces systèmes sont malheureusement destinés à des experts, bien qu'étant utilisés par une grande variété de personnes. Beaucoup de systèmes interactifs et en particulier des systèmes de recherche d'information ont ces caractéristiques. Les facteurs suivants contribuent au fait qu'ils sont difficiles à utiliser pour des gens inexpérimentés.

- L'ensemble des commandes n'est pas structuré, ou si il l'est, cette structure n'est pas visible.

- Les commandes et leur spécification doivent être mémorisées; il faut connaître un nombre minimum de commandes pour accéder au système de manière satisfaisante.

- Les commandes sont pénibles à entrer (la plupart des utilisateurs sont de mauvais dactylographes).

- Les noms prêtent à confusion. De plus, ils sont souvent abbrégés selon certaines règles obscures.

- Les erreurs de syntaxe sont possibles.

Une autre confusion intervient dans les systèmes interactifs de RI: dans la plupart des cas, les noms de commande et les termes-index, utilisés pour décrire l'information stockée, ont une apparence identique lorsqu'ils sont tapés sur le clavier. Les commandes et les termes sont exprimés au moyen de chaînes de caractères. Aujourd'hui, la plupart des activités de la RI sont encore accomplies au moyen de langage de commandes ayant ces déficiences (e.g. DIALOG, ORBIT). Les récents développements, dans le sens d'une structuration du jeu de commandes et de l'emploi de touches (hard ou soft) pour spécifier ces commandes (e.g. STAIRS), n'ont eu jusqu'à présent que peu de conséquences dans la pratique. Le problème réside dans le fait que les systèmes de RI sur le marché doivent encore pouvoir accepter les caractéristiques des terminaux du style "télétype" pour pouvoir connecter les utilisateurs avec l'ordinateur.

De nos jours, les systèmes interactifs, et en particulier ceux de la RI, sont utilisés quasi exclusivement par des spécialistes de ces systèmes. Dans un proche futur, ces systèmes seront utilisés directement par l'utilisateur final, c'est-à-dire la personne qui *a besoin de l'information.* C'est pourquoi notre premier objectif a été de développer un système RI aussi adapté que possible à cet utilisateur, pour qu'il puisse effectuer ses recherches sans l'aide d'un spécialiste de la RI. Lorsque l'on parle d'adaptation à l'utilisateur, cela ne signifie pas forcément que le langage de commande est facile à

apprendre. Notre but a été plutôt de construire un système qui s'explique de lui-même sans avoir à compulser des manuels pour l'utiliser. Les manuels écrits sont des outils inadaptés à la descrition de systèmes interactifs hautement flexibles.

Développer un système adapté à l'utilisateur final est un but ambitieux. Cela signifie avant tout effectuer un pas dans le processus d'évolution, qui part des langages de commande pour arriver aux interfaces utilisateurs. C'est pourquoi nous avons étudié le système complet du point de vue de l'utilisateur final. L'option a été de penser que le système, comprenant également les données stockées, devait être présenté à l'utilisateur aboutissant ainsi à un interface utilisateur et non pas à un nouveau langage de commande.

L'abandon de la vision caractère-par-caractère de l'information a été une importante décision dans la conception du système. L'utilisateur s'occupe plutôt de *concepts* d'information en laissant de côté le fait que de tels concepts sont normalement décrits par des termes-index ou des mots-clés sous la forme de chaînes de caractères. De plus, la structure de ces concepts d'information doit être présentée à l'utilisateur, comme c'est le cas dans les systèmes les plus nouveaux.

Nous avons conclu que la propriété la plus importante d'un système orienté vers l'utilisateur était sa transparence; après la phase d'initialisation, le système doit présenter lui-même les services et l'information à l'utilisateur. Les services comprennent toutes les actions qu'il est possible d'effectuer en cours de session (p.ex. feuilleter les objets contenus dans le système via des index, préparer des requêtes, effectuer des recherches, combiner des ensembles d'objets, ...). L'*information* se compose de tous les objets stockés dans le système y compris les index qui décrivent le contenu de ces objets. Contrairement à la plupart des systèmes de RI actuels, notre système doit clairement montrer les structures des services et de l'information pour que l'utilisateur puisse effectuer un choix parmi les possibilités qui lui sont offertes. Nous pensons que seul un tel système est capable d'aider efficacement l'utilisateur final dans sa manière de chercher de l'information, puisque, dans la plupart des cas, la stratégie de recherche n'est pas très systématique. Ainsi que l'a déjà remarqué Bush [Bush45], puisque la réflexion est souvent guidée par des associations, son cheminement peut être dévié par de nouvelles informations puis revenir par la suite au courant principal. De cette manière, l'utilisateur collecte de l'information pièce après pièce jusqu'à ce qu'il ait atteint ses buts. Ceci n'est toutefois possible que si les services et l'information sont présentés de manière claire.

Les services et leur représentation

Bien que beaucoup de services d'un système interactif soient conceptuellement divisés en plusieurs sous-services, très souvent les commandes qui permettent d'exécuter ces sous-services sont au même niveau que les commandes principales. L'utilisateur occasionnel rencontre alors le problème suivant : si un système est sophistiqué, il possède un grand nombre de commandes et l'utilisateur doit investir beaucoup de temps pour mémoriser un sous-ensemble particulier de commandes afin

d'utiliser le système; plusieurs langages de commande consistent en une simple liste linéaire et, dans ce cas, il est difficile de savoir à chaque moment quelles sont les commandes appropriées ou exécutables. Lorsqu'une commande est exécutée, un autre problème intervient si les services d'un système interactif sont structurés: le système entre dans un autre *mode*, l'environnement précédent est donc abandonné. L'ensemble des commandes possibles change, ce qui signifie que de nouveaux noms de commandes sont disponibles ou d'anciens changent de signification. Les difficultés rencontrées par l'utilisateur proviennent principalement du manque d'information donnée par le système. Il paraît évident que si les services sont structurés, cette structure doit apparaître sur l'écran.

Les services que l'on peut trouver dans un système de recherche sont structurés, il est donc possible d'employer une méthode graphique pour les représenter. Ceci s'écarte de la liste linéaire de commandes et possède un certain nombre d'avantages. Un service est une partie du système et constitue une activité bien définie qui correspond aux besoins de l'utilisateur. C'est pourquoi seuls les sous-activités et les commandes appropriées lors de cette activité doivent être présentes et exécutables. Dès qu'un autre service est appelé, d'autres sous-activités et d'autres commandes deviennent disponibles. Par exemple le processus *Search* est divisé en plusieurs sous-processus pour créer et modifier une question, chercher et étudier des objets et imprimer ou sauver des résultats. La figure ci-dessous présente la forme d'un arbre de commande tel qu'on pourrait le trouver dans un système de recherche de données. Le service Search, par exemple, est subdivisé en cinq commandes.

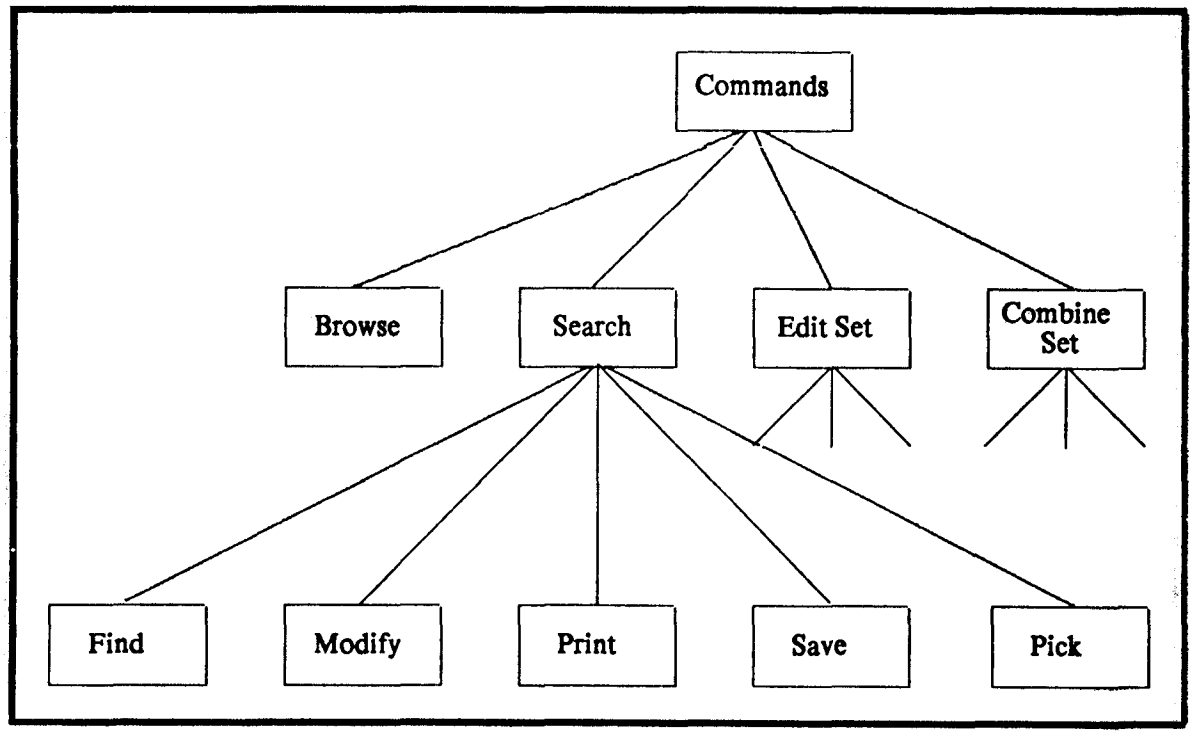


Fig 6.11 Les Commandes

Comme on peut le voir ci-dessus, il s'agit des cinq services (Find, Modify, Print, Save

et Pick). Eux-même peuvent à nouveau être subdivisés en plusieurs commandes. On voit que Browse n'est plus subdivisé en d'autres commandes. Nous reviendrons dans le chapitre suivant sur la signification de ces commandes.

Langage de commande - grammaire formelle

Tout langage de commande correctement structuré possède une grammaire qui en définit sa syntaxe. Ceci permet d'une part de contrôler automatiquement si une séquence de commandes appartient au langage et, d'autre part, de garantir la consistance du langage puisque l'on peut s'assurer ainsi qu'il ne contient pas de contradictions. Une théorie sur les grammaires formelles et les automates qui permettent de les contrôler est décrite par Hopcroft et Ullman [Hopc69]. On définit généralement ces grammaires par la forme de Backus-Naur (BNF) [Grie81]. Voici la grammaire proposée pour une séance de recherche d'information.

```
<session de RI>     ::= {<opération>}.
<opération>         ::= <spécification> <exécution>.
<spécification>     ::= <opérande> <assertion>.
<opérande>          ::= "désignation d'un objet sur l'écran" |
                        "entrée d'un nom au clavier".
<assertion>         ::= "touche du clavier" | "touche fonction" |
                        "touche d'une unité d'entrée".
<exécution>         ::= <"réaction du système"> {<opération>}.
```

Une session est une suite d'opérations. Chaque opération comporte deux parties, la spécification et l'exécution. La spécification consiste à préciser ce que l'on désire exécuter. Ceci peut se faire soit en tapant sur le clavier les différentes lettres qui forment un nom, soit en choisissant le champ d'un menu, soit encore en montrant un objet sur l'écran au moyen d'un pointeur. Dans ces deux derniers cas, la position choisie est l'*opérande*. L'opération est complètement spécifiée lorsque l'utilisateur presse sur une touche quelconque pour confirmer son choix (assertion) et ainsi démarrer l'exécution effective. Le système réagit en changeant son état et éventuellement en proposant un autre sous-ensemble d'opérations possibles. Des sous-opérations peuvent ainsi être effectuées. En précisant que l'exécution peut à nouveau contenir une suite d'opérations, on définit une structure hiérarchique.

La spécification

Il existe deux manières de spécifier des opérations :

- Pointer sur un endroit spécifique de l'écran où se trouve un objet, par exemple le noeud d'un arbre ou une touche présente sur l'écran. C'est une méthode rapide et précise. Il suffit de placer un curseur sur la position désirée puis de presser un bouton pour indiquer que l'exécution du service défini par cette position peut avoir lieu.

- Presser sur un bouton de l'unité qui sert à déplacer le curseur sur l'écran ou presser sur une ou plusieurs touches du clavier et ceci indépendamment de la position du curseur.

On peut remarquer que seules les opérations qui ont des conséquences irrémédiables, comme détruire un objet, doivent être exécutées en utilisant les touches du clavier. Presser sur un bouton d'une unité d'entrée (touche-fonction ou bouton sur un curseur) est une méthode très rapide pour la spécification, particulièrement lorsqu'il n'y a pas besoin de placer le curseur à un certain endroit de l'écran. C'est pourquoi cette méthode doit être réservée aux opérations n'ayant pas de conséquences graves si elles sont entrées par erreur.

Très souvent une combinaison des deux méthodes peut être utilisée. Il s'agit alors de positionner le curseur puis de presser sur l'un des boutons du curseur ou éventuellement une touche du clavier. Dans ce cas chaque bouton peut avoir une signification différente. Les effets des boutons peuvent aussi être fonction de la position du curseur sur l'écran. L'utilisateur doit donc être informé en tout temps de la signification actuelle des boutons. On emploira plus volontiers les boutons se trouvant sur le curseur pour des raisons de rapidité d'entrée, bien que l'on perde un grand nombre de touches par rapport au clavier. Il est très gênant de devoir déplacer constamment la main qui tient le curseur pour accéder au clavier après avoir donné une position. D'autre part, l'utilisation de l'autre main placé sur le clavier reste du domaine réservé aux virtuoses ce qui ne rentre pas dans le cadre de notre travail.

Les opérandes

Lors de la spécification nous avons placé les opérandes avant les commandes. Il aurait été possible de les placer après mais cette solution permet de terminer plusieurs opérandes par une assertion, donc de ne pas limiter à priori le nombre de ces opérandes. D'autre part, dans le cas du positionnement sur l'écran, on diminue d'un le nombre de fois qu'il s'agit de presser sur une touche. En effet, placer le curseur sur une position puis presser une touche permet en même temps de désigner l'objet et de spécifier l'opération. Dans le cas contraire, il faudrait tout d'abord presser une touche pour spécifier l'opération, puis en presser une autre pour désigner l'objet sur lequel on s'est positionné. Ceci ralentirait la phase de spécification et diminuerait la clarté des opérations.

Dans le cas d'une représentation hiérarchique graphique, il est possible de définir une position courante, par exemple un noeud de l'arbre ou le champ d'un menu, comme l'opérande par défaut. Le système doit représenter cet emplacement de manière différente des autres afin que l'utilisateur sache que cette position est prise comme opérande. Ceci évite, dans certains cas clairs, de devoir positionner le curseur plusieurs fois au même endroit.

L'assertion

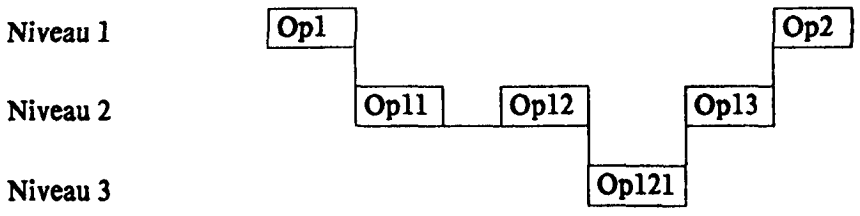
La méthode classique d'effectuer une opération consiste à taper au clavier les différentes lettres qui forment son nom. La plupart des systèmes interactifs l'utilisent et elle a déjà fait ces preuves plusieurs fois. Elle requiert toutefois de l'utilisateur d'une part une parfaite connaissance du système, en particulier les noms possibles et, d'autre part, une aptitude à la dactylographie pour spécifier ces noms rapidement. Une tendance actuelle consiste à utiliser des touches fonction programmables présentes sur les claviers. L'inconvénient majeur de ces touches réside dans le manque d'information que l'on a à propos de leur signification. Une autre possibilité consiste à représenter ces touches sur un écran de manière à ce que l'utilisateur puisse voir leur signification et ainsi en user judicieusement. Cette solution a l'avantage d'être facile et rapide à utiliser même pour un utilisateur occasionnel. Deux cas sont possibles:

- Le système présente sur l'écran un schéma des touches fonctions qui existent sur le clavier et il présente la signification actuelle des touches. Il suffit alors de presser sur la touche correspondante du clavier pour spécifier cette commande.

- Le système définit un jeu de touches que l'on peut activer au moyen d'un curseur et exécuter en pressant sur une touche d'une unité d'entrée, après avoir placé le curseur sur la position choisie sur l'écran. Dans ce cas, le placement du curseur est considéré comme l'opérande de l'opération qui est confirmée en pressant sur la touche de l'unité d'entrée. Cette solution a l'avantage de ne pas limiter le nombre de touches à celui défini par les touches du clavier qui peut être relativement restreint.

L'exécution

Une fois l'opération spécifiée et confirmée, le système réagit en modifiant son état, en présentant la nouvelle situation à l'utilisateur et en proposant éventuellement d'autres opérations. Cette phase de réaction du système n'appartient pas directement à la grammaire du langage de commande puisqu'elle ne requiert aucune action de l'utilisateur. Elle est plus là pour clairement définir l'enchaînement des différentes phases de la session. Vient ensuite à nouveau une suite d'opérations qui peuvent être considérées comme des sous-opérations de l'opération actuelle. La séquence d'opérations peut se schématiser de la manière suivante:



Chaque opération au même niveau représente une séquence d'opérations les unes après les autres. Le passage d'un niveau à un niveau inférieur représente l'exécution de sous-opérations de l'opération du niveau supérieur. En particulier, les opérations

Op11, Op12 et Op13 ne peuvent pas être exécutées si l'opération Op1 n'a pas eu lieu. Nous donnerons dans le chapitre suivant un exemple de grammaire spécifique à la recherche d'information qui suit ce schéma. Des exemples de séquence d'opérations permettront d'éclaircir le rôle et la signification des opérations.

Conclusion

En résumé, voici les différents points importants des interfaces utilisateurs et des méthodes qu'il s'agit d'appliquer pour conserver une interaction simple et claire entre le système et l'utilisateur:

- Structure hiérarchique de commandes
- Présentation de cette structure
- Rapidité de spécification de commandes
- Utilisation de touches-fonction
- Message constant à l'utilisateur sur l'état du système
- Définition d'une grammaire formelle comme base du langage de commande
- Séquence opérandes-opérations

Voyons maintenant comment peut être défini et implémenté un système, suivant les règles qui ont été stipulées.

7. Caliban, une tentative d'implémentation

7.1 Introduction

Caliban est un système de recherche d'information développé sur un ordinateur personnel. Le but de ce système est d'essayer de démontrer qu'il est possible à un utilisateur occasionnel de rechercher sa propre information sans passer par un intermédiaire. De plus, le système doit être suffisamment simple pour qu'il puisse l'employer sans devoir suivre de longues explications et perdre un temps qu'il n'a souvent pas envie d'investir. Caliban devrait également démontrer qu'il est possible de concevoir un système destiné à un utilisateur occasionnel qui convienne encore à un professionnel. Pour ce faire, nous nous sommes basés sur une représentation systématique des objets afin de montrer dans quel état se trouve le système et quelles sont les informations disponibles. Ce chapitre devrait donner une vue d'ensemble des possibilités du système.

Pour démarrer, l'utilisateur doit taper le nom *Caliban* au clavier. Le système s'annonce et effectue une phase d'initialisation qui se termine par l'apparition des commandes sur l'écran (Fig 7.1).

Pour distinguer les différents concepts lorsque l'on utilise Caliban, l'écran est toujours subdivisé en trois fenêtres distinctes: la fenêtre *Information*, la fenêtre *Command* et la fenêtre *Message*.

La fenêtre *Information* est la plus grande des trois et contient aussi bien les structures d'information (index) que les objets eux-mêmes. Les résultats de la commande Browse ainsi que ceux de Search sont montrés dans cette fenêtre.

La fenêtre *Command* contient l'ensemble structuré des services disponibles. Pour invoquer un service de Caliban, il faut le sélectionner parmi les services présentés dans cette fenêtre. De plus, cette fenêtre présente constamment le service qui est actif. C'est une grande aide pour l'utilisateur qui revient au système après avoir été interrompu.

La fenêtre *Message* contient différentes sortes de messages du système. En premier lieu, ces messages sont destinés à aider l'utilisateur pendant qu'il travaille avec le système. C'est la raison pour laquelle cette fenêtre peut être ignorée par un utilisateur expérimenté. Par exemple, elle contient toujours un dessin représentant la souris, que l'on emploie comme outil d'entrée, avec la signification des effets de ses trois boutons. On sait ainsi en tout temps quel serait l'effet de la commande si l'on appuie sur l'un des boutons. Cette information est relativement peu intéressante pour une personne connaissant bien le système. (Comme nous le verrons plus tard, la signification des boutons change).

Pour exécuter des commandes ou pour choisir des objets sur l'écran, l'utilisateur dispose de la souris, avec laquelle il contrôle le curseur qui se trouve sur l'écran. Le

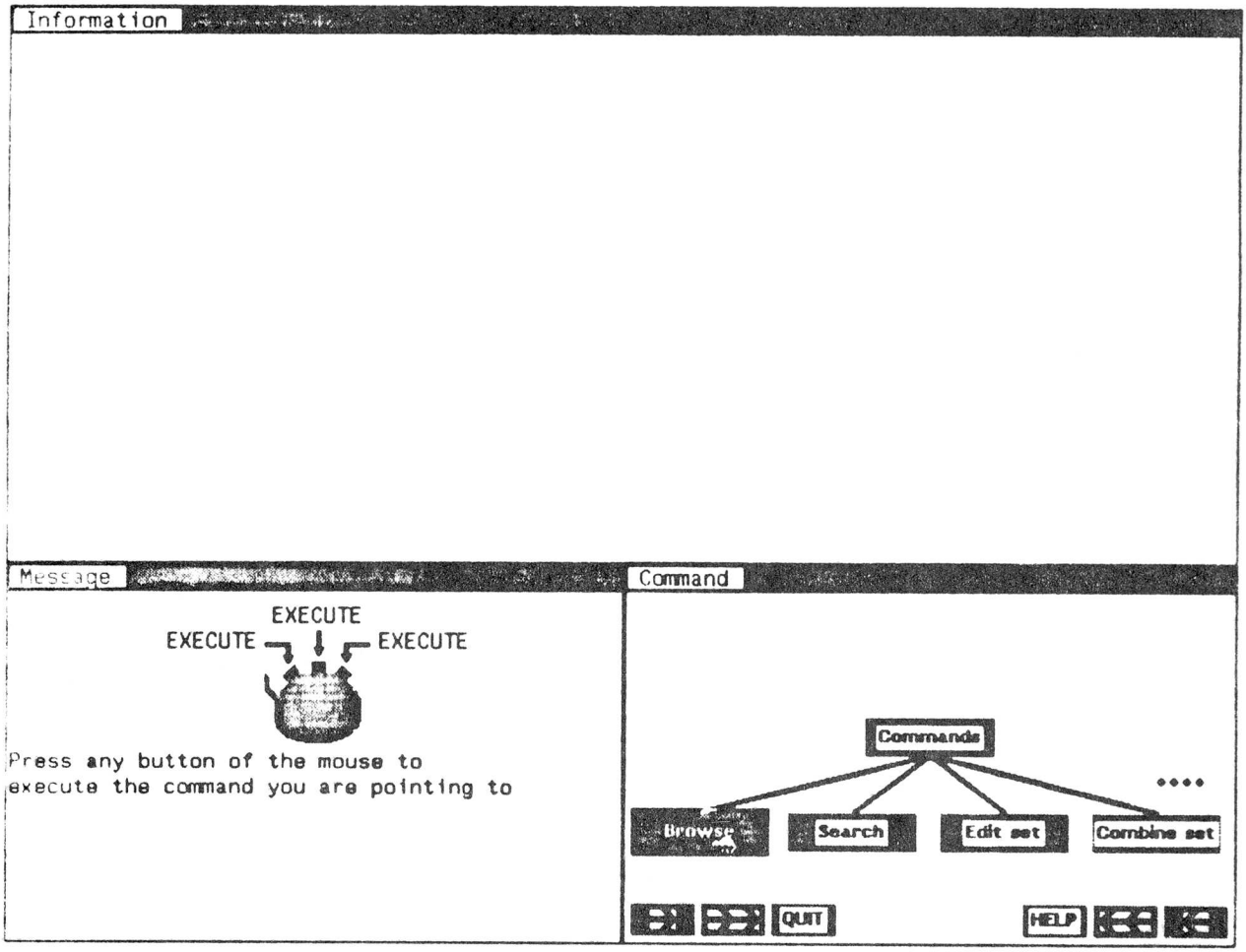


Fig 7.1

curseur de Caliban, que l'on distingue dans la fenêtre *Command*, est représenté en forme de souris. Pour déplacer le curseur sur l'écran, il suffit de bouger la souris dans la direction choisie. Dès que le système reconnaît un objet pointé par le curseur, il en change la représentation. C'est ce que l'on peut observer à la figure 7.1 où la commande Browse, actuellement pointée par le curseur, est représentée différemment des autres commandes. Pour exécuter cette commande, il suffit à l'utilisateur de presser un des boutons de la souris. A cet instant, comme on le voit dans la fenêtre *Message*, chaque bouton a le même effet, à savoir EXECUTER la commande sur laquelle se trouve le curseur.

Nous verrons quelques exemples d'utilisation du système, mais auparavant voyons comment sont représentés les structures d'accès.

7.2 La visualisation des structures d'accès

Comme nous l'avons vu au chapitre précédent, il y a plusieurs manières de présenter l'information. Caliban se base sur les possibilités graphiques de la machine pour présenter une structure qui à la fois reflète l'organisation logique des données et permette à un utilisateur de s'orienter facilement. Les trois structures principales sont les listes, les arbres et les réseaux. Parmi ces derniers, on peut faire une subdivision supplémentaire en distinguant les multihiérarchies des réseaux. Une multihiérarchie est définie comme un arbre où chaque noeud peut posséder un ou plusieurs pères. C'est une généralisation des arbres sans pour autant être un réseau au sens général du terme.

La représentation de base choisie dans Caliban est une structure hiérarchique. Il est en effet toujours possible de représenter une liste, un arbre ou un réseau de cette manière. Le cas des données organisées en forme de listes ou d'arbres se règle ainsi de manière triviale.

En ce qui concerne les réseaux, nous avons décidé de ne conserver pour la représentation qu'une structure hiérarchique. Cet arbre est présenté à l'utilisateur de manière graphique en laissant de côté les autres connections qui forment le réseau. Toutefois, pour ne pas supprimer une grande partie des connections et ainsi perdre de l'information, nous donnons à l'utilisateur la possibilité de visualiser ces autres arêtes. Il peut à tout moment réclamer les différentes relations qui ne sont pas affichées. Si l'on prend le cas du thesaurus, ces connections se situent sur plusieurs niveaux. Tout d'abord, chaque noeud est susceptible d'avoir plusieurs pères. Il est possible, par une simple commande, de visualiser cette multihiérarchie en plus du père qui est déjà sur l'écran. Ensuite, les termes liés par un lien plus faible que la relation père-fils peuvent aussi être représentés. Dans ce cas, il est nécessaire d'éliminer les autres noeuds puisqu'ils ne peuvent avoir place tous en même temps sur l'écran.

En raison de la taille réduite de la fenêtre dans laquelle nous présentons l'information, seuls trois niveaux de l'arbre sont visibles. Les noeuds sont représentés sous forme de rectangles avec leur noms écrits au centre et les arêtes entre pères et fils sont explicitement dessinées. Le noeud au centre de la fenêtre est appelé le noeud courant; il est représenté différemment des autres de manière à pouvoir le distinguer. Seul le père, deux frères et au plus sept fils sont affichés simultanément. La figure 7.2 montre comment cette vue partielle est extraite de l'arbre entier; les noeuds à l'intérieur de la ligne foncée apparaissent sur l'écran. Les points qui se situent au-dessus du fils le plus à droite ou le plus à gauche signifient que d'autres noeuds sont présents dans cette direction.

Dans nos données de test, une des structures d'accès est un thesaurus. On en voit une partie à la figure 7.3 avec le terme *Information science* comme noeud courant. Un thesaurus est un exemple de réseau ayant une structure hiérarchique principale et un nombre d'autres relations telles que BROADER TERMS, RELATED TERMS, etc.

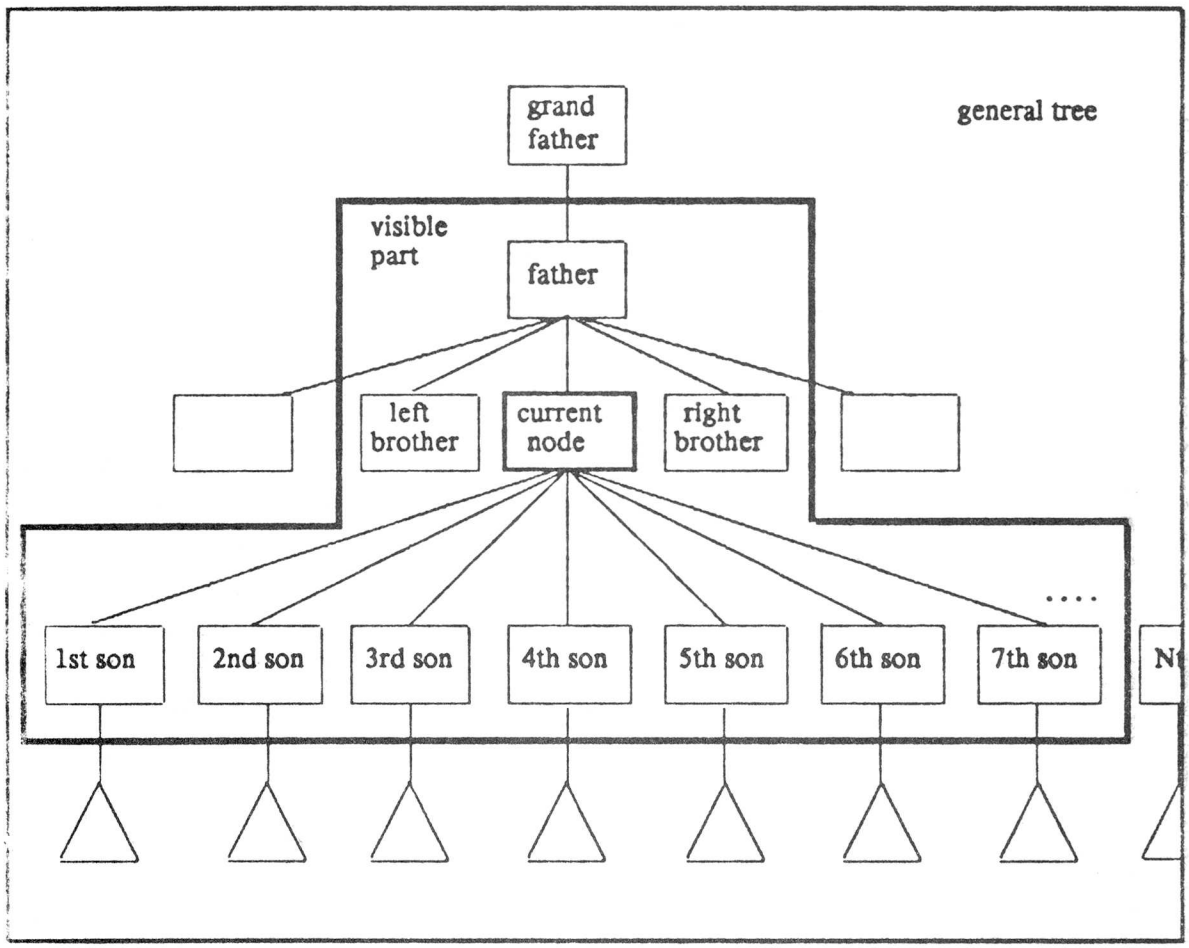


Fig 7.2 Vue partielle d'un arbre

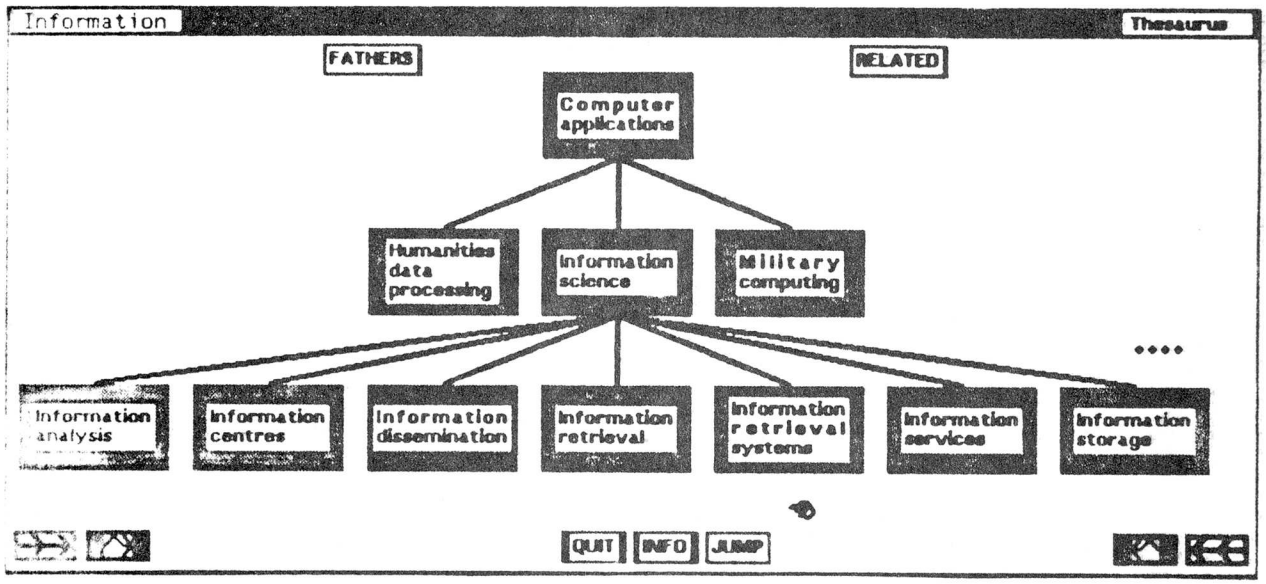


Fig 7.3

BROADER TERMS représentent des concepts plus généraux en plus du terme qui tient lieu de père. Les termes liés (RELATED TERMS) sont ceux qui ont une

relation plus faible que celle intervenant dans la structure hiérarchique.

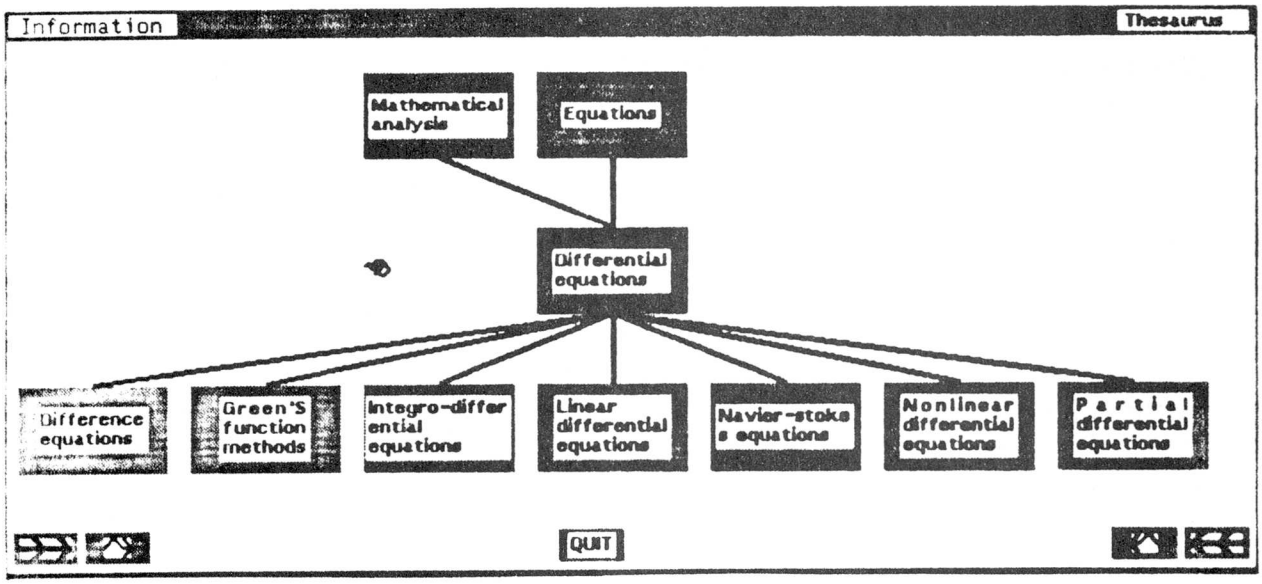


Fig 7.4

La représentation d'une multihiérarchie se fait de la manière suivante. Le noeud courant ainsi que ses fils restent affichés. Les frères à gauche et à droite disparaissent de manière à permettre de dessiner les traits qui conduisent aux différents pères du noeud courant sans que des lignes s'entrecroisent. Ces pères sont dessinés au niveau le plus haut présenté sur l'écran au côté du père déjà existant (Fig 7.4).

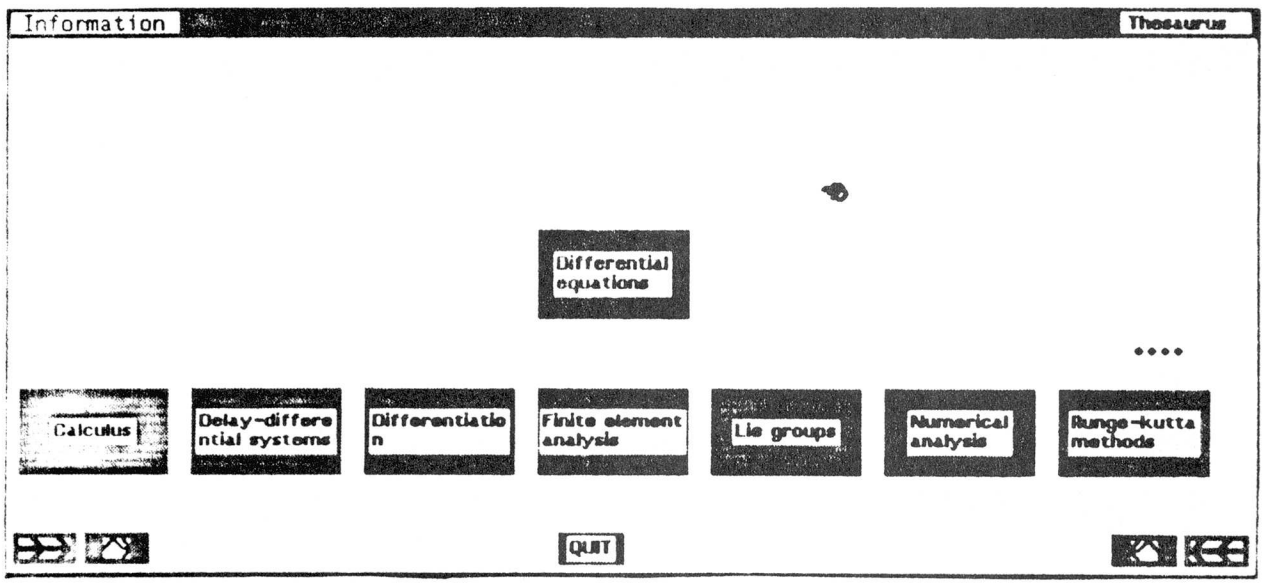


Fig 7.5

Enfin les noeuds qui forment la structure de réseau sont visualisés simplement en montrant le noeud courant au milieu de l'écran et au-dessous de lui les termes qui lui

sont rattachés mais cette fois-ci sans dessiner un trait entre eux pour montrer que la connection est différente de celle que l'on affiche avec l'arbre (Fig 7.5).

Pour permettre à l'utilisateur d'avoir une vue d'ensemble, il doit pouvoir se déplacer et voir les parties de la structure qui n'apparaissent pas explicitement sur l'écran. Cette méthode de déplacement dans les structures est appelée Navigation.

7.3 La navigation dans les structures

La navigation donne la possibilité de se déplacer dans la structure. On peut ainsi voir d'autres positions et visualiser leur environnement. Si l'on prend une liste linéaire, la navigation se limite au déplacement à l'élément suivant ou précédent. Dans le cas d'une structure hiérarchique, l'ordre dans lequel les éléments sont organisés définit un algorithme de traversement. Les deux exemples classiques sont la méthode préfixée et la méthode postfixée. Dans le premier cas, on traverse l'arbre en visitant tout d'abord le père puis ses fils. Dans le second cas, on commence par visiter les fils pour finir par le père. La navigation, au sens strict du mot, permet de passer aux frères à gauche et à droite, de revenir au père ou enfin d'atteindre un des fils de la position courante. En ce qui concerne les réseaux, il est possible de se déplacer vers n'importe quel point directement relié à la position courante.

Caliban permet ce genre de navigation. Tout d'abord on peut toujours accéder aux noeuds qui se trouvent sur l'écran en pointant dessus. Ces noeuds représentent toujours l'environnement direct de la position courante. Dès qu'un noeud a été choisi, il prend la place du noeud courant au centre de la fenêtre. Le nouvel environnement de cette position est alors visible, en particulier les fils de la position choisie sont présentés. Tous les fils d'un noeud ne peuvent cependant pas toujours être affichés simultanément. Des touches à gauche et à droite de la fenêtre, dans laquelle est représentée la structure, permettent d'amener le reste de l'environnement du noeud courant. Le premier type de touches amène un nombre fixe de noeuds cachés en positionnant le noeud le plus à gauche tout à droite ou inversément celui le plus à droite sur l'extrême gauche. Ce processus montre six autres noeuds dans l'actuelle version de Caliban. Etant donné que ce genre de déplacement peut se révéler assez fastidieux si l'on a beaucoup de noeuds cachés et que l'on désire voir le dernier, un autre type de touche a été prévu. Lors de l'activation de cette touche, on voit apparaître une barre de saut qui donne à la fois des renseignements quant au nombre de fils rattachés au noeud courant et quant à la proportion actuellement sur l'écran (Fig 7.6). Cette représentation de la partie visible peut être déplacée au moyen de la souris vers la position que l'on désire voir. On peut ainsi passer de la position à l'extrême gauche à la position à l'extrême droite sans avoir à visualiser tous les noeuds se trouvant au milieu. Lors du déplacement de la partie visible à l'intérieur de la barre de saut, la position actuelle des noeuds reste toujours visible. Ceci permet de se déplacer de manière relative par rapport à cette position courante, par exemple de voir les noeuds qui se trouvent juste à côté de ceux que l'on voit.

En ce qui concerne les multihiérarchies, un système semblable peut être mis au point pour afficher les pères de la position courante. Il serait facile de prévoir des touches de déplacement en haut de la fenêtre pour pouvoir présenter les pères qui n'apparaissent pas sur l'écran. Toutefois, l'expérience nous a montré que le nombre de pères est généralement suffisamment petit pour pouvoir toujours tous les afficher (c'est du moins le cas dans un thesaurus). La version actuelle de Caliban affiche

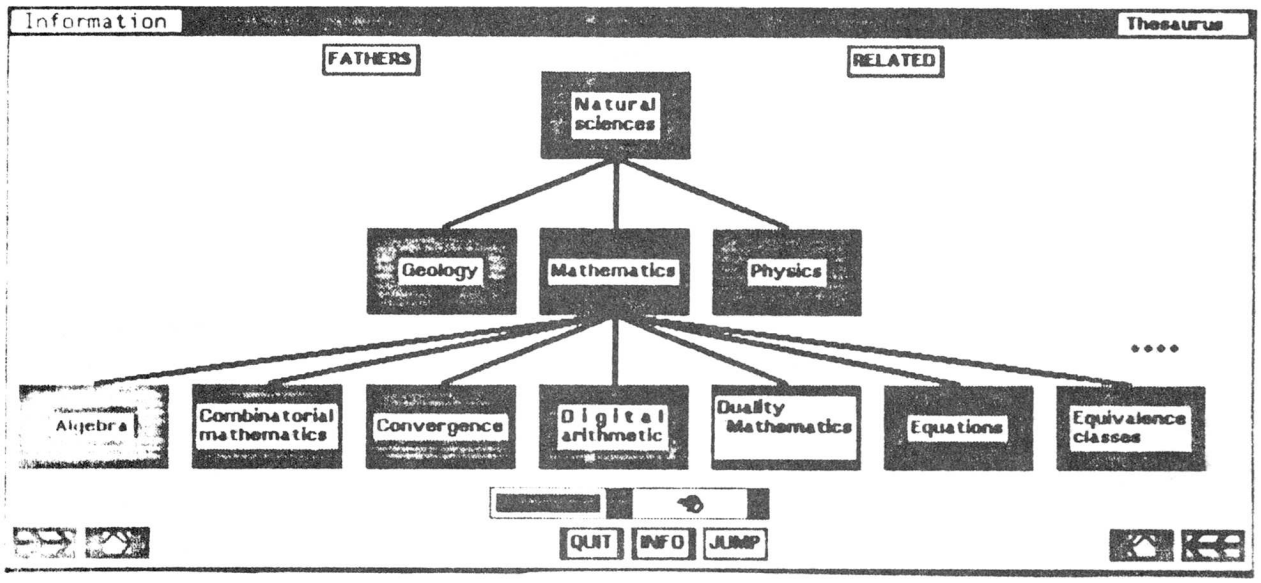


Fig 7.6

toujours tous les pères de la position courante lorsque l'utilisateur désire les voir. Ceci permet de sauter vers l'un de ces pères et de visualiser une nouvelle partie de la structure. Etant donné que la structure de base est une hiérarchie, les pères du noeud courant ne se trouvent pas forcément à proximité immédiate de l'endroit où l'on se trouve. En choisissant un des pères comme but de déplacement, on peut se retrouver à un endroit quelconque de la structure hiérarchique où est situer le père choisi. C'est pourquoi le système garde en mémoire l'endroit que l'on vient de quitter pour permettre d'y revenir ultérieurement.

Cette technique de mémorisation de l'emplacement d'où l'on part est également employée lors du déplacement vers les noeuds qui forment un réseau. Chaque fois que l'on saute vers un endroit qui n'appartient pas à l'environnement direct dans la structure hiérarchique du noeud courant, le système mémorise cette position. C'est le cas pour la multihiérarchie et le réseau ainsi que pour une méthode supplémentaire de déplacement qui permet de sauter à un endroit que l'on connaît.

Toutes les méthodes de déplacement citées jusqu'à présent ne permettent de se mouvoir que relativement à la position courante. Les fils, les pères ou d'autres noeuds reliés d'une manière logique peuvent être atteints. Si l'on connaît le nom d'un noeud dans la structure et qu'on désire l'atteindre, il faut suivre un chemin qui peut s'avérer long et complexe. De plus, il n'est pas certain que l'on connaisse exactement la route par laquelle on doit passer pour atteindre cet emplacement. Caliban permet de sauter directement à un endroit précis d'une structure en utilisant la commande JUMP que l'on a au bas de la fenêtre. A son activation, le système demande l'adresse à laquelle on veut se déplacer et l'utilisateur doit entrer au clavier le nom du noeud qu'il désire atteindre. Ceci représente une exception à la règle qui a été donnée dans les principes du système. L'utilisateur occasionnel n'a toutefois aucunement l'obligation de se déplacer de cette manière et cette technique permet à une personne

connaissant déjà le système, et avant tout les structures d'accès, de gagner du temps et de diminuer l'irritation que pourrait provoquer l'obligation de passer par tous les points d'un chemin qu'elle connaît déjà.

Ce genre de déplacement élargit la notion de navigation dans une structure étant donné qu'il ne s'agit plus de l'environnement immédiat de la position courante. Elle permet de se déplacer beaucoup plus rapidement à un point connu.

Voyons maintenant, par un exemple, comment on peut appliquer ces méthodes de déplacement pour rechercher de l'information.

7.4 Le survol, exemple de dialogue

Un des objectifs du système est de contrôler et même de suggérer des associations d'idées à l'utilisateur et de lui fournir un moyen de suivre les différents chemins qui se présentent. Ceci est particulièrement important lorsque l'utilisateur n'a qu'une vague idée de ce qu'il recherche. Dans ce cas, il a besoin d'un outil qui lui permette de survoler l'information et de collecter des termes qu'il pourra employer dans la formulation d'une requête. La représentation graphique des structures sur l'écran facilite la recherche d'idée pour un usage ultérieur. De plus, l'utilisateur acquiert une impression juste de la structure lorsqu'il se déplace explicitement vers le père, le fils ou vers une autre position reliée à l'endroit où il se trouve actuellement. Cette manière de survol de l'information est particulièrement bien adaptée à la manière associative de penser des êtres humains.

Le survol permet également l'inspection de documents liés aux termes index. Ceci donne une idée plus précise de la signification d'un terme particulier et correspond à l'attitude d'une personne qui se trouve dans une petite bibliothèque et qui feuillette quelques livres en passant devant les rayons. Dans plusieurs cas, un survol judicieux va amener l'utilisateur directement aux documents qu'il recherche sans avoir à formuler une requête précise.

Pour illustrer ce processus de survol, prenons une personne qui recherche de l'information sur les ordinateurs et leur utilisation. Supposons que cette personne est incapable de préciser exactement ce qu'elle recherche et qu'elle ne peut formuler une question précise. Pour cette raison, elle invoque l'utilitaire Browse en exécutant la commande du même nom. Le système lui offre alors le choix entre cinq structures. Notre utilisateur sélectionne le thesaurus en placant son curseur sur la case correspondante et en pressant sur le bouton SPECIFY INDEX (Fig 7.7). A cet instant, le thesaurus apparaît. Etant donné qu'il y aurait eu trop de termes au premier niveau, des noeuds intermédiaires ont été insérés pour permettre un accès plus rapide aux termes eux-mêmes (Fig 7.8). Chaque case indique le domaine couvert par les termes qui lui sont rattachés. On remarque que les noeuds sont ordonnés alphabétiquement. Notre utilisateur détermine que le terme *Computers* doit se trouver entre *Coi* et *Control o*. Pour y accéder, il décide d'inspecter la structure dans cette direction. Il place son curseur sur le noeud *Coi TO Control o* et presse le bouton de la souris désigné par MOVE. Chaque fois que l'utilisateur place la souris sur un noeud de l'écran et presse le bouton MOVE, la structure est redessinée avec le noeud choisi au centre de la fenêtre. Dans notre cas, *Coi TO Control o* s'incrit maintenant au milieu (Fig 7.9), remplaçant ainsi *Thesaurus* qui apparaît un niveau plus haut. On peut alors voir le nouvel environnement de *Coi TO Control o*, en particulier les noeuds qui lui sont rattachés.

Les flèches, à gauche et à droite en bas de la fenêtre, permettent de visualiser les fils du noeud courant qui ne peuvent être présentés sur l'écran. Plus précisément dans notre exemple, la flèche tout à droite permet d'amener le noeud *Communication* de sa

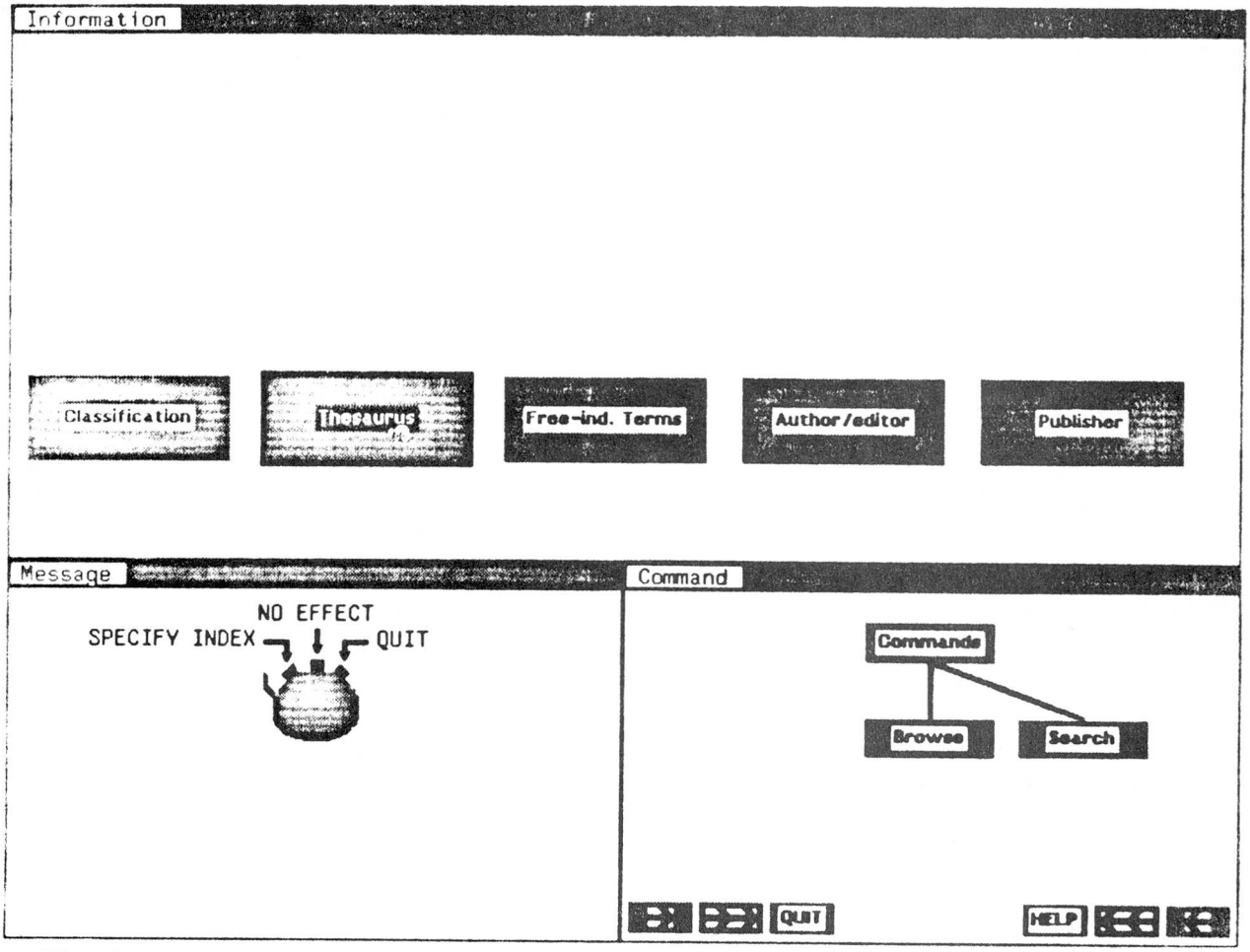

Information
Classification
Thesaurus
Free-ind. Terms
Author/editor
Publisher
Message
NO EFFECT
SPECIFY INDEX
QUIT
Command
Commands
Browse
Search
QUIT
HELP

Fig 7.7

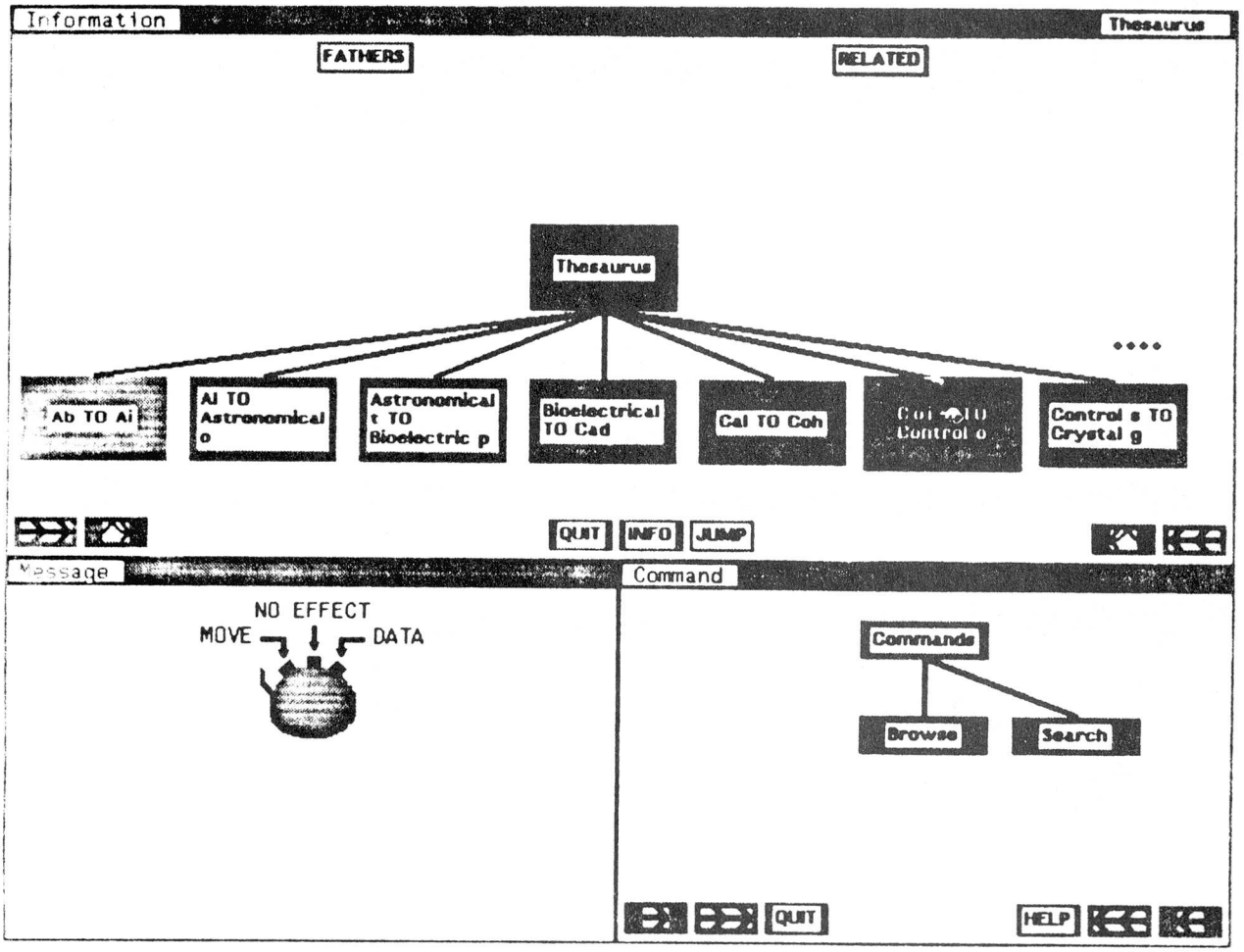

Information
Thesaurus
FATHERS
RELATED
Thesaurus
Ab TO Ai
Al TO Astronomical o
Astronomical t TO Bioelectric p
Bioelectrical TO Cad
Cal TO Coh
Control o
Control s TO Crystal g
QUIT
INFO
JUMP
Message
NO EFFECT
MOVE
DATA
Command
Commands
Browse
Search
QUIT
HELP

Fig 7.8

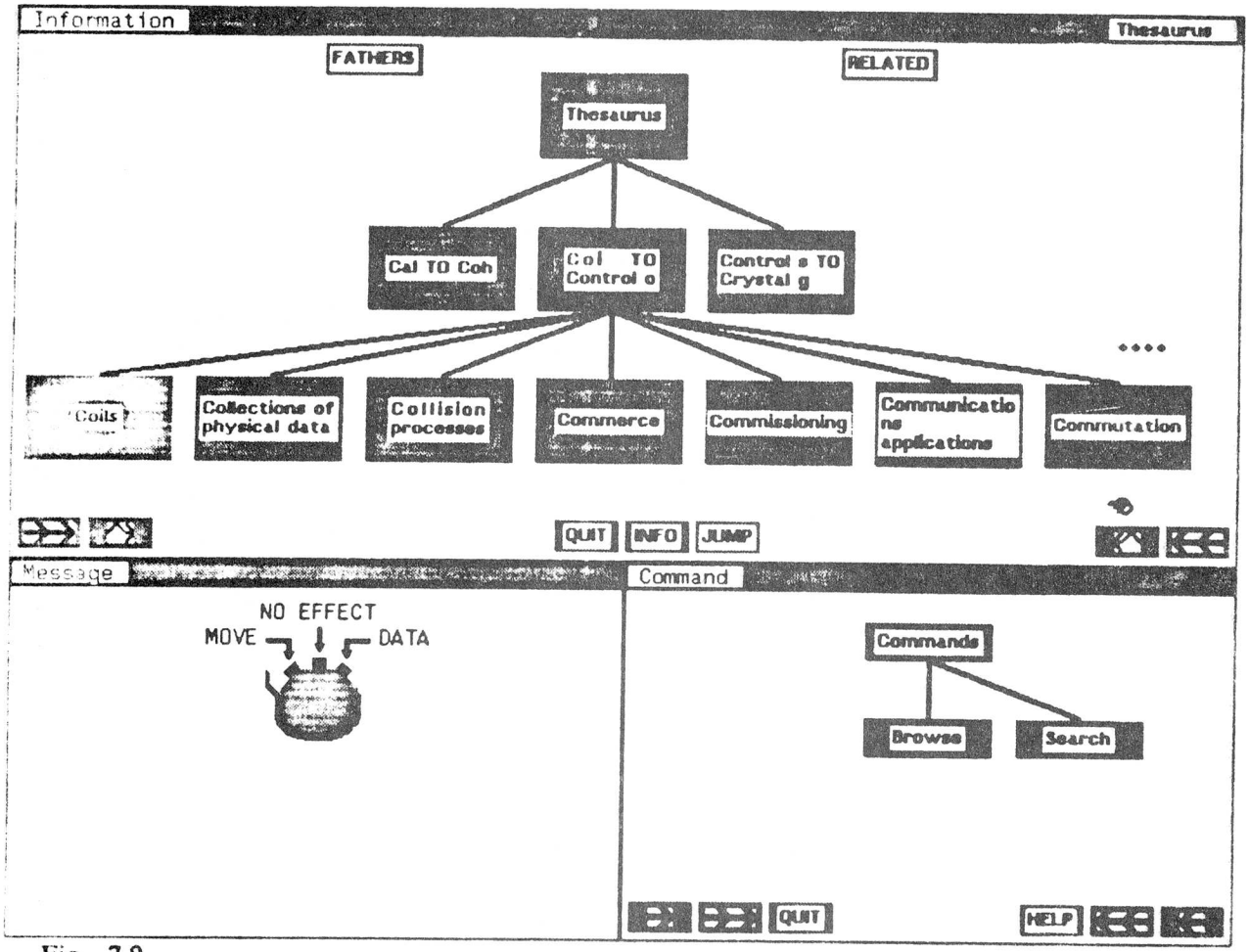


Fig 7.9

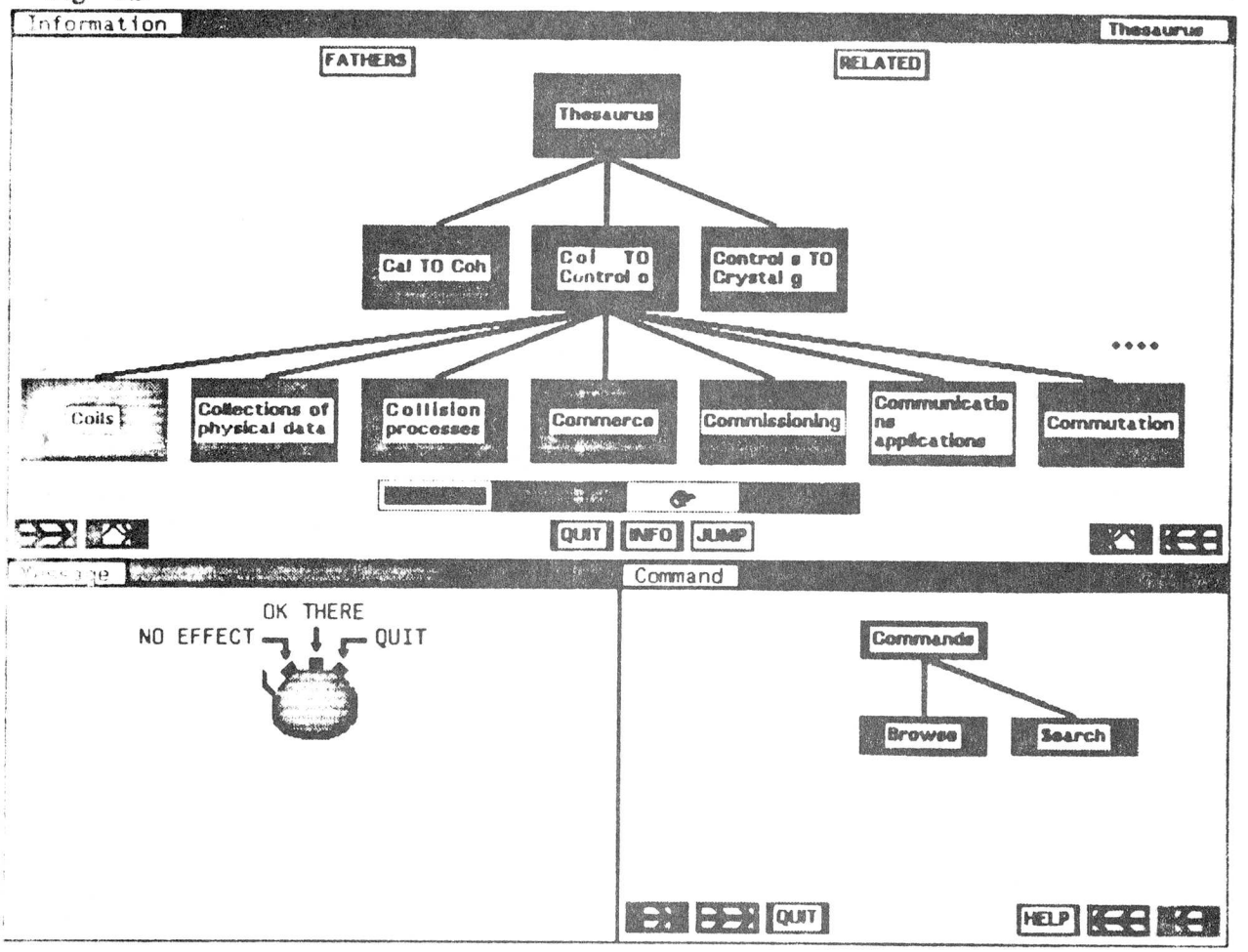


Fig 7.10

position la plus à droite de la fenêtre à celle le plus à gauche et ainsi de montrer six noeuds supplémentaires qui se trouvent à sa droite. La seconde flèche à droite fait apparaître une barre de saut (jump bar) qui représente à la fois l'ensemble des fils attachés au noeud courant et la portion actuelle sur l'écran (Fig 7.10). Dans notre cas, les premiers noeuds tout à gauche sont sur l'écran (rectangle avec le bord clair) et l'utilisateur se propose de visualiser ceux qui se trouvent environ au milieu de l'ensemble (position désignée par la souris). De cette manière, l'utilisateur atteint, à la figure 7.11 ,le noeud *Computers* et, en plaçant ce noeud au centre de l'écran (Fig 7.12), il peut voir les trois termes qui lui sont rattachés (*Analogue, Digital* et *Hybrid Computers*). En supposant que le domaine couvert par ces termes ne le satisfasse pas, il peut essayer de visualiser d'autres termes reliés au terme courant pour autant que la structure qu'il parcourt soit en forme de réseau tel que c'est le cas du thesaurus. En exécutant la commande RELATED, d'autres termes, qui ne sont pas directement liés par une structure hiérarchique mais qui ont tout de même un rapport avec le noeud courant, vont apparaître sur l'écran tel qu'on peut le voir à la figure 7.13. Dans ce cas, aucune ligne ne relie les termes étant donné que la relation est plus faible qu'auparavant. Le déplacement se fait toutefois de la même manière en pointant sur le terme que l'on désire. La visualisation des termes, qui n'apparaissent pas dans la fenêtre mais qui sont également liés à *Computers*, se fait exactement de la même façon que pour les fils, c'est-à-dire au moyen des flèches à gauche ou à droite. Notre utilisateur décide alors de s'intéresser à *Computer interfaces* en pensant que cela pourrait avoir un rapport avec la manière d'utiliser un ordinateur. Il choisit donc ce noeud comme but et se retrouve quelque part dans le thesaurus à l'endroit où se situe le terme *Computer interfaces* (Fig 7.14). Il n'a, en fait, aucune idée où il se trouve actuellement dans la structure. Il peut être très près ou très loin de l'endroit d'où il est parti. C'est pourquoi le système se souvient de la position de départ pour lui permettre de revenir en arrière au moyen de la commande BACK qui est apparue au moment où il a sauté. La représentation hiérarchique normale lui permet d'étudier à nouveau l'environnement du terme choisi. Il s'avère en fait que ce terme concerne les interfaces hardware des ordinateurs. Il s'en rend d'ailleurs compte en voyant le terme *Camac*, comme fils du noeud courant, qui est un type d'interface entre ordinateurs. Il décide alors de revenir à l'endroit où il se trouvait avant de sauter. Il utilise pour ce faire la commande BACK qui lui permet de retrouver sans peine l'endroit d'où il vient. Il peut ainsi étudier d'autres termes liés à *Computers* pour trouver ce qu'il cherche. *Programming languages* lui paraît approprié dans le domaine de l'interaction avec un ordinateur et il décide de voir à cet endroit si quelques termes peuvent le mettre sur une voie intéressante. Les fils de *Programming languages* ne paraissent pas spécialement être dans le domaine de sa recherche mais il voit *Programming*, le noeud à gauche du terme courant, qui pourrait lui donner quelques renseignements (Fig 7.15). Il pointe donc sur ce terme pour voir son environnement, en particulier ses fils, qui enfin lui apporte ce qu'il désire, à savoir *Interactive Programming* (Fig 7.16). La commande DATA, signalée dans la fenêtre *Message* sur le bouton de droite de la souris, permet d'accéder aux documents connectés au noeud sur lequel pointe la souris. Pour chaque noeud présent sur l'écran, il est possible de voir les documents correspondants en plaçant simplement le curseur à cet endroit et en pressant le

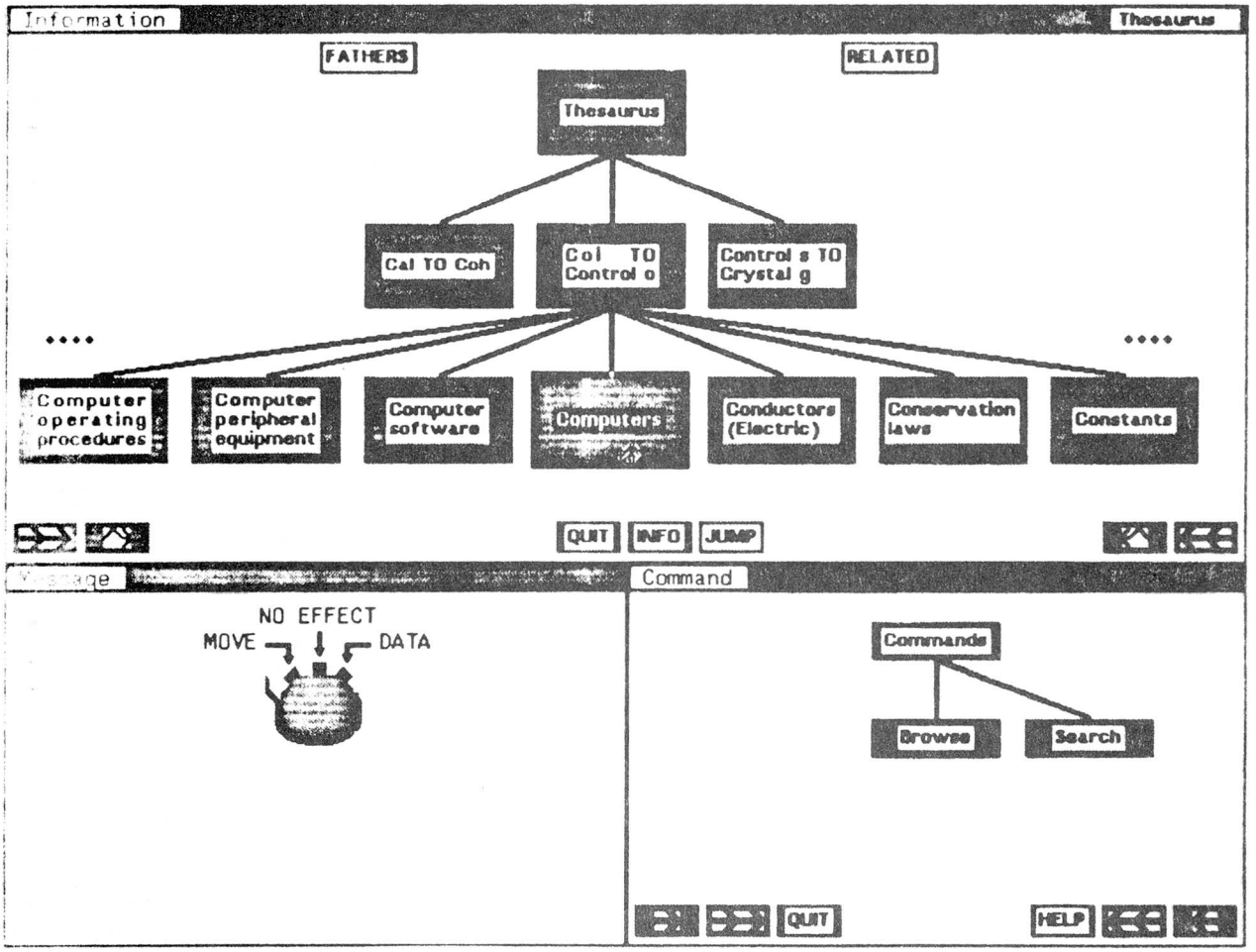


Fig 7.11

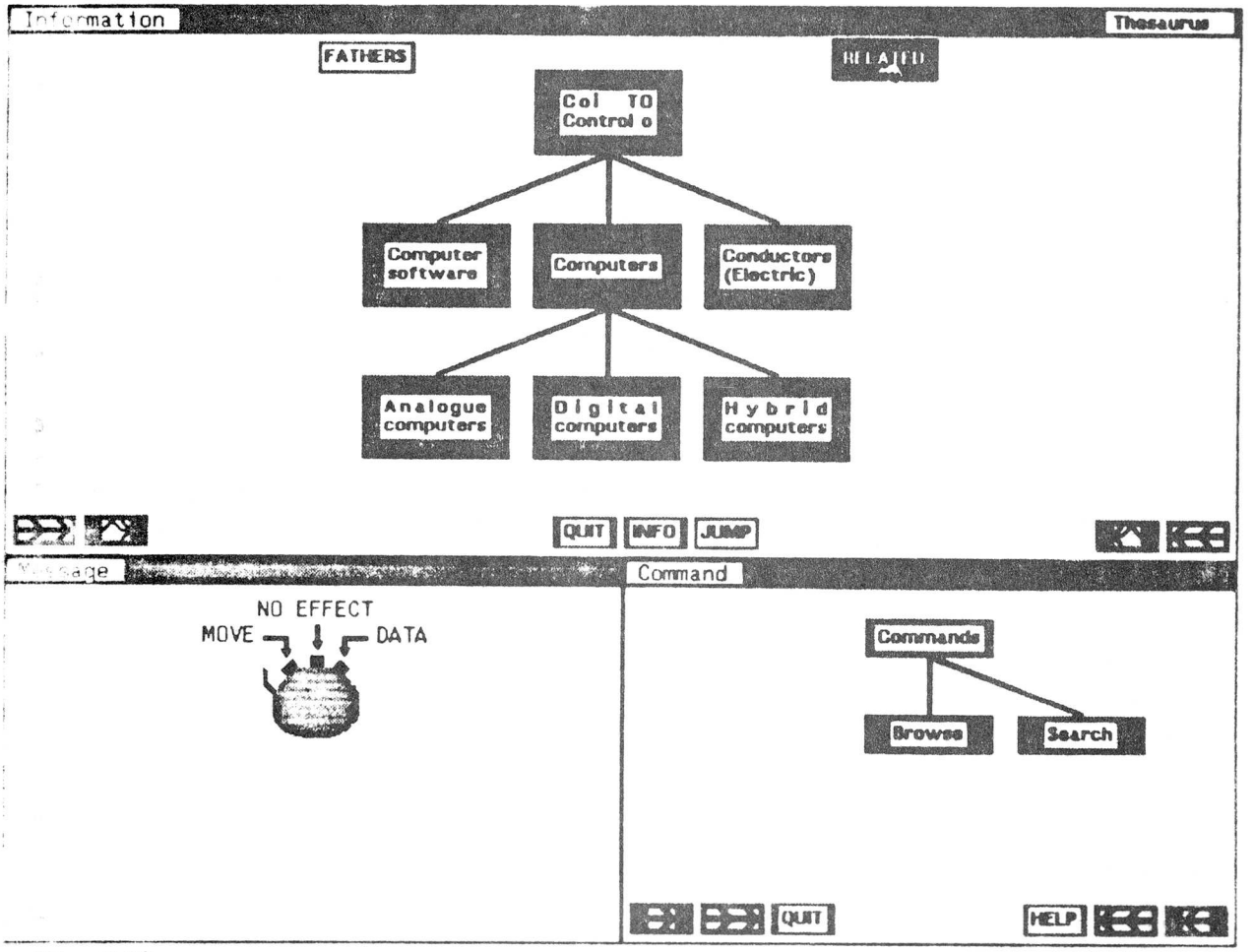


Fig 7.12

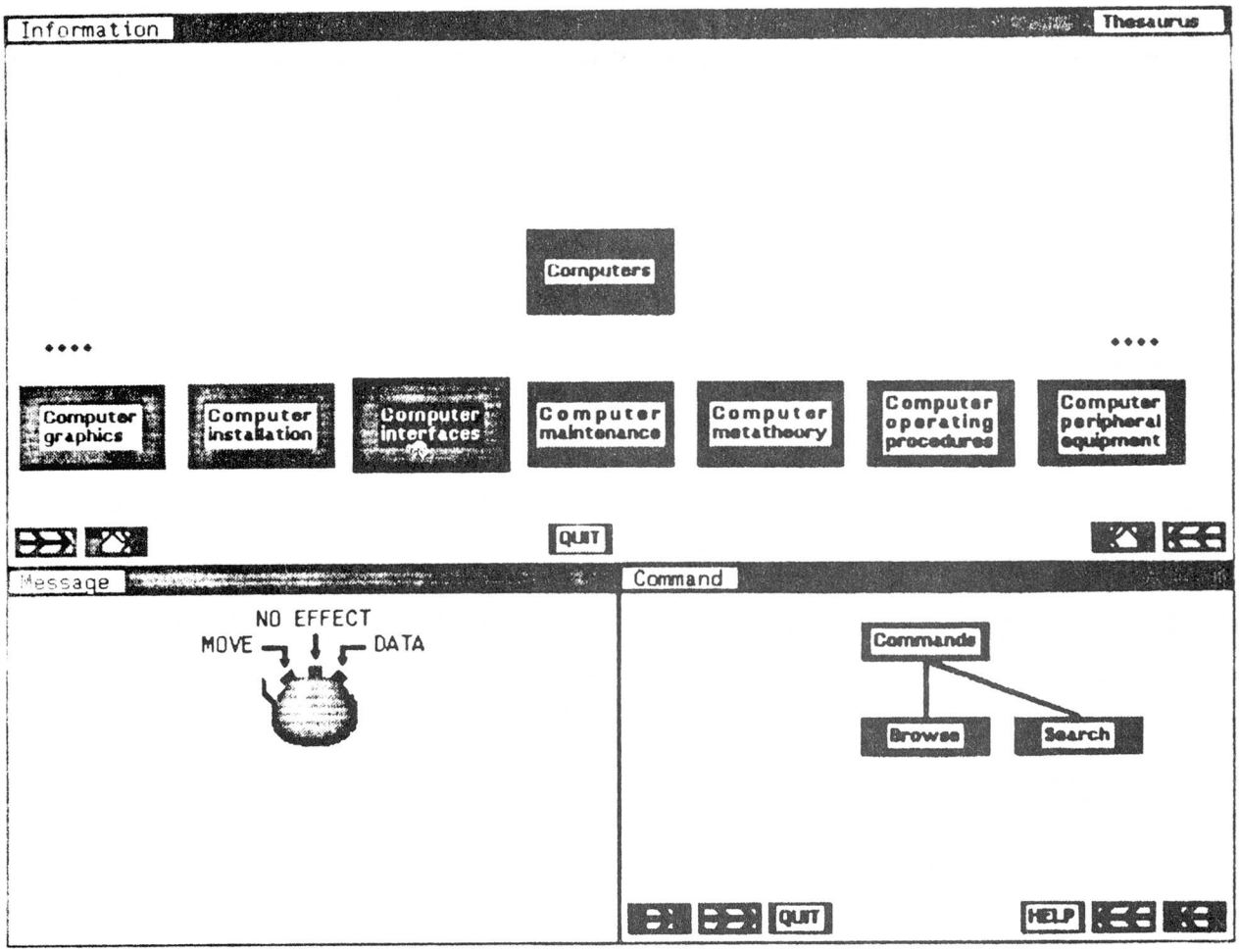


Fig 7.13

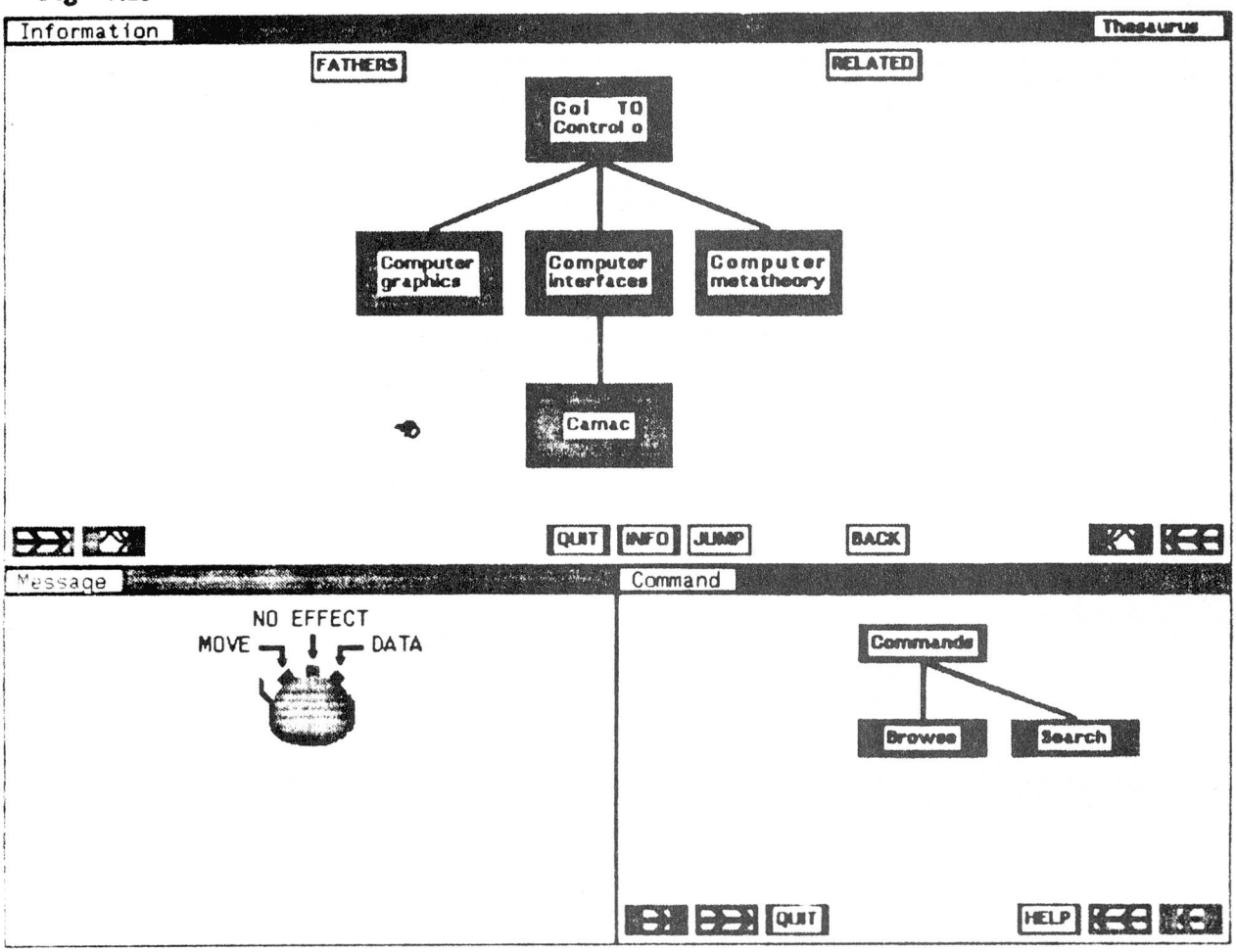


Fig 7.14

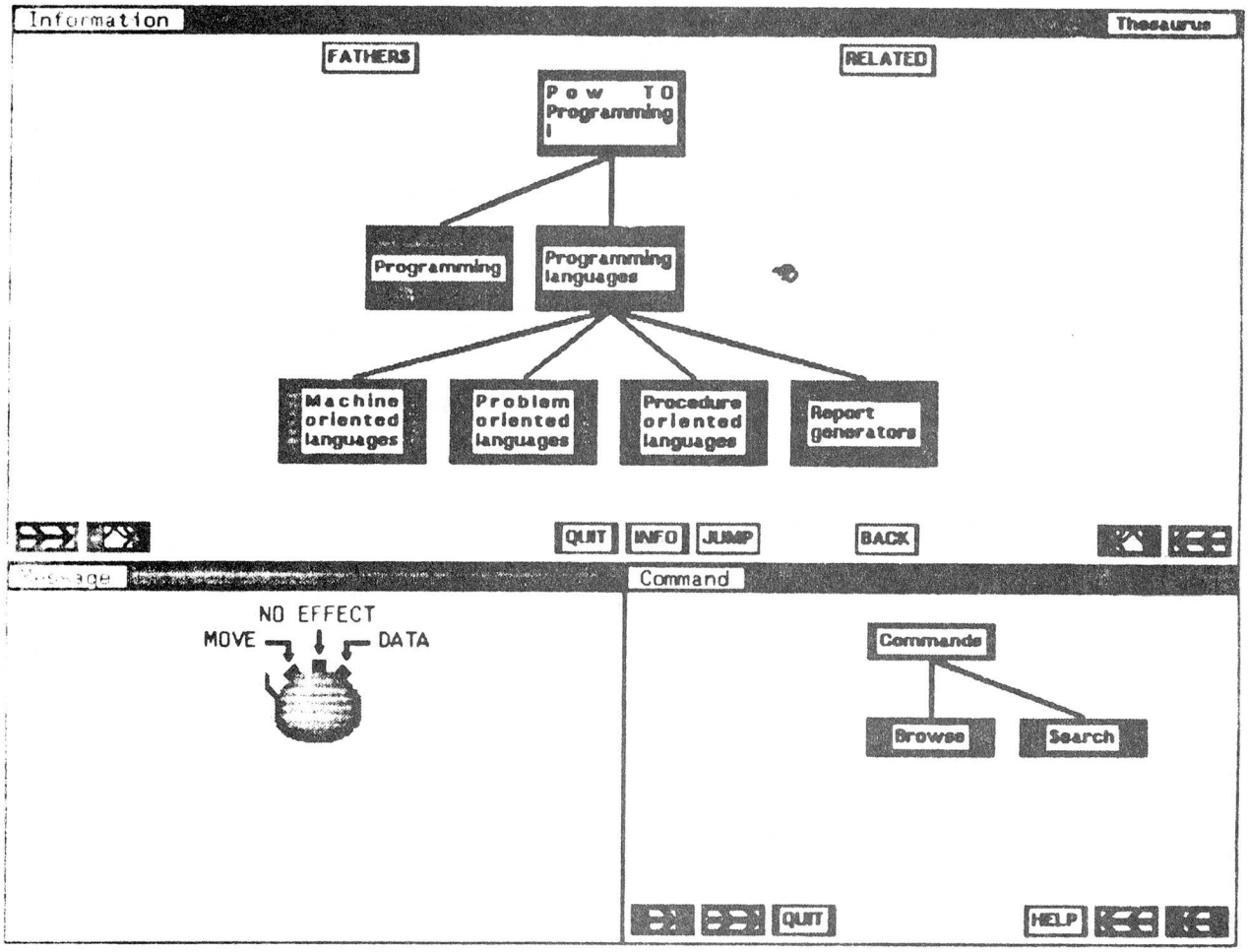


Fig 7.15

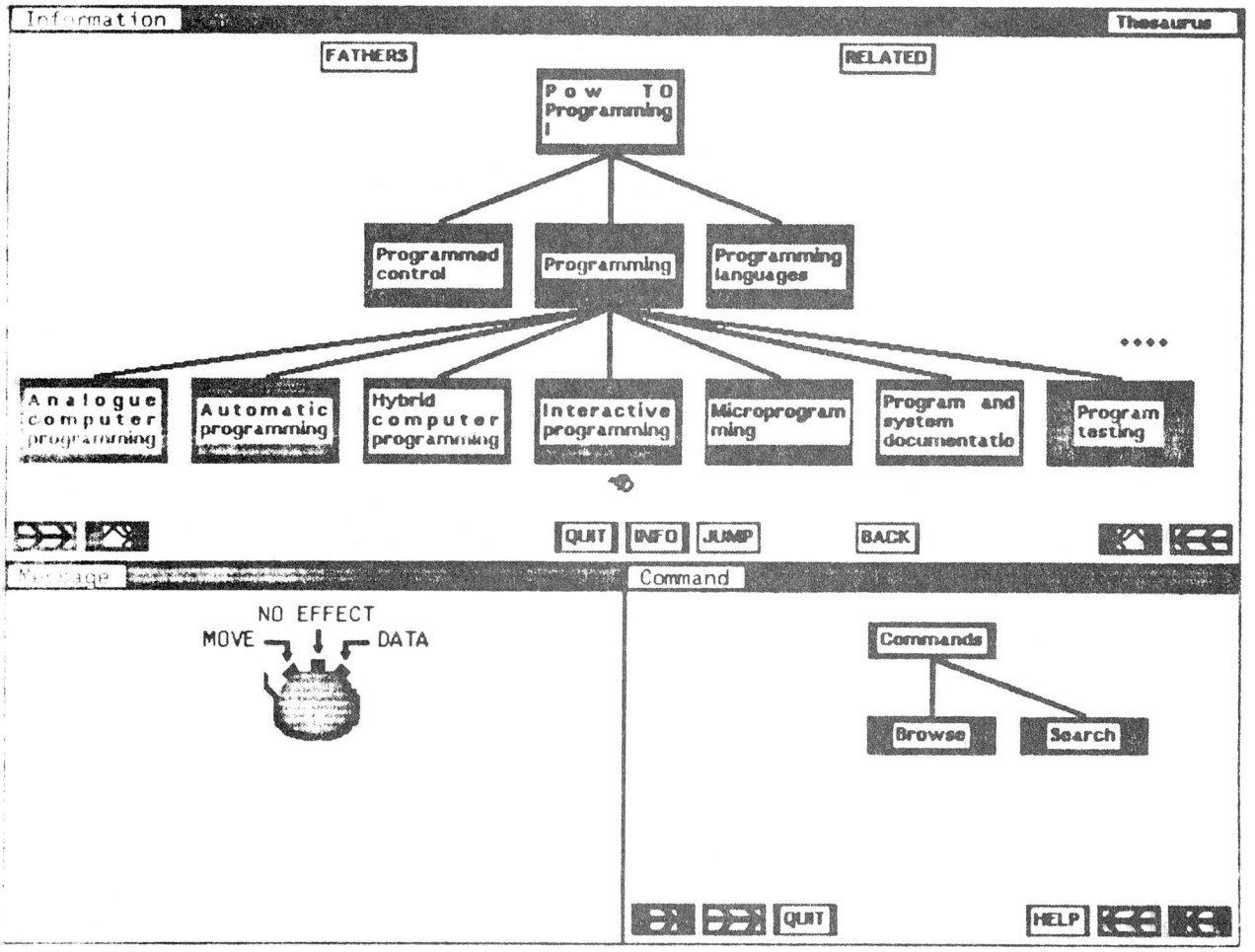


Fig 7.16

bouton DATA. La figure 7.17 montre le résultat de cette commande alors que l'utilisateur pointait sur le terme *Interactive Programming.* La dimension de l'ensemble des documents concernés par ce terme est affichée dans la fenêtre *Message.* Un des documents est affiché complètement dans la fenêtre *Information* et l'on voit le titre du document suivant. Tous les documents de l'ensemble ont, dans le champ thesaurus de leur description, le terme *Interactive Programming.* L'utilisateur peut alors étudier ces documents les uns après les autres. Nous reviendrons plus tard sur les techniques de visualisation des documents. Cet accès aux données n'interrompt pas le survol des structures. En utilisant la commande QUIT, on revient exactement à la position d'où l'on avait demandé les documents et l'on peut continuer le survol pour chercher d'autres termes et étudier d'autres documents.

Cet exemple montre les possibilités de recherche d'information par une technique de survol. Bien que cette technique permette souvent de trouver directement l'information que l'on recherche, elle possède deux caractéristiques qui la distingue des autres formes de recherche.

- La recherche dépend exclusivement des décisions particulières faites lors du déplacement dans les structures d'index. L'utillisateur peut suivre ses associations d'idées ce qui représente souvent une technique supérieure à la méthode traditionnelle de recherche. Toutefois, une grande quantité d'information peut échapper à l'utilisateur étant donné que la recherche se fait selon des chemins plus ou moins aléatoires.

- Le processus de survol ne permet pas d'accéder aux données depuis plusieurs points de la structure simultanément. La recherche est limitée à l'accès aux données depuis un point de la structure seulement.

Nous verrons plus loin une approche totalement différente pour rechercher de l'information, mais voyons tout d'abord quelles sont les techniques de visualisation des objets.

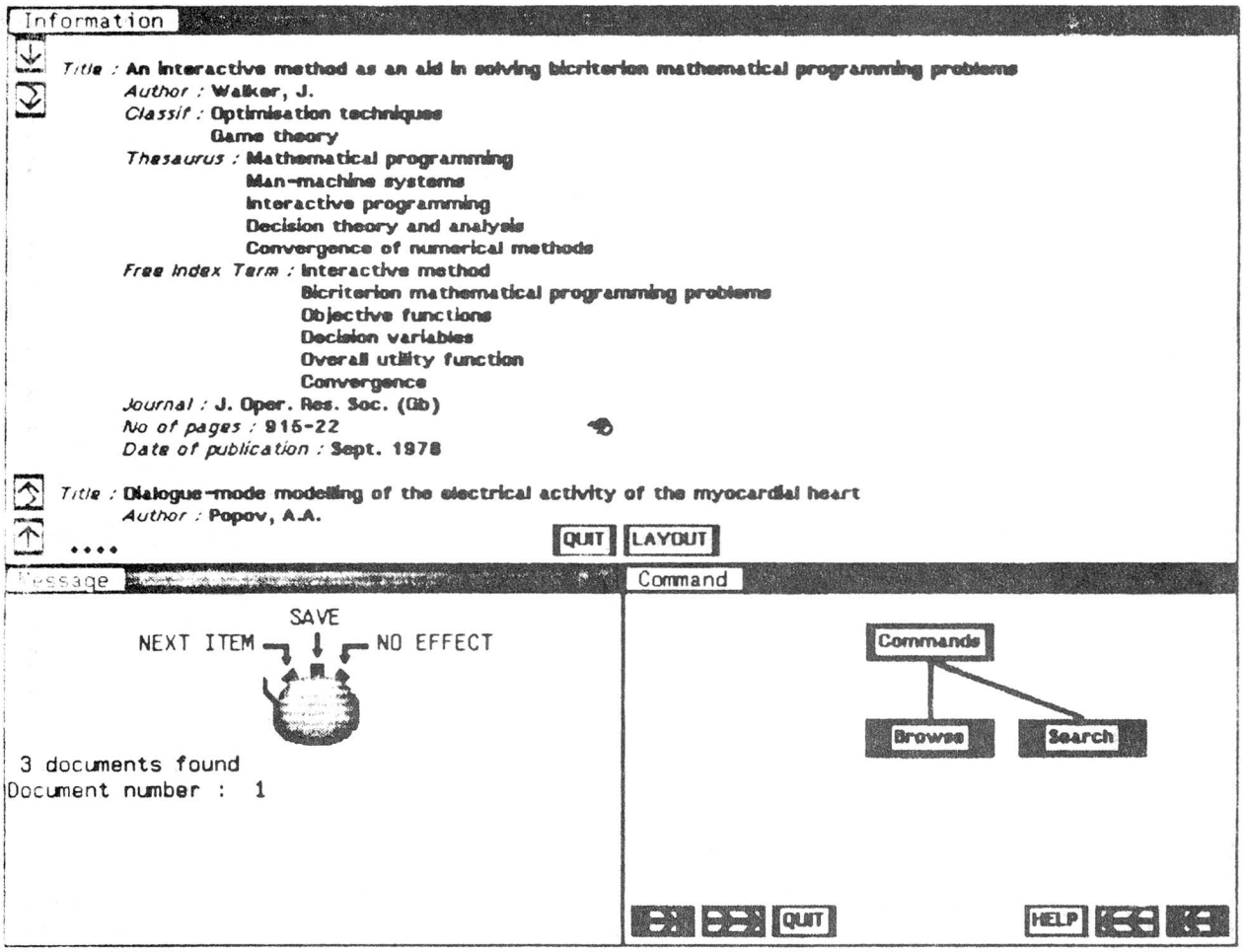


Fig 7.17

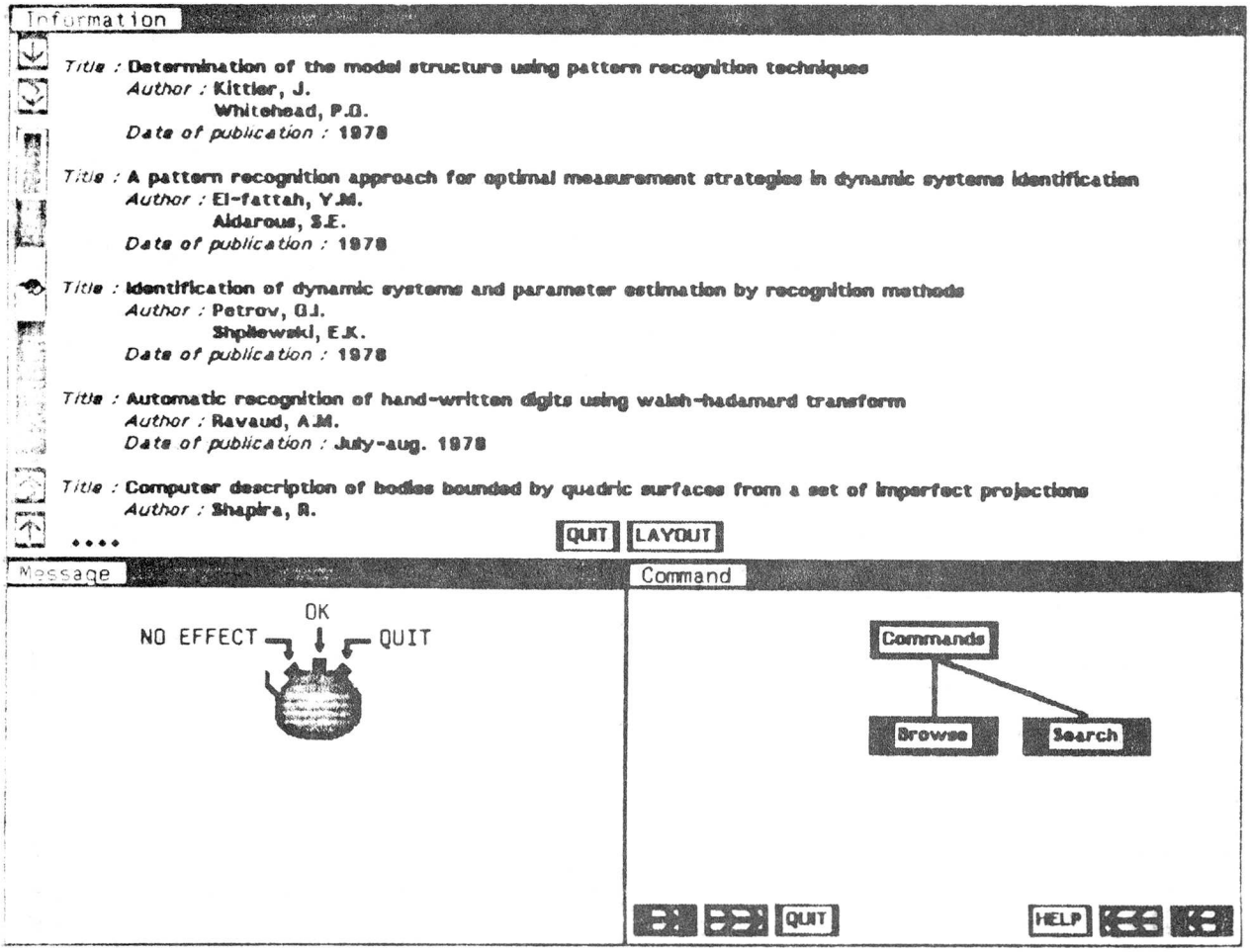


Fig 7.18

7.5 La visualisation des documents

Le but final d'un système de recherche d'information est de trouver parmi une grande quantité d'objets celui ou ceux qui sont intéressants. Si la visualisation des structures d'accès qui permet d'atteindre ces objets est très importante, la possibilité d'étudier les objets eux-mêmes est également primordiale. Le résultat des recherches fournit souvent un très grand nombre d'objets qu'il faut étudier pour en sortir ceux qui conviennent le mieux à la requête.

Nos données de test sont des références bibliographiques. Chaque référence est présentée à l'utilisateur dans la fenêtre *Information*. Elles sont constituées de champs ayant une ou plusieurs valeurs. Un titre est généralement unique tandis que plusieurs mots-clés décrivent le contenu de l'oeuvre originale. La représentation sur l'écran se fait de la manière suivante. Chaque champ est signalé par un en-tête écrit en italique. Puis vient le contenu de ce champ qui peut s'étaler sur plusieurs lignes (c'est le cas d'un résumé par exemple). Le titre de l'ouvrage est décalé par rapport aux autres champs, d'une part parce que c'est le champ le plus utilisé pour identifier l'oeuvre, et d'autre part parce que cela permet de séparer les références qui se trouvent les unes derrière les autres.

L'utilisateur n'est pas toujours intéressé par tous les champs de la référence. Il peut décider de n'en lire que le titre, l'auteur et le résumé pour se faire une idée du contenu des ouvrages trouvés. Il est toutefois très gênant de devoir situer l'information qu'on recherche parmi des données superflues. C'est pourquoi Caliban offre la possibilité de décider interactivement quels sont les champs que l'on désire voir. Ainsi l'utilisateur peut examiner les références de manière complète jusqu'au moment il décide de ne plus examiner que le titre de l'oeuvre. Si tout à coup un titre lui paraît intéressant, il peut immédiatement réclamer le reste de l'information concernant la référence. Ce changement se fait par la commande LAYOUT qui permet de modifier la vue que l'on a des références. Chaque champ peut être éliminé individuellement de cette vue par une simple commande. Le retour à la vue totale de la référence se fait également au moyen d'un seul bouton de la souris.

Comme nous l'avons dit plus haut, les références viennent à la suite l'une de l'autre. Il faut donc un moyen de se déplacer sur l'ensemble des objets que l'on a trouvés pour pouvoir tous les visualiser.

7.6 La navigation dans les documents

Si un document est trop long pour tenir entièrement place dans la fenêtre, un système de flèches, analogues à celles que l'on a de manière horizontale pour les structures d'accès, permet de voir le reste de la référence qui n'est pas affichée. Des points au bas de la fenêtre indiquent que le document n'est pas complètement sur l'écran. L'utilisation de la flèche tout en bas de la fenêtre permet de déplacer la dernière ligne affichée sur la première ligne en haut, ceci pour autant que le document soit suffisamment long. Dans le cas contraire, la dernière ligne du document se placera sur la dernière ligne de la fenêtre. Lorsque la vue du document a été déplacée, des points apparaissent en haut de l'écran pour signifier que l'on ne voit pas le début de la référence. La flèche tout en haut permet de ramener les lignes invisibles dans la fenêtre.

Si le document est complètement représenté sur l'écran, ces flèches de déplacement deviennent inutiles. C'est pourquoi elles servent, dans ce cas, à visualiser la référence suivante ou précédente. Le dernier document, partiellement affiché sur l'écran, est amené en haut de la fenêtre par la flèche du bas, tandis que le document précédant l'actuel en haut de la fenêtre peut être visualisé par la flèche du haut. Ceci permet de visualiser séquentiellement les documents qui ont été fournis par le système de recherche et représente la navigation au sens strict au sein d'une liste.

Toutefois, il se peut que l'on désire se déplacer de manière absolue dans l'ensemble des documents, lorsque l'on désire voir le dernier élément de l'ensemble ou revenir au premier. Le second jeu de flèches permet ce déplacement absolu en présentant une barre de saut, tout comme dans les structures d'accès mais verticale cette fois-ci, où l'on voit la position actuelle de l'information présentée sur l'écran et où l'on peut déplacer au moyen de la souris, l'emplacement de la fenêtre (Fig 7.18). Ce positionnement sera d'autant moins précis que le nombre de documents est grand. La longueur de la fenêtre est fonction du pourcentage d'information que l'on voit actuellement. Si quatre documents forment l'ensemble des données et que l'on voit actuellement un document entièrement sur l'écran, alors la hauteur de la partie visible occupera un quart de la barre de saut. Si deux documents de cet ensemble sont affichés, la partie visible prendra la moitié de la barre de saut. En ayant un ensemble de documents beaucoup plus grand, disons une centaine de documents, la hauteur de la partie visible est réduite en proportion et l'on ne distinguera qu'un simple trait au cas où un seul document apparaît dans la fenêtre. Dans ce cas, le positionnement absolu ne peut pas se faire de manière très précise, au document près, puisqu'il est difficile d'estimer où se trouve un document quelconque au milieu de l'ensemble. Seul le déplacement au début et à la fin de l'ensemble se fait de manière précise. On travaillera sinon de manière approximative, en approchant le plus près possible de l'endroit qu'on désire atteindre puis on utilisera les autres flèches pour se déplacer en avant ou en arrière pour obtenir le document désiré.

7.7 La formulation de requêtes, exemple de dialogue

Pour formuler une requête, l'utilisateur doit remplir les champs d'un exemple de document plutôt que d'utiliser un langage de commande. A la figure 7.19, il est en train de préparer ce que nous avons appelé un document virtuel. Cette technique peut être apparentée à la méthode de Zloof [Zloo75], dans son langage QBE (Query By Example), où il faut remplir les colonnes d'une relation qui sert d'exemple de ce que l'on recherche. Chaque champ correspondant à une structure d'index, tel que auteur, maison d'édition, classification, thesaurus et index libre dans nos données, peut être laissé vide ou rempli par des termes qui correspondent à ce que l'utilisateur recherche. Si il désire obtenir des documents sur un certain sujet écrit par un auteur de sa connaissance, il doit remplir les champs correspondants. La question est alors soumise au système de recherche qui fournira un ensemble de documents correspondant au mieux à celui proposé. Toutefois, l'utilisateur ne doit pas taper au clavier de longues séquences de caractères. Les structures d'index sont disponibles pour l'assister dans la préparation de sa question (SHOW INDEX TREE). Des termes peuvent être choisis sur l'écran et inclus dans le document virtuel.

La commande MOVE CURSOR (Fig 7.19) permet de sélectionner un champ du document virtuel pour le remplir de l'information qui va servir à la recherche. Un curseur de texte, en forme de rectangle, désigne le champ en cours de préparation. Si l'on désire choisir un autre champ, on approche la souris de ce champ et l'on presse le bouton MOVE CURSOR. Le rectangle se placera alors à cet endroit.

La structure d'accès, correspondant au champ désigné par le rectangle, peut être visualisée en exécutant la commande SHOW INDEX TREE (Fig 7.20). Bien que l'on soit en phase de préparation de requête, l'utilisateur peut utiliser toutes les commandes de BROWSE que l'on a vu auparavant. Le survol devient une sous-activité de la préparation des requêtes. Cette fois-ci, les termes intéressants peuvent être automatiquement insérés dans la requête grâce à la commande CHOOSE que l'on peut activer avec le bouton du milieu de la souris. Les trois commandes MOVE, CHOOSE et DATA s'appliquent à n'importe quel noeud actuellement présent dans la fenêtre. A la figure 7.20, les trois termes *Computers*, *Microcomputer* et *Minicomputers*, provenant du thesaurus, ont été choisis pour être inclus dans la requête, c'est-à-dire dans le document virtuel. En fait un nombre arbitraire de termes, accessibles par les commandes de déplacement à l'intérieur de la structure, peuvent être sélectionnés et inclus dans le document virtuel. Caliban confirme les choix faits en affichant les termes dans la fenêtre Message. On quitte la structure d'index en activant la commande QUIT qui nous ramène à la requête en préparation. Chaque terme, choisi lors du survol, apparaît maintenant dans le champ correspondant. L'utilisateur peut à nouveau choisir une structure d'index (éventuellement la même) pour collecter d'autres termes. Dans notre exemple, la classification a été parcourue et l'utilisateur a inclus également les classes *Artificial intelligence* et *Pattern recognition* (Fig 7.21).

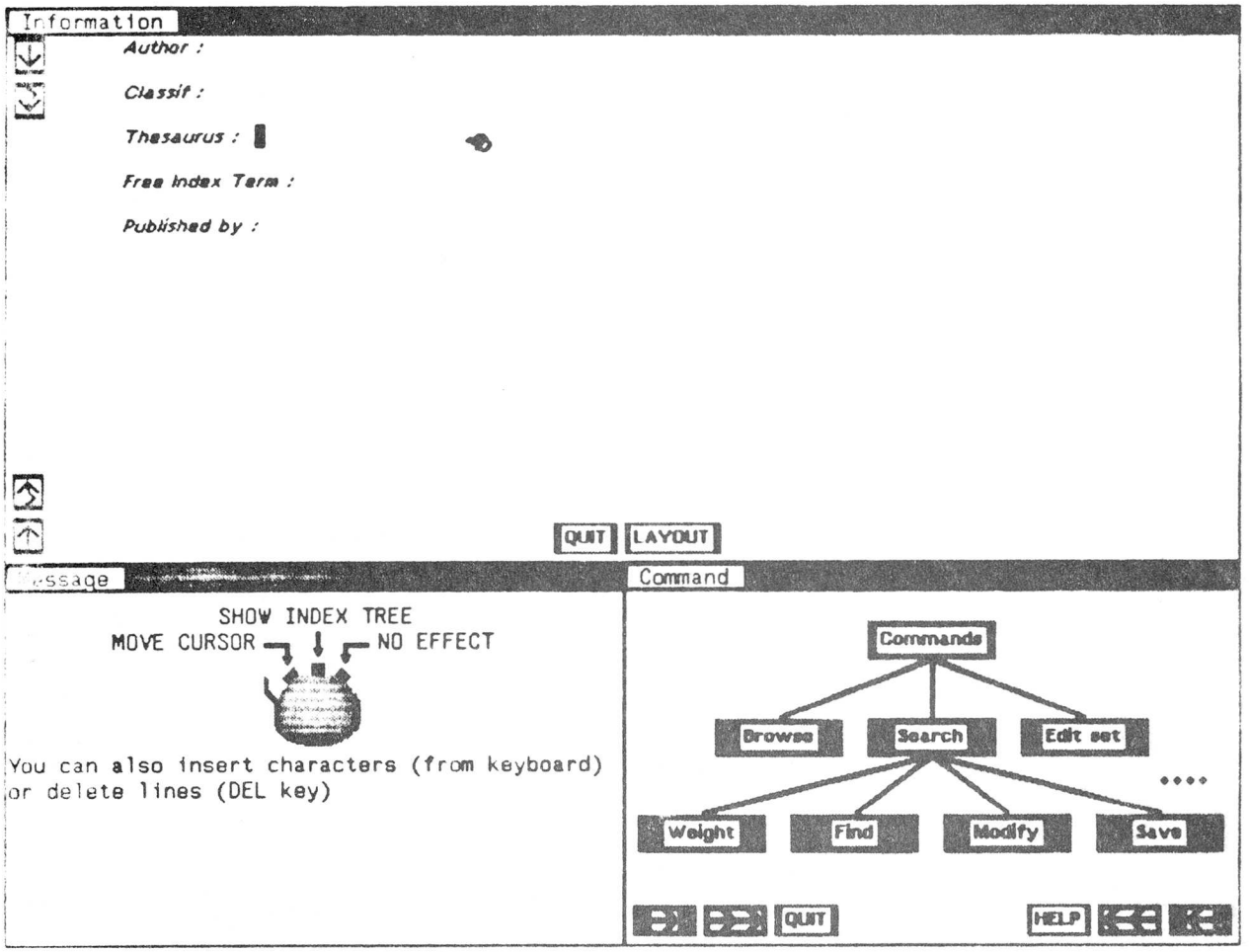


Fig 7.19

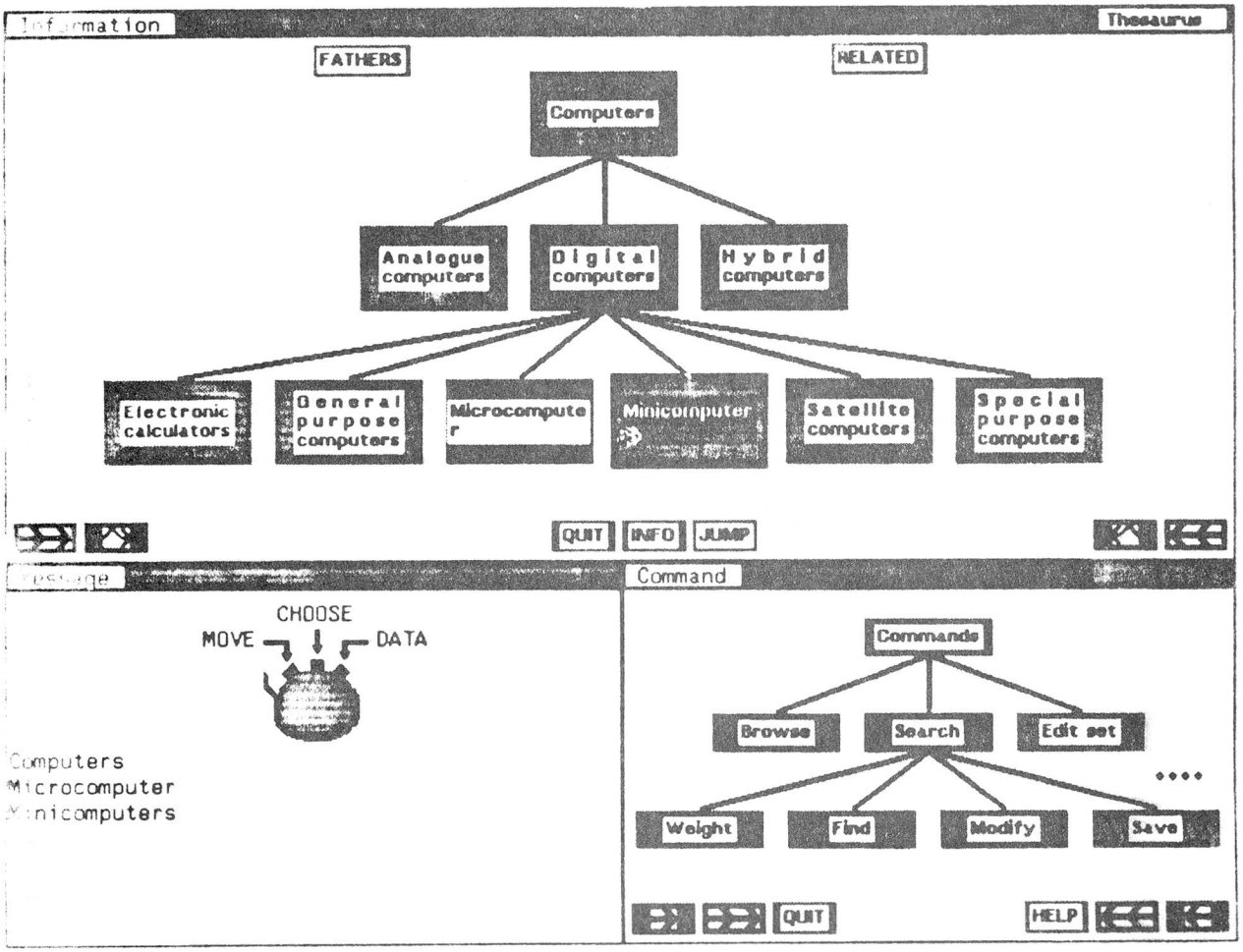


Fig 7.20

Une fois que la requête est remplie de manière satisfaisante, l'utilisateur quitte la phase de préparation (QUIT) ce qui le ramène au dialogue dans la fenêtre *Command*. Find lui permet de rechercher les documents correspondant à sa requête. Il lui suffit d'exécuter cette commande pour obtenir la liste de tous les documents ayant un rapport avec le document virtuel. L'algorithme de recherche fournit les documents qui ont un degré de similarité avec la requête plus élevé qu'une certaine valeur. Cet algorithme assigne à chaque élément de l'ensemble un rang correspondant au degré de similarité entre la requête et l'élément. Les documents fournis par l'algorithme de recherche ne se présentent donc pas dans un ordre quelconque, mais sont ordonnés de manière décroissante selon leur degré de similarité. Cet algorithme de recherche est décrit dans [Bärt84].

Avant de soumettre sa requête à l'algorithme, l'utilisateur peut choisir l'importance qu'il attribue à chacun des termes de la requête. En effet, il peut s'avérer que les termes choisis n'interviennent pas tous au même degré dans la question. Le terme *Artificial intelligence* peut paraître plus intéressant que *Pattern recognition*, bien que les deux termes présentent un intérêt. L'utilisateur a la possibilité de donner un poids à chaque terme en fonction de l'importance qu'il lui accorde et que ce terme doit jouer dans le processus de recherche. Il le fait en activant la commande Weight (Fig 7.21) qui affiche la requête avec un rectangle pour chaque terme du document virtuel. La couleur de ce rectangle donne le poids actuel du terme dans la requête. Plus la couleur du rectangle est foncée, plus le terme a d'importance. Ces rectangles sont disposés sur un fond de couleur neutre pour permettre de donner un poids négatif à un terme. Ainsi l'utilisateur peut éliminer de sa question les documents décrits par un certain champ. On peut imaginer l'exemple de quelqu'un qui recherche de l'information sur un certain sujet mais qui désire éliminer les ouvrages écrits par un auteur dont il connaît toute l'oeuvre. Si le poids d'un terme est identique à la couleur de fond, ce terme ne jouera pas de rôle dans le processus de recherche.

Ces poids sont utilisés à la fois par l'algorithme de recherche et par celui qui ordonne les documents dans l'ensemble qui forme le résultat. Le premier ne retourne que les éléments qui ont une valeur au-dessus d'une certaine limite. L'algorithme d'ordonnancement assigne un rang élevé aux éléments qui possèdent une grande similarité avec la requête.

Cette méthode d'évaluation de l'importance des termes paraît plus naturelle à l'utilisateur occasionnel. En effet, cela lui permet de juger de l'importance relative des termes l'un par rapport à l'autre sans avoir à se soucier des priorités entre les opérateurs booléens. De plus, le système n'exige pas une évaluation des termes étant donné qu'il attribue automatiquement un poids maximum lors du choix d'un terme dans la structure d'accès. Ainsi une requête n'ayant pas passé par le processus Weight a un poids maximum identique pour tous ses champs.

Le résultat de la commande Find donne une liste de documents dont on peut voir le premier à la figure 7.22. La première requête ne satisfait généralement pas l'utilisateur. Comme nous l'avons déjà dit, la recherche est un processus itératif et le

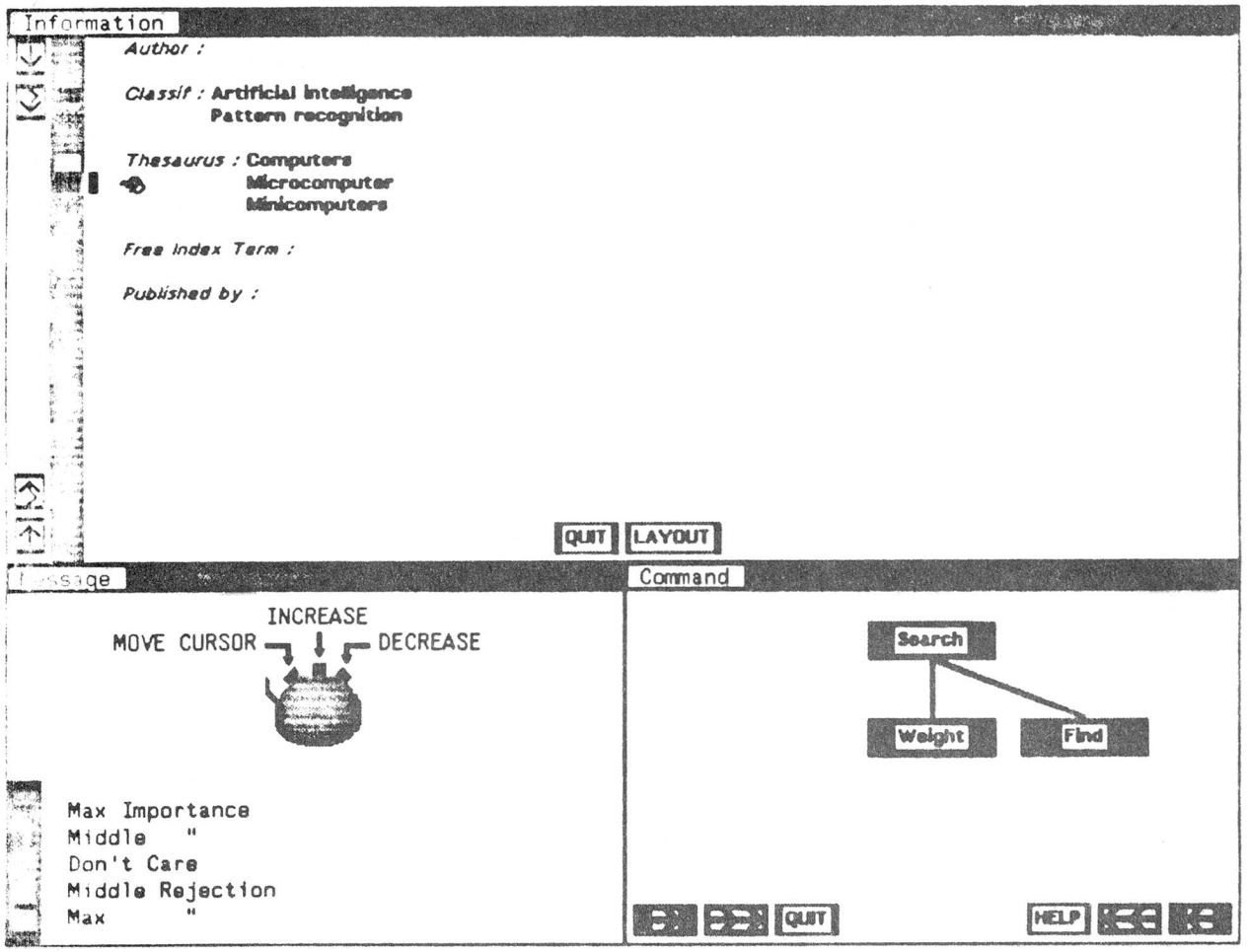


Fig 7.21

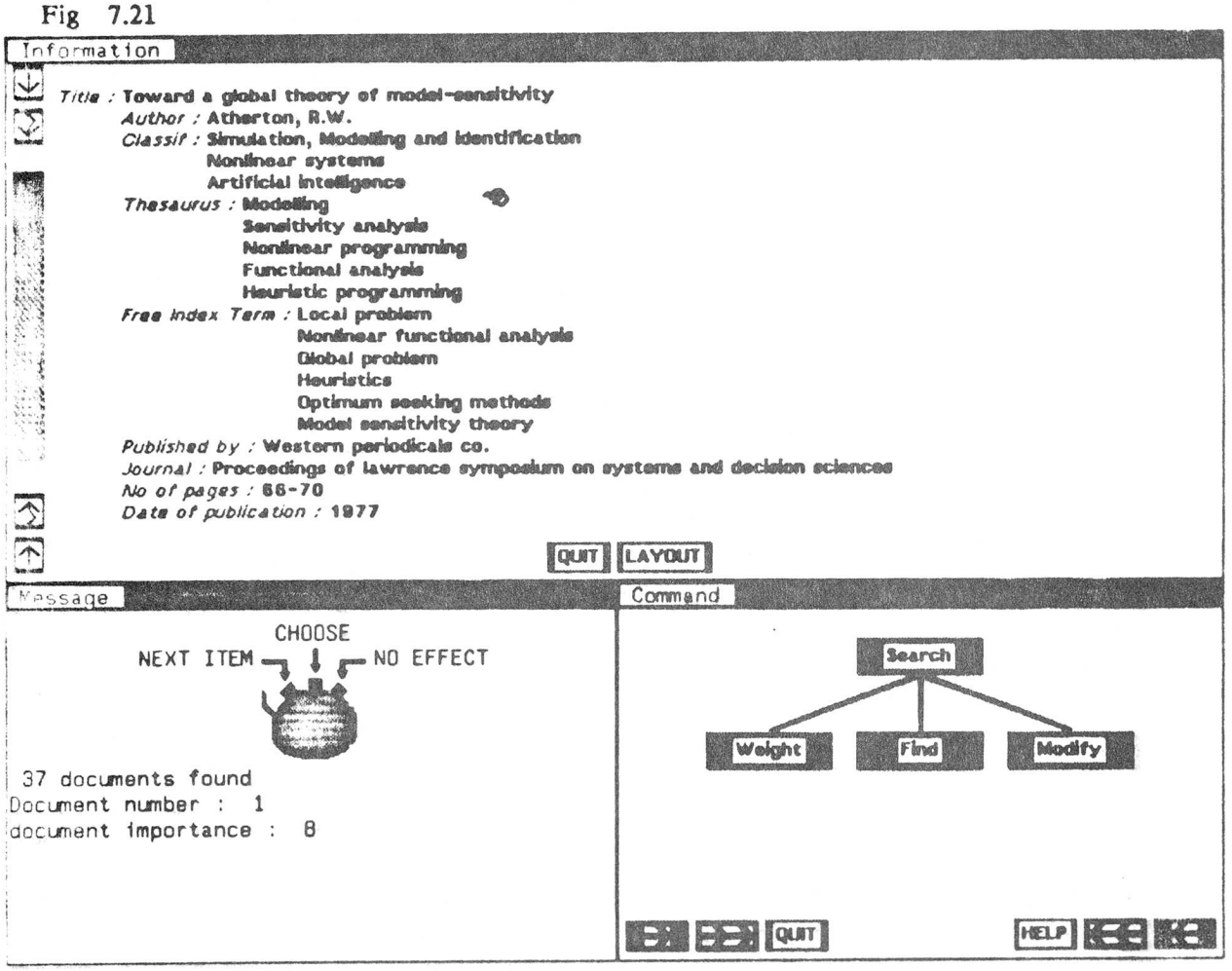


Fig 7.22

système doit fournir le moyen d'accomplir cette amélioration. Lorsque l'ensemble des documents réels est affiché, il est possible d'en choisir un pour l'utiliser comme requête. Ceci est rendu possible grâce à l'identité de la structure d'une requête (un document virtuel) et de celle d'un document (document réel). Une fois le document choisi, on peut le modifier grâce à la commande Modify. Des termes peuvent être éliminés ou d'autres, tirés des structures d'accès, incorporés au document qui devient à nouveau un document virtuel (puisqu'il n'existe pas comme tel dans les données). Lorsque l'utilisateur soumet cette requête au système, celui-ci va fournir, comme résultat, une liste de documents dont le premier sera vraisemblablement le document qui a servi à construire la requête (ceci n'est pas obligatoirement le cas étant donné que l'on peut modifier considérablement le document original ou attribuer des poids négatifs aux termes de la question ce qui produira une valeur de similarité avec le document lui-même relativement faible). Mais il trouvera éventuellement dans ce nouvel ensemble, un autre qui correspondra encore mieux à son idée.

Ce processus d'amélioration n'est pas le seul possible dans Caliban. Il peut arriver que l'ensemble de documents ne satisfasse pas du tout l'utilisateur et qu'il ne soit pas possible de prendre un autre document comme requête. Tout d'abord, le système conserve l'ancienne requête aussi longtemps que celle-ci n'est pas remplacée. Cela signifie entre autre que l'on peut revenir à son ancienne question et la modifier légèrement pour obtenir d'autres résultats (insérer d'autres termes tirés des structures d'accès). Enfin l'utilisateur peut trouver dans la réponse des termes intéressants disséminés au travers des documents qu'il étudie. Pour ne pas l'obliger à rechercher la position de ce terme dans la structure, Caliban permet de choisir directement des termes contenus dans les résultats et de les inclure automatiquement dans la requête en cours. Par exemple, *Modelling*, que l'on voit à la figure 7.22 dans le champ thesaurus, peut être pris comme terme dans la question de la figure 7.19.

Après la visualisation des documents, on revient au niveau de commande avec QUIT et le système affiche dans la fenêtre *Information* la requête en cours (l'ancienne modifiée ou non, ou la nouvelle). L'activation de l'utilitaire Modify nous ramène au niveau d'édition que l'on a déjà eu lors de la préparation de la première requête. On peut éliminer ou insérer des termes qui paraissent attractifs que l'on tire des structures d'accès. Un point important est apparu au cours du développement de Caliban. Si un terme a été pris dans un document et intégré dans la requête, l'utilisateur ne sait pas où se trouve ce terme dans la structure. Il peut évidemment utiliser la commande JUMP pour l'atteindre mais cela peut être relativement pénible à faire (Les termes du thesaurus peuvent avoir jusqu'à plus de 120 caractères !). C'est pourquoi nous avons mis au point le système suivant : lorsque l'utilisateur positionne le curseur de texte (rectangle) sur un terme de sa requête et qu'il demande à voir la structure d'index du champ dans lequel se trouve ce terme. Le système va présenter la structure avec ce terme comme noeud courant (au centre de la fenêtre). Ceci permet de retrouver l'emplacement dans la structure du terme choisi dans un document et de visualiser l'environnement de ce terme dans la structure ce qui peut à nouveau apporter de nouvelles idées pour la requête suivante. L'utilisateur pouvait

avoir oublié toute une partie de la structure, n'ayant pas pensé à un terme particulier, et la retrouver facilement par ce biais-là.

Cette idée de prendre en compte des résultats de recherche précédente a déjà été soulignée par Oddy [Oddy77] qui décrit un système permettant d'insister sur des termes appartenant à un document de l'ensemble des résultats ou d'éliminer des termes de la requête qui semble hors du domaine de recherche. Toutefois le système THOMAS ne va pas aussi loin dans l'intégration et l'utilisation de termes pour les requêtes suivantes. En particulier il n'est pas possible de revenir à l'environnement du terme dans sa structure d'accès.

Le dernier point important dans un processus de recherche est de montrer à l'utilisateur quelle est la qualité des résultats qu'il a obtenus suite à une requête. En effet, supposons que l'ensemble des documents soit formé de 100 éléments. La répartition de la qualité de ces documents est primordiale pour savoir comment continuer la recherche. Si parmi ces 100 documents, les 80 premiers ont une similarité avec la requête qui est relativement égale et les 20 derniers présentent un moindre intérêt, alors l'utilisateur sera tenté d'améliorer et de préciser sa requête pour réussir à former un ensemble plus petit qu'il pourra étudier dans sa totalité. Si, dans un cas contraire, les 5 premiers documents de l'ensemble présentent une similarité beaucoup plus grande que le reste, il pourra décider d'étudier directement ces documents. Les deux points importants à connaître pour un ensemble donné sont:

. Le taux de similarité des documents avec la requête
. La répartition de la similarité parmi les documents

Caliban donne une indication approximative des résultats obtenus. Nous avons vu que la hauteur de la barre de saut représente la totalité des documents de l'ensemble lorsqu'on essaye de positionner la partie visible. Caliban utilise cette représentation pour indiquer la répartition des documents. A chaque document de l'ensemble est attribuée une valeur indiquant sa similarité avec la question (Retrieval Status Value) [Bärt84]. Caliban attribue une "couleur" aux différentes valeurs qui interviennent dans l'ensemble et présente à l'utilisateur cette répartition à l'emplacement de la barre de saut (Fig 7.22). De plus, si l'on utilise les flèches de positionnement pour visualiser un autre document que celui actuellement sur l'écran, la couleur de fond de la barre de saut ne sera plus uniforme, comme c'était le cas lors du survol, mais représentera justement cette répartition. Cela permet de voir quelle est la valeur du document que l'on a actuellement sous les yeux et surtout de ne visualiser que les documents qui ont une valeur de similarité supérieure à une certaine limite. Cela permet également de reconnaître les deux cas de l'exemple donné plus haut lorsque 80 ou 5 documents retournent la meilleure valeur de similarité.

7.8 Définition de la grammaire formelle

Au chapitre précédent, nous avons montré la définition d'une grammaire formelle qui pouvait s'adapter à un langage de recherche d'information. Cette grammaire devait aider à formaliser le langage et surtout éliminer les parties inconsistantes qui pouvaient subvenir dans la définition du langage. La grammaire a été définie plus spécialement pour le cas de Caliban. Elle se présente de la manière suivante:

```
<session de RI>        ::= { <opération> } .
<opération>            ::= <spécification> <exécution> .
<spécification>        ::= <soft key> | <opérande> <assertion> .

<soft key>             ::= "activation d'une clé sur l'écran".
<opérande>             ::= "choix d'un noeud de l'arbre de commande" |
                           "choix d'un noeud de la structure d'accès" |
                           "choix d'un terme d'un document" |
                           "entrée d'une chaîne de caractères au clavier" .
<assertion>            ::= <mouse button> | <keyboard> .
<mouse button>         ::= "activation d'un bouton de la souris" .
<keyboard>             ::= 'N' | 'Y' | 'ESC' | 'CR' .

<exécution>            ::= <réaction du système> { <opération> } .

<réaction du système>::= "présentation d'une structure" |
                         "présentation d'un objet" |
                         "présentation de la barre de saut".
```

Une session de Recherche d'Information est définie comme une suite d'opérations. Chaque opération consiste tout d'abord en une spécification (où l'utilisateur précise ce qu'il veut effectuer) suivie de l'exécution de cette opération. La phase d'exécution comprend la réaction du système (cette phase ne fait pas partie intégrante du langage mais sert à améliorer la compréhension de la grammaire) suivi à nouveau d'opérations qui sont maintenant des sous-opérations de la première effectuée. La partie principale du langage est définie par la spécification. Ceci peut se passer de deux manières différentes : soit l'utilisateur emploie un des boutons présents sur l'écran (soft keys) et dans ce cas il peut utiliser un bouton quelconque de la souris pour terminer sa spécification, soit il désigne un noeud qui se trouve sur l'écran, en particulier dans une structure d'accès ou dans l'arbre de commande, et il certifie ce choix par un des trois boutons de la souris en fonction de l'opération qu'il veut effectuer. Il faut remarquer que la sémantique de l'opération, dans ce dernier cas, se situe dans le choix du bouton de la souris que l'on a choisi alors que dans le cas d'un bouton sur l'écran, cette sémantique réside dans la position de ce bouton.

On voit, en examinant cette grammaire, qu'il est possible d'implémenter Caliban sur une machine ne possédant pas une souris munie de trois boutons mais qu'il suffit d'avoir un pointeur qui permet de désigner des objets sur l'écran; la signification des

commandes pourrait se faire au moyen d'autres "soft keys" prévues à cet effet ou encore être entrée via les touches du clavier. Du reste, l'utilisation de la souris pour entrer des commandes s'est révélée très agréable pour un utilisateur occasionnel. Par contre, une personne qui connaît bien le système a tendance à préférer l'emploi du clavier si chaque opération peut être accomplie au moyen d'une seule touche. Il est d'ailleurs prévu d'introduire cette méthode dans une version ultérieure de Caliban. Le problème qu'il s'agit de résoudre dans cette solution est le suivant : chaque spécification de commande peut être accomplie soit par la souris, soit par le clavier, mais il faut fournir un moyen à l'utilisateur occasionnel d'apprendre la relation qu'il existe entre les deux modes (surtout dans une phase transitoire lorsqu'il ne connaît pas par coeur toutes les commandes). Le déplacement dans la structure peut se faire par exemple en donnant le numéro du fils que l'on veut avoir comme noeud courant. Pour revenir au père on peut prendre la lettre "p" comme commande correspondante au mouvement que l'on fait normalement en pointant sur ce noeud dans la fenêtre. Il faut trouver un moyen satisfaisant de permettre à l'utilisateur d'apprendre que ce mouvement se fait par la lettre "p" sans pour autant donner de l'information sur l'écran qui pourrait s'avérer incongrue à un utilisateur ne s'intéressant pas à cette possibilité. Ce dédoublement des moyens d'entrée de commandes a déjà été testé dans un système relativement analogue à Caliban qui sert à modifier interactivement les structures d'accès. Ce programme n'est pas destiné à des utilisateurs occasionnels mais utilisé de manière interne pour le développement et la modification des données. Il ne fournit malheureusement pas le moyen d'apprendre quelles sont les touches qui correspondent aux moyens d'entrée par le curseur. De plus, il n'a pas été possible de savoir quels étaient les préférences des utilisateurs. La plupart du temps une combinaison des deux modes d'entrée est employée, en particulier le déplacement dans la structure se fait plus volontiers à l'aide du curseur en pointant sur le noeud où l'on veut se rendre et les autres opérations se font à l'aide du clavier.

Prenons maintenant un exemple de session effectué avec le système en détaillant exactement chaque mouvement fait par la personne qui utilise le système. Cet exemple montre une recherche faite par un survol dans la classification où les documents décrits par la classe *Systems and Control Theory* sont visualisés. Les lignes qui sont écrites en caractères gras ne font pas partie intégrante du langage. C'est ce qui a été désigné par "réaction du système" dans la grammaire formelle et qui n'appartient pas aux opérations que doit faire l'utilisateur.

. Curseur sur Browse |
. Bouton 1 (EXECUTE) | spec
. **5 Possibilités**
. Curseur sur *Classification* |
. Bouton 1 (SPECIFY) | spec
. **1er niveau de l'arbre**
. Curseur sur *Control* |
. Bouton 1 (MOVE) | spec
. **2eme niveau de l'arbre**
. Curseur sur *Systems and control theory* |
. Bouton 3 (DATA) | spec
. **1er document sur l'écran**
. Curseur sur flèche |
. Bouton 1 | spec
. **Document suivant**
. Curseur sur flèche |
. Bouton 1 | spec
. **Document suivant**
..... (jusqu'au dernier document)
. Curseur sur QUIT |
. Bouton 1 | spec
. **3eme niveau de l'arbre**
. Curseur sur QUIT |
. Bouton 1 | spec
. **Retour à la fenêtre Command**

Chaque spécification est suivie de l'exécution de l'opération. L'exécution est représentée de manière décalée sur la droite pour que l'on voit le temps qu'elle dure. En particulier, celle de Browse se poursuit jusqu'à la dernière ligne. Chaque exécution peut comprendre à nouveau une série de spécifications et d'exécutions. C'est le cas lorsque l'utilisateur active la commande DATA pour visualiser les documents. D'autres exemples pourraient être donnés, comme la préparation d'une requête, où l'on choisit des termes dans une structure d'accès, suivi de la visualisation des résultats.

Ceci achève la présentation du système de recherche d'information Caliban qui montre une tentative d'application des principes généraux énoncés au chapitre 6. Il est intéressant de remarquer que ce système suit de manière relativement stricte les principes de Gebhardt et Stellmacher que nous avons présentés au chapitre 3. Bien que ces principes ne nous aient pas été connus lors de la mise au point du système, nous pouvons dire que Caliban est une réalisation satisfaisante des divers points cités. Ce système va même plus loin dans les domaines que l'on a vus au chapitre 6. Il est relativement difficile de présenter sur le papier un système basé fortement sur des principes interactifs. Nous avons toutefois tenté d'en donner une illustration qui devrait montrer le caractère attractif du système et les innovations qu'il représente par rapport à l'état actuel des systèmes de recherche d'information traditionnels.

8. Détails d'implémentation

8.1 Les modules de Caliban et leur chargement

Le système Caliban est écrit en Modula-2 [Wirt82], langage qui permet un développement modularisé des programmes. Nous voulons, dans ce chapitre, donner une vision d'ensemble de l'implémentation du système, en énumérant tout d'abord les différentes parties de Caliban avec leurs interconnections, puis en présentant quelques aspects de programmation qui nous paraissent intéressants. Le but de ce chapitre n'est pas de donner une description complète du système mais plutôt d'en tirer l'essentiel et de présenter quelques techniques de programmation qui ont été suivies lors du développement de Caliban.

La machine sur laquelle tourne le système est de la classe des ordinateurs personnels (relativement puissante toutefois). Ceci nous a amenés à faire quelques compromis entre les temps de réponse, la place en mémoire centrale et les possibilités du système. En ce qui concerne l'interface utilisateur, le poids a toujours été mis sur la rapidité de réaction du système. Il est en effet primordial, sur ce genre d'ordinateur, que l'utilisateur ait une très grande interaction avec le système. Nous avons dû cependant, pour des raisons de place, définir une structure avec des "overlays" qui perd certainement un peu de temps au niveau du chargement mais qui permet de ne pas avoir constamment en mémoire des modules utilisés rarement.

Cette structure de chargement apparaissant à la figure 8.1. SEK (Sequential Executive Kernel) est la partie centrale du système d'exploitation Medos-2 qui tourne sur Lilith [Knud83]. C'est elle qui sert de base au chargement du module principal Caliban qui ne fait qu'initialiser le système de fichiers hiérarchiques (TFS, Tree File System [Suga82]) et afficher une page de titre au début et à la fin de la session. Cette initialisation du système de fichiers se fait en appelant CalibanInterpreter, l'interpréteur de commandes du système, qui charge les modules de TFS. C'est lui qui par la suite s'occupera de l'analyse des commandes et de la gestion des différents modules qui pourront être chargés.

Le premier module appelé par l'interpréteur est la phase d'initialisation qui a pour but premier de préparer les fenêtres de visualisation et la représentation des structures qui vont intervenir par la suite. En effet, presque toutes les structures d'accès sont préparées à l'avance de manière à permettre une apparition instantanée lorsque l'utilisateur les demande. Deux autres modules ne sont pas chargés constamment dans le système et requièrent un certain temps de réaction lorsqu'ils interviennent (de l'ordre de 3 à 4 secondes). Il s'agit tout d'abord du processus de recherche qui entre en jeu lorsque l'utilisateur soumet une question complexe (une requête) pour obtenir les documents qui lui sont similaires. A cet instant, la personne se rend compte de la complexité de la question et est prête à payer le prix du temps de chargement. Le processus d'évaluation de la question prend lui-même un temps de l'ordre de quelques secondes (ceci est naturellement fonction du nombre de

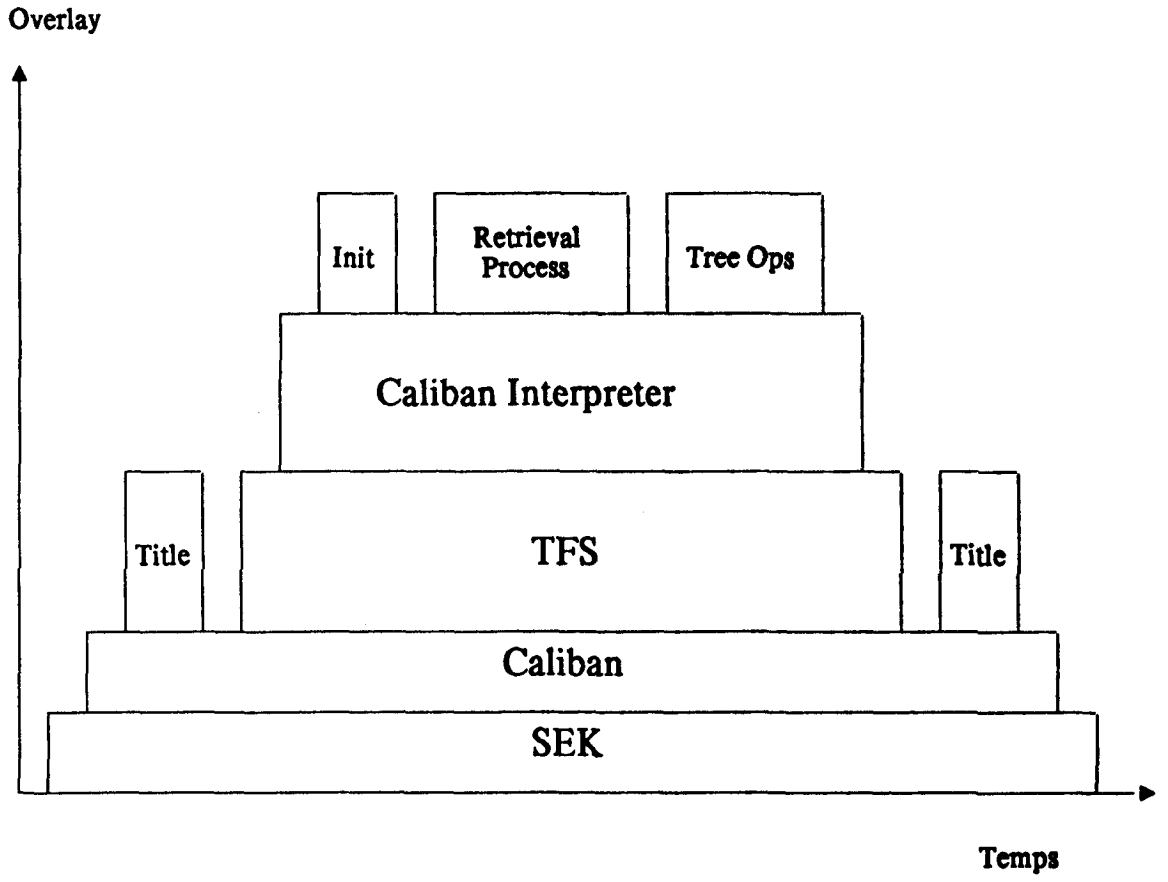


Fig 8.1 La structure de chargement

documents qui va former la réponse mais en général en-dessous de 5 secondes).

Le deuxième module qui n'est pas chargé constamment dans Caliban possède une série d'utilitaires qui permet d'effectuer des manipulation au niveau de la structure des données. En particulier l'insertion et la destruction de noeuds dans la structure hiérarchique définie par TFS. Ce processus n'intervient que quand l'utilisateur désire sauver un ensemble de documents, ou un document tout seul (une requête par exemple) qu'il désire utiliser par la suite. Ce sauvetage se fait en insérant dans une hiérarchie un noeud qui va représenter l'ensemble des documents sauvés. L'utilisateur donne un nom à cet ensemble qui va jouer le rôle de représentant de l'ensemble et qui va s'inscrire dans un noeud d'une structure hiérarchique visible sur l'écran. L'utilisateur peut ainsi conserver les documents qu'il a déjà trouvés auparavant et ainsi ne pas perdre de temps à les rechercher une seconde fois. Caliban lui donne la possibilité de gérer ses données en insérant de nouveaux ensembles et en éliminant d'autres. Ce processus n'étant toutefois pas utilisé très fréquemment, il est chargé en "overlay".

L'ordre dans lequel sont montrés ces deux processus dans la figure 8.1 n'est donc pas relevant puisqu'il dépend des choix de l'utilisateur. Les autres modules sont eux dans l'ordre chronologique exact tel qu'il apparaît si l'on parcourt le schéma de gauche à droite.

La figure 8.2 présente tous les modules qui font partie intégrante du système Caliban. Elle est à observer de bas en haut étant donné que les modules les plus bas sont ceux qui représentent les utilitaires de base. Chaque module ne fait appel qu'à des procédures définies dans d'autres modules du même niveau que lui ou d'un niveau inférieur. On voit évidemment apparaître CalibanInterpreter au sommet du diagramme puisque c'est lui qui gère le système et qui appelle les procédures en fonction des désirs de l'utilisateur. D'autres, tels que DisplayItems et MultiTreeDisplay qui s'occupent de la représentation des objets (documents et structures d'accès), interviennent plus bas dans la hiérarchie.

Les modules que nous allons examiner un peu plus en détail sont CalibanInterpreter qui est le centre de toutes les actions effectuées, MultiTreeDisplay qui définit la représentation graphique des structures, les procédures de RetrieveItems qui est le module de recherche, ListManager qui est un utilitaire de traitement de listes et qui définit la structure interne d'un document, et enfin HMString qui s'occupe du traitement des chaînes de caractères qui jouent un très grand rôle dans un tel système.

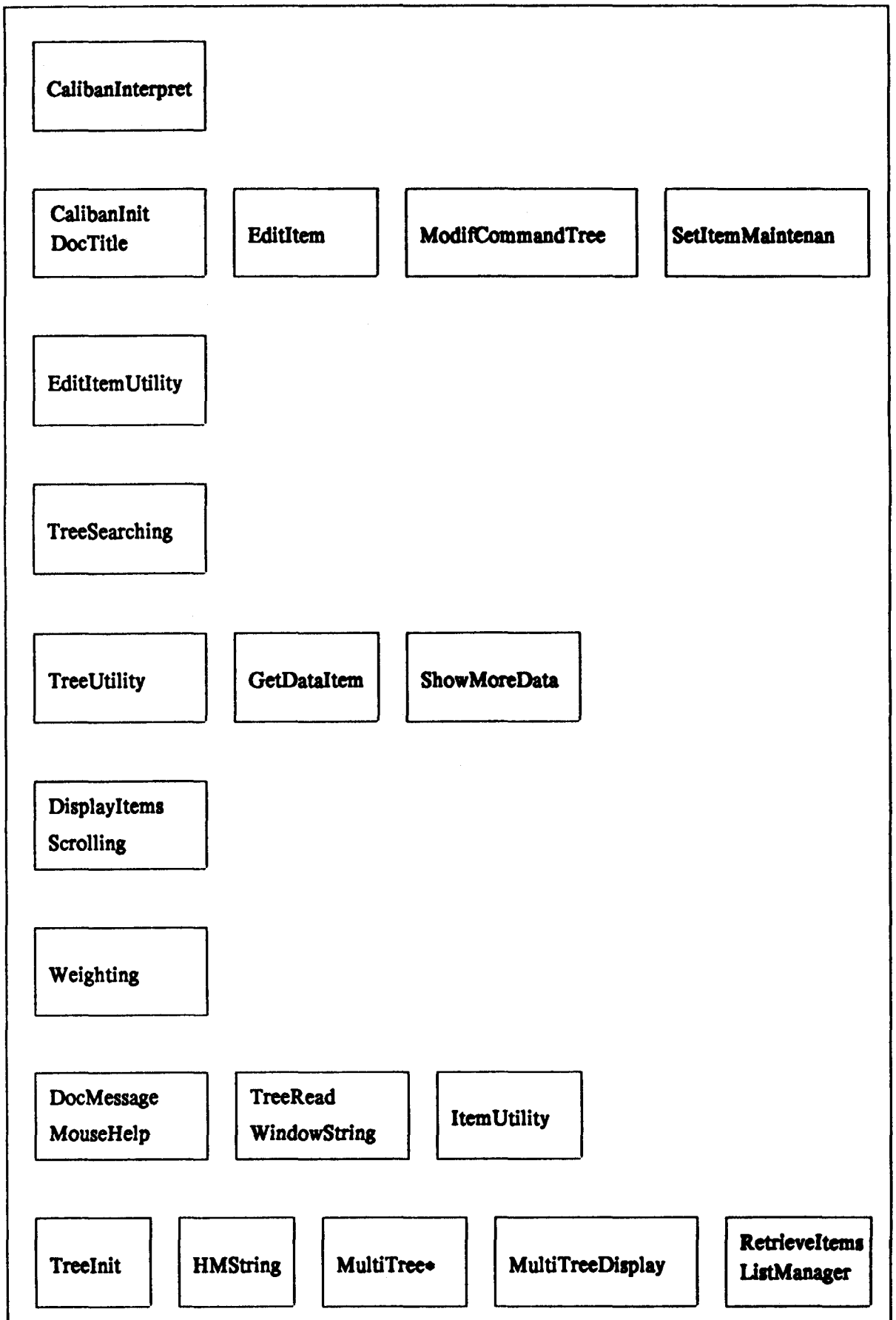


Fig 8.2 Les modules de Caliban

8.2 CalibanInterpreter, l'interpréteur de commandes

Ce programme procède d'abord à quelques initialisations avant de partir dans une boucle qui demande une commande à l'utilisateur pour ensuite l'exécuter. La structure générale du programme se présente comme suit :

```
MODULE CalibanInterpreter;

TYPE IRCommands = (Browse, Search, Weight, Find, Save, ......);

VAR  cmd       : UserCommand;
     IRCmd     : IRCommands;

BEGIN
  InitializeCommands;
  InitializeTrees;
  ShowCommandTree;
  LOOP
    GetCommand(cmd, IRCmd);
    UpdateCommandTree;
    CASE IRCmd OF
         Browse : ExecuteBrowseCmd;
            |
         Search : ExecuteSearchCmd;
            |
         .......
    END; (* case *)
  END; (* loop *)
END CalibanInterpreter.
```

La procédure InitializeCommands sert à définir les commandes possibles dans le système. En effet, les commandes et leurs structures sont conservées dans un fichier de TFS, ce qui permet de garder une certaine indépendance des commandes face à l'interpréteur. En effet, le nom des commandes qui apparaît dans la fenêtre *Command* est conservé dans un fichier sur disque; on peut ainsi modifier ce nom interactivement sans avoir à modifier le nom correspondant dans le programme. Les noms de commandes qui forment le type IRCommands ne sont utilisés que de manière interne. Caliban donne la possibilité de définir les noms de commandes que l'on désire avoir (par exemple pour une version allemande ou française) sans devoir recompiler tout le système. Il en va de même pour tous les messages que donne le système et qui se trouvent dans un fichier de texte modifiable avec un éditeur de texte normal. Tous les messages apparaissant dans la fenêtre *Message* (y compris la signification des boutons de la souris) sont répertoriés dans ce fichier. Lorsque le système doit afficher un texte, il accède à ce fichier pour présenter à l'utilisateur le texte qu'il a trouvé à la référence correspondante.

La deuxième initialisation est celle de tous les arbres qui apparaissent sur l'écran. La représentation de l'arbre de commande et des différentes structures d'accès est faite à

ce moment-là; ce qui permet d'afficher un arbre (par exemple ShowCommandTree) très rapidement et sans délai pour l'utilisateur. La représentation des cinq structures d'accès est donc déjà prête au moment où l'utilisateur en demande une. Cette préparation consiste à initialiser la structure interne de la représentation graphique et à lire sur disque le nom des 11 noeuds qui peuvent apparaître sur l'écran (le noeud courant, le père, deux frères et sept fils au maximum).

Vient enfin la boucle principale qui demande à l'utilisateur d'entrer une commande (soit en tapant sur le clavier, soit en pointant sur un objet de l'écran) puis qui exécute le code correspondant. Il faut distinguer deux types de commande. La variable cmd représente l'action de l'utilisateur, elle contient le type de commande qui a été donnée de manière physique (caractère tapé au clavier ou position sur l'écran). La variable IRCmd donne la correspondance de cette opération avec la commande de recherche d'information. Il faut en effet établir la signification logique de l'objet désigné par l'utilisateur. Par exemple, si l'utilisateur désigne le premier noeud à gauche rattaché au niveau de commande le plus haut, alors la commande exécutée sera Browse. Cela signifie que la commande n'est pas fonction du nom qui se trouve inscrit dans le noeud présenté à la fenêtre *Command* mais de sa position. Il n'est donc pas possible de modifier la structure de l'arbre sans réajuster l'interpréteur de commande.

8.3 MultiTreeDisplay, La représentation des hiérarchies

La définition des arbres représentés sur l'écran est une partie relativement complexe. Chaque hiérarchie ou multihiérarchie est définie au niveau du module MultiTreeDisplay. Une liste de tous les arbres est conservée pour pouvoir déterminer quels objets sont actuellement sur l'écran et sur quel noeud l'utilisateur vient de pointer. De plus, pour pouvoir immédiatement inverser les objets qui sont sur l'écran, le contrôle du curseur (en particulier de son déplacement) doit se faire dans le même module. C'est pourquoi MultiTreeDisplay permet d'une part de définir des hiérarchies et des multihiérarchies mais aussi de recevoir les commandes entrées par l'utilisateur. Voyons tout d'abord la définition des hiérarchies.

Un arbre est un enregistrement tel qu'on peut le voir à la figure 8.3. Le premier champ sert à relier tous les arbres entre eux. Chaque arbre est inséré dans la liste au moment de sa définition. Le pointeur "Forest" sert d'en-tête de liste pour pouvoir accéder aux différents arbres. Il faut signaler que la définition d'un tel enregistrement est caché au module qui a défini l'arbre (Hidden Type). Il n'a en fait pas besoin de connaître ces différents champs; la manipulation et la mise à jour se font via les procédures exportées par MultiTreeDisplay. Modula-2 permet en effet ce genre de définition de types abstraits tels qu'ils ont été définis par B. Liskov [Lisk74]. Lors de la définition de l'arbre, MultiTreeDisplay retourne une variable de type "VisibleTree" qui sert au module appelant pour référencer l'arbre qu'il désire manipuler.

Les champs suivants (de TreeWindow à TreeSons) indiquent les caractéristiques propres à la représentation de l'arbre (dans quelle fenêtre il se trouve, quel jeu de caractères on emploie pour écrire le nom des noeuds, quelle est sa position relative dans la fenêtre , etc). Ces champs sont remplis lors de la définition de l'arbre et sont statiques pendant la durée de vie de l'arbre.

NameProc et NSonProc sont deux champs de types PROCEDURE. Le module appelant peut définir lui-même les procédures qui vont servir à lire le nom des noeuds et à donner le nombre de fils actuellement accrochés au noeud courant. Cela signifie entre autre que MultiTreeDisplay est absolument indépendant d'un système de gestion de fichiers quelconques (en particulier de TFS). Seul le format de ces deux procédures est défini dans MultiTreeDisplay (le nombre et le type des paramètres) et toute procédure ayant les caractéristiques de cette définition peut servir à donner le nombre de fils ou le nom des noeuds. La définition de ces procédures est la suivante:

```
NodeNameProc = PROCEDURE(  (* Tree : *)       VisibleTree,
                           (* Node : *)       NodeRelativePosition,
                           VAR (* SonName : *)String,
                           VAR (* Exists : *)BOOLEAN);

NumSonsProc = PROCEDURE(   (* Tree : *)       VisibleTree,
                           VAR (* NumSons : *)CARDINAL);
```

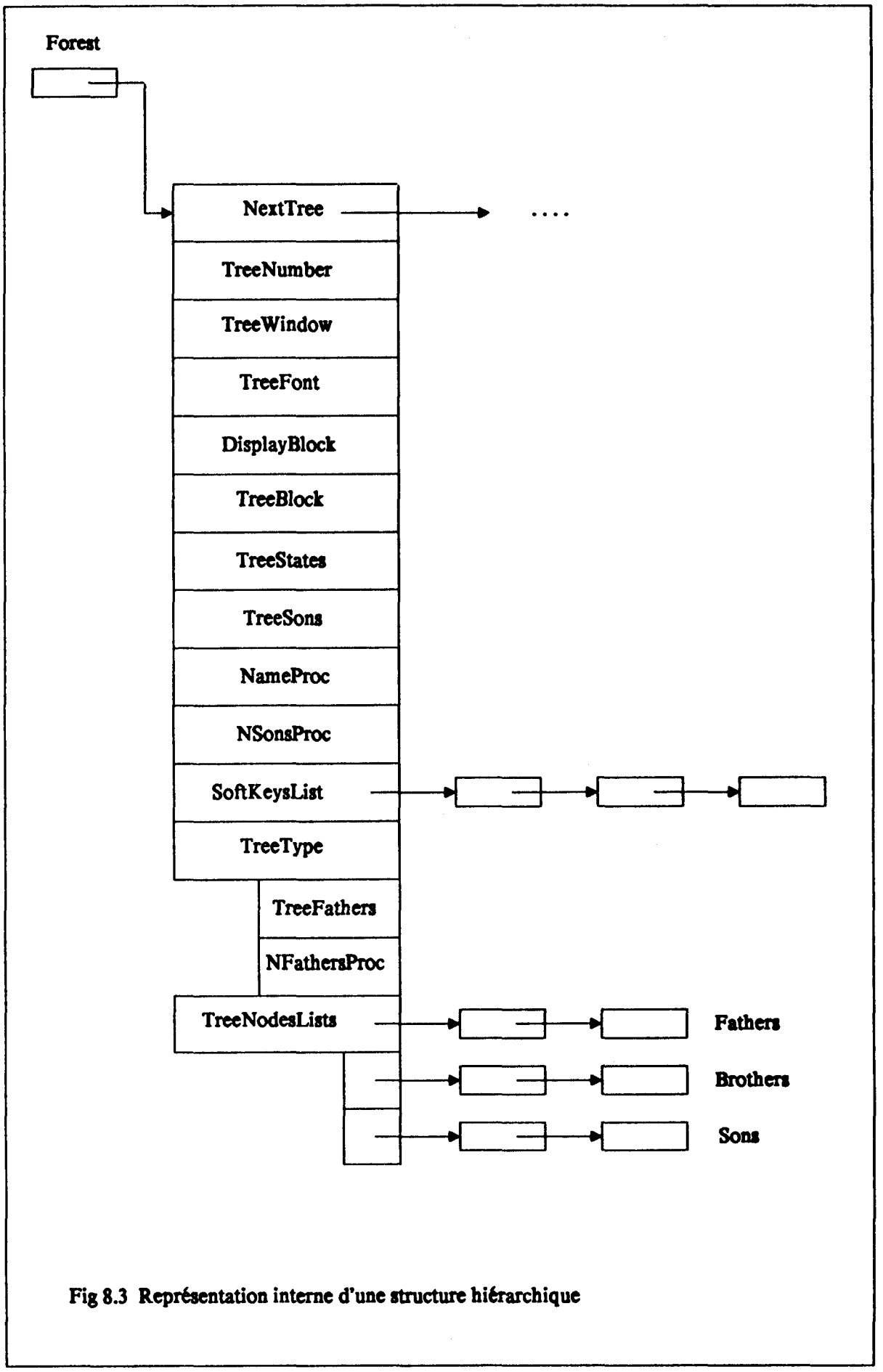


Fig 8.3 Représentation interne d'une structure hiérarchique

La première sert à définir le nom des noeuds qui vont être affichés sur l'écran. Lors de la visualisation des structures dans la fenêtre, MultiTreeDisplay appelle cette procédure pour chaque noeud qui apparaît sur l'écran. Il donne pour cela l'arbre dans lequel se trouve le noeud et la position relative de ce noeud par rapport au noeud au centre de la fenêtre. La procédure, donnée par le module appelant lors de la définition de l'arbre, est alors exécutée et elle doit indiquer si ce noeud existe (Exists) et, si oui, retourner une chaîne de caractères (String) qui indique le nom du noeud en question.

La deuxième procédure sert à donner le nombre de noeuds rattachés au noeud courant. Cette procédure est appelée pour savoir si il est possible de présenter tous les fils du noeud courant (au maximum 7 dans notre cas) ou si certains doivent être gardés en réserve.

Le champ SoftkeysList décrit tous les boutons qui font partie de la représentation de l'arbre (par exemple les flèches de déplacement). Ces boutons sont spécifiés indépendamment de la définition de l'arbre mais sont rattachés à l'enregistrement qui définit l'arbre. On peut donc enlever ou rajouter des boutons lors de l'utilisation de l'arbre (procédures MakeButton et RemoveButton).

Vient ensuite la définition du type de l'arbre, à savoir une hiérarchie simple ou une multihiérarchie. Dans le second cas, il est en effet nécessaire d'avoir à nouveau une procédure variable (NFathersProc) qui donne le nombre de pères du noeud courant; La limite maximum de pères visibles est définie par TreeFathers.

Enfin trois listes linéaires décrivent la position sur l'écran des noeuds des trois niveaux de l'arbre représenté. Il est à noter que le nombre trois est défini dans MultiTreeDisplay comme une constante qui pourrait être modifiée pour présenter plus de niveaux de l'arbre.

Pour permettre une interaction valable entre l'utilisateur et la représentation de l'arbre, il faut que, lorsque l'on déplace le curseur sur un des noeuds ou un des boutons de la représentation, le système réagisse immédiatement en montrant qu'il a reconnu l'objet (en l'inversant par exemple). Cela signifie que le contrôle du curseur doit être effectué par le module qui possède la définition de l'arbre. La relation entre les mouvements de la souris et le curseur sur l'écran est donc faite également par une procédure de ce module. Il s'agit de la procédure GetCommand qui a la définition suivante :

```
PROCEDURE GetCommand (      VAR Cmd  :   Command;
                            Wait     :   BOOLEAN);
```

Les modules de haut niveau (comme CalibanInterpreter) font appel à elle lorsqu'ils désirent que l'utilisateur final désigne un noeud de l'arbre ou entre une commande au clavier. Le paramètre "Wait" sert à indiquer si la procédure doit attendre que l'utilisateur final ait désigné un objet judicieux actuellement sur l'écran (en pointant sur un noeud de l'arbre et en pressant sur un bouton de la souris) ou si elle retourne

au module appelant dès que la souris a bougé en donnant les coordonnées actuelles de celle-ci. Comme résultat, "GetCommand" fournit les indications relatives à la position et à l'état de la souris actuellement. Le type "Command" est défini de la manière suivante :

```
TYPE Command = RECORD
                  CASE Type : CmdType OF
                    Move :
                       MTree : VisibleTree;
                       MCmd  : NodeRelativePosition;
                     |
                    Button :
                       BTree : VisibleTree;
                       BCmd  : CARDINAL;
                     |
                    Keyboard :
                       KChar : CHAR;
                     |
                    UserDefined :
                       Ux,
                       Uy    : CARDINAL;
                  END; (* case *)
                  MouseBut : CARDINAL;
               END;
```

La commande peut être de quatre types (Move, Button, Keyboard, UserDefined). Chaque type précise l'action qu'a effectué l'utilisateur final. "Move" signifie qu'il a désigné un noeud de l'écran, "Button" qu'il a utilisé une des "Softkey", "Keyboard" qu'il a tapé un caractère sur le clavier et "UserDefined" dans tous les autres cas, c'est-à-dire qu'il n'a pas touché un objet précis sur l'écran mais qu'il a pressé un des boutons de la souris. Le dernier champ "MouseBut" indique quel est le bouton qui a été pressé.

8.4 Définition et opérations sur les documents

Les documents sont, avec les structures d'accès, les objets principaux de Caliban. Aussi bien la définition que la manipulation et la représentation sont des aspects primordiaux dans le système. Pour conserver une structure simple de définition, nous avons choisi une liste linéaire pour implémenter les documents. Chaque document est une liste circulaire avec, comme élément de la liste, différents champs lui appartenant. Un enregistrement de cette liste est défini comme suit :

```
TYPE Parameter = RECORD
                   NextParam     :   POINTER TO Parameter;
                   posx,posy     :   CARDINAL;
                   Filled        :   BOOLEAN;
                   Displayed     :   BOOLEAN;
                   TypeParam     :   Items;
                   Weight        :   INTEGER;
                   Field         :   String;
                   Key           :   String;
                   ItemList      :   InfoItemList;
                 END;
```

Chaque enregistrement contient une ligne d'un document (pour des données bibliographiques). Certains champs servent à la représentation sur l'écran (posx, posy, filled, displayed). "TypeParam" donne le type d'objet que l'enregistrement contient. Ce type est également un type caché (hidden type) pour garder une certaine indépendance du système par rapport aux données. On peut ainsi rajouter de nouveaux types sans avoir à modifier tout le système. Ces types sont définis, dans nos données de test, par une énumération qui contient entre autre Auteur, Editeur, Titre, Date de parution, Classification, Thesaurus, etc. Chaque enregistrement peut aussi bien décrire une ligne d'un document que le contenu d'un noeud d'une structure d'accès. C'est ce qui permet d'intégrer directement dans un document virtuel des termes trouvés dans une structure d'accès.

Le champ "Field" détient la chaîne de caractère (String) qui forme le terme en question. Il se peut que ce champ soit doublé par une clé d'accès (Key). La classification décimale est un exemple de ce fait. Chaque classe est définie par un numéro et un nom de classe. Pour le stockage de l'information, il est plus pratique d'utiliser le numéro alors que pour l'utilisateur, ce numéro n'a aucune signification et qu'il doit voir le nom de la classe pour se faire une idée des objets qu'elle contient. Un autre exemple est la description du nom des périodiques. Ces noms ont un système d'abréviation international (CODEN). Chaque titre de périodique possède un code, il est donc judicieux de conserver ce code pour le stockage mais à nouveau l'utilisateur a besoin du titre complet pour retrouver le périodique que le système de recherche lui a fourni.

Le champ "Weight" fait partie de la description d'un document et donne le poids défini par l'utilisateur dans sa requête pour le champ concerné. Cette valeur est de

type entière (négative ou positive) puisque certains poids peuvent avoir une valeur de négation (rejet); une correspondance directe est établie entre la couleur que l'utilisateur a donnée à ce terme et la valeur du champ "Weight".

Enfin "ItemList" donne la liste des documents qui peuvent être rattachés à l'enregistrement en question. Si cet enregistrement contient un terme du thesaurus ou une classe de la classification, "ItemList" désigne tous les documents rattachés à ce champ. Lorsque l'utilisateur choisit un terme dans une structure d'accès, le système remplit simultanément le champ "Field" et le champ "ItemList" pour permettre une évaluation plus rapide lors de la soumission d'une requête au processus de recherche.

Les documents, formés d'enregistrement de type "Parameter", sont des listes linéaires circulaires. Un module de traitement de telles listes a été développé pour permettre leur manipulation. Ce module définit des procédures qui permettent d'insérer ou d'éliminer un ou plusieurs éléments dans la liste, de détruire ou de copier une liste complète et de traverser partiellement ou complètement une liste en effectuant une opération sur chaque élément de la liste. Cette opération est donnée comme procédure en paramètre de la procédure de traversement de la liste. Ceci permet par exemple d'appeler la procédure "Traverse" en lui demandant d'initialiser un champ quelconque de tous ses éléments et d'utiliser cette même procédure "Traverse" pour afficher les champs "Field" de toute la liste dans une fenêtre quelconque.

Lorsque l'utilisateur prépare une requête et la soumet au système de recherche, il définit en fait une telle liste linéaire. Cette liste sert donc d'entrée au module de recherche qui est indépendant de l'interface utilisateur. En effet, Caliban peut supporter plusieurs d'algorithmes de recherche différents pour autant que le module qui définit ces algorithmes se plie aux conventions adoptées entre l'interface utilisateur et le module de recherche. Ces conventions précisent la forme de la question (un document virtuel), la forme des résultats (un ensemble de document du type "InfoItemList") et quelques procédures qui permettent une interaction entre l'interface utilisateur et le module de recherche. Parmi ces procédures on trouve par exemple :

```
PROCEDURE Search (Query        :   POINTER TO Parameter;
                  VAR ItemList :   InfoItemList;
                  VAR done     :   BOOLEAN);
```

Query est un pointeur sur la liste linéaire qui contient la requête de l'utilisateur. La procédure Search retourne une liste de documents (ItemList) au cas où l'évaluation a été possible (done).

Les conventions donnent également des procédures qui permettent d'obtenir, à partir d'un ensemble de documents, les éléments qui la composent, étant donné que la définition de "InfoItemList" n'est pas connue de l'interface utilisateur.

8.5 HString, les chaînes de caractères

Comme nous l'avons vu, Caliban utilise beaucoup de chaînes de caractères. Que ce soit pour l'affichage des noms des noeuds des structures d'accès ou pour la visualisation des documents ou encore pour les messages du système, chaque chaîne de caractères doit, à un moment ou à un autre, résider en mémoire centrale de l'ordinateur. Pour éviter de conserver en mémoire ces caractères pendant toute la durée de la session, un module a été mis au point pour allouer de manière dynamique la place nécessaire en mémoire.

Une première version de Caliban définissait des chaînes de caractères de manière dynamique dans la partie basse de la mémoire. Chaque fois qu'une chaîne était définie, la place nécessaire était réservée dans la partie d'allocation dynamique du système (Heap). Comme de nombreuses chaînes étaient utilisées simultanément dans Caliban, la place occupée dans la mémoire accessible directement était très grande, produisant parfois un débordement de mémoire et empêchant l'utilisateur de travailler normalement.

Une deuxième version du module HMString, qui définit les chaînes de caractères, a été développée. Elle utilise la partie haute de la mémoire de Lilith, qui n'est pas accessible directement, pour stocker les caractères. Seul un pointeur, donnant l'adresse de la chaîne, est gardé dans la partie basse de la mémoire. Etant donné que l'espace disponible en mémoire haute est relativement grand et que le module du système, qui sert à allouer des blocs en mémoire haute, est particulièrement lent, il a fallu écrire un module qui améliore la rapidité d'allocation et de manipulation des chaînes en mémoire haute.

Des blocs de longueur fixe sont alloués en mémoire haute pour stocker les caractères. Un pointeur indique l'endroit où les nouvelles chaînes doivent être écrites à l'intérieur du bloc. Dès que ce pointeur atteint la fin du bloc, un nouveau bloc est alloué en mémoire haute. Les chaînes sont donc toujours écrites à la suite de l'endroit occupé précédemment. Lorsque une chaîne n'est plus utilisée, et donc sa place en mémoire de nouveau libre, un compteur, donnant le nombre de caractères actuellement dans le bloc, est décrémenté de la place rendue libre. Si le pourcentage d'occupation d'un bloc arrive en-dessous d'une certaine limite, les chaînes restantes sont écrites dans le bloc qui est en train d'être rempli et la place de ce bloc est à nouveau rendue disponible.

Un tel algorithme emploie certainement beaucoup de place en mémoire haute mais permet de conserver un processus d'allocation et de désallocation très rapide (environ 10 fois plus rapide que si l'on alloue un bloc par chaîne !).

Dans la version actuelle du module, la grandeur du bloc est de 508 mots de 16 bits et la limite inférieure d'occupation est de 5%. Ces deux valeurs peuvent être modifiées pour une autre application car elles ont été définies et testées spécialement pour le cas de Caliban.

HMString définit un type caché (String) qui est en réalité un pointeur sur un enregistrement qui contient l'adresse de la chaîne en mémoire haute ainsi que sa longueur. Toutes les opérations sur les chaînes (définition, copie, concaténation, visualisation, désallocation ou modification au niveau des caractères) sont effectuées via des procédures mises à disposition par le module HMString.

Un gain de place très net a été rendu possible par ce système d'allocation en mémoire haute sans pour autant perdre un temps de chargement précieux qui aurait gêné l'utilisateur.

9. Conclusion

J. Opel, chairman d'IBM, a écrit récemment un article où il parle, entre autre, des besoins en information [Opel83]. " Tandis que les gens semblent avoir des besoins limités pour des articles courants tels que les souliers ou les voitures, ils semblent avoir un appétit insatiable pour l'information. Je n'ai encore personne entendu dire qu'il ne pourrait employer plus d'information. Cela signifie que la demande en traitement de l'information n'est peut-être pas infinie, mais elle est énorme."

Ceci contredit quelque peu la loi de Mooers [Mooe60], donnée au début de ce travail, où il prétend que certaines personnes peuvent parfois arriver à saturation d'information. Il faut toutefois remarquer qu'entre ces deux citations s'étend une période de plus de 20 ans et que les besoins et les habitudes des gens ont certainement évolués. L'ordinateur est en train de dépasser les prévisions de Bush et remplir les conditions que Heumann proposait lorsqu'il disait [Heum61] : "Ce que je veux, c'est une petite machine personnelle et mécanique avec laquelle je puisse stocker et rechercher mes fichiers personnels et qui coûte moins de mille dollars". Ceci n'est pas encore tout à fait vrai et la machine sur laquelle tourne le système Caliban, bien que pouvant satisfaire les besoins de stockage et de recherche d'information d'Heumann, n'est pas dans cette gamme de prix.

Ce travail montre qu'il est possible de définir des méthodes et un système de recherche d'information qui s'adaptent à la technologie actuelle et qui évitent les pièges cités par Nievergelt [Niev82]. De plus, il précise la voie véritable que doivent suivre les ordinateurs et les programmes qui sont développés. Cette voie est très bien résumée par Rouse [Rous82] : "Du point de vue de la conception de l'interaction entre l'homme et la machine, l'élément-clé est de définir un software tel qu'il soit le complément des capacités de l'être humain tout en l'aidant à surpasser ses limitations".

Nous avons comparé les principes de Caliban à ceux proposés par Gebhardt et Stellmacher qui ont été présentés au chapitre 3. Chaque principe a été confronté à leurs exigences pour permettre de voir quels aspects de Caliban couvraient beaucoup d'exigences de Gebhardt et Stellmacher et inversément quelles exigences étaient particulièrement bien implantées. Il ressort de cette analyse que les facteurs prépondérants de Caliban sont :

- La représentation hiérarchique des commandes
- La représentation hiérarchiques, multihiérarchiques ou en réseau des structures d'accès
- La similarité de forme entre les requêtes et les objets
- Le principe de navigation dans les structures et parmi les documents

De plus, les exigences suivantes sont particulièrement bien remplies :

- Commandes simples
- Emploi des structures
- Langage de sortie
- Evidence du système et réutilisation

Aucun fait, dans les concepts de Caliban, ne contredit fondamentalement les exigences mentionnées.

Voyons enfin quelles sont les possibilités de développement et d'application de Caliban. Elles se situent dans deux domaines.

Tout d'abord, Caliban peut servir tel quel de système de recherche dans une petite bibliothèque ou dans le cadre d'un bureau. Il s'agirait, dans ce cas, d'un système totalement indépendant qui permettrait de retrouver l'information stockée dans la machine elle-même. Un développement de cet aspect devrait intégrer un système de stockage et d'indexation de documents basé sur les mêmes principes que ceux de Caliban. L'indexation des documents correspond exactement au processus de survol où l'utilisateur recherche des thèmes que couvre le document à indéxer. D'autre part, le stockage d'un document ou d'une référence à ce document s'apparente à la formulation d'un document virtuel suivi de son sauvetage sur une mémoire auxiliaire. Les principes de base peuvent donc être les mêmes pour procéder à ce développement.

Le second domaine serait de développer Caliban comme "Front-End" d'un plus gros système existant sur le marché. Il est possible, et nous l'avons montré avec les données d'INSPEC, de conserver, de manière locale, les structures d'accès ainsi que quelques documents représentatifs des termes contenus dans les structures. Ceci permettrait une formulation de question locale, à l'aide des structures et des documents à disposition, qui serait ensuite soumise au système de recherche en télétraitement. L'indépendance de l'interface utilisateur face au module de recherche permettrait d'effectuer cette conversion. L'évaluation des résultats ainsi que le processus itératif d'amélioration des requêtes pourraient se faire de manière locale avec Caliban.

Et Caliban dit :

" ... Souvenez-vous
D'abord de vous emparer de ses livres; car sans eux
Ce n'est qu'un sot comme je suis, et qui n'a pas
Un seul esprit à ses ordres .... "

La Tempête, Acte III, Scène II.
Shakespeare

Glossaire

Classification
Subdivision en différentes classes qui permet d'organiser et de structurer certaines données, en particulier des données bibliographiques. La classification d'INSPEC, dont les données couvrent l'Electronique, la Physique et le Contrôle, comprend environ 2600 classes.

Document
Objet réel qui est à la base des données traitées par Caliban (livres, lettres, mémos, proceedings, etc). Par abus de langage, mentionne parfois une référence à un document.

Document virtuel
Désigne une requête préparée par l'utilisateur. Le mot virtuel vient du fait qu'il ne s'agit pas d'un document existant dans les données mais que cette requête, ayant la forme d'un document, reflète les désirs de la personne effectuant une recherche.

Fenêtre
Subdivision de l'écran d'un terminal qui permet de séparer les données ayant différentes significations. Caliban possède trois fenêtres fixes *Command*, *Message* et *Information*.

Hardware
Ensemble des éléments physiques employés pour le traitement des données (en français : Matériel).

Interface
Jonction entre deux matériels ou logiciels leur permettant d'échanger des informations par l'adoption de règles communes physiques ou logiques.

Interface utilisateur
Jonction entre un utilisateur et un système de traitement de l'information qui permet à l'utilisateur d'entrer des données et au système de fournir des résultats suivant un schéma défini.

Module appelant
Unité contenant un programme ou une procédure qui appelle une procédure définie dans un autre module.

Navigation
Méthode de déplacement dans une structure. Ce déplacement se fait à partir d'une position courante dans l'environnement logique défini par la structure. Pour une liste linéaire, par exemple, cet environnement comprend les éléments suivant et précédent la position courante. Pour une structure hiérarchique, il s'agit du père et des fils du noeud courant.

Noeud courant
Position actuelle dans une structure. Dans Caliban, cette position est toujours située au milieu de la fenêtre où se trouve la structure en question.

Objet
Elément abstrait manipulé par un système informatique. Dans le cas de

Caliban, il peut s'agir aussi bien d'un document, d'une référence que d'un mot-clé tel qu'un terme du thesaurus, d'une classe de la classification ou du nom d'un auteur.

Référence

Elément construit manuellement à partir d'un document. Dans le cas d'un document bibliographique, une référence donne, par exemple, le titre, l'auteur, un ensemble de mots-clés, la date de parution ou un résumé tiré de l'ouvrage original.

Software

Ensemble des programmes, procédés et règles et éventuellement de la documentation, relatifs au fonctionnement d'un ensemble de traitement de l'information.

Structure d'accès

Ensemble d'éléments ordonné selon certains critères. Chaque élément donne accès à des objets (par exemple des documents) et possède lui-même un rapport avec d'autres éléments dans la structure. Ces structures peuvent avoir la forme d'une liste, d'une hiérarchie, d'une multihiérarchie ou d'un réseau. La classification et le thesaurus sont deux exemples de structures d'accès dans Caliban.

Terme

Mot général qui désigne aussi bien une classe de la classification ou un élément du thesaurus qu'une des entités faisant partie d'une référence à un document.

Thesaurus

Ensemble de termes qui sont liés entre eux et qui forment un vocabulaire couvrant un ou plusieurs domaines (p.ex. Physique ou Electronique). Chaque terme du thesaurus est formé d'un ou plusieurs mots qui définissent un concept précis. De plus, des connections entre les termes définissent une structure ordonnée. Dans le cas du thesaurus défini par INSPEC, ces connections s'appellent "Broader Term", "Narrower Term", Top Term", "Related Term" et définissent une structure de réseau. D'autres connections, externes à la structure, peuvent former un ensemble encore plus complexe (dans INSPEC, à chaque terme du thesaurus correspondent une ou plusieurs classes de la classification).

Bibliographie

[Bärt78] Bärtschi M. **Befehlssprachen bibliographischer Dialog - Informations - Systeme.** ETH Zürich, interner Bericht, 1978.

[Bärt82] Bärtschi M., Frei H.P. **A Data Organization for Information Retrieval on a Personal Computer.** Personal Computing, Teubner Stuttgart, 1982.

[Bärt84] Bärtschi M. **Term Dependence in Information Retrieval Models.** ETH-Dissertation, à paraître, Zürich 1984.

[Bell80] Bell C. L., Jones K.P. **The Development of a Highly Interactive Searching Technique for MORPHS.** Information Processing and Management, Vol 16, 1980.

[Berg58] Berge C. **La théorie des graphes et ses applications.** Dunod 1958.

[Blic83] Blick A.R. **Information Science Research versus the Practionner.** IRFIS 5, Heidelberg, 1983.

[Bour79] Bourne C., Anderson B. **DIALOG Lab Workbook.** 2nd edition, Lookheed Information Systems, 1979.

[Bush45] Bush V. **As we may think.** The Atlantic Monthly 176, 1945.

[Crof82] Croft W. B. **An overview of Informations Systems.** Information Technology: Research and Development, Vol 1 No 1, January 1982.

[Crof83] Croft W. B. **Applications for Information Retrieval Techniques in the Office.** ACM SIGIR, 1983.

[Date77] Date C.J. **An Introduction to Database Systems.** 2nd edition, Addison-Wesley, 1977.

[Dial82] DIALOG Information Services Inc., 1982.

[Delo82] Delobel C., Adiba M. **Bases de Données et Systèmes Relationnels.** Dunod, 1982.

[Fisc82] Fischer H.G., Zeidler A. **User Friendly Interaction with an Integrated Data Base Information Retrieval System.** Proceedings International Zurich Seminar on Digital Communications, 1982.

[Frei82] Frei H.-P. **Dokumentationsysteme.** ETH Zürich, Vorlesung Sommersemester 1982.

[Frei83] Frei H.-P., Jauslin J.-F. **Graphical Presentation of Informantion and Services: A User-Oriented Interface.** Information Technology: Research and Development, Vol 2, 1983.

[Gebh78] Gebhardt F., Stellmacher I. **Design Criteria for Documentation Retrieval Languages.** JASIS, Vol 29 No 4, 1978.

[Geis83] Geissmann L. **Separate Compilation in Modula-2 and the Structure of the Modula-2 Compiler on the Personal Computer Lilith.** ETH-Dissertation No 7286, Zürich, 1983.

[Grie81] Gries D. **The Science of Programming.** Springer-Verlag, 1981.

[Heap78] Heaps H.S. **Information Retrieval, Computational and Theoretical Aspects.** Academic Press, 1978.

[Hero80] Herot C.F. **Spatial Management of Data.** ACM Transactions on Database Systems, Vol 5, Dec. 1980.

[Heum61] Heumann K.F. **New Device Needed.** American Documentation, 12, 157, 1961.

[Hopc69] Hopcroft J.E., Ullman J.D. **Formal Languages and their Relation to Automata.** Addison-Wesley Publishing Company, 1969.

[Ingw83] Ingwersen P. **Online Man-Machine Interaction Facilities : a Cognitive View.** IRFIS 5, Heidelberg, 1983.

[Knu83] Knudsen S.E. **Medos-2: A Modula-2 Oriented Operating System for the Personal Computer Lilith.** ETH-Dissertation No 7346, Zürich, 1983.

[Lanc73] Lancaster F.W., Fayen E.G. **Information Retrieval On-Line.** Melville Publishing Co, Los Angeles, 1973.

[Lexi76] Mead Data Control. **LEXIS Quick Reference.** New York, 1976.

[Lili82] Lilith Handbook. **A Guide for Lilith Users and Programmers.** Institut für Informatik, ETH-Zürich, 1982.

[Lisk74] Liskov B.H., Zilles S. **Programming with Abstract Data Types.** SIGPLAN, Vol 9 No 4, April 1974.

[Medl79] National Library of Medicine. **MEDLARS, the Computerized Literature Retrieval Services of the NLM.** Education and Welfare, 1979.

[Mill68] Miller R.B. **Response Times in Man-Computer Conversational Transactions.** Procceedings Spring Joint Comp. Conf. AFIPS Press, Montvale N.J., 1968.

[Mill77] Miller L.A., Thomas J.C. **Behavioral Issues in the Use of Interactive Systems.** Int. J. Man-Machine Studies 9, 1977.

[Mink77] Minker J. **Information Storage and Retrieval - a Survey and Functional Description.** SIGIR Forum 12, 1977.

[Mooe60] Mooers C.N. **Mooers' Law or why some Retrieval Systems are used and others are not.** American Documentation, Vol 11 No 3, July 1960.

[Niev82] Nievergelt J. **Errors in Dialog Design and How to Avoid Them.** Proc. International Zurich Seminar on Digital Communications, 1982.

[Nils80] Nilsson N.J. **Principles of Artificial Intelligence.** Palo Alto, California : Tioga, 1980.

[Oddy77] Oddy R.N. **Information Retrieval through Man-Machine Dialogue.** J. of Documentation, Vol 33 No 1, 1977.

[Opel83] Opel J. **The Colossus that Works.** TIME, 11 July 1983.

[Orbi75] SDC Search Service. **ORBIT User Manual.** Santa Monica, 1975.

[Rand77] Randall D., Buchanan B., Shortliffe E. **Productions Rules as a Representation for a Knowledge-Based Consultation Program.** Artificial Intelligence 8, 1977.

[Raph76] Raphael B. **The Thinking Computer: Mind inside Matter.** W.H. Freeman and Co, San Francisco, California, 1976.

[Rijs79] van Rijsbergen C.J. **Information Retrieval.** 2nd edition, Butterworths, 1979.

[Rijs83] van Rijsbergen C.J. **Information Retrieval: New Directions: Old Solutions.** ACM SIGIR, 1983.

[Roth69] Rothman J. **The New York Times Information Bank.** New York Times, N.Y., 1969.

[Rous82] Rouse W.B., Rouse S.H., Morehead D.R. **Human Information Seeking : On Line Searching of Bibliographic Citation Networks.** Information Processing and Management, Vol 18 No 3, 1982.

[Salt71] Salton G. **The SMART Retrieval System - Experiments in Automatic Documents Processing.** Prentice Hall Inc, 1971.

[Salt83] Salton G., McGill M.J. **Introduction to Modern Information Retrieval.** McGraw-Hill, 1983.

[Sela82] Selander S.E. **Several Approaches for Improving User Cordiality in the Design of On-Line Systems for Novice Users.** Computer Personnel, Vol 99 No 2, Nov 1982.

[Shne80] Shneiderman B. **Software Psychology, Human Factors in Computer and Information Systems.** Winthrop Publishers Inc, 1980.

[Smit82] Smith D.C. et al. **Designing the Star User Interface.** Byte, Vol 7 No 4, 1982.

[Stai76] **Storage and Information Retrieval System/Virtual Storage, STAIRS/VS.** Reference Manual, Sindelfingen, 1976.

[Stew83] Stewart T. **The Software Interface.** Ergonomic Principles in Office Automation. Ericsson Info. Syst. AB, Sweden, 1983.

[Suga82] Sugaya H. **Tree File : A Data Organization for Interactive Programs.** ETH-Dissertation No 6944, Zürich, 1982.

[Tsic77] Tsichritzis D.C., Lochovsky F.H. **Data Base Management Systems.** Academic Press, New York, 1977.

[Weiz76] Weizenbaum J. **Computer Power and Human Reason.** W.H. Freeman and Co, 1976.

[Will83] Williams G. **The Lisa Computer System.** Byte Vol. 8 No. 2 1983.

[Wins77] Winston B. **Artificial Intelligence.** Addison Wesley Publishing Company, Reading, Mass., 1977.

[Wirt81] Wirth N. **Lilith : A Personal Computer for the Software Engineer.** Proc. of the 5th Conf. on Software Engineering, IEEE Comp. Soc. San Diego, 1981.

[Wirt82] Wirth N. **Programming in Modula-2.** Springer-Verlag, 1982.

[Zloo75] Zloof M. **Query By Example.** Proceedings of the NCC, AFIPS Press, Montvale N.J. 1975.

Curriculum Vitae

Je suis né le 31 juillet 1954 au Locle (Neuchâtel). J'ai suivi tout d'abord l'école primaire dans cette ville puis à Auvernier (Neuchâtel). J'ai ensuite fréquenté l'école secondaire régionale de Neuchâtel pendant quatre ans puis le gymnase cantonal, en section littéraire, pour obtenir, après trois ans et demi, la maturité fédérale type B. En 1973, j'ai entrepris des études de mathématique à l'Université de Neuchâtel où j'ai terminé, en 1978, une licence ès sciences. Dès 1976 déjà, j'ai été engagé comme assistant en informatique à l'Université de Neuchâtel, où je suis resté pendant quatre ans. J'ai pu, grâce à ce poste, approfondir mes connaissances en informatique, en particulier dans le domaine des langages de programmation. Pendant cette période, j'ai suivi des cours de 3ème cycle d'informatique à l'Ecole Polytechnique Fédérale de Lausanne et je me suis intéressé, dans le cadre d'une collaboration avec l'industrie, au domaine du développement assisté par ordinateur (CAD). En octobre 1980, une place d'assistant en informatique à l'Ecole Polytechnique Fédérale de Zurich m'a été proposée pour l'étude d'un projet sur un système de recherche d'information.