Information Technology: Research and Development (1983), *2* (23–42)

GRAPHICAL PRESENTATION OF INFORMATION AND SERVICES: A USER-ORIENTED INTERFACE

H. P. FREI AND J.-F. JAUSLIN

Swiss Federal Institute of Technology (ETH), Zurich, Switzerland

(*Received 17 August 1982, revised 22 October 1982*)

ABSTRACT

A novel user-machine interface for information retrieval (IR) is described which supports the associative way of human thinking and suggests associations of thoughts by displaying and opening query-related paths. Both information and commands are structured in a tree-like fashion; these trees are displayed on a graphics screen in separate windows and can be explored by means of motion commands. The two principal services offered by the system enable browsing through both information structures and information items as well as formulating and processing queries. However, the traditional query formulation is replaced by the concept of a virtual information item. The services of the information retrieval system are both informally described and presented by means of sample dialogues. The system is implemented on a relatively small personal computer and the software is written in a high-level system programming language.

1. INTRODUCTION

As early as 1945—before the existence of modern computers—V. Bush described his visionary device *memex* as a mechanized private tool enabling its users to store, process and retrieve almost infinite amounts of data (Bush, 1945). In his article Bush also makes clear that the human mind operates by *associations*, that it *follows* previously built-up *trails* and that it instantly *swaps to new items* as soon as this is suggested by the associations of thoughts.

Despite the progress made in technology in general and in interactive computer systems in particular, today's systems have not yet lived up to Bush's nearly 40-year-old visions. While many of the technical problems of storing data have been solved in the meantime, most interactive systems are still unable to provide enough flexibility and ease of use to effectively support the associative way of thinking of their users.

This also holds for present-day information retrieval (IR) systems: users are still dealing with the baroque tools of so-called command languages, are still typing rather complicated, lengthy character sequences into a keyboard and—frequently—have to decipher linear character strings where a pictorial output would be more appropriate. Getting useful information after *a simple tap of a key*, or browsing through large amounts of data by simply *deflecting one of the levers to the right* as Bush (1945) envisaged, is still wishful thinking.

Unfortunately, we are not in the position to present a technical realization of the proposed device *memex*. However, the system that was the basis for this study of man-machine communication has a few characteristics of *memex*: the hardware very much resembles Bush's ideas as the system is implemented on a personal computer with a pointing device as the predominant input medium; selected information items and information structures immediately show up on the screen—often in a two-dimensional graphical fashion—and the user is able to jump from one information item to the other at will; information items exhibiting a close association can be tied together by establishing *paths* and may, subsequently, be retrieved and instantly displayed by following such *paths*.

Although conventional commercially available bibliographic data are used to test the functions and performance of our system, the system itself departs in two essential points from many of today's IR systems:

1. *Data organization*: Stored information can structurally be described as trees and networks. A tree file system is used to implement these structures and even in the case of a network, an hierarchical organization is emphasized by means of a tree. This allows an easy visual representation on a graphics screen.
2. *User-machine interface*: The dialogue between the user and the system is conducted via up-to-date hardware; particularly important are the so-called mouse as input device and the graphical refresh screen as output device. However, most notable is the novel organization of the dialogue. Both the structure of the information and the structure of the available command set are graphically displayed—in separate windows—on the screen. In addition, the user may build up hypothetical information items to be used as the input to retrieval algorithms.

This paper discusses the philosophy underlying our user-machine interface. On top of that, the features at the user's disposal and the reactions of the system to requests entered by the user are presented. Putting together such a description is a cumbersome undertaking as it is difficult to convey the flavour of a dialogue with a highly dynamic interactive system by means of a linear sequence of characters.

The IR system *Caliban* is implemented on the personal computer *Lilith* and has been written in the high-level system programming language *Modula-2* (Wirth, 1982). We refrain from presenting the hardware and basic software of *Lilith* in detail as this has been done elsewhere (Wirth, 1981). Instead, we only mention those facts absolutely necessary to understand the user-machine interface. The data organization underlying *Caliban* is described elsewhere (Bärtschi and Frei, 1982). Rather than putting together our own small test collection, we tested *Caliban* with commercially available bibliographic data on Physics, Electronics, and Control. These test data were purchased from INSPEC in machine-readable form.

In section 2 the fundamental ideas underlying *Caliban's* user interface are presented and it is pointed out how the approaches of *Caliban* and conventional IR systems differ. Section 3 explains how the stored information is structured and how these structures are displayed to the user. Section 4 discusses the services available to the user and how these services are represented. Finally, in sections 5 and 6 the two principal services *Browse* and *Search* are informally described as well as presented in the form of sample dialogues.

2. FROM COMMAND LANGUAGES TO USER INTERFACES

Command languages are considered crucial parts of interactive systems as they are the means of accessing the services rendered by the system. Frequently, users even tend to identify a system with its command language and the power of a system is often judged by merely assessing its command language. Examining present-day command languages and the usage of the systems they control reveals that the average user 'is often placed in the position of an absolute master over an awesomly powerful slave, who speaks a strange and painfully awkward tongue', as Miller and Thomas (1976) put it. Let us first examine the main characteristics of a typical command language, then we will list some of the principal arguments that might have caused Miller's and Thomas' provoking quotation, and finally we will present alternative ideas, namely the ones that influenced *our* approach to interfacing the user with a machine.

The typical command language of a conventional interactive system consists of a fairly large set of commands, usually arranged as a linear list. Such a system may therefore be viewed as a one-state machine: after a command has been executed, the system returns to the same state it was in before the command was entered. From the user's point of view this looks as follows: each time the user issues a command, he or she does a *selection* out of the very long—but invisible—list of commands. Note that the list of available commands is hardly ever displayed to the user; in most cases the list would be too long to fit on a screen anyhow.

Commands usually have to be invoked by typing a character string into a keyboard. Especially when the command name is long, this is a cumbersome and error-prone procedure. Only the more recently developed interactive systems provide special keys, switches or so-called function keys to invoke the most frequently occurring commands.

In short: command names, possibly followed by some arguments, have to be typed into a keyboard; the command names and their meanings have to be learned before they can be used and the more commands are memorized and the faster they can be invoked, the better the system is mastered. Unfortunately, such systems only fit the needs of some sort of expert user although they are often used by a wide variety of other persons. Many interactive systems as well as many of today's frequently used interactive IR systems have these or similar characteristics. The following factors contribute to the fact that they are difficult to use for a non-expert:

1. The set of commands is not structured or when it is the structure is not visible.
2. Commands and their meaning have to be memorized; a minimum number of such commands is necessary when a system is to be used in a meaningful way.
3. Many commands are cumbersome to invoke (most users are untalented typists).
4. Many commands have confusing names; in addition, the names are often abbreviated but sometimes according to obscure rules.
5. Syntax errors are possible.

An additional confusion arises with interactive IR systems: in most cases command words and index terms used to describe the stored information are identical in appearance when typed into the keyboard. Both commands and terms have to be expressed by means of character strings. At the present time, most IR activities are still performed employing command languages with the above-mentioned deficiencies (e.g., DIALOG, ORBIT). More recent developments towards both better structuring command sets and using hard or soft keys to invoke commands (e.g., STAIRS) have had few consequences on the practice of IR so far. Part of the problem is that many commercial IR systems still have to cope with the characteristics of the teletype terminal as the sole means of connecting users and computer systems.

At the present time, interactive systems in general and IR systems in particular are almost exclusively operated by systems specialists. In the not too distant future such systems will be used by the end user directly, i.e., by the *person who needs* the information to be searched for. This is why our prime objective was to design an IR system which would be as user friendly as possible, thereby enabling the end user to search without the help of an information specialist. Referring to user friendliness does not only mean that the command language is easy to learn. Rather, our aim was to build a system which is self-explanatory to the extent that there is very little or even no need for a written manual. Written manuals are *inadequate tools* for describing highly flexible interactive computer systems.

This goal of having an IR system tailored to the end user directly is ambitious. First of all, it means making another step in the evolution process from command languages to user interface. For this reason, we studied the *entire system* from the point of view of the end user. The belief was that the *system*, including the stored data, has to be exposed to the user resulting in a *user interface* rather than another new command language.

An important decision when designing *Caliban* was to get rid of the character-by-character view of information retrieval. Rather, the *Caliban* user may deal with information *concepts* disregarding the fact that such concepts are usually described by index terms or keywords in the form of character strings. In addition, the structure of these information concepts is to be disclosed to the user, as is the case with the more innovative of the currently available IR systems.

We concluded that the most important property of a user-oriented system is its *transparency*; after the system's initialization *Caliban itself* offers both the available services and the information to the user. The *services* comprise all the actions a user may perform employing the IR system, i.e., browsing through the stored objects via content-describing indexes, putting together and editing queries, processing queries, building up and combining object sets. The *information* consists of all the objects contained in the system as well as of the content-describing indexes. In contrast to most existing IR systems, *Caliban* clearly displays the *structure of both the services and the information* so that the user has an unconstrained choice from what is offered.

Our belief is that only such a system is able to supplement efficiently the end user's way of searching for information, as in most cases the human search strategy is not very systematic. Bush (1945) has already pointed out that, as the thoughts of the end user are guided by associations, their train may be deviated by newly gained knowledge and may return to the mainstream followed beforehand, thus jumping back and forth. In this way the end user is collecting piece after piece of information until his or her needs are met. However, this is only possible when both the services and the information are presented in a lucidly arranged manner.

To distinguish the different concepts when *Caliban* is used, the screen is partitioned into three distinct windows at all times: *Information* window, *Command* window, and *Message* window. The information window is the largest of the three windows and contains both information structures (indexes) and information items.

Actual results of browsing as well as results of queries are displayed in this window.

The command window contains the structured set of available services. In order to invoke a service of *Caliban* it has to be selected from the services displayed in this window. In addition, the command window continuously exhibits the currently active service when the system is in use. This is a valuable aid to the user, e.g., when getting back to the system after having been interrupted by a phone call or the like.

The message window contains various kinds of system messages. Primarily, these messages are thought to support the user while working with *Caliban*. This is why the message window can be ignored by an experienced user most of the time. For example, it always contains a sketch of the mouse, used as input device, indicating the meaning of the three push buttons—definitely unnecessary information for an experienced user. (As we shall see later, these buttons change their meaning quite often.)

3. DISPLAYING INFORMATION STRUCTURES

Very often humans tend to structure information in an hierarchical manner: broad concepts are defined and partitioned into groups of narrower concepts; these narrower concepts are again partitioned until finally specific information items are defined. A well-known hierarchical partitioning of knowledge is the decimal classification. Genealogical trees or a table of contents in a book are typical examples of this kind of representation. It seems to be natural for human beings to split general concepts into more detailed sub-concepts. Furthermore, hierarchical representations are used widely enough to allow someone, who did not define a particular structure, to easily understand its meaning.

Unfortunately, the nature of information is sometimes more complex than a pure hierarchy. A network is often the appropriate structure. A typical example is a thesaurus where there are a number of connections (e.g., to related terms) which do not fit into the hierarchy. *Caliban* is able to handle both hierarchies and networks.

As many information structures (e.g., classifications, thesauri) are of significant size, the problem arises how to present them to the user of a system. There are two limiting factors. First of all, the amount of information a user is able to perceive is restricted. Secondly, the dimension of a screen is limited and only parts of a sizable structure can be displayed; in addition, if a graphical presentation is chosen, many of the edges of a structure would lead to nodes outside the screen. This applies to both hierarchies and networks. However, edges of an hierarchy crossing the boundaries of a screen invariably lead to a node of the immediate structural neighbourhood: a father or a son. Contrarily, edges of a network crossing the boundary of a screen may lead to an arbitrary—e.g., structurally far distant—node of the network. As we felt that such edges would confuse the user, we decided not to display them on the screen.

We ended up capturing the important relationships of a network as an hierarchy. Only the hierarchy is presented to the user in a graphical manner despite the fact that in many instances there are further connections of information structures. In this way, *graphically* represented trees are displayed to the user instead of networks. However, the user may request additional relationships in a *non-graphical* representation.

Because of the restricted size of the window in which we display information, only three levels of a tree can be graphically displayed. Nodes are represented as rectangles with their names written inside and the father-to-son edges of the structure are explicitly drawn as line segments.

The node in the centre is called the *current node*; it is highlighted to distinguish it from the others. Only the father of the current node, two brothers and at most seven sons are displayed at the same time. Figure 1 shows how this partial view is extracted from the entire tree: all the nodes appearing inside the dark line are displayed. This view of the tree can be changed in two different ways: either a new current node is chosen—which will then appear in the centre of the window—or invisible sons of the actual current node are made visible. Dots appearing above the leftmost or the right-most son of the visible part signify that more nodes are available in that direction.

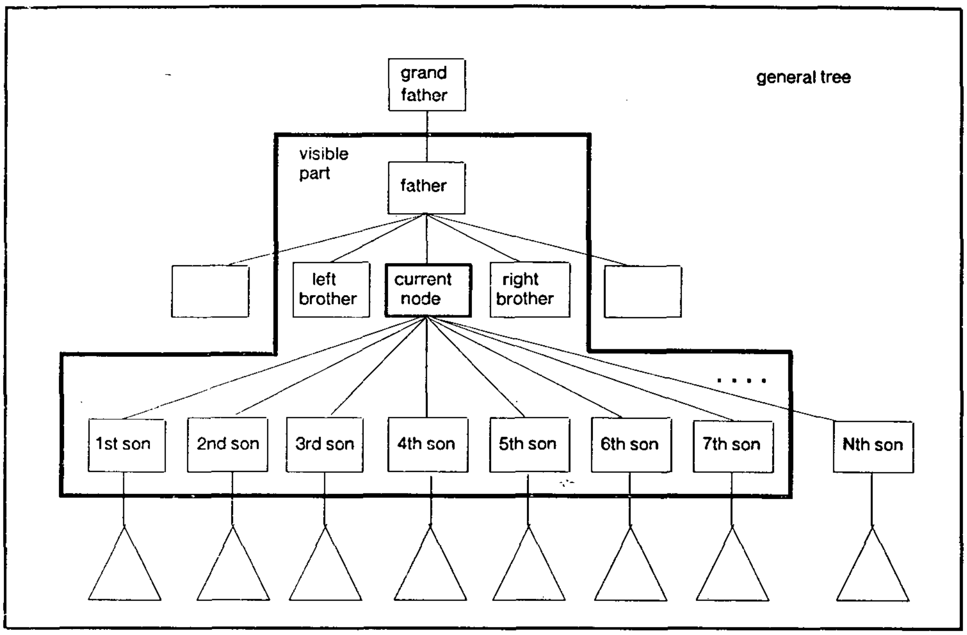


FIG. 1. Visible part of a tree

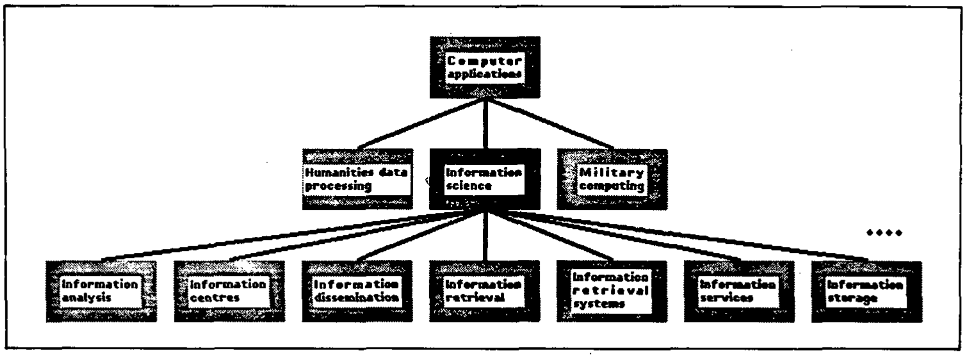
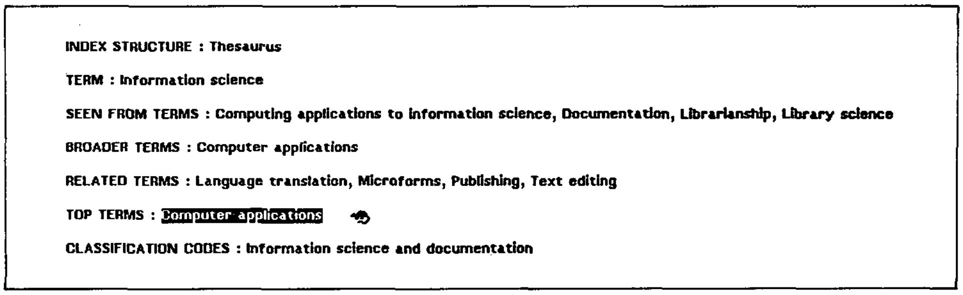


FIG. 2. Part of a thesaurus as it is displayed on the screen*

* We apologize for the quality of this and the following figures (laser beam printer output from the computer screen)

In our test data one of the index structures is a Thesaurus. Part of it is displayed in Figure 2 with the term *Information science* as the current node. A thesaurus is a typical example of a network with an hierarchical main structure and a number of other connections such as connections to BROADER TERMS, RELATED TERMS, TOP TERMS and CLASSIFICATION CODES. The broader terms represent more general concepts in addition to the father term which exists all the time. Related terms are terms with a direct relation to a particular node term other than the nodes directly connected through the tree structure. Top terms are the possible ancestors and classification codes represent connections to the outside of the index structure, namely to a classification tree.

In contrast to the pictorial representation of the hierarchical part of the structure (cf., Fig. 2) these additional connections are represented in a textual manner (Fig. 3), i.e., still in a character-by-character representation.



INDEX STRUCTURE : Thesaurus

TERM : Information science

SEEN FROM TERMS : Computing applications to information science, Documentation, Librarianship, Library science

BROADER TERMS : Computer applications

RELATED TERMS : Language translation, Microforms, Publishing, Text editing

TOP TERMS : Computer applications

CLASSIFICATION CODES : Information science and documentation

FIG. 3. Additional connections are presented textually

4. SERVICES AND THEIR REPRESENTATION

Although many of the services of an interactive system are conceptually divided into several sub-services, very often the sub-service commands are on the same level as the service commands. The casual user encounters the following problems: if a system is powerful, there are a great number of commands available and it takes a long time to memorize a particular command subset for using the system in a simple way; as many of the command sets consist of simple linear command lists, it is difficult to find out in which state the system is, i.e., which commands are meaningful and/or executable. If the services of an interactive system are structured, another problem arises when a command is given: the system generally enters a new mode, i.e., the previously active environment is left. The set of possible commands changes which means that either new command names become available or old ones change their meaning. The difficulties for the user originate mainly from the lack of information provided by the system. It is our belief that, if the services of a system are structured, this structure should appear on the screen.

The *Caliban* services are structured and—as we had already a tree representation for the data—we decided to use the same graphical method to represent our services. This is in contrast to referring to a linear list of commands and has a number of advantages. A service is a part of the system and constitutes an exactly defined activity, a *state of mind* of the user. Therefore, only those sub-services and commands relevant to the particular service have to be active and executable. As soon as another service is entered, other sub-services and commands become relevant and active. In this way, the *Search* process is divided into the process of creating and modifying a query, the process of finding and inspecting items and the process of printing and saving results.

Figure 4 shows the general form of the command tree. The *Search* service, for example, is subdivided into five commands. Each of these commands is directly executable just by pointing to it.

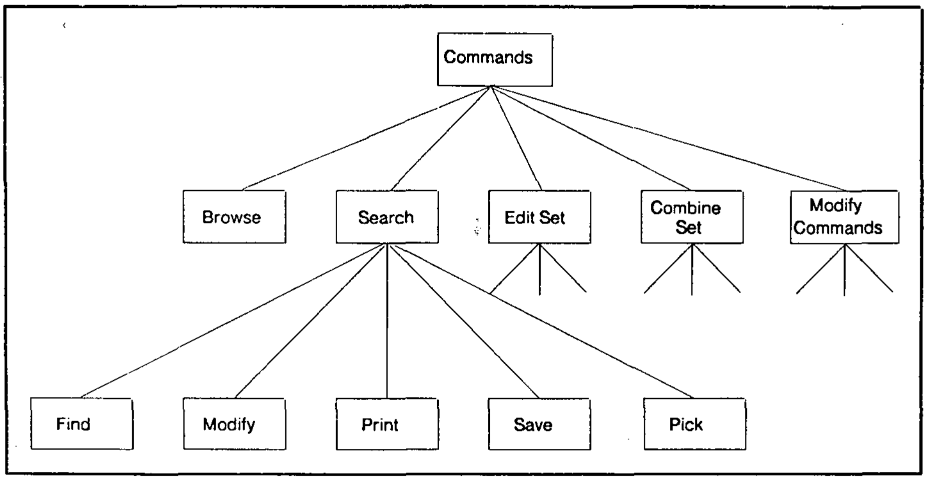


FIG. 4. The command tree

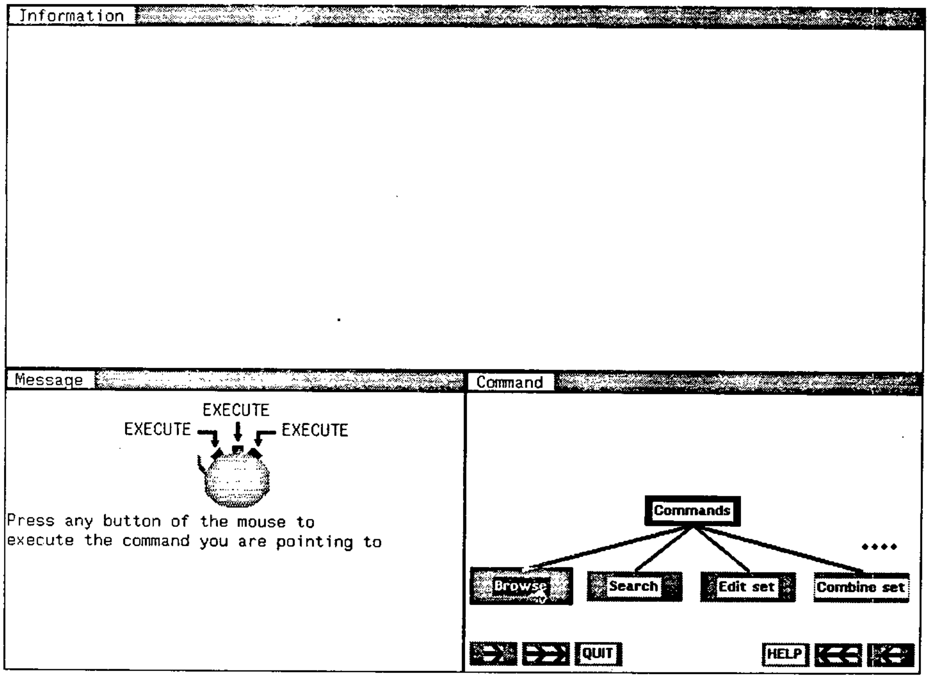
There are two ways of invoking commands in *Caliban*:

1. To point to a specific position on the screen, say a tree node or a so-called soft key. This quick and powerful method is accomplished by first putting the cursor at the desired position and then pressing any of the buttons of the pointing device. In order to eliminate any kind of doubt, the system always illuminates the object when the cursor is placed on it. In this way, the user knows exactly in which state the system is and to where he or she is pointing.
2. By pressing a hard key either on the keyboard or on the pointing device independent of the position of the cursor.

In *Caliban*, pressing hard keys on the keyboard is reserved for commands which have irremediable consequences, say deleting an item. Pressing a mouse button is a very fast way of invoking a command, especially when there is no need to position the cursor to a specific location. This mode of execution is reserved for commands having minor consequences if they are entered by mistake. Most of the time the combination of positioning the cursor and pressing a mouse button is used in *Caliban*. As different meanings can be assigned to the buttons of the mouse when pointing to different locations of the screen, an explanation of the three buttons is always given to the user in the message window, which is a useful aid for the casual user.

At the outset, after the system has been started, all the three buttons of the mouse have the same meaning as shown in Figure 5: they serve to EXECUTE a command.

As already mentioned, the node the cursor is positioned on is always highlighted: in Figure 5 it is the node representing the command *Browse.*



Press any button of the mouse to
execute the command you are pointing to

FIG. 5. Situation after starting *Caliban*

Caliban provides two different ways of getting information:

1. *Browse*, meaning to move about index trees similar to inspecting the books on the shelves of an open library.
2. *Search* to prepare a query and submit it to the system in order to retrieve items. *Edit set* and *Combine set* allow the combination of the retrieved sets of items.

5. BROWSING

One of the main objectives of the system is to support and even suggest associations of thoughts to the user and to supply a means of following the different paths opened. This is particularly important when the user has only a vague idea of what he or she is looking for. In this case the user needs a browsing facility which allows collection of terms that can be used in a query formulation. The graphical representation of the tree structures on the screen facilitates finding ideas for future use. In addition, the user gets an impression of structure when explicitly moving to the father or to one of the sons of the current term or when inspecting further relationships. This kind of browsing through information supports an associative way of thinking by humans.

Browsing also allows inspection of real information items attached to the index terms. This gives a more precise idea of the meaning of a particular term and is similar to what the patron of a small library can do by opening a few books when inspecting the shelves. Furthermore, we are convinced that in many cases a convenient browsing facility serves to find the requested documents directly without formulating a formal query.

5.1 *Test data*

As mentioned in section 1, we employ bibliographic test data purchased from INSPEC. Among the five index structures *Caliban* provides, there is a classification and a thesaurus. Every node symbolizes an index term and there is always a set of documents attached to it. In contrast to the classification, which is a regular hierarchical structure containing about 2600 nodes, the structure of the thesaurus is a network. It was difficult to implement the thesaurus tree with more than 6000 nodes because it has a very large fan-out, namely 600 first level sons. An intermediary level of alphabetically ordered nodes has been inserted to simplify the access to these nodes. In addition to these two index structures purchased from INSPEC, we built three others: free-indexing terms, author/editor and publisher. They simply consist of trees with nodes that are alphabetically ordered on each level. An intermediate level of indexes was introduced where deemed necessary.

5.2 *Sample dialogue*

In order to illustrate the browsing process, we assume a person looking for information about *Computers.* Furthermore, we assume that the person is unable to express this wish exactly, i.e., to formulate a query. For this reason the user enters the service browse (Fig. 6). The system then offers a choice between the five available index structures. Our particular user selects the thesaurus by placing the mouse-shaped cursor on that node and pressing the left button of the mouse, i.e., the button SPECIFY INDEX (cf., message window).

After the invocation of this command the thesaurus tree appears. As there would be too many direct sons, an intermediate level of alphabetic nodes is displayed (Fig. 7). Our user determines that the term *Computers* is located in the subtree of the node ‘Coi TO Control o’. In order to find it, this subtree has to be inspected. The user first moves to the node ‘Coi to Control o’ by placing the cursor on the node and then pressing the mouse button MOVE. Whenever the user points to a node and presses MOVE the tree is updated so that the indicated node moves to the centre of the screen (i.e., the indicated node becomes the *current node*). In our sample dialogue ‘Coi TO Control o’ becomes current node (Fig. 8). The arrows at the left and right bottom of the information window allow movement to invisible sons of the current node.

More specifically, in our example the double arrow in the lower right hand corner of the window enables the node *Commutation* to be shifted to the leftmost position on the screen, thus displaying six more sons. The jump arrow pointed to in Figure 8 causes a jump bar to appear on the screen (Fig. 9), which represents both the range of the entire set of sons and the range of the presently displayed visible ones. In this way the user may estimate how many nodes are invisible and what portions of invisible sons are located to the left and to the right of the information window. The range of visible nodes may be moved to the left or to the right by continuously holding down the respective mouse button. In this way, the term *Computers* is finally reached in Figure 10 and the three sons of that term are displayed in the window.

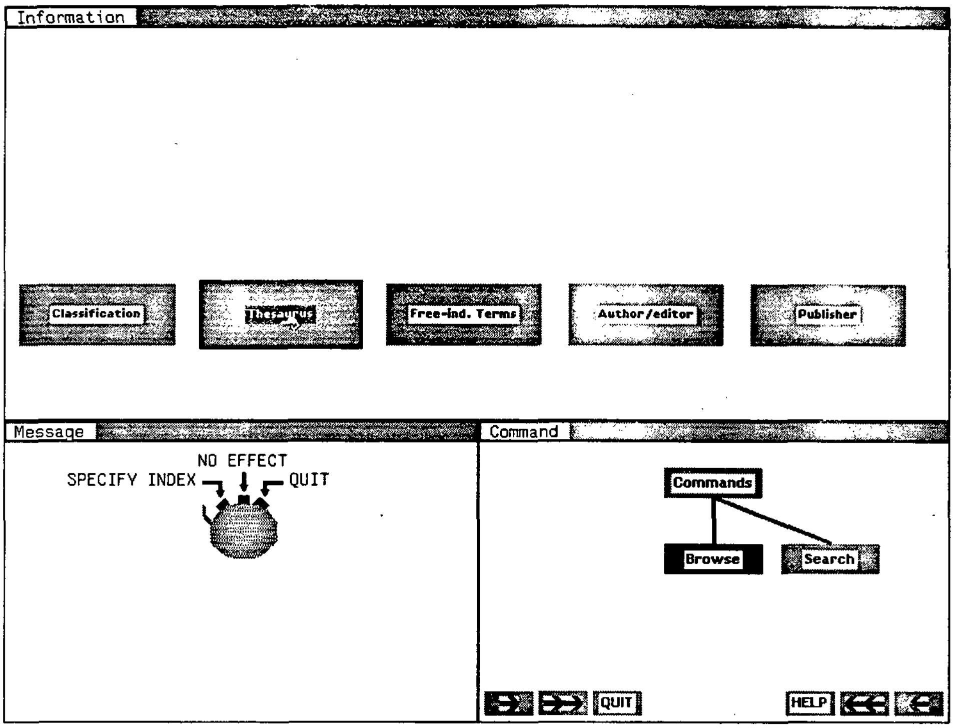


FIG. 6. Choosing the index structure *Thesaurus*

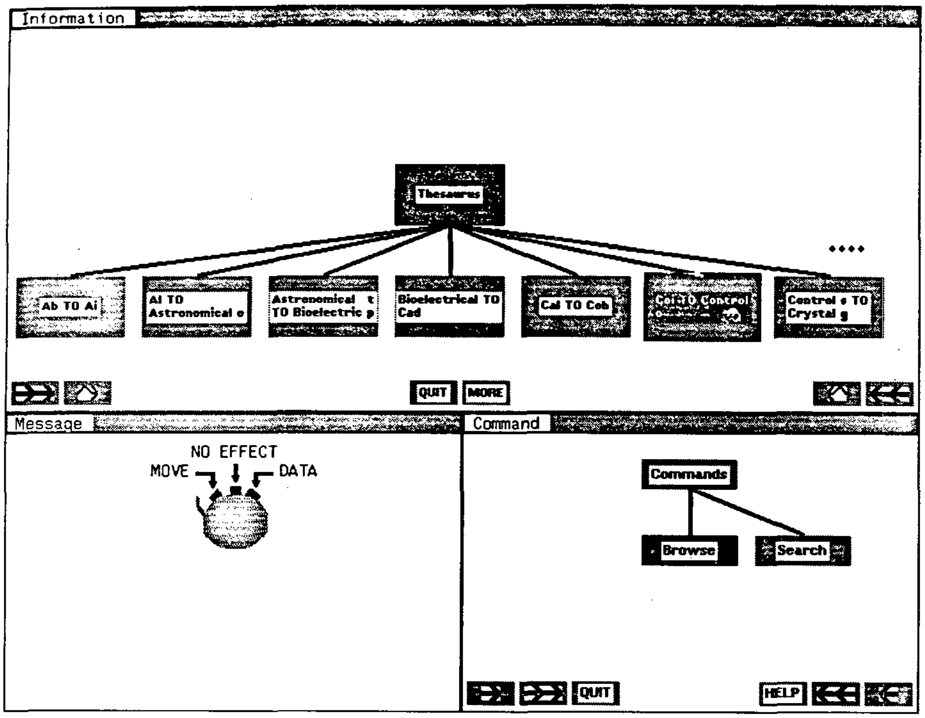


FIG. 7. First level of sons

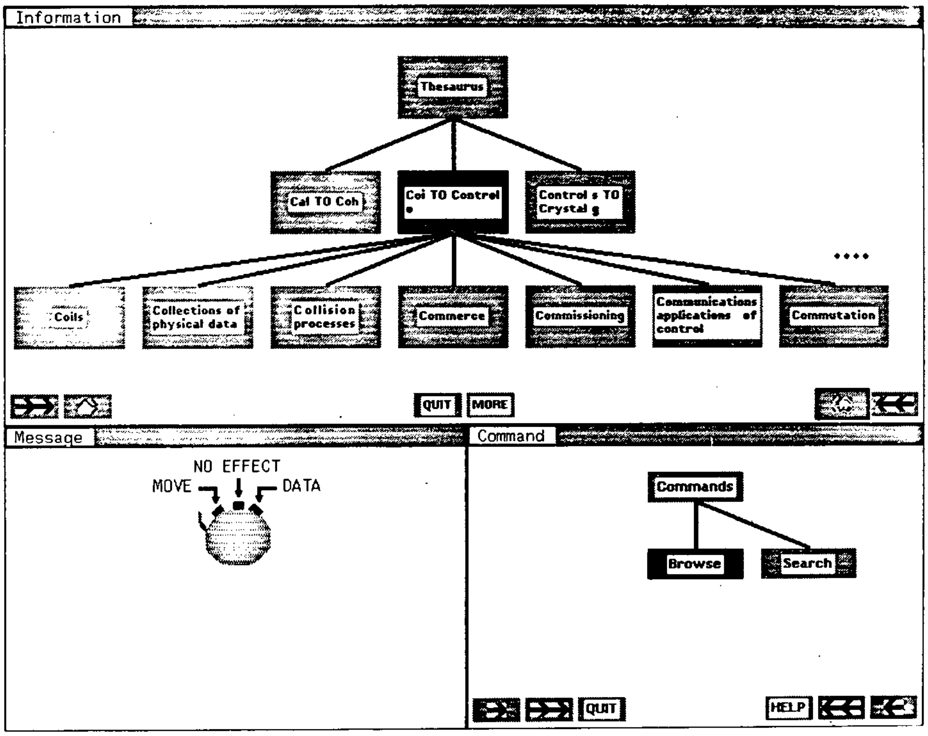


FIG. 8. Moving down to find *Computers*

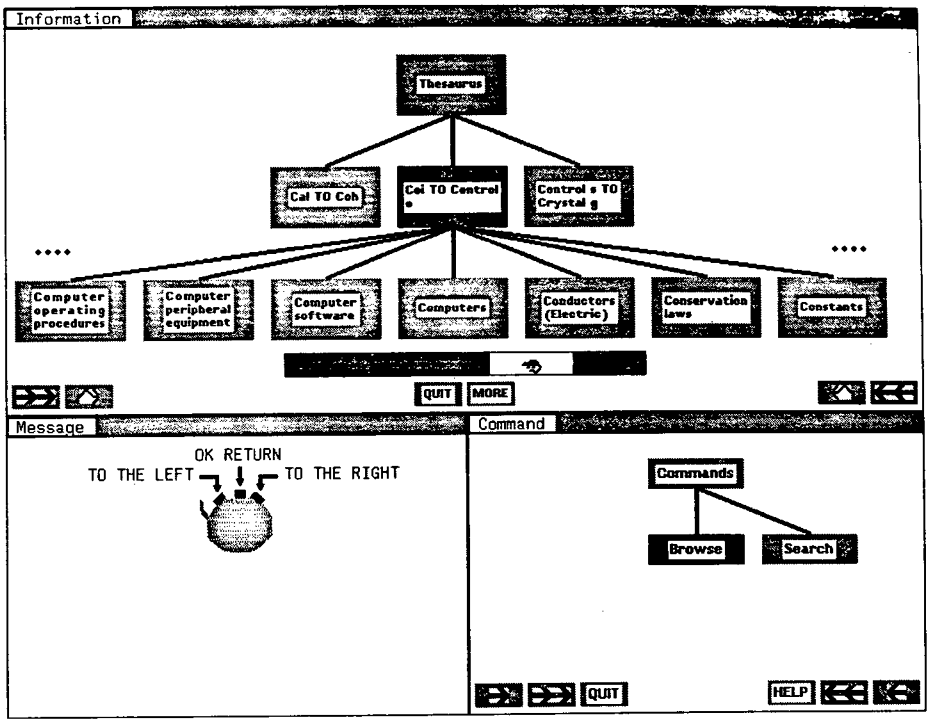


FIG. 9. Invisible sons can be fetched by a jump bar

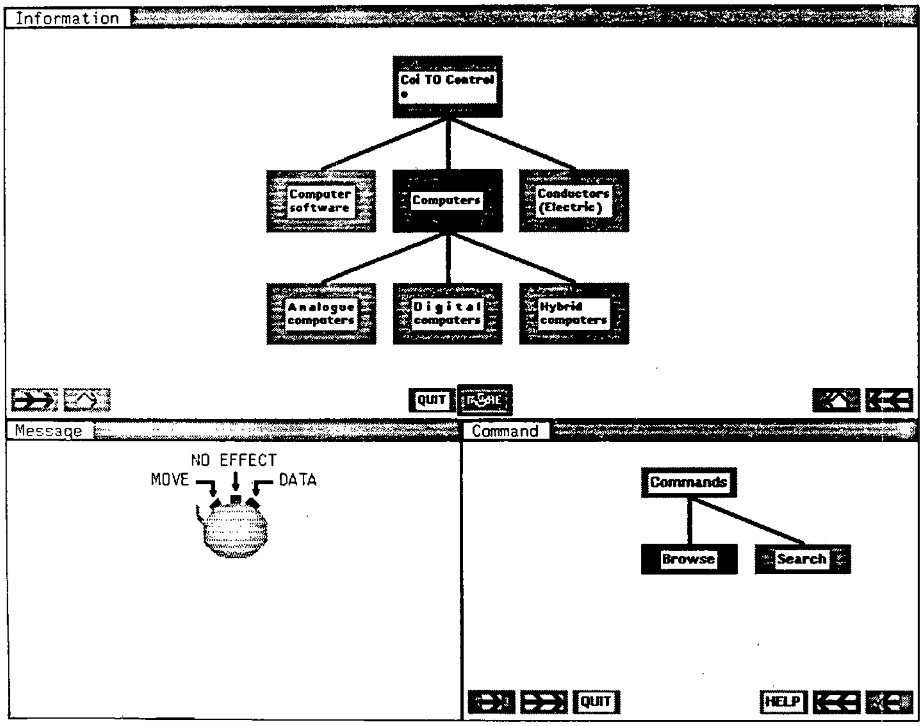
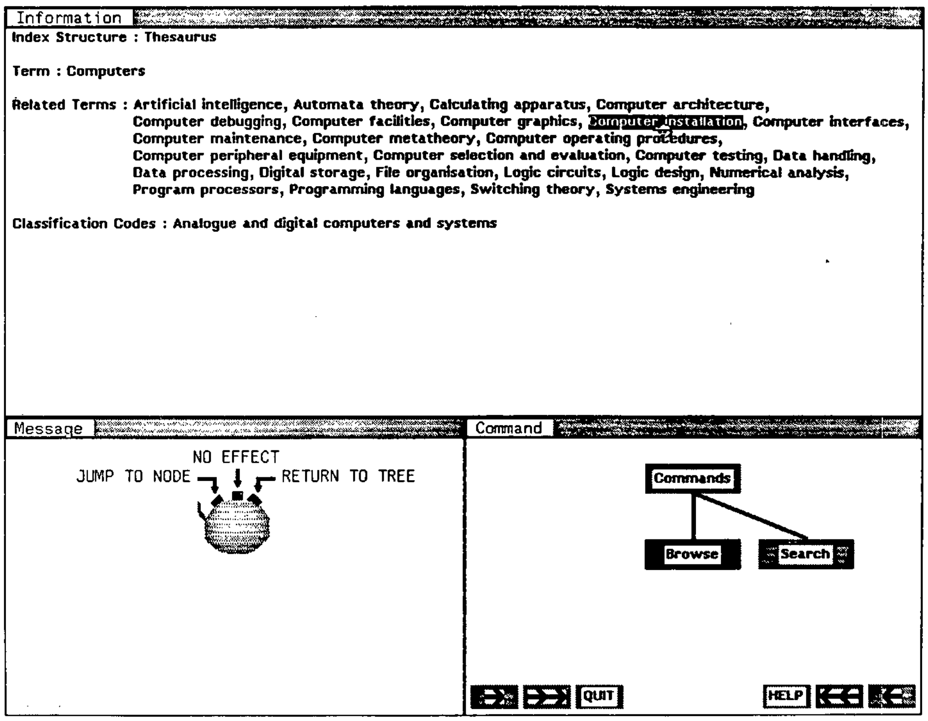


FIG. 10. *Computers* as current node


Index Structure : Thesaurus

Term : Computers

Related Terms : Artificial intelligence, Automata theory, Calculating apparatus, Computer architecture, Computer debugging, Computer facilities, Computer graphics, Computer installation, Computer interfaces, Computer maintenance, Computer metatheory, Computer operating procedures, Computer peripheral equipment, Computer selection and evaluation, Computer testing, Data handling, Data processing, Digital storage, File organisation, Logic circuits, Logic design, Numerical analysis, Program processors, Programming languages, Switching theory, Systems engineering

Classification Codes : Analogue and digital computers and systems

FIG. 11. Cross connections are displayed textually

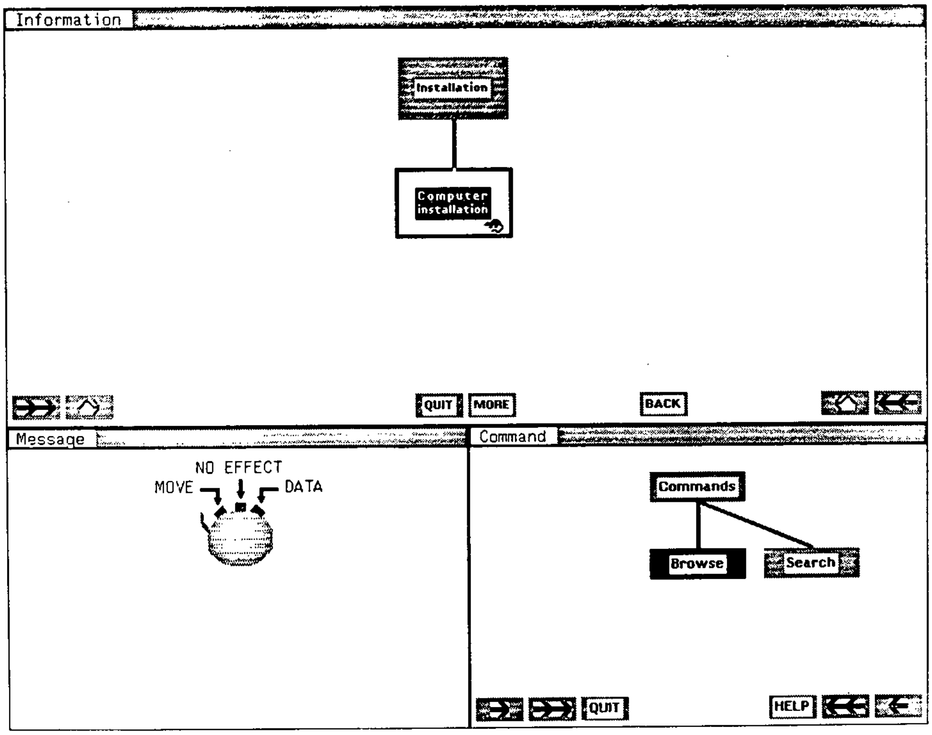
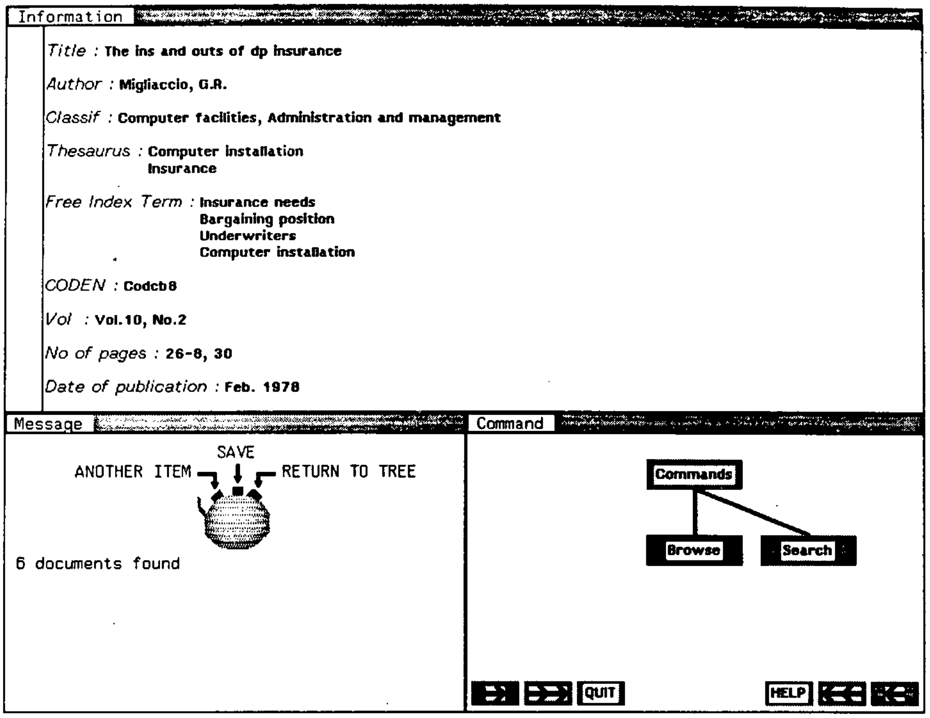


FIG. 12. Jumped to *Computer installation*



Title : **The ins and outs of dp insurance**

Author : **Migliaccio, G.R.**

Classif : **Computer facilities, Administration and management**

Thesaurus : **Computer installation**
Insurance

Free Index Term : **Insurance needs**
Bargaining position
Underwriters
Computer installation

CODEN : **Codcb8**

Vol : **Vol.10, No.2**

No of pages : **26-8, 30**

Date of publication : **Feb. 1978**

6 documents found

FIG. 13. One of the items of the data set belonging to *Computer installation*

If the index structure is a network, as in the case of our thesaurus, the user can request more structural information about the current node (MORE). The non-hierarchical relations will appear on the screen, as it is shown in Figure 11 (cf., Fig. 3). If one of the related terms seems to be of interest, the JUMP TO NODE command causes the selected term to become the current node and displays this term including its neighbourhood in the usual tree-like fashion. Selecting such a term is similar to selecting a term in a tree: as soon as the cursor is placed close to the screen position where the term appears, the system inverts the character string. In our case, the user decides to jump to *Computer installation* as it is a domain he or she is interested in (Fig. 12). This neighbourhood happens to be a very simple one: *Computer installation* is a leaf node, it is the only son of the node installation. However, the user has no idea *where* this visible neighbourhood is located in the index structure: it may be very close or far distant to where he or she jumped off. This is why the system keeps track of the jump-off position, enabling the user to directly jump back by means of the soft key BACK shown in Figure 12.

The command DATA allows direct access to source data belonging to the node pointed to. For every node displayed on the screen attached data can be requested by just placing the cursor on it and pressing the button DATA. Figure 13 shows the situation after this command has been executed while pointing to the term *Computer installation.* The size of the retrieved set is indicated in the message window: 6 documents found. One of the retrieved documents is displayed in the information window. Its description contains the term *Computer installation* in the thesaurus field, as will all of the other five retrieved documents.

As mentioned earlier, the IR system *Caliban* has been designed to treat various kinds of information such as office information, letters, memos, technical reports, personal data, etc. We demonstrated the capabilities of the system using our current test data: bibliographic material. Although other data would certainly be structured in a different way and therefore represented differently, the browsing facility on the corresponding index structures would be the same.

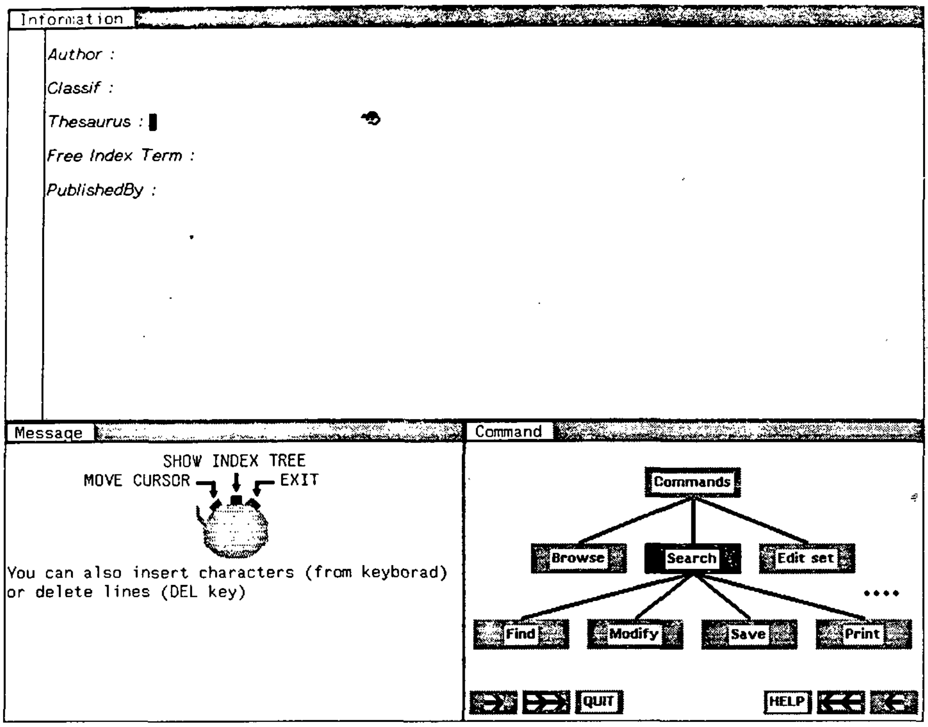
Browsing is to be distinguished from other forms of retrieving information:

1. Retrieval exclusively depends on the particular decisions made by the user while moving about the index data; however, the user may follow his or her associations of thoughts which is, in many cases, a concept superior to asking plain questions. On the other hand, a great deal of information can be missed when the retrieval depends on more or less random paths.
2. The browsing process does not allow one to access data from several parts of index structures at the same time: retrieval is limited to asking for data connected to one term only. The retrieved items are linked only by that term.

The next section explains a completely different approach to retrieving information items.

6. QUERY FORMULATION

When formulating a query, the user has to fill in the fields of a query template rather than using a command language. In Figure 14 the user is preparing what we call a *virtual information item*, similar to filling in a row of a sample relation in query by example (Zloof, 1975). Each searchable field of the stored information items, such as author, publisher, classification, thesaurus or free indexing terms in our test data, may be left blank or filled in according to the information the user is looking for. If a user is looking for items on a specific subject written by a specific author, the corresponding fields have to be filled in. Then, the virtual information item is submitted as query to the system which will return a set of real information items corresponding to the virtual one. However the user does not need to type long and complicated sequences of characters: the tree-structured indexes are always available (SHOW INDEX TREE) to assist the user when preparing a virtual information item. Terms may be picked directly from the screen and included into the virtual item.

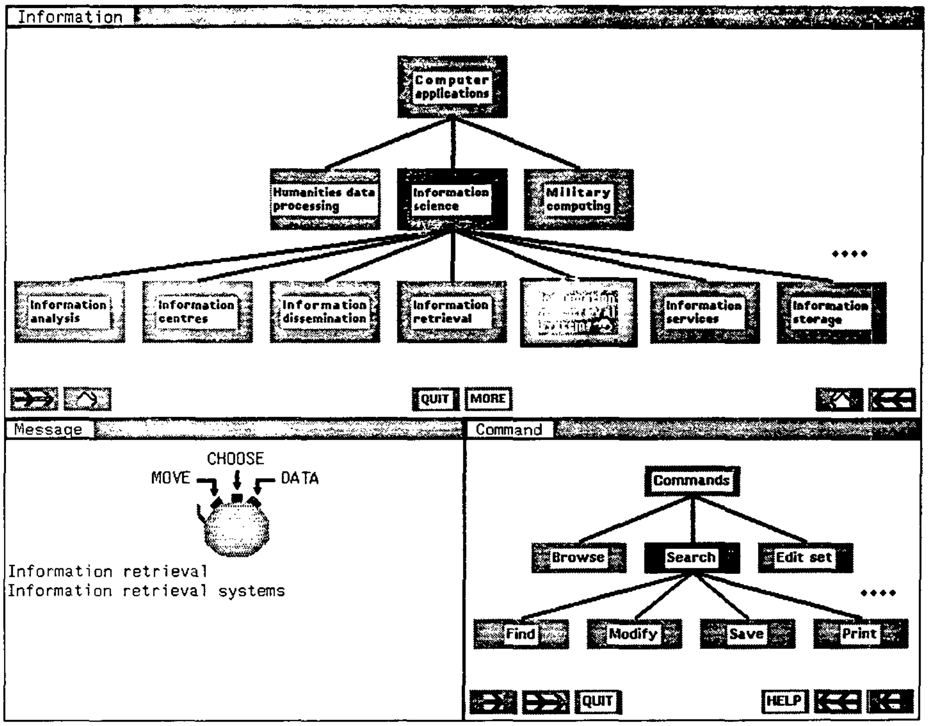


Author :
Classif :
Thesaurus :
Free Index Term :
PublishedBy :

You can also insert characters (from keyborad) or delete lines (DEL key)

FIG. 14. The query template

The MOVE CURSOR command (Fig. 14) allows explicit selection of a field of the template in order to fill it with query information. No matter to which field the cursor points, the index structure corresponding to that particular topic may be displayed by invoking the command SHOW INDEX TREE. Although in search mode, the user may use all the BROWSE facilities described above, browsing becomes a sub-activity of searching. (Whether browsing should also formally be treated as a sub-activity of searching and appear as a brother node of *Find, Modify*, etc. in the command tree is debatable.) However, this time, interesting terms can automatically be included in the query formulation by activating the command CHOOSE. In Figure 15 the two thesaurus terms *Information retrieval* and *Information retrieval systems* have been selected for inclusion into the query formulation, i.e., into the virtual information item. In fact, an arbitrary number of terms accessible through MOVE commands could be selected and included in the virtual item. *Caliban* acknowledges the choice made by displaying the terms in the message window.



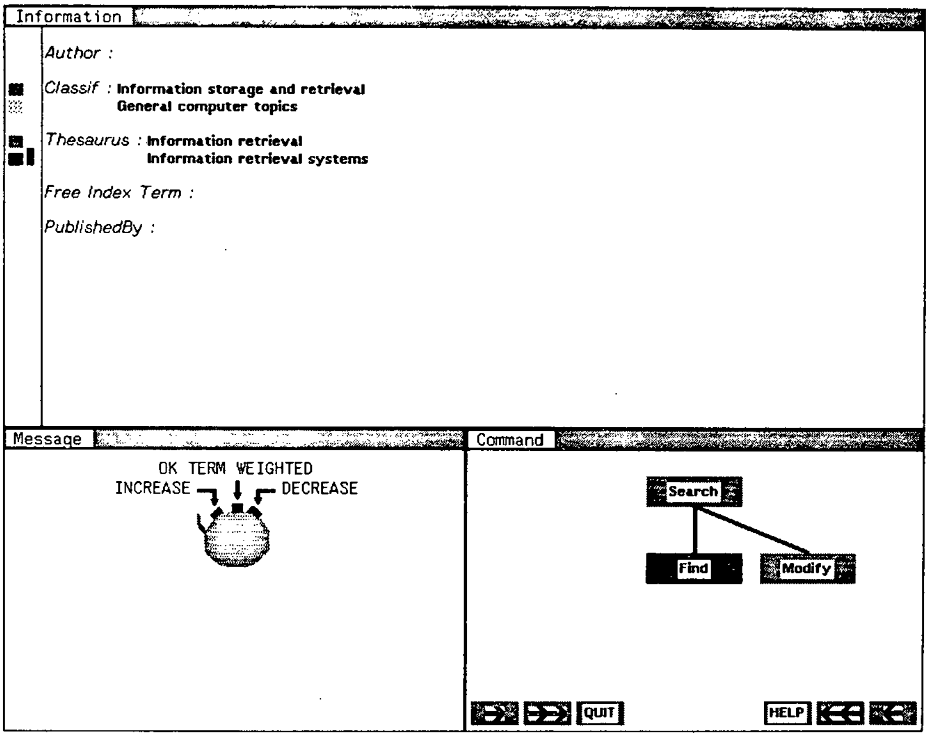
```
Information retrieval
Information retrieval systems
```

FIG. 15. Choosing terms to be included in the query

The index tree is left by the QUIT command and all the items chosen appear in the template. The user could now select any other index structure (or even the same again) in order to CHOOSE more index terms. In our sample dialogue he or she selects the classification tree and picks two other terms, namely *Information storage and retrieval* and *General computer topics.*

Once the template is satisfactorily filled with query information, the command *Find* is activated to retrieve real information items. Prior to executing the actual query algorithms, the user gets the opportunity to weight every field according to the importance a term should play in the retrieval process. This is done by 'colouring' a rectangle for every term: the darker the rectangle, the higher the weight and the more important the term (Fig. 16). These weights are used by both the retrieval and the ranking algorithms. The retrieval algorithm retrieves those information items which exhibit a higher similarity to the virtual information item than a certain threshold. The ranking algorithm ranks the set of retrieved documents by assigning higher ranks to items with higher similarity to the query.

After a set of real information items has been displayed to the user it is possible to choose an arbitrary item as the next query (Fig. 17). This is made possible by the fact that the structure of a *Caliban* query is *identical* to the structure of a real



Author :

Classif : Information storage and retrieval
General computer topics

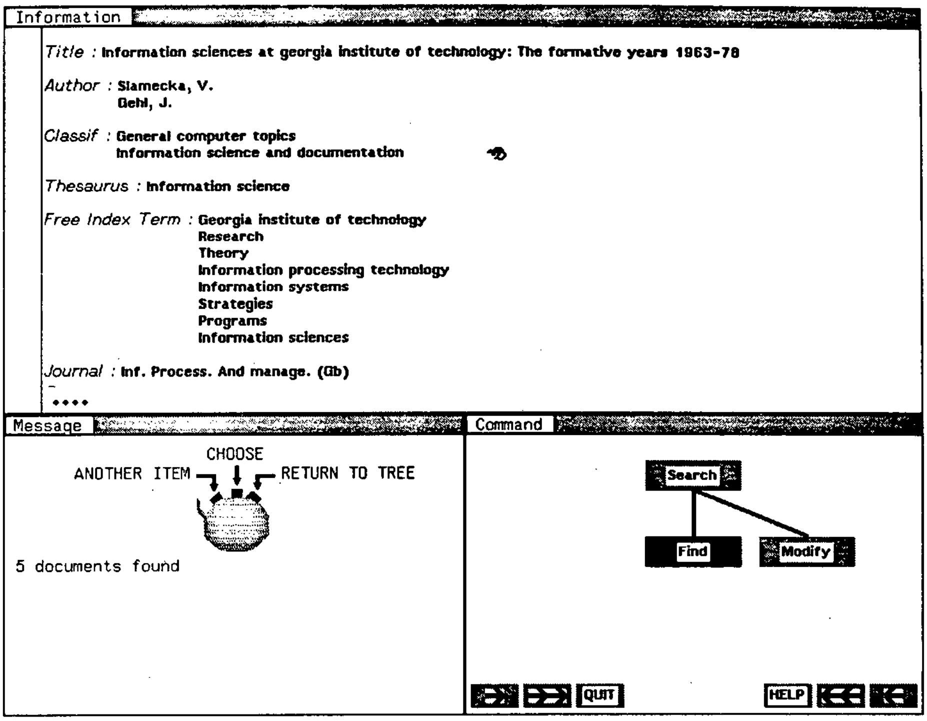
Thesaurus : Information retrieval
Information retrieval systems

Free Index Term :

PublishedBy :

OK TERM WEIGHTED
INCREASE DECREASE

FIG. 16. Weighting the terms in the query formulation



Title : Information sciences at georgia institute of technology: The formative years 1963-78

Author : Slamecka, V.
Gehl, J.

Classif : General computer topics
Information science and documentation

Thesaurus : Information science

Free Index Term : Georgia institute of technology
Research
Theory
Information processing technology
Information systems
Strategies
Programs
Information sciences

Journal : Inf. Process. And manage. (Gb)

CHOOSE
ANOTHER ITEM RETURN TO TREE

5 documents found

FIG. 17. One of the retrieved items

information item. In addition, the user may select any terms found in any of the items and include them in the template initially prepared. Other modifications like including new keywords or deleting old ones, can improve the quality of the search. In this way, the quality of retrieved item sets can continuously be upgraded in an *iterative process.* The idea of directly taking into account results of previous searches has already been reported by Oddy (1977), who describes a system which enables one to insist on terms belonging to a retrieved item or remove those terms from the query which seem to be outside the domain of the search. There are two major advantages of such a retrieval process:

1. A query may be prepared by directly employing index structures.
2. The query is very *similar* to the information item.

7. CONCLUSIONS

In the not too distant future computers will be used in many more sectors of the economy than is presently the case; they will even be used in the private home. Collecting, storing and making information available will be prominent tasks performed by these computer systems. As their users will not be computer specialists, we agree with Crawford who—in a recent paper (Crawford, 1981)—quotes B. Mahon: 'New search and retrieval techniques are a must.'

The *user interface* described in this paper is an attempt to remove the still existing barrier between the persons seeking and the systems containing information. This is why only *easy-to-understand concepts* underlie our man-machine interface. These concepts include the notion of *browsing* through both information structures and information items. Another such concept is the *virtual information item* which seems to be a natural way of expressing a user's information needs.

In order to make flexible browsing and efficient searching possible, powerful interactive systems are necessary. They can be designed and built employing those modern hardware and software tools emerging at the present time. However, in terms of modern hardware not only faster processors are to be mentioned but also *user-friendly* input and output devices. Software-wise the advent of *high level system programming* languages is the notable event. We made use of all these novel tools when developing the IR system *Caliban.*

ACKNOWLEDGMENTS

We would like to thank N. Wirth and his group for providing the hardware and systems software which enabled the development of *Caliban.* We are indebted to M. Bärtschi and P. Lamb for providing numerous valuable ideas, overall support, and—most important—a great deal of software. We are also grateful for many constructive comments by a referee who must have spent much time doing a thankless job.

REFERENCES

Bärtschi, M. and Frei, H. P. (1982) A data organization for information retrieval on a personal computer. In *Personal Computing.* (Ch. Schlier, ed.) pp. 75–91. Stuttgart: B. G. Teubner.

Bush, V. (1945) As we may think. *Atlantic Monthly 176,* 101–108.

Crawford, G. W. F. (1981) The hardware approach to information retrieval systems and its impact on the information market in the 1980's. In *The Design of Information Systems for Human Beings.* (K. P. Jones and H. Taylor, eds.) pp. 49–56. London: ASLIB.

Miller, L. A. and Thomas, J. C. (1976) *Behavioral issues in the use of interactive systems.* Yorktown Heights, NY: IBM Research Report (RC 6326).

Oddy, R. N. (1977) Information retrieval through man–machine dialogue. *Journal of Documentation 33*, 1–14.

Wirth, N. (1981) A personal computer for the software engineer. *Proceedings of the 5th International Conference on Software Engineering*, San Diego.

Wirth, N. (1982) *Programming in Modula-2.* Berlin, Heidelberg: Springer-Verlag.

Zloof, M. M. (1975) Query by example. *Proceedings of the NCC.* pp. 431–438. Montvale, NJ: AFIPS.