A BROWSING APPROACH TO DOCUMENTING DOCUMENTATION

Y. Leclerc, S.W. Zucker, and D. Leclerc

McGill University
Montreal, Quebec

ABSTRACT

Although every research facility contains a wealth of software, user information, and documentation, the extent and variability of this information make it difficult to retrieve. The most common medium - hard-copy - is especially difficult to use, with the resulting burden of information dissemination placed on the shoulders of a small group of expert users. This burden can be reduced with the aid of an on-line information retrieval system. Such a solution is attractive for a restricted research facility since the amount of information is limited enough for on-line storage to be reasonable, while its variability is limited enough for standard organizational principles to be applicable. The system we have designed provides a human-engineered framework within which a user can retrieve information organized around a loose hierarchy of keywords; he can either traverse the hierarchy or he can match keywords. Experience with an implementation of the system has shown that the two retrieval mechanisms are used differently. The former mechanism is used by novices for browsing through an area to determine what is available, while the latter mechanism is used by experts for quick access to specific technical information. As might be expected, a given user might be a novice in one area and an expert in another. Such typical users take advantage of both retrieval mechanisms. In short, the system provides a comfortable and efficient interface to documentation. And, perhaps more importantly, it demonstrates that such interfaces can be built and maintained with only modest expenditures in manpower and resources.

RÉSUMÉ

S'il est vrai que toute installation de recherche contient un logiciel, une information pour les utilisateurs et une documentation des plus riches, l'ampleur et la variété de cette information en rend l'extraction difficile. L'emploi du support le plus courant - la sortie sur papier - impose un fardeau particulièrement lourd sur les épaules d'une poignée d'utilisateurs spécialistes. On peut alléger ce fardeau en se servant du système d'extraction en ligne de l'information. Une telle solution est avantageuse pour une petite installation de recherche, étant donné que le volume de l'information est suffisamment réduit pour justifier la mémorisation en ligne alors que sa variabilité est passablement limitée pour permettre l'application de principes organisationnels standard. Le système que nous avons élaboré offre un cadre technique qui est conçu à l'image de l'esprit de l'homme et au sein duquel l'utilisateur peut extraire l'information disposée autour d'une vague hiérarchie de mots-clé. En expérimentant l'implantation du système, nous avons constaté que les deux mécanismes d'extraction servent de façon différente. Le premier mécanisme sert aux novices qui survolent un domaine pour déterminer ce qui s'y trouve, alors que le deuxième mécanisme sert à offrir aux spécialistes un accès rapide à une information technique particulière. Bien entendu, un utilisateur donné peut être novice dans un domaine et spécialiste dans un autre. Ce genre d'utilisateurs tirent parti des deux mécanismes d'extraction à la fois. En résumé, le système assure une interface aisée et efficace avec la documentation. Ce qui est peut-être plus important encore, il prouve que les interfaces de ce genre peuvent être incorporées et mises à jour, moyennant de modestes dépenses en main-d'oeuvre et en ressources.

> "There are those who would argue that the OS/360 six-foot shelf of manuals represents verbal diarrhea, that the very voluminosity of manuals represents a new kind of incomprehensibility. And there is some truth in that.
>
> Frederick P. Brooks, Jr. [1]

1.0 INTRODUCTION

As Brooks implies in the above quote, the sheer volume of information required for research facilities to function makes the information difficult to retrieve and difficult to comprehend. His response to this difficulty (in the case of OS/360) was to have a carefully organized set of manuals through which specific information, such as details of a programming language or parameters to a subroutine, could be easily found. While the time-honoured technology of manually searching through extensive indices is adequate for the retrieval of such information, it does not lend itself to the simple and leisurely discovery of that information. In other words, it is difficult to browse through the information in order to discover what is available. Computerized documentation systems, such as the VAX/VMS HELP facility [2], offer more flexibility in their information retrieval capabilities than hard-copy manuals, but are also not well-suited to browsing. The purpose of this paper is to describe one experiment in the design of a documentation system that provides mechanisms for both needs.

The decision to design and implement a documentation system with browsing capabilities was made when it was noticed that many members of our laboratory (*) tended to "re-invent the wheel" by needlessly duplicating the efforts of others. Even though previous efforts were documented in various ways, most people were unaware that a particular topic (e.g., Fast Fourier Transforms or pseudo-colouring of images) had been explored by another member of the laboratory, much less precisely what that other member had done. Also, whenever it was known that a particular topic had been explored by a particular person, that person spent considerable time explaining where the documentation for the topic was to be found, and providing general information about the topic. It was hoped that providing a documentation system that allowed people to both discover and quickly retrieve information about the resources of the laboratory would alleviate the above problems. It is also our experience that these problems of information dissemination are not unique to our laboratory, but rather are more often the case than the exception.

(*) The McGill University Computer Vision and Graphics Laboratory (CVaGL).

2.0 SYSTEM DESIGN

The primary goal of the Browsing System was to provide a means for browsing through and/or quickly retrieving information about the laboratory's resources. It was important for our needs, however, that the implementation, and especially the maintenance, of the system require a minimum of man-power and on-line storage.

The design decisions that were made to meet these goals are described in the following sections.

2.1 Accessibility

Since executable programs are an important part of the resources of the laboratory, it was felt that, ideally, the documentation system should be accessible from within other programs, and that other programs should be accessible from within the documentation system. For example, the former might be useful when editing a program (e.g., to find the parameters to an external subroutine), and the latter would allow a person to discover what programs are available without leaving the environment of the documentation system. We have incorporated only the latter capability in the current implementation, but we see no major problems in incorporating the former in future implementations.

2.2 Information Organization

A capability for browsing requires both an appropriate presentation of individual items of information and an organizational context that will facilitate the browsing. The first need was met by naming each item of information explicitly, so that it could be retrieved immediately once its name was located. In other words, each item of information, be it an executable program or some form of text, was assigned one or more keywords. The second need was met by making each entry short enough that many could fit on one page of a normal display terminal, and these keywords were then embedded in a structure whose category names were also keywords.

We chose a multiple-parent hierarchy (i.e. a directed acyclic graph with a single root node) as the structural organization of the

keywords for its simplicity and flexibility. A strict hierarchy was considered to be too inflexible, because it forces each keyword to be placed in a specific category. This often leads to difficulties both in searching and in categorizing information. At the same time, an arbitrary network of keywords, although very flexible, is difficult for a person to grasp and follow. The compromise of a hierarchy with multiple parents allows items of information to be found through multiple paths, and yet is simple enough that traversal of the graph is natural.

The structure is implemented as a doubly linked list of variable length records containing the keyword name, the file name, and the links to the fathers and sons of the keyword.

2.3 Separation of Organization and Information

The hierarchy of keywords is a separate file, with the information pointed to by file names. This has several important advantages over storing the information within the structure itself:

1 - Once a keyword has been entered for a particular item of information, the information can be updated by the person responsible without the intervention of a "librarian".

2 - Information need not be modified in any way to be incorporated into the Browsing System.

3 - The information can be created independently of the system without the need for special editors, thereby reducing the implementation cost.

4 - Since information is not stored explicitly, the structure is relatively small and can be quickly and efficiently traversed by the system, allowing quick traversal of the graph by the user.

2.4 Human Interface

A constraint on our design of the human interface to the Browsing System was the requirement that it be useable from a standard video terminal with minimal graphic capabilities. Inspired by the standard LISP pretty-printed form of displaying lists, we chose the indented form of displaying a hierarchy illustrated in figure 1. The desire for quick traversal of the graph led to single keystroke commands, which are listed at the bottom of the screen as a reminder to the user. Figure 2 illustrates the initial display of the Browsing System. (The design of the human interface was also inspired, in part, by an application of the Zog system [3,4] to browsing through books and technical reports [5]. In fact, a version of the Browsing System could have been implemented in Zog, but this conflicted with our requirements for simplicity of implementation and efficiency of computation. See also [6], where the issues of a document retrieval language are described in detail.)

Note that only one level of the hierarchy is initially visible in Figure 2, with ellipses to indicate that further levels exist. This is to simplify the user's initial view of the system, but can be changed to view up to 4 levels of the hierarchy simultaneously. If the displayed hierarchy does not fit completely on the screen, the user can scroll the display up or down. Figure 3 illustrates a case in which 2 levels of a hierarchy are displayed simultaneously, and figure 4 shows the displayed hierarchy scrolled down several lines.

The user moves down the hierarchy by moving the pointer (in the top left-hand corner of figure 2) up or down on the screen to the desired keyword, then typing "s" to view its sons. The user can move up the hierarchy one level at a time (retracing his steps down the hierarchy), or he can move directly back to the top. The user can also locate a specific keyword by entering the full keyword, or just an abbreviation. The Browsing System then searches for the first matching keyword from the top of the hierarchy (using a depth-first search). If the matched keyword is not the one that the user wanted, he can force the system to go on to the next match.

Once the user has placed the pointer at an appropriate keyword, the associated information can be displayed on the screen, or the associated program can be run. The user can always return to the Browsing System via a control character, giving him the freedom to experiment.

2.5 Creating and Modifying a Network

Any user can create and modify his own Browsing Network, although a password is required to modify the (default) system network. A different program is used for this which is a superset of the normal Browsing System. That is, every command in the Browsing System is also available in the Browsing Editor, along with extra commands to create, insert, delete, and modify keyword entries. Figure 5 illustrates the initial display of the Browsing Editor, along with a brief summary of the available commands.

The approach used in building the system

network was to proceed from the abstract to the concrete. The top level of the hierarchy is the most abstract, representing various points of view that a user might have when entering the system: a person interested in seeing the type of research carried on in the laboratory might start searching through the "AREAS OF RESEARCH" keyword; a visitor to the laboratory might first search through the "DEMONSTRATION PROGRAMS" keyword to get a glimpse of the current work; while a veteran user might start with the "LATEST ENTRIES" keyword to see what has recently been added to the network. As one goes down the hierarchy, the keywords become increasingly more specific, with the lowest level keywords typically pointing to programs, subroutine sources, technical reports, etc. The multiple-parent feature of the system was used extensively to allow keywords to be found through many different paths, easing the discovery of information by the user.

3.0 SUMMARY AND CONCLUSIONS

The Browsing system has been implemented as described in this paper. It has been used primarily, though not exclusively, to document software developed in our laboratory. This effort has been made in conjunction with a prior commitment to the use of a standard page preceding every piece of software created in our laboratory. Thus, for the most part, the Browsing System is pointing to files starting with a standard header page, which makes for a consistent view of the available software.

We have found that people do use the Browsing System for the two major reasons initailly planned for:

1 - To browse through available documentation, be it for hardware, software, technical reports, etc., in order to discover what is available.

2 - To quickly find technical documentation about a specific piece of software (e.g. for the parameters to a subroutine).

The Browsing System has been very helpful in alleviating needless duplication of effort by providing a means for discovering, in a simple manner, what other people had done. And, perhaps more importantly, the development and maintenance costs were quite reasonable (2-3 man-months for designing and implementing the system), demonstrating the feasibility of implementing useful documentation systems at reasonable costs. When these costs are compared with the time that our expert users no longer have to expend to simply disseminate information, the investment seems emminently worthwhile.

REFERENCES

[1] Brooks, F.P. Jr. (1975). "The Mythical Man-Month - Essays on Software Engineering", Addison-Wesley Co., Reading, Massachusetts (1975), p. 134.

[2] VAX/VMS Command Language User's Guide (1980). Order No. AA-D023B-TE. Digital Equipment Corporation. Maynard, Massachusetts.

[3] Robertson, G., Newell, A., and Ramakrishna, K. (1977). "Zog" A Man-Machine Communication Philosophy.", Carnegie-Mellon University Technical Report, August 5, 1977.

[4] Robertson, G., McCracken, D., and Newell, A. (1979). "The ZOG Approach to Man-Machine Communication", Carnegie-Mellon University Technical Report, October 23, 1979.

[5] Fox, M.S., and Palay, A.J. (1979). "The BROWSE System, Part I: An Introduction", Carnegie-Mellon University Technical Report.

[6] Gebhart, F., and Stellmacher, I. (1978). "Design Criteria for Documentation Retrieval Languages", J. Am. Soc. Information Science, Vol. 29, No. 4, July 1978, pp. 191-199.

ACKNOWLEDGEMENTS

The authors are grateful to John Mohammed, David Kashtan, Harold Hubschman, and Peter Sander for their help in designing the Browsing System, and for their critical comments and helpful suggestions concerning this paper.

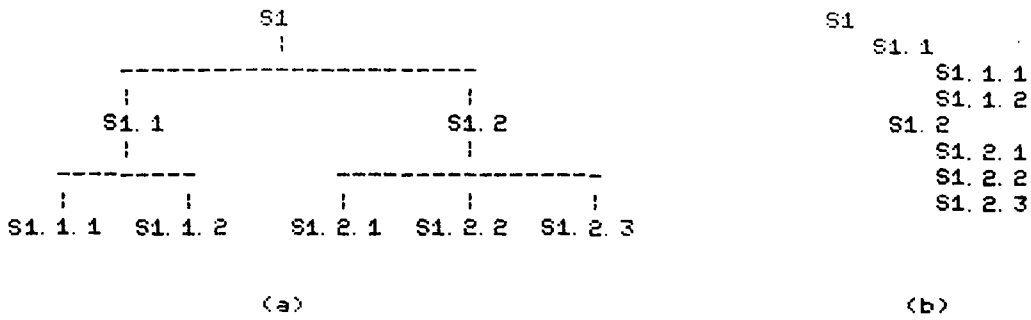

Figure 1. (a) A small hierarchy. (b) The same hierarchy displayed in the indented form used in the Browsing System.

```
Press "i" for information.        n =
↑
!

   LATEST ENTRIES ...
   AREAS OF RESEARCH ...
   SUPPORTED SOFTWARE ...
   AVAILABLE HARDWARE ...
   REPORTS & THESES ...
   DEMONSTRATION PROGRAMS ...
   PEOPLE POINTERS ...
   SUGGESTION BOX ...



back exit father generations help information locate next page run sons top
```

Figure 2. The initial display of the Browsing System. The ellipses following the keywords indicate that the keyword has sons. Attached to each of these keywords is an arbitrary text file, which can be retrieved by moving the pointer (currently pointing to the topmost keyword) to the appropriate keyword and then pressing "i". The set of available commands are listed at the bottom of the screen.

```
SUPPORTED SOFTWARE                    n =
↑  CHARACTER STRING MANIPULATION
¦     DISCARDING TRAILING BLANKS ...
      FILE NAME MANIPULATION ...
      FREE FORMAT DECODING ...
      KEYWORD MATCHING ...
      TRANSLATING CASES ...
   CVAGL UTILITY PROGRAMS
      DISPLAY
      PRETTY_PRINT
      SCANNER
      TEST PATTERN GENERATOR
      TYPE_VIDEO ...
   GRAPHICS
      PROGRAMS ...
      SUBROUTINES ...
   HUMAN ENGINEERING AIDS
      KEYWORD MATCHING INPUT STREAM ...
      VIDEO TERMINAL OUTPUT ...
   IMAGE PROCESSING
      PROGRAMS ...
      SUBROUTINES ...
   SYSTEM UTILITIES                                                  MORE...
back exit father generations help information locate next page run sons top
```

Figure 3. Two levels of the keyword SUPPORTED SOFTWARE. The "MORE..." indicates that the two levels could not fit completely on the screen.

```
SUPPORTED SOFTWARE                    n =
      TRANSLATING CASES ...                                          MORE...
   CVAGL UTILITY PROGRAMS
      DISPLAY
      PRETTY_PRINT
      SCANNER
      TEST PATTERN GENERATOR
      TYPE_VIDEO ...
   GRAPHICS
      PROGRAMS ...
      SUBROUTINES ...
   HUMAN ENGINEERING AIDS
      KEYWORD MATCHING INPUT STREAM ...
      VIDEO TERMINAL OUTPUT ...
   IMAGE PROCESSING
      PROGRAMS ...
      SUBROUTINES ...
   SYSTEM UTILITIES
      COMMAND LANGUAGE INTERPRETER
   VIDEO TERMINAL OUTPUT
      MCG_TERM_TYPE
      NEW_PAGE
-->   PRINT_HELP ...                                                  MORE...
back exit father generations help information locate next page run sons top
```

Figure 4. Two levels of the keyword SUPPORTED SOFTWARE scrolled down several lines. Notice that the arrow now points to the keyword PRINTHELP, and that the system is indicating that more of the hierarchy can be displayed by the two symbols "MORE..."

```
Press "i" for information.      n =
↑
:




   LATEST ENTRIES ...
   AREAS OF RESEARCH ...
   SUPPORTED SOFTWARE ...
   AVAILABLE HARDWARE ...
   REPORTS & THESES ...
   DEMONSTRATION PROGRAMS ...
   PEOPLE POINTERS ...
   SUGGESTION BOX ...



add bck crt del exi fat gen hlp inf kil loc mod nxt pag quit run son top updt ?
```

(a)

```
Add          - Add the Input node to the network as the son of the "-->" node.
Back         - Move the "-->" back one page (22 lines) when possible.
Create       - Create a new node (call this the Input node).
Delete       - Delete the "-->" node, and all nodes isolated by this deletion.
"downarrow"* Move the "-->" downwards (when not available, use linefeed).
Exit         - Exit from the program, saving all modifications in a new file.
Father       - Go to the father (super-category) of the current node.
Generations* Specifies depth of sub-categories seen simultaneously. Max (4).
Help         - Print this message.
Information- Print the information associated with the "-->" node.
Kill link    - Kill (delete) the link from the Input node to the "-->" node.
Locate       - Search for the first match of the specified string in the net.
Modify       - Modify the "-->" node, and make it the new Input node.
Next         - Search for the next match of the string in the net.
Page         - Move the "-->" forward one page (22 lines) when possible.
Quit         - Exit from the program without creating a new file.
Run          - Run the program associated with the "-->" node.
Sons         - List the sons (sub-categories) of the "-->" node.
Top          - Go to the top of the tree.
"uparrow"    * Move the "-->" upwards (when not available, use backspace).
Update       - Update the network file.
?            - Print the names of the files associated with the "-->" node.
```

(b)

Figure 5. (a) The initial display of the Browsing Editor.
(b) A summary of the commands available in the Browsing Editor.