# Designing a Menu-based Interface to an Operating System

Thomas S. Tullis

Burroughs Corporation
Irvine, California 92718

## ABSTRACT

The development of a large menu-based interface to an operating system posed a number of interesting user interface questions. Among those were how to determine the user's view of the relationships among the myriad of functions in the system, and how to reflect those relationships in a menu hierarchy. An experiment utilizing a sorting technique and hierarchical cluster analysis was quite effective in learning the user's perception of the relationships among the system functions. A second experiment comparing a "broad" menu hierarchy to a "deep" menu hierarchy showed that users made significantly fewer inappropriate menu selections with the broad hierarchy.

## INTRODUCTION

This paper describes some of the user interface issues that were addressed during the development of a menu-based interface to the operating system for Burroughs large mainframe systems. This interface, called Menu-Assisted Resource Control, or MARC, is designed to be an easy way for both new users and experienced users to access any of the features provided by the operating system. These include such features as file maintenance (e.g., copy, remove, change title), program execution, access to system news, sending messages, and access to various kinds of system information (e.g., status of jobs). In addition, if the user has the appropriate privilege, these features include all of the capabilities normally associated with system operators, such as control of job scheduling, control of peripherals, and manipulation of a variety of system parameters.

In short, MARC is a very large menu-based system. The latest version of it includes the capability of performing over 700 unique functions.



The development of such a large menu-based system posed many interesting user interface questions. We tried to answer some of these questions by collecting empirical data from representative users. The two main questions to be addressed in this paper are as follows:

1. How can one determine the logical relationships among the myriad of functions in such a system so that they can be presented to the user in a coherent fashion?

2. Given that the logical relationships between the functions have been determined, how does one best present these functions in a menu hierarchy (i.e., the "depth" vs. "breadth" question)?

## EXPERIMENT 1

One of the first problems we were faced with in developing MARC was how to organize the hundreds of functions that the system can perform in such a way that they make sense to the user. Clearly, the goal is to present the functions in a manner that reflects (as closely as possible) the user's perception of the logical relationships among the functions. To do that, one must try to understand the user's perception of those relationships. That was the goal of Experiment 1.

### Method

Trying to learn about perceptions of relationships among a set of objects is a very common problem in psychological research. Traditionally, the problem is addressed by asking subjects to judge the similarity of all possible pairs of objects. Such an approach would have meant asking subjects to make about 245,000 comparisons among the functions in MARC. Even the most agreeable subject would probably balk at the 68 hours or so that would take. Instead, a sorting technique was used (see Rosenberg and Kim, 1975). This involved asking subjects to sort the functions into mutually exclusive groups based on their similarity.

The basic procedure was as follows:

1. Short, one-sentence descriptions of the functions to be provided by MARC were written on index cards-- one function per card. For

example, one function was described as follows: "Copy all of the data on one disk pack to another disk pack". Originally, about 500 such cards were developed.

2. Five pilot subjects, composed of system operators, programmers, and technical writers who are familiar with the operating system, then sorted those cards into groups based on similarity. This took each subject about 4 hours and was quite tedious. Based upon these groupings, the set of cards was reduced to 271 by retaining only one function to represent any set of functions that all five of the pilot subjects grouped together. This made the sorting task more manageable.

3. Fifteen Burroughs employees then participated individually in the main sorting task for about two hours each. All of these subjects had some familiarity with the operating system. They included system operators, programmers, and technical writers. They were instructed to sort the cards into "logically related" groups and to assign a label to each group. No additional guidance about how many groups they should use or what "logically related" means was provided. After that, they were asked to sort the groups of cards they had created into still larger groups (called categories), again based on how logically related they are.

4. For each subject, a matrix of "perceived distances" between all pairs of functions (cards) was derived by applying the following rules:

   (a) If two functions were assigned to the same group, they were given a distance of 0.

   (b) If two functions were assigned to different groups, but the same larger category, they were given a distance of 1.

   (c) If two functions were assigned to different categories, they were given a distance of 2.

5. These distance matrices for the fifteen subjects were then summed to create an overall distance matrix. The entries in this matrix could range from 0 (if all subjects put that pair of functions in the same group) to 30 (if all subjects put that pair in different categories).

6. The overall distance matrix was then analyzed using a hierarchical clustering technique. Specifically, a technique based upon the "Minimum Method" described by Johnson (1967) was used.

## Results

The hierarchical clustering resulted in 18 levels of clustering, ranging from the lowest level with 261 clusters to the highest level with 1 cluster. Given the large number of functions involved, it is not practical to present the entire output from the hierarchical clustering, but Figure 1 shows an example of one portion of that output. The individual functions are represented by numbers for

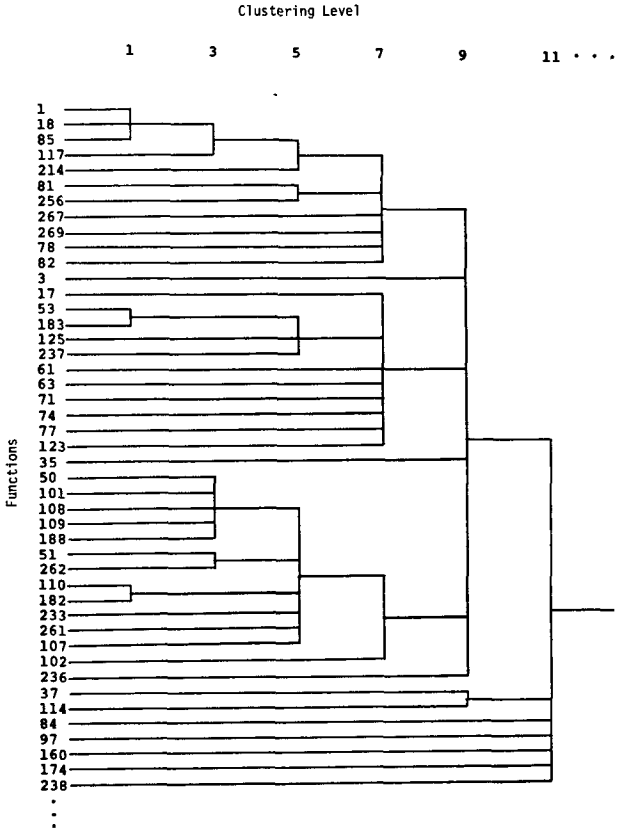


Figure 1. A portion of the output from the hierarchical clustering analysis of 271 MARC functions

ease of reference. The cluster levels are shown across the top; for simplicity, only every other level is shown. The lines in Figure 1 represent the levels at which the functions merged into clusters.

For example, functions 1, 18, and 85 merged at level 1, indicating they were viewed as being very closely related. This is reasonable, since those three functions correspond to requests for displays of jobs that are active on the system, recently completed jobs, and tasks running under jobs. As another example, Figure 1 shows that functions 97 and 160 (near the bottom) are not as closely related, since they did not merge until level 11. Although both of those functions involve actions on programs that are currently running, one of them "locks" the program so it cannot be accidentally interrupted, and the other displays the contents of a memory location being used by that program.

## Discussion

For the purposes of this paper, the importance of Experiment 1 lies primarily in its methodology rather than the specific outcome of the hierarchical clustering. The sorting technique and hierarchical clustering proved to be quite an effective means of learning the logical

relationships that users perceive among a large set of functions.

One of the specific outcomes of the clustering, however, may be of general interest. Many of the functions performed by the system involve operations on various peripheral devices such as line printers, disk packs, and magnetic tape drives. A relatively standard set of actions can be performed on those devices, including acquiring a new device, displaying the status of a device, and reserving a device for maintenance work. The subjects in Experiment 1 could quite reasonably have grouped these functions either by type of action (e.g., acquiring, displaying status, reserving) or by type of device (e.g., printer, disk pack, magnetic tape drive).

From a strictly linguistic perspective, one might predict that the subjects would group these functions by action, since you say "reserve a disk pack" rather than "disk pack, reserve". That is also what one might predict from the existing command language interface to the operating system, which conforms to a "verb-object" syntax. However, the subjects in this experiment uniformly chose to group these functions by device rather than by action. This implies that when they think about performing one of these functions (like acquiring a new printer), their attention is focused on the device rather than the action. This sort of "object" as opposed to "action" orientation is consistent with the approach currently being used in many window- and icon-based systems, where the icon (representing an object) is selected first, followed by selection of an appropriate action from a pop-up menu.

## EXPERIMENT 2

Given that the logical relationships among the functions to be included in MARC have been determined, the next problem is to translate those relationships into a menu hierarchy. The basic problem is deciding how many levels of menus to use. For example, one could take the output from the hierarchical clustering at face value and implement 17 levels of menus. Thus, in Figure 1, every vertical line connecting a set of horizontal lines would represent a menu. For example, at the lowest level, there would be a menu containing just functions 1, 18, and 85.

This problem is certainly not a new one in the design of menu systems. The trade-off between "depth" of menus (number of levels) and "breadth" (number of selections per menu) has been studied recently by Miller (1981); Snowberry, Parkinson, and Sisson (1983); and Kiger (1984).

Miller studied the effects of depth vs. breadth by having subjects find target words in one of four menu hierarchies, each of which contained 64 target words. The four hierarchies ranged from a "deep" structure with six levels of menus having two choices each, to a "broad" structure having 64 choices on only one menu. He found that time to reach the target word was basically a U-shaped function with number of display choices: the very "deep" and "broad" conditions resulted in the longest times while the two intermediate conditions (3 levels of 4 choices each, and 2 levels of 8 choices each) resulted in the shortest times.

Snowberry et al. (1983) replicated Miller's finding using the same menu hierarchies, but they added another version of the "broad" condition with 64 choices on one menu. They pointed out that Miller's menu with 64 choices was the only one in his study that did not retain any of the information about the categories these words belonged to; it was essentially a random display of words. Snowberry, et al, added a 64-choice menu in which the words were visually separated into their logical categories. They found that this "broad but organized" menu resulted in the shortest time to find the target word.

Kiger (1984) studied five different menu structures, each of which contained 64 target items. The deepest menu structure used six levels of menus with only two choices each. At the other extreme, there were three broad menu structures that used only two levels of menus but different numbers of choices per menu: 8 and 8, 4 and 16, or 16 and 4. He found that the three broad menu structures resulted in the shortest times and fewest errors in finding the target. Further, the "balanced" broad menu structure (8 and 8) seemed to be preferred by the subjects.

These studies would lead the designer of a menu system to strive for breadth in a menu hierarchy, as long as categorical information can be retained. But the generalization from studies of relatively simple menu hierarchies with only 64 target conditions to complex hierarchies with several hundred target conditions is tenuous at best. It seemed wise to evaluate depth vs. breadth in the context of the large number of functions to be provided by MARC.

### Method

Two different menu hierarchies for MARC were developed. These represent two different ways of "slicing" the results of the hierarchical clustering from Experiment 1. Both versions would probably be considered rather "broad" by the standards of the studies just described. The two versions were as follows:

1. Single-column Menus - This version limited the number of selections on one menu to those that could fit in one column, 15. The "home" or "root" menu that a system operator would see for this version is shown in Figure 2. Unlike the previous studies, where all of the target conditions always occurred at the same level, the results of the hierarchical clustering suggested that the MARC functions should occur at various levels in the hierarchy. Thus, some functions could be accomplished by viewing only two menus, while others required three or four menus.

2. Multi-column Menus - This version was "broader" in that it expanded the number of selections on one menu by allowing two and three columns, up to a maximum of 45 selections per menu. The "home" menu for this version is shown in Figure 3. With this version, all functions could be accomplished by viewing only two menus.

```
        HOME - OPERATOR SELECTIONS                                    MARC
Action: [                                                              ]
        HOme  PArent  GO  PRevious  QUit          Press SPCFY for Help

        JOB     Job/Task Management
        PS      Printing System
        DEV     Devices
        SYS     System Management
        LOG     Logging
        MCP     MCP Control
        DT      Date and Time
        HLI     Halt/Load and Intrinsics
        CCS     Compilers, Control, and Supervisor Programs
        USER    Usercodes and Privilege
        DIR     Directories and File Maintenance
        LIB     Libraries, Functions, Subsystems, and System Files
        DC      Data Communications
        NET     BNA Network Control
        COMS    Communications Management System

Choice: [                                                              ]
```

Figure 2 - The "home" menu for the single-column menu prototype

```
        HOME - OPERATOR SELECTIONS                                    MARC
Action: [                                                              ]
        HOme  PArent  GO  PRevious  QUit          Press SPCFY for Help

DJ    Display Jobs      CR    Card Reader       HLI   H/L & Intrinsics
CJ    Control Jobs      CP    Card Punch        SP    Special Prog's
JQ    Job Queues                                USER  Usercodes
                        PK    Disk Pack         PRIV  Privilege
PS    Print System      DK    Fixed Disk
                        DT    Diskette          DISK  Disk File Mngmt
CON   System Config     MT    Magnetic Tape     TAPE  Tape File Mngmt
SM    System Mngmt                              ACC   Access Structure
DUMP  Dumps             LP    Line Printer      LIB   Lib's, Subsystem
DIAG  Diagnostics       IP    Image Printer
LOG   Logging                                   DC    Datacomm Control
SWAP  Swapper           MM    Memory Modules    NET   BNA Control
MCP   MCP Control       PROC  Processors        COMS  COMS Information
DATE  Date and Time     SC    System Console    SESS  Session Control
                        OTHER Other Devices     SEND  Send Message

Choice: [                                                              ]
```

Figure 3 - The "home" menu for the multi-column menu prototype

As an example of the differences between the two menu hierarchies, notice the "Devices" selection on the single-column home menu in Figure 2. This selection leads to another menu that contains basically the same selections as are contained in the center column of the multi-column home menu in Figure 3 (i.e., "Card Reader", "Card Punch", "Disk Pack", etc).

Prototype versions of MARC were developed on a Burroughs B20 computer workstation to represent these two menu systems. The prototypes recognized two types of user entries:

1. Menu selections - made by entering the desired mnemonic selection key, such as "DEV" for "Devices", in the "Choice" field at the bottom of the screen. If the selection led to another menu, that menu was then displayed. If the selection would have resulted in the execution of a function in the real system (like deleting a file), the prototype simply returned a message like, "The number is 267". This number was different for each function. The subjects were instructed simply to record this number.

2. Actions - made by entering, in the "Action" field at the top of the screen, at least the first two letters of one of the words listed just beneath that field. "Home" was used to return to the home, or root, menu; "parent" was used to return to the menu that contains a selection leading to the current menu; "go" followed by the name of a screen (shown on the first line of each screen) was used to go directly to that screen; "previous" was used to redisplay the last screen shown; and "quit" was used to end the session.

The experiment was a between-subjects design, with 20 subjects using the single-column menus and 17 subjects using the multi-column menus. The subjects included system operators, programmers, and technical writers. They were randomly assigned to the two groups. Their degree of experience with the Burroughs operating system ranged from about 1 month of use to over 15 years.

Each subject was presented with the same set of 24 tasks to accomplish using the prototype. These tasks were selected to represent the kinds of tasks that the operator of a system might be presented with, including printing a directory of files on a tape, sending a message to a user, powering up a disk pack, and checking the status of a line printer. The subjects were given no guidance in how to accomplish the tasks; they were left to their own devices to find the appropriate menus and selections.

The following measures were recorded automatically by the prototype for each subject:

1. Total number of steps taken in accomplishing the set of 24 tasks. A step was defined as pressing the "transmit" key. This could have been either a menu selection, an action like returning to the home menu, or an invalid entry.

2. Total time to accomplish the set of tasks.

3. Total number of errors committed. An error was defined as an entry that the prototype could not interpret. Note that "logical errors", like selecting a valid, but inappropriate, menu item are not included in this count. Such "logical errors" would tend to inflate the total number of steps taken.

**Results**

Every subject in both groups eventually accomplished all 24 tasks, although some took a more circuitous path than others. Table 1 presents the results for both groups, including the mean number of steps taken, mean time to complete the tasks, and mean number of errors committed.

Not surprisingly, the total number of steps was significantly greater with the single-column menus (134.8) than the multi-column menus (102.3) as indicated by a t-test: $t\ (35\ df) = 4.83$, $p < .001$. This is to be expected from the very nature of the menus: the greater depth of the single-column version necessitates more steps to accomplish the same functions.

One can also determine the minimum, or optimum, number of steps that a user would have to take in order to accomplish the set of tasks, if no "side trips" were taken down incorrect menu paths. As shown in Table 1, for the single-column menus the optimum number of steps is 80, while for the multi-column menus it is 70. Thus, any steps beyond the optimum number can be viewed as extra, or unnecessary, steps. The interesting finding is that single-column menus resulted in significantly more extra steps than did multi-column menus (54.8 vs. 32.3): $t\ (35\ df) = 3.34$, $p < .001$.

In spite of the fact that subjects took significantly more steps with the single-column version, they did not take significantly longer to perform the set of tasks (37.8 vs. 35.8 minutes): $t\ (35\ df) = 0.53$. Obviously, this means that the subjects were taking more time per menu with the multi-column version, as one might expect. However, the extra steps for the single-column

Table 1. Results of Experiment 2

| | Total # Steps | Time (mins) | Total # Errors | Optimum # Steps | Extra # Steps |
|---|---|---|---|---|---|
| Single-column Menus | 134.8 | 37.8 | 8.7 | 80 | 54.8 |
| Multi-column Menus | 102.3 | 35.8 | 6.0 | 70 | 32.3 |

version did result in a marginally significant increase in the number of errors, or unrecognizable entries, made with that version (8.7 vs. 6.0): $t$ (35 df) = 1.79, $p < .10$.

## Discussion

Given the fact that the subjects were able to perform a set of tasks in essentially the same amount of time with both versions, it seems clear that the best menu hierarchy to use is the one that required fewer steps-- the multi-column version. This version has the added advantage that it resulted in marginally fewer errors, obviously due to the fact that there were simply fewer opportunities for error with the smaller number of steps. Another advantage of this version, from a total system perspective, is that the smaller number of steps taken by users translates to less processor time in handling user input and displaying output.

The results of Experiment 2 support the notion that designers of menu hierarchies should strive for breadth rather than depth in designing their systems. The main disadvantage of a deeper structure like the single-column menus in this study appears to be that users have difficulty predicting from a high-level menu (like the home menu in Figure 2) what low-level functions fall under each of those selections. Consequently, they sometimes start down inappropriate menu paths and may not discover their error until they reach a low-level menu. This is the behavior that resulted in the greater number of "extra" steps with the single-column version. On the other hand, a broader structure like the multi-column menus brings more of the low-level functionality closer to the surface, making it easier to predict which path to take.

## ACKNOWLEDGEMENT

I wish to thank Pamela Whittington for her invaluable assistance in collecting and analyzing the data from both of these experiments.

## REFERENCES

Johnson, S. C. (1967).
Hierarchical clustering schemes.
Psychometrika, 32,
p. 241-254.

Kiger, J. I. (1984).
The depth/breadth trade-off in the design of menu-driven user interfaces.
International Journal of Man-Machine Studies, 20
p. 201-213.

Miller, D. P. (1981).
The depth/breadth tradeoff in hierarchical computer menus.
Proceedings of the 25th Annual Meeting of the Human Factors Society,
p. 296-300.

Rosenberg, S. and Kim, M. P. (1975).
The method of sorting as a data-gathering procedure in multivariate research.
Multivariate Behavioral Research, 10,
p. 489-502.

Snowberry, K., Parkinson, S. R., and Sisson, N. (1983).
Computer display menus.
Ergonomics, 26,
p. 699-712.