As businesses increase their emphasis on productivity, data processing departments face rising demand for computer services from people with no data processing training or background. To be effective, these services must be easy to use. This article discusses some usability considerations and how they were applied in developing an end user system. It relates experiences and observations in developing a system that is marketed in Canada under the name Interactive Extension Facilities. This system is an extension to VM/370 and CMS and was developed by the IBM Canada Limited Laboratory (Toronto) to enhance the ability of business professionals to do their work without becoming data processing specialists.

Improving system usability for business professionals

by G. A. Helander

A consequence of increased emphasis by businesses on productivity is that data processing departments are experiencing rising demand for computer services from people who have no data processing training or background.[1,2] To be effective for general users, these services must be easy to use. Considerable work has been done on the human factors characteristics of applications[3-6] from which specific attributes for the user interface[7-9] have been suggested. The particular problem that we studied was that of making a full-function data processing system usable by business professionals without requiring them to pursue data processing training. Discussed in this article are system usability considerations and their application to end user systems.

In the past, full-function data processing systems have supported trained data processing professionals and users of specialized applications.[1,2] There are, however, a number of useful tools like APL,[10] Document Composition Facility (DCF or SCRIPT),[11] A

Departmental Reporting System (ADRS),[12] and Query-by-Example,[13] that can enhance the ability of a business professional to apply a computer system to his work. When such a user approaches the system, he must know about, understand, select, and apply to his application a number of different tools available in the system. Each tool can have a different and inconsistent interface, requiring training to access and use. There are, for example, over fifty Conversational Monitor System (CMS)[14] commands and over twenty-five DCF options. Education on how to use a tool, however, is often available only in a classroom. In many cases, reference material is difficult to obtain and advice and assistance are spotty.

Common aids to CMS system usability are EXECs that relieve complexity and provide specific applications. (An *EXEC* is a named combination of computing functions for a frequently performed task.) There are problems with this approach, however. Typically, EXECs force the user through a fixed command sequence, thereby making him do things he may not want to do. Also, there may be a large number of EXECs which force the user to memorize them. Unless the EXEC names are meaningful to the user, he must memorize names that are unknown to him. Another alternative is to put some prompting logic into the EXEC. The problem here is how to cue for commands and still provide needed flexibilities. A sequence of commands entered yesterday may not be sufficient for today because the requirements and applications may have changed. The order of processing may change because priorities have changed, or one may want to suspend an active application and invoke another that has greater urgency. For a user to work in a way that is natural, the system must be one that the user controls.

In addition, as users gain experience with a system they create ideas on improving it and on new applications to assist them.[14] They want to tailor or personalize the system to their unique and individual needs. In short, they want a system that can expand and change with their needs. It is important, therefore, that a system be so designed that a non-data-processing user can take part in all stages of design and development of his own applications that run on the system.

Interactive Extension Facilities

A system that we call Interactive Extension Facilities[15] is designed to facilitate user control, user assistance, and cooperative application development by organizing system-user interactions into dialogues and sessions. A *session* structures full-screen IBM 3270 display panels, commands, helpware, and generalized system utilities so as to help users perform a particular activity, such

as writing memoranda and reports or developing programs. Each session is a screen that contains commands that work with particular types of objects—reports, memoranda, or programs. The commands available depend on the desired result and user's experience.

To support the concept of a session we have built the following three components based on VM/370 and CMS:

- *Dynamic Management Facility* (DMF) extends CMS and makes it a session processor. It also includes a Problem Management Component that allows on-line problem reporting and management; problem reports are maintained in a central library.
- *Dynamic Development Facility* (DDF) provides a session that can be used to modify sessions and develop new ones.
- *Dynamic Application Environment* (DAE) is a set of documents and end user sessions based on DMF, DDF, and other IBM software that form a complete end user service. It can be extended to meet unique end user requirements.[16]

The components are organized in this way to give an installation flexibility in configuring a system. The Dynamic Management Facility (DMF) is the engine that drives the other two components. Typically a data processing department uses the Dynamic Application Environment (DAE) and Dynamic Development Facility (DDF) to create an end user system that suits their unique needs. When completed, multiple copies of a system containing only the Dynamic Management Facility (DMF) and Dynamic Application Environment (DAE) can be distributed to multiple remote systems. The Problem Management Component of DMF can be used to support remote systems from a central location as well as be used solely by a data processing department to do system support and problem management.

System development

This system has evolved over time,[14] beginning with a prototype in 1978. At that time, we built a complete end user system and tested it in a user department in a large corporation.[17] Based on their reports and those of other users, we modified and improved the prototype. The developmental process we followed was an important factor in the product development. The process combined standards, benchmark sessions, reviews, and tests.

We searched the publications of other authors[3,8,18,19] for ideas for the design of our system. Based on this research, we established standards for such things as panel format, commands, sentence phrasing, command names, helps, and FIRSTUSE and EXPLAIN content. Developers then worked within these standards.

As each new session was developed, it was often patterned after a benchmark or reference session. This enhanced the consistency of the appearance of the panels as well as the consistency of the dynamic patterns of usage. Thus, the sessions looked the same and reacted in the same way in similar situations.

Each session was reviewed twice for usability in addition to our usual defect-removal reviews. At the first review, end users actually tried prototypes and mock-ups of a session to see whether it worked the way they wanted it to. Their suggestions and changes were incorporated before any development took place. The second review concentrated on standards, system-wide consistency, and the information presented on the display panels. For example, within each class of display panels, the same information had to appear in the same location. System responses to commands also had to be identical. Finally, the wording was reviewed. Sentences had to be short, in the active voice, and free of data processing jargon. A user-invokable dictionary of terms was developed for complex words. In addition to the defect-removal steps, the system was tested with real end users who were invited to suggest improvements, which we often found to be quite valuable. For example, during one of our early tests, one person suggested that sample text and Dialogue Composition Facility (SCRIPT) commands be supplied when a new document is created. The user could then replace the sample text with his own. This suggestion is the basis for the model facilities that are now standard in the system.

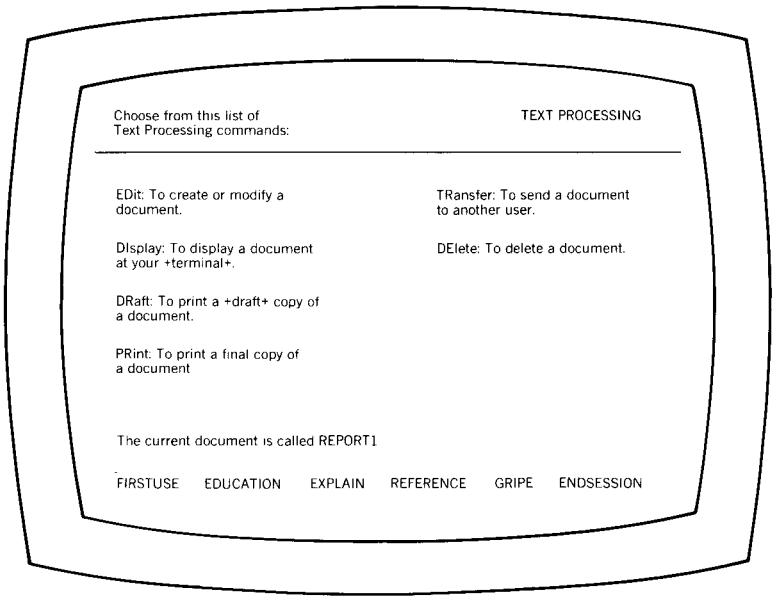
System description

The system is now introduced in much the same way a system coordinator introduces a new user to the system. Discussed first are sessions, their organization and command entry. Discussed next are facilities that assist in understanding and using the system. The description ends with a discussion of expanding and changing the system.

controlling the system

In designing the system, we first studied the intended users, and observe that they are most familiar with objects they work with daily, such as reports, memoranda, documents, and programs. Since users also work on objects (i.e., write memoranda, produce reports, and test programs), we were led to view an application as a set of operations to be performed on a logical object. For each application that can be described as a set of operations on a logical object, a session can be developed. A *session* is a set of commands that allow a user to operate on a class of objects. For example, a set of commands that operate on documents are said to form a text processing session, as shown in Figure 1.

Figure 1 Basic text processing session—menu display panel



Sessions are organized by object class. The specific operations performed on an object depend on the desired result and the end user's experience. Different operations can be made available to accommodate differing levels of user experience. Since operations can be specific, they can minimize the effort necessary to obtain the required results. The command structure bears a logical relationship with the user's experience, a fact that aids comprehension. Users can select sessions by determining whether the session works with the object they want to work with. Thus the system is extensible by adding new operations. New sessions can be added in turn, to operate on new objects. Figure 1 is presented here to illustrate a menu display panel from a simple text processing session. A menu panel displays the available commands (operations) for a class of objects, and the user chooses from that menu. The user does not have to memorize any commands, and the system does not assume the user's next operation. If, for example, a user completes an EDIT operation, the system does not assume DISPLAY to be the next operation. The user controls the system solely by issuing commands.

command format

All applications follow the session command format: operation object—name. The commands and narrative are defined externally, and can use terminology appropriate to the application and familiar to the user. The commands are also consistent across applications. Thus the command PRINT as used in text processing is the same as PRINT used in application development. In this

way, the system builds on consistent recurring elements, and when a user learns to use one application he can move quickly to others.

All menu display panels follow the format shown in Figure 1. The instructions are at the top left and commands are arranged in two half-display columns of 36-character lines rather than in full 80-character lines. For a given point size of type and leading between lines and characters, reading speed and comprehension favor the narrow column.[20] The status message telling the user that the current document is REPORT1, the command input line (as indicated by the cursor), and the service line are all together at the bottom of the display panel. All text is in upper and lower case type because mixed-case text is easier to read than all upper- or lower-case text.[20] The commands, such as EDIT, DISPLAY, and EXPLAIN, are intensified so that an experienced user can easily choose among them without reading all the text.

session objects

The user can enter commands by keyboard, selector pen, or program function keys. On keyboard entry, the command can be truncated to the letters indicated by the upper-case portion of the command. If the user does not specify a document, he is shown a list of documents from which to choose one. Then the system remembers the current document and works with it until another document is specified. The system thus minimizes user interaction in obtaining the required result. For example, to modify and print REPORT1 using CMS, the user has to enter the following commands:

```
EDIT REPORT1 SCRIPT A1

SP PRT CL 2

SCRIPT REPORT1 (TW CO B(5 5) PRO(SSPROF) I SY(H YES X NO)
```

Contrast these commands with the following comparable text processing operation (which remembers the document name and spools the printer):

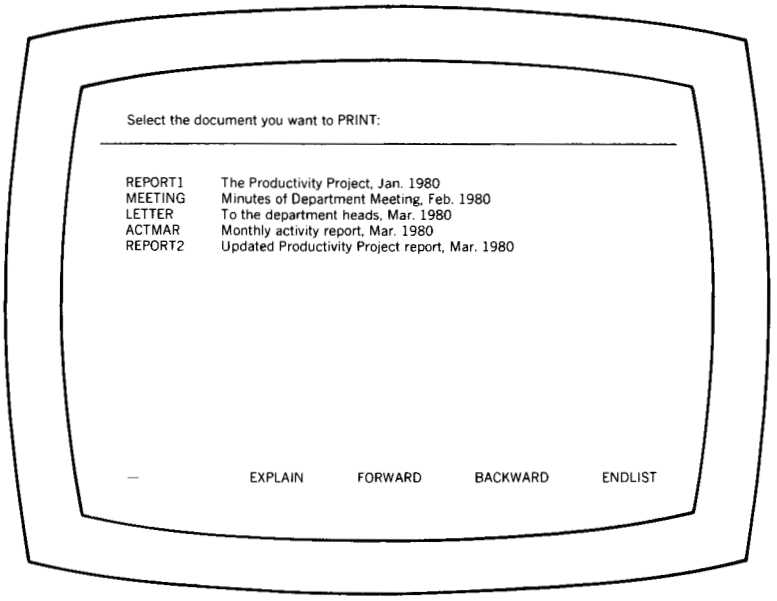
```
ED REPORT1

PR
```

A command with an object name of * (the default) means that the user is not sure of the name of the object he wants to work with. In this case, the system presents him with a list of existing objects and allows him to select one. Figure 2 shows a list of five documents resulting from the command PRINT *. Notice that each object name is followed by the user's description of the object.

The user assigns a name to each object as he creates it. This assignment can only be done by entering a complete command.

Figure 2 Basic text processing session—object selection panel



For example, to create a document called NEWNAME from the panel in Figure 1, the user enters ED NEWNAME. When creating a new object, the session automatically initializes it with a skeleton or model to work from because it is easier to create a new document by changing sample text in a model than it is to create one from memory. The model for text processing is a typical text document. Here the user simply replaces the sample text with his own. The model also reminds the user of Document Composition Facility (DCF) commands that are appropriate to control the format of his document. In more sophisticated sessions, one chooses a skeleton from all available documents. Thus, one can quickly construct a new document by combining and tailoring sections of existing documents.

Understanding the system

A detailed usage analysis of twenty-nine users of a prototype of the system indicated that it was critical that users be successful in their first use of the system.[17] Users who received early guidance and answers to questions continued to use the system; others stopped almost immediately. Those who were able to achieve their application objectives, such as producing a report or document the first time, were much more likely to continue with the system than those who did not. Based on this analysis and our own experience, we found that peers are an effective way of

introducing a new user to the system. The system coordinator gives the new user a brief introduction to the system and the terminal, helps him sign on the first time, shows him how to use the *Terminal User's Reference Guide*, and directs him to the course *Introduction to the System*. The system coordinator also introduces the new user to other members of the user group to whom he can turn when he has questions or problems while learning to use the system. Thus a new user quickly enters an established group and a supportive environment for the critical first use of the system.

A line of service words appears along the bottom of each display panel. Each word in the service line is a selectable command providing on-line access to help facilities. This framework provides direct and immediate access to the specific help. A design alternative would have been to have a monolithic help facility in which the user would be prompted to select the help he wants. Because of a person's short-term memory, however, the more interactions he has before receiving help, the more likely he is to forget what his problem is.[8,18] Also, a monolithic approach would force one through known material, thereby causing frustration and confusion. We have found that the following facilities are the minimum necessary for a complete system.

FIRSTUSE is a narrative session that helps in understanding the application. One normally selects FIRSTUSE for the first use of a session because it discusses features and benefits and gives examples of the object it works with. FIRSTUSE also directs one to available education courses and reference material.

EDUCATION provides the user access to on-line courses appropriate to the application.

REFERENCE provides access to appropriate on-line publications and documents for each application.

EXPLAIN assists a user who does not understand a panel.

DICTIONARY explains complex terms required to understand the system or application. The + symbols around "draft" and "terminal" in Figure 1 indicate that the system dictionary contains definitions of the words. To obtain a definition, the user places the selector light-pen where the word appears in the text or types in +terminals+, for example.

GRIPE aids one when all help facilities have been exhausted or the system requires data processing services. GRIPE allows one to enter a description of the

problem in one's own words and to submit it to a system coordinator. This increases satisfaction and improves report reliability. The problem report is placed in a problem library, where it can be reviewed by the submitter, the system coordinator, and the person responsible for its resolution. The parties involved can add or request additional information about problems, and the entire process can be closely tracked.

ENDSESSION terminates the text processing session.

FIRSTUSEs, EXPLAINs, definitions, and courses are all implemented as narrative sessions containing textual information, examples, and exercises. The user controls narrative sessions as he controls other applications. He can step through a narrative topic at his own pace, and is free to review topics, skip sections, see examples, and try exercises as needed.

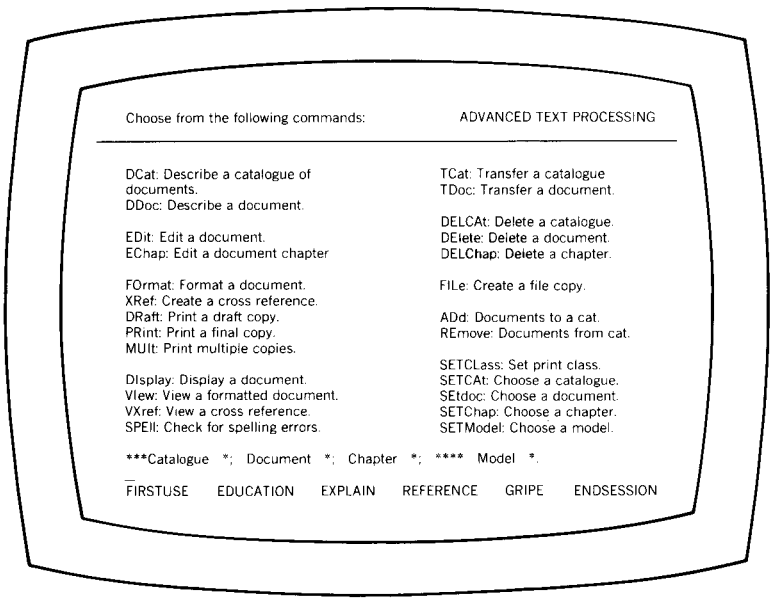
Expanding and changing the system

Another important characteristic is the ability to expand and change the system[1,2,6] both in its functional content and in its use. System developers must listen to the end users and incorporate their requirements and suggestions when expanding or changing the system.[4] An example of changing the way the system is used by individuals is that of stacked commands and procedures. A string of commands can be executed without displaying a panel. Another example is recursion, in which a user can suspend one session and invoke a second without losing his place in the first session. Thus one can respond to a priority request without terminating a current activity. Users can also enter a CMS or CP command from any display panel. The system does not impede the experienced user.

Because the system is independent of the text displayed, sessions have no dependency on display panel format or national language. Therefore, one can change a command or message to something that is meaningful. All session commands, display panel content, and message text are stored in external files. The isolation provided by CMS minidisks allows users to change display panels, commands, and messages without affecting other users. This requires that individuals keep private copies of parts of the system; it also complicates maintenance when fixing errors and migrating to new versions of the system.

End users can use the Dynamic Development Facility (DDF) to make simple changes to display panel and message wording. Developers can use the facility to modify existing sessions, create

Figure 3 Advanced text processing session—menu display panel



new ones, and design all types of on-line applications. The Dynamic Development Facility can also be used to create prototypes or mock-ups of applications and to test these applications.

In our experience, users typically start with very simple applications, such as the one shown in Figure 1. They quickly outgrow these applications and create a family of sessions, each having a richer set of operations than the previous one, but still working on the same object. An advanced text processing session involving a rich set of operations is shown in Figure 3. Another interesting observation is that the more experienced users demand that all commands of a session be displayed on one panel. That is, a single menu is preferred to a multiple-panel hierarchy.

Another consideration is that of starting the novice with a session that relates to his skill and understanding. The basic text processing session shown in Figure 1 is suitable for this purpose, whereas the advanced text processing session shown in Figure 3 might be discouraging to the beginner.

Concluding remarks

The architecture of the system discussed in this paper is based on sessions and objects. A way to extend the system is to add new

objects. The challenge is to correctly identify the object the user wants to work with. Once this is done, the operations follow quite easily. At first, it is often obvious neither to the system designer nor to the user what the object really is. Another concern is to provide different versions of a session and object for different levels of user experience so that the user moves to more advanced sessions as he gains experience. This must be done while maintaining the display panel consistency and dynamic interaction consistency of the base system.

Development of an end user system is iterative. Users must be involved at all stages, and the system designer must continuously seek suggestions and reactions from those who use the system. We have found that a good way to begin is by creating application prototypes and obtaining comments on them from users.

Organizing sessions around objects rather than commands focuses the user on the work product. Models serving as templates can directly improve productivity because recognition is easier than recall. Very often, new reports, memoranda, or programs can be created by combining and modifying existing information. This is more productive than attempting to create without reference to existing information. The most critical time for a novice is his first encounter with the system. Peers in the user group play an important role in system acceptance and in developing patterns of successful use.

The system improves the man-machine interface by providing a structure for business professional applications. Whereas previously these users provided an implicit structure through their knowledge of CMS commands and their application, we have defined and specified one that suits many needs and provides significant advantages. The architecture of the system centers around objects that are derived from daily experience and work. Sessions bring together commands that manipulate an object class. Thus users, through sessions, can perform their work. Finally, the system recognizes that everyone is an individual and that the suitability of the system can be determined only by direct experience. The system provides the tools to change and personalize the system to each user's unique need.

ACKNOWLEDGMENTS

The success of the system is in no small part due to the hard work, dedication, and innovative thinking of the development team: Clancy Marchak, Roger Tessier, Susan Williams, Robert Dunn, and Martin Marchyshyn. I would also like to thank Clancy Marchak for his invaluable assistance in preparing this paper.

CITED REFERENCES

1. *Data Processing in 1980–1985*, John Wiley & Sons, Inc., New York (1976).

2. E. B. James, "The user interface," *Computer Journal* **23,** No. 1, 22–28 (1980).
3. L. A. Miller and J. C. Thomas, Jr., "Behavioral issues in the use of interactive systems," *International Journal of Man-Machine Studies* **9,** 509–536 (1979).
4. B. Shneiderman, "Human factors experiments in designing interactive systems," *Computer* **12,** No. 12, 9–19 (December 1979).
5. J. Martin, *Design of Man-Computer Dialogues*, Prentice-Hall, Inc., Englewood Cliffs, NJ (1973).
6. J. L. Bennett, "The user interface in interactive systems," *Annual Review of Information Science and Technology* **7,** C. Cuadra (Editor), 159–196 (1972).
7. W. Dzida, S. Herda, and W. D. Itzfeldt, "User-perceived quality of interactive systems," *IEEE Transactions on Software Engineering* **SE-4,** No. 4, 270–276 (1978).
8. S. E. Engel and R. E. Granda, *Guidelines for Man/Display Interfaces*, IBM Poughkeepsie Technical Report TR 00.2720 (December 1975).
9. R. B. Miller, *Human Ease of Use Criteria and Their Tradeoffs*, IBM Poughkeepsie Technical Report TR 00.2185 (April 1971).
10. *A Programming Language (APL)*, IBM General Information Manual, order number GH20-9064, available through IBM branch offices.
11. *Document Composition Facility*, IBM General Information Manual, order number GH20-9158, available through IBM branch offices.
12. *A Departmental Reporting System (ADRS)*, IBM User's Guide, order number SH20-2165, available through IBM branch offices.
13. *Query-By-Example (QBE)*, IBM Program Description and Operations Manual, order number SH20-2077, available through IBM branch offices.
14. F. P. Brooks, Jr., "The computer 'scientist' as a toolsmith—studies in interactive computer graphics," *IFIP Conference Proceedings*, North-Holland Publishing Company, Amsterdam (1977), pp. 625–634.
15. *Interactive Extension Facilities: Introduction*, order number GB09-4001, available through IBM branch offices.
16. *Interactive Extension Facilities: System Designer's Guide*, order number SB09-4002, available through IBM branch offices.
17. Private communication from John G. MacKay, June H. Ziola, and Victor G. Zachovay, IBM Data Processing Division.
18. J. D. Foley and V. L. Wallace, "The art of natural graphic man-machine conversation," *Proceedings of the IEEE* **62,** No. 4, 462–467 (April 1974).
19. W. J. Doherty and R. P. Kelisky, "Managing VM/CMS systems for user effectiveness," *IBM Systems Journal* **18,** No. 1, 143–163 (1979).
20. C. B. Avery and L. Pelstring (Editors), *Words Into Type, Third Edition*, Prentice-Hall, Inc., Englewood Cliffs, NJ (1974).

GENERAL REFERENCES

M. E. Atwood, H. R. Ramsey, J. N. Hooper, and D. A. Kullas, *Annotated Bibliography on Human Factors in Software Development*, U.S. Army Research Institute for the Behavioral and Social Sciences, Alexandria, VA (June 1979).

IBM Virtual Machine Facility/370: Introduction, order number GC20-1800, available through IBM branch offices.

L. H. Seawright and R. A. MacKinnon, "VM/370—a study of multiplicity and usefulness," *IBM Systems Journal* **18,** No. 1, 4–17 (1979).

Display Input/Output Facility, IBM Program Description and Operations Manual, order number SB11-5329, available through IBM branch offices.

The author is located at the IBM Canada Limited Laboratory, 1150 Eglinton Avenue East, Don Mills, Ontario M3C 1H7, Canada.

Reprint Order No. G321-5150.