USING THE COMPUTER TO COMMUNICATE: CREATING A COMPUTERIZED CONFERENCING SYSTEM ON UNIX

Michael M.L. Pearson
James E. Kulp

November 1980
WP-80-159

INTERNATIONAL INSTITUTE FOR APPLIED SYSTEMS ANALYSIS

PREFACE

This paper was prepared for the International Symposium on Computer Message Systems, Ottawa, Canada, April 6-8 1981. It is one of a series produced by IIASA's Survey Project in cooperation with the Computer Services Department on the topic of using the computer to communicate. Our aim is to make the IIASA computer more accessible to IIASA scientists for a wide range of people-oriented activities: computer-based text editing and text formatting, decision support systems, computerized conferencing, multi-author manuscript preparation, and other operations promoting international team research among widely dispersed researchers.

For readers unfamiliar with IIASA and the Survey Project we offer the following two paragraphs for a better understanding of the institutional context of this paper.

The **International Institute for Applied Systems Analysis (IIASA)** is an interdisciplinary, nongovernmental research institute, chartered in October, 1972 upon the initiative of the academies of science or equivalent institutions of twelve nations (there are now seventeen National Member Organizations). By applying systems analysis, its staff of scientists from East and West is seeking to gain a better understanding of important contemporary problems resulting from scientific and technological development. IIASA conducts much of its research in cooperation with other research and policymaking organizations worldwide.

The IIASA **Survey Project**—a project to survey the state of the art of applied systems analysis—was established to promote the development of applied systems analysis and to disseminate its methods and approaches. The Project seeks to encourage the widespread and better application of systems analysis to problems of international relevance; to improve analytical techniques and their usefulness to decision processes; to contribute to the education in systems analysis of the expert and the interested nonexpert. To pursue these objectives it is: 1) publishing a series of books on applied systems analysis, 2) writing a *Handbook of Applied Systems Analysis* in three volumes, and 3) conducting research into the craft of applied systems analysis.

USING THE COMPUTER TO COMMUNICATE: CREATING A COMPUTERIZED CONFERENCING SYSTEM ON UNIX

Michael M.L. Pearson, and James E. Kulp

INTRODUCTION

This paper discusses a computerized conferencing system being developed at our institute. It does not describe the system. For a description we refer the reader elsewhere (see Pearson and Lathrop, 1980). The system, which we call Telecenter, is similar to others of its genre; using it one can create conferences, add participants, enter new conference comments, modify old comments, learn who has seen which comments in any given conference, get an overview of one's own status in all one's own conferences, and so forth.

Telecenter's current form, however, is not the topic of this paper. What we wish to address here is how and why Telecenter was developed. The "how" has to do with our UNIX* operating system and is the subject of Part Two. The "why" has to do with our users and their needs and is addressed in Part One. Regarding both users and UNIX, we have tried to conform to certain general principles recognized by others involved in designing and implementing computerized conferencing systems, office automation and decision support systems. One result, we believe, is that we have made considerable strides with surprisingly few resources. For example, Telecenter itself was created with approximately one man-week's programming effort. This is noteworthy considering the usual requirements for developing such a system. Murray Turoff, a pioneer in the field of computerized conferencing writes in *The Network Nation:*

*UNIX is a trademark of Bell Laboratories.

> At the moment, most of the (computerized conferencing) systems that have been implemented represent many person-years of programming effort. There is no suitable higher-level language that allows a concise and appropriate specification of a human communication process. All current efforts at using FORTRAN, APL, BASIC, etc. are like describing a picture of the Mona Lisa in words only... Currently we are beginning to understand the basic functions that characterize conferencing systems and it can be expected that a computer language will evolve that can allow these systems to be created in a few person-months of effort (Hiltz and Turoff, p. 391).

The kind of higher-level language for human communication processes that Turoff speculates about has yet to be invented.* Relevant to this statement, however, is the fact that software tools used to construct conferencing systems a few years ago are inferior to tools developed more recently. That Telecenter was made operational so quickly is primarily the result of its having been fashioned under an interactive and comparatively state-of-the-art time-sharing operating system, but it is also partly the result of our attitude toward user needs.

USERS' NEEDS: THE "WHY" OF TELECENTER

Since the research project** where one of the authors works is primarily a publishing activity, his first involvement with computerized conferencing was in the specialized form he terms "computerized manuscript conferencing," the use of computer conferencing mechanisms to facilitate the joint authorship of manuscripts (see Pearson, 1980a). Presently his interests have broadened to include any application of computerized-conferencing-type tools to facilitate applied systems analysis and institute research, what he, for want of a better term, calls "using the computer to communicate" (see Pearson, 1980a, Pearson and Lathrop 1980 and Pearson 1980b). Manuscript conferencing remains central to the project's teleconferencing interests since a large percentage of the institute's research involves writing, but it is now a part of a more general activity whose goal is to explore the possibilities of a new kind of

---

*The closest thing in spirit to it with which we are familiar is a proposal by Richard Miller to use Backus Normal Form (BNF) representations (the formalism originally created to describe ALGOL) for functional descriptions of an electronic message system (EMS). Miller stresses the need for a formal description, or at least some consistent set of definitions, of the functions and interactions of electronic message systems, and sees the BNF-type formalism as a starting point for a "common language in which various (electronic message) systems can be conceived, characterized, compared and evaluated." He goes on to say, "A formal representation of the user environment could benefit the EMS field to the same extent that compiler theory 20 years ago made an impact on the rapid development of high-level computer languages."

**The Survey Project, a part of the institute's General Research program (see Preface).

computing/communications environment–one which should be able to significantly expedite scientific work among institute staff separated in time and/or space, and at the same time be accessible to the average researcher.

It is our belief that the best way to explore "using the computer to communicate" for scientific research at our institute is to facilitate it for the applications of individual scientists. Thus as a tool-environment, Telecenter offers a set of potentials that we do not fully understand and will not fully understand until we involve *users* and their own potentials and see how matters evolve. It is these users, the IIASA scientists and their applications, that we must serve in order to make this medium serve applied systems analysis.

Under no circumstances do we want to create an elaborate piece of machinery on the basis of our, undoubtedly premature, perceptions as to what is best for institute staff. Hence the need for modules, adaptable functions, that we can offer as preliminary tools to individuals for whom it is not fully clear at the outset in which direction their activity will take them.

We agree entirely with what Uhlig, Farber and Bair say regarding the implementation of computer systems for the "Office of the Future" and think it especially applicable to systems such as Telecenter:

> Too often systems are built to be used by and for their designers–not by the ultimate users. We can see this in many current systems. Also we tend to assume that the world will stop and wait for our systems to be fixed and that the users will be tolerant of loss of information. Many of the systems of the future will be used not by computer people but by executives, by the public, and by policy makers. (Uhlig, *et al.*, p. 153)

It should be stressed that we are by no means certain that a computerized conferencing system offers the best solutions to the problems of facilitating joint research electronically. Our primary objective is facilitating research, not creating a conferencing system. It may be a mistake, for example, to try and enforce the structured notion of conferences–topic oriented discussions–on the normally fluid communications habits of most researchers. Or it may be futile only in some cases, not in others, depending on the individuals involved and the task at hand. It is most likely that we need an environment that integrates conferencing, computer-mail functions and other computer-based tools–text processing, database use, computational routines, and so forth. The important thing is to have a facility up and running that can be adapted to user needs.

Thus, Telecenter is not primarily the kernel of a teleconferencing system, but, rather, a set of modifiable functions that can serve as a starting point in searching for a suitable environment for "using the computer to communicate". It is important that these functions can be packaged in a single, consistent, easy-to-use system such as Telecenter but

even more important that they exist as modules that can be modified and combined in different ways for different purposes. Of all the computer-based, person-to-person communications tools we are familiar with, computer conferencing seems the most accessible to people unfamiliar with computers.* It is for this reason we began with a general computer conferencing schema.

SERVICE AS ANOTHER WORD FOR AVAILABILITY

Our point of departure in exploring the potential of "using the computer to communicate" at our institute has been the user. Similar kinds of attitudes have emerged from decision support system (DSS) research. Consider Gambino and Keen's admonitions to learn the language of the group you are supporting, and develop command functions for it, "build a capability rather than a product"; support first and extend later (as inexpensively as possible); select innovative, creative users with something at stake; support the person rather than solve the problem or build a model; encourage feedback and respond to user ideas and requests (see Gambino).

And it is our experience that, providing the user is motivated and has an important job to do, the single most important factor influencing the success or failure of an activity is *availability*. This concept runs as a common thread through all phases of implementing the computer/communications environment mentioned above–from the simple existence of Telecenter (before it we had nothing we could use locally) to documentation of facilities, guaranteeing acceptable response time (the single biggest problem to date in promoting use of the system; other imperfections in the system have been insignificant factors compared to this) and access to terminals.

A corollary to availability is *adaptability*. If a user needs a certain feature available in order to do the job at hand better, Telecenter must provide it–adapt to the user's need. This fact, even more than the issue of cost, precluded our turning to already existing conferencing systems. Most were not portable: they were written in impossible-to-modify assembler language or in large, unwieldy programs in higher-level programming languages that were unsatisfactory. Some constituted whole systems–worlds unto themselves–and none was suitable for integration into our institute's computing environment. We needed something we could quickly implement, understand and adapt.

---

*Turoff's EIES system, in particular, has demonstrated how with forgiving software and conscientious on-line consultancy services a system of this kind can acquire a large constituency, including a large number of people who have never used a computer before (see Hiltz and Turoff). The commercial computerized conferencing service, Infomedia, has also achieved success in this area. Infomedia's approach has been to offer very easy-to-use software to its customers (see *Planet News*, its monthly newsletter).

An individual will only use a computer system such as Telecenter when it is preferable to competing tools such as pencil, telephone, letters, telex, assistants, and so on. In DSS terminology, the system is discretionary--it is at the discretion of the user/decision maker whether or not to employ it (see Sprague).

We can learn a number of lessons from DSS designers and implementors. Telecenter-type functions could, we think, evolve into a communications tool with significant DSS components, although at present it is not a DSS; its primary purpose is not offering models and databases to help managers make decisions. Nevertheless, Telecenter's relation to its users is analogous to that of a DSS to its users. The DSS notion of adaptive design, for example, is applicable to our approach. As Sprague says, it involves:

(1) Identifying a critical subproblem

(2) Building an initial system

(3) Using and evaluating

(4) Adding and deleting capabilities

(5) Repeating 3 & 4

Similarly, as far as Telecenter's development is concerned, the concept of a DSS generator, and that of DSS tools, both have close parallels with, respectively, the script mechanism of the UNIX command interpretor (see below) and the UNIX operating systems itself. In DSS terminology, the DSS generator is the vehicle that permits the builder of a specific DSS do his job quickly and efficiently without having to have the computer expertise of a systems programmer. Thanks to the DSS generator, the builder can devote himself exclusively to decision tasks at hand--analyzing applications issues and responding to them. "DSS tools" are the products of the systems programmer (the "toolsmith") that permit the existence of the DSS-generator facility.

An important issue here is willingness to trade off responsiveness to user needs against polish in final design. Instead of dedicating all resources to the creation of a perfect, fully designed system, the approach is to devote, say, fifty percent of one's resources to initial system design and implementation then devote the remaining fifty percent to a labor intensive service activity that modifies the system while simultaneously working with end users. Thus we have found that a large component of research into the usefulness of this new medium is service--the promotion of availability in its broadest sense. Implementators must have as their primary perspective, and responsibility, both facilitating and experimenting with the use of such tools to assist researchers. The luxury of a flexible and adaptable system does not spare one the expense of human time spent directly assisting users during many sessions.

USING UNIX TOOLS: THE "HOW" OF TELECENTER

After pointing out the significance of declining costs of hardware in the microelectronics field for forthcoming developments in "Office of the Future" technology, Uhlig, Farber and Bair go on to say:

> However in order to effectively utilize all this cheap logic we must reduce the cost of the software. We must create simpler systems that can be built reliably with small software investments. Some of the things that contribute to declining costs are utilizing single-task-machine approaches. This vastly simplifies the cost of software. In addition having simple interfaces simplifies the overall task. Furthermore, the adoption of simpler program structures will vastly ease the task of the programmer. The use of highly modular code may increase the use of cycles (which are cheap) but will greatly reduce the cost of software complexity. The use of structured coding technology will also contribute to the reduction of the cost of programming. But perhaps the greatest contribution to reducing software costs in the Office of the Future will be to create a better software development environment, i.e., more and better tools presented in a fashion that makes them always available to the programmers. (Uhlig, *et al.*, p. 173)

As far as the creation of Telecenter is concerned, we believe that UNIX offered one such "better software development environment" and that four of its features deserve mention in this regard:

(1) its *utility programs* and the philosophy behind them,

(2) its *command interpreter*,

(3) *transparency* between functions implemented at different levels of software, and

(4) its hierarchical *file system*.

Utilities

Although he is focussing primarily on tools for making compilers (for example, a lexical analyzer and a parser) Stephen C. Johnson in *Language Development Tools on the UNIX System* makes points about the UNIX philosophy of software tools relevant to the building of Telecenter. We feel these points are of interest here, first, because Telecenter incorporates UNIX utilities and, second, because it has a highly modular design.

> Over the years, a number of general utility programs have grown up on the UNIX system. Many of these tools may be viewed as an encapsulation of a piece of specialized knowledge–anything from driving a piece of hardware to performing a utility task.

> They may be as simple as a *sin* subroutine, or as complex as a disk driver. By using tools, programmers are able to gain access to facilities and algorithms that they do not have to understand in detail. Not surprisingly, constructing programs by using available tools and pieces has proved to be very successful:
>
> - The resulting products are produced quickly.
> - They are likely to work correctly.
> - They are often quite flexible and adaptable as applications change.
> - Inter-machine portability is often enhanced by using tools.
> - Tool usage encourages a natural modularity in the resulting program.
> - Using tools constructed by experts, programmers get the use of expert algorithms.
>
> (Johnson, p. 16)

UNIX has many utility programs written to do simple things to text files, like printing lines that contain a specified pattern, eliminating duplicate lines in a sorted file, or printing the first or last few lines of a file. All these simple "tools" are designed to be flexibly combined at the command level to perform more complex tasks, and to use them in this way requires no programming and little effort.

Most of the facilities for doing Telecenter's basic tasks already exist under UNIX in the form of utilities–for example, a simple screen-oriented editor program* *edx* for creating and modifying text, a utility called *grep* for searching files for strings of interest, the utility *date* for inserting the date and time into conference comment headers, notifications of login and logout times, and so forth. They also structure information (see the paragraph below on file system).

Command interpreter

The command interpreter–called the "shell" in UNIX–is the primary mechanism through which time-sharing users interact with UNIX. A shell is a normal, un-priveleged user program that typically reads commands typed by the user at his or her terminal and executes functions either through code compiled into the shell itself, by running other programs, or by interpreting text files containing shell commands. The shell used to implement Telecenter is called the C shell and is the shell used by all UNIX users at our institute (see Joy). For implementing Telecenter, the

*For users with hardcopy terminals a line-oriented editor is optional.

C shell was used as a very high level, block structured, interpretive programming language which provided convenient access to built-in functions, utilities and the various programming features of the C shell itself, including variables, simple numeric expressions, pattern matching, macros and file system manipulation.

Except for one or two trivial compiled programs, all aspects of Telecenter are small (ideally thirty-five to seventy lines of code each) C-shell "programs" (sometimes called scripts) that are easy to write and modify. Since the basic elements of this "programming language" are normal interactive commands, sometimes parametarized and/or conditionalized, powerful functions using many utility programs combined in various ways can be implemented by users with little or no programming experience. Furthermore, since this language is interpretive, it is easy to debug and experiment with programs, or parts of them, merely by typing them interactively at the terminal.

Transparency

Since there is great consistency between built-in functions, compiled programs and C-shell scripts, all can be combined within C-shell programs or made directly available to the user without knowing or caring how a given function is implemented. This allows modules to be easily replaced by more efficient implementations for efficiency reasons once the required functionality is well-known, presumably determined through a flexible, very high level, interpretive "first attempt".

In a first stage of development, where we are now, appropriate functionality can be explored by user-oriented people able to modify shell-scripts. In a second stage, efficiency can be achieved for those functions deemed desirable by having shell-scripts partially or wholly converted to either compiled programs or functions built into the shell. In the meantime all components of the system are compatible with one another--all modules interconnect. We are confident about the second stage of Telecenter's evolution because UNIX supports a modern, structured programming language called "C" (see Kernighan and Ritchie) that is superior to FORTRAN and other earlier programming languages and offers the kind of access to the operating system that has been shown to be essential for developing these kinds of systems.*

---

*In a review of both hardware and software requirements for implementing computerized conferencing systems, Hiltz and Turoff have pointed out, "The current trend toward insulating the higher-level languages from the operating system needs to be reversed, so that the conference program in that language can control the operating systems. This means that the status and control of interrupts and I/O have to be incorporated at the higher level..." and that "Most conference packages today have to make frequent transitions into the domain of the operating system command language. All user I/O (break key, etc.) has to be removed from the operating system to the conference system so there is no way for the user to get at the operating system." (Hiltz and Turoff, pp. 390-1)

File system

The structuring of information in Telecenter was facilitated by the fact that UNIX has a simple yet powerful file/directory system that supports the organization of files into an arbitrarily deep hierarchy. Every directory can contain files and/or more directories. A directory entry for a file (called a link) can be placed in more than one directory, thus allowing a file to appear in many directories at once.

Organizing Telecenter users, conferences and conference participants was done simply by writing C-shell scripts that manipulate the appropriate directories–a directory hierarchy for users and a directory hierarchy for conferences. Thus when a participant submits a conference comment he or she does so by calling a function that creates and links files in appropriate directories. The Telecenter functions that do these things incorporate standard UNIX utilities for making, linking, moving among, and removing directories as well as those for making, linking, editing and removing files. For example, keeping track of who has seen what is simply a matter of establishing links for each participant to a single comment file, removing a link when someone has viewed a conference comment and then, for any given conference, counting who still has links and how many.

In effect, UNIX's directory/file structure provides a simple database management system for implementors and users of Telecenter.

CONCLUSION

The phrase "think globally, act locally" was the theme of a recent world futures conference in Canada. It seems to us that these words summarize well what we feel to be of value in the design approach to the UNIX-based computer conferencing system described above. By responding to the needs of actual users, we think that we have created a useful collection of communications functions. By creating them on the technical basis of a standard, widely available and modern operating system such as UNIX, we hope that our approach will be of use to a wider community as a small example of adaptive design in an experimental but useful situation.

REFERENCES

Gambino, T.J. (1980) Implementation Research and DSS. Paper delivered by Gambino on work done together with Peter Keen. Delivered at the 5th Annual Information Systems Summer Workshop held in Williamstown, Mass., U.S.A., whose theme was Strategies for Supporting Management of the Future. (Quoted material on the basis of notes prepared by Alain Barbarie, Management and Technology Area, IIASA.)

Hiltz, S. R., and M. Turoff (1978) *The Network Nation.* Addison-Wesley.

Johnson, Stephen C. (1980) Language Development Tools on the UNIX System. *Computer.* Vol. 13, No. 8, August, pp. 16-21.

Joy, William (1980) An introduction to the C shell. Computer Science Division, Department of Electrical Engineering and Computer Science, University of California, Berkeley, Berkeley, California 94720.

Kernighan, B. W. and D. M. Ritchie (1978) *The C Programming Language.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Miller, Richard (1980) Representing the Message System User Environment: A Report to the User Environment Sub-Group, IFIP Working Group 6.5. Contribution to International Federation for Information Processing workshop on computer message systems held May, 1980 at the Gesellschaft fuer Mathematik und Datenverarbeitung in Bonn, FRG, and organized by IFIP Working Group

6.5 on international computer message systems. Will be part of a forthcoming workshop proceedings.

Pearson, M.M.L. (1980a) *Using the Computer to Communicate: An Introduction to Computerized Conferencing* WP-80-72. Laxenburg, Austria: International Institute for Applied Systems Analysis.

Pearson, M.M.L. (1980b) *Using the Computer to Communicate: An Introduction to Text Processing at IIASA–the edx and nroff programs* WP-80-111. Laxenburg, Austria: International Institute for Applied Systems Analysis.

Pearson, M.M.L. and C.L. Lathrop (1980) *Using the Computer to Communicate: A User's Guide to Telecenter* WP-80-109. Laxenburg, Austria: International Institute for Applied Systems Analysis.

Sprague, Ralph (1980) An Overview of Decision Support Systems. Contribution to Task Force meeting on decision support systems held at the International Institute for Applied Systems Analysis (IIASA), Schloss Laxenburg, Laxenburg, Austria on June 23-25, 1980. To be published as part of a IIASA Conference Proceedings.

Uhlig, R.P., D.J. Farber, and J.H. Bair (1979) *The Office of the Future* North-Holland Publishing Company.