Interactive Computer Support of Geographically Separated Design Teams

David J. Perreault and John G. Kassakian

Laboratory for Electromagnetic and Electronic Systems
Massachusetts Institute of Technology
MIT Rm. 10-050, Cambridge, MA 02139, USA

Abstract

A prototype shell system is described which gives single-user applications multi-user interfaces by linking workstations together. The system is designed to be independent of any particular application program, and to operate over low-bandwidth communications links. The limitations of existing techniques are overcome through use of a parallel processing approach. To assess the potential of the approach, a preliminary evaluation of the resulting user interface is made.

Introduction

The recent explosion in the availibility of computers has heightened interest in computer-based tools which support cooperative work. Computer support of group collaboration may have the most utility in geographically distributed workgroups, where face-to-face communications are impractical [1]. It is hoped that the development of computer tools which support remote interaction will lead to improved efficiency and better cooperation among distributed workgroups.

This paper describes a prototype system which enables collaborators to interact within the environment of an existing application program, while working on separate workstations. The system is designed to be independent of any particular application program, and to operate over low-bandwidth communications links, regardless of the amount of graphics output or database changes produced by the program. This is achieved by controlling the execution of the application program at each workstation such that the programs behave identically.

The first section of this paper presents the motivation for the development of such a system. The existing approach to this type of problem is then presented, and its limitations are discussed. The third section presents an overview of the approach taken for the prototype system. A preliminary evaluation of the approach is then presented, based on experiences with the prototype system. Finally, conclusions are drawn, and suggestions for future research in the area are made.

Motivation

Recent research in the area of Computer-Supported Cooperative Work (CSCW) has examined the needs of teams of collaborators working together. A study of information exchange within engineering design teams indicated that graphics capabilities and fully synchronous communications would probably be necessary for an engineering collaboration tool to be effective [2]. Similarly, researchers studying the workspace activity of design teams "observed two key features that the designers utilized in developing ideas: a) being able to readily try out representations of ideas in the public workspace, and b) having those representations gradually evolve into distinct artifacts, often through other participants building on and modifying them." [3]. In the same vein, the importance of shared drawing surfaces for developing and representing ideas was established in [4]. What may be learned from these studies is that the ability to interact synchronously in a shared workspace and the ability to easily represent ideas are desireable attributes of a remote collaboration tool.

Single-user workspaces for representing and developing ideas are a fundamental part of many applications programs. A rich supply of single-user programs for tasks such as Computer-Aided Design (CAD), financial analysis, and document editing has developed in recent years. These programs provide graphical workspace environments optimized for developing and representing ideas in *task-specific* formats. However, because of their single-user orientation, these existing tools have not been fully exploited as part of the collaborative process.

In light of this information, we propose a 'shell' system, which gives existing single-user application programs a multi-user interface by linking two work-

0-8186-2760-3/92 $03.00

stations together. When the shell is running, each station shows the same data, and any action taken on one station happens at all of them. This is a variation of the WYSIWIS ("What You See Is What I See" - pronounced whizzy-whiz) interface described in [5],[6].

A software shell which allows users to interact in the environment of a task-specific program has several benefits. First, it provides the shared workspace and synchronous interaction desired in a remote collaboration tool, with the shared workspace inherently optimized for a specific task. Second, it allows the enormous base of existing single-user application software to be applied in a collaborative setting, without redeveloping each program for a multi-user environment. Furthermore, a shell-based system permits users to maintain the same software standard, eliminating problems inherent in adopting different data formats. Also, users can take advantage of their knowlege of established single-user programs, instead of having to learn a new tool exclusively for collaboration. Finally, it has been pointed out that "task-oriented tools designed to facilitate the completion and integration of specific work products" are one class of tools necessary to support the needs of distributed work teams [7]. A shell program designed to be used with existing task-oriented software may help fill this niche in the CSCW arena more rapidly than otherwise possible.

Background

CSCW software can be classified by whether it is designed for local or distant use, and also by whether it is designed to be used synchronously or asynchronously [8],[9]. As can be seen from [8] and [10], much of the work in computer-supported cooperative work has been devoted to either wide-area asynchronous communication tools, or local-area synchronous tools. Some wide-area synchronous tools do exist, including a PC-based graphics system produced by Optel Communications, Inc., and a conferencing/editing tool by Group Technologies, Inc. (both of which allow operation over telephone links). However, construction of such systems requires ground-up development of each interactive environment, implying that there will be no *immediate* proliferation of task-oriented tools of this nature. A background search has been conducted for experimental or commercial software which could be used to implement a shell system, and only one related type of software has been found. This class of software, availible from a number of manufacturers, is used for remote control of a PC system. As shown in Fig. 1, it functions by running the application program on one system, and using the remote system as a dummy. All video information is shipped from the application system to the dummy, and keyboard input from both systems is used by the application system. This attempts to give the remote user the illusion of working on the local machine in tandem with the local user. However, as discussed in [11] and [12], currently availible software of this type is inadequate for graphics-based applications such as CAD. Furthermore, this approach is not generally feasible for graphics-based applications. This is because the amount of graphics data involved requires excessive time lags with the bandwidth availible (over standard modem connections). This limitation is severe, since graphical representation of ideas is an important attribute of effective collaboration, as previously discussed. Thus, no software suitable for constructing a shell has been found, and the currently existing approach to this type of problem is inadequate.

Approach

The approach taken for developing a prototype system, the RESCOL (*Re*mote *S*ynchronous *Col*laboration) shell program, is geared towards the low end PC/AT environment under the DOS operating system. The PC type system has been selected due to its widespread use and low cost. Furthermore, the RESCOL shell system is designed to operate over low bandwidth communications links such as modems, allowing remote interaction at a minimum of expense and hardware requirements. It should also be noted that the shell program works best in combination with direct audio links (e.g. telephone). The results of [1] and [7], as well as personal experience with the prototype system suggest the benefits of these additional links.

To handle the large amounts of graphics data necessary, the RESCOL system uses a parallel processing approach. As shown in Fig. 2, each workstation runs the application software independently, with user input at each system delivered globally. All graphics and database changes are thus generated locally, reducing the required data transmissions to an acceptible level for low bandwidth communications. Control is handled in a master/slave format, which allows the system to be expanded to an arbitrary number of workstations. Using this approach, the prototype system runs quite effectively over point-to-point modem connections at 2400 baud.

The prototype RESCOL shell is a Terminate and Stay Resident (TSR) program which performs two major tasks. First, the software captures and distributes the user input at each workstation over the entire RESCOL network. This common input connection is what allows the users to interact in a WYSIWIS manner. Input from both keyboards and pointing devices (such as mice) are handled by the RESCOL system. As a second task, the software monitors the workstations to ensure that they are actually

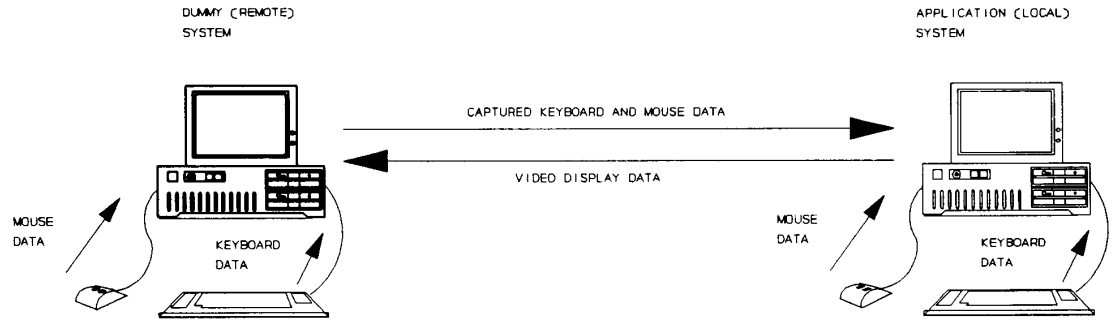

Figure 1 Operation of remote takeover software. One workstation acts as a terminal, while the other handles the computing. Transmission of video data is the limiting factor for graphics based applications.

working in parallel. Such a mechanism is necessary to prevent the data on the different systems from diverging. This task is accomplished by regulating the environment in which the application program executes, as developed in [13]. For example, one technique used for doing this is observing and controlling software access to external devices and files, to ensure that each program receives identical information. Thus, by managing the execution of programs on multiple workstations, the RESCOL system allows people in different locations to work synchronously and interactively via low bandwidth communications links.

Evaluation

The need for computer-based tools which support remote collaboration has motivated the development of the RESCOL 'shell' approach. In spite of the known difficulties in appraising such multi-user interfaces [14], a preliminary evaluation of the shell approach has been made using the prototype system as a test vehicle. The goal of this preliminary evaluation was to gain a basic understanding of the benefits and liabilities inherent in a shell-based interactive environment.

Evaluation Methodology

The exploratory study consisted of four interactive sessions between pairs of collaborators, each lasting no longer than one hour, and each separated by about a week. The study was set up such that the collaborators were colleagues who had worked together previously, and who were each familiar with the tasks at hand.

The four sessions were conducted over 1200 baud serial links, with the addition of two-way audio communication. Each collaboration session was conducted in the environment of a different commercially available software package, with which at least one of the users was experienced.

The first collaboration session involved joint editing and review of a partially completed scientific document. The session was carried out in the environment of a word processing package with WYSIWYG and graphics capabilities. While both collaborators were familiar with the word processor, only one of them was versed in its equation editing tools, providing an opportunity to examine the instructional potential of the shell-based system.

The second collaboration session involved editing and review of technical illustrations in the environment of a popular spreadsheet / graphics package. This session provided a good opportunity to see the performance of a shell-based system in a situation where a shared view of graphical images was central to the task at hand.

The third collaboration session was carried out in the environment of a popular CAD package, with the intent of performing preliminary "brainstorming" for a graphic design. No predefined data were brought into the design session, except for a vague verbal specification of the desired result. The object of this session was to examine the usefulness of a shell-based system in situations where the two collaborators were attempting to initiate a project, rather than extend existing work.

The final collaboration session involved review and discussion of a partially completed circuit design, in the environment of a widely used electrical CAD package. This session provided an opportunity to examine the potential of a shell based system in an engineering design context, where graphical communication is often important.

General Observations

The four formal collaboration sessions, in conjunction with more casual use of the prototype system, yielded a

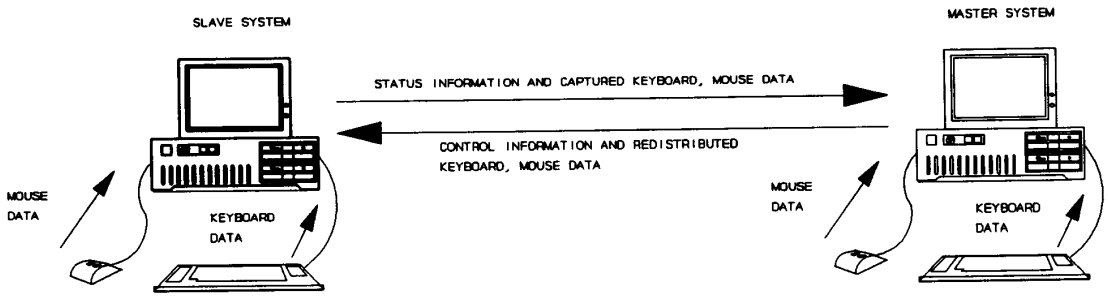

Figure 2 Operation of a RESCOL system based on parallel execution. Since each workstation generates its own video and database changes, the system can function with graphics-based applications.

wealth of information about shell-based collaboration and its potential advantages and liabilities. One conclusion that can be drawn from preliminary experiences with the prototype system is the need for separate audio communications links between collaborators. Early in the project, it was hypothesized that effective collaboration might be feasible using only on-screen *chat boxes* (textual conversation) or multiplexed voice/data. With experience, it became clear that an independent voice link was a highly desirable addition for the system to be effective. The addition of an independent voice link allows simultaneous action (in the workspace) and verbal discussion, which greatly facilitates the communication of ideas. This finding corroborates the conclusions of [2] and [7] that multiple channels of communication are necessary for effective computer-mediated collaboration support. While there is an apparent need for a separate audio connection, this does not seem to pose any problem, since all that is required is another low-bandwidth (phone) link. As long as two-way audio communication was provided, the prototype system allowed users to communicate in an acceptable manner.

One question that was of great concern when developing the prototype system was whether control locking and passing mechanisms would be necessary for the system to be effective. Because a shell-based system only permits a single entry point into an application program (which must be shared among users), there was concern that users trying to act simultaneously would consistently interfere with one another, perhaps significantly degrading the quality of interaction. If this were the case, then some type of control locking and passing mechanism would be needed to allow only one user to enter data at a time. Experience with the system has shown that, at least for small groups, formal control passing is unnecessary. Two collaborators working together within the framework of a program rapidly develop a sensitivity for alternately acting within the shared environment. This can be likened to the experience of a dialogue over a two-way radio, where the two users naturally alternate speaking and listening. In addition, it was found that informal rules for sharing the workspace rapidly developed between users, further reducing the need for control passing. For example, when editing a document jointly, it rapidly became accepted that a collaborator could add, mark or rearrange text, but could not delete text without permission. Thus, it has been found that for (small) groups of (nonantagonistic) collaborators, control locking and passing is not necessary, and would probably only hinder collaboration in a shell based system.

The apparent utility of the prototype system was observed for various functions associated with collaboration. Brainstorming is one collaborative function that is often considered when developing collaboration tools [6],[9]. As considered here, a brainstorming session "involves the initial generation of ideas" [6], where users start with a clear workspace and attempt to formulate new approaches or concepts. Preliminary experience with the prototype system has indicated that the shell-based approach is not well suited to this particular type of activity. The major reason for this seems to stem from the nature of collaborative interaction at the genesis of a project. When a project begins, each user often needs to spend relatively large amounts of time to create representations of ideas to share with collaborators. During this period of time, it is difficult to build upon each others' ideas, since their representations have not been sufficiently fleshed out. Therefore, at this stage there is a need for collaborators to work semi-independently (and preferably simultaneously). In a RESCOL shell system, the users must all share a single entry point into a program's workspace (usually a single cursor), which restricts users

to taking turns at expressing themselves and pursuing ideas. Because this restriction conflicts with the needs of collaborators at the beginning stages of a project, a shell-based system will not perform well for that situation. Thus, it seems from our experience that the nature of a shell-based user interface inhibits its use for pure brainstorming interactions. In contrast to its performance as a brainstorming tool, the prototype system was found to be very effective for development and extension of existing collaborative projects. This finding seems to support the prediction in [1] that "computer-mediated communication will be more valuable for coordinating already existing collaborative projects than for starting new ones". The reason performance is so much better when collaborating on an existing project seems to be related to the focus of the interactions. When two collaborators operate semi-independently (such as when brainstorming), the single entry point interface of a shell-based system obstructs the flow of ideas. However, when both collaborators are focussed on a single object or section within the shared workspace (which is more prevalent when extending existing work), a single data entry point does not seem to inhibit interaction or development of ideas. In fact, by enforcing a single focus within a shared workspace, the shell approach seems to help maintain the "sense of closeness" necessary in a remote collaboration tool [3],[4]. (Other researchers have also noted this benefit of enforced turn taking; see [6], for example). Thus, the shell-based interface shows great promise for supporting ongoing remote collaboration, with the benefits of synchronous interaction and a shared workspace intact.

The shell approach also performs well in instructional situations, where one collaborator is learning from another how to use an application program's environment. The major reason for this seems to be the identical "view" the collaborators have into the program environment: one collaborator can observe the other, and easily give a "helping hand" when necessary. It is not surprising that this type of identical view interface is good for instructional purposes. The Remote Takeover Software described in [11] and [12] provides a qualitatively similar (but more limited) interface, and is often used for these purposes. Thus, instructional use is one promising application for shell-based systems.

Finally, it should be noted that this type of interface seems to yield the most benefit when the information is inherently graphical in nature. Thus, the shell technique may hold great promise in fields such as engineering design, where information is best represented graphically.

The observations made here have been drawn from a somewhat limited exploratory study. The tasks at hand were relatively short, and executed under somewhat controlled conditions. Additionally, the collaborators were extremely familiar with working together, and were operating in familiar program environments. However, it is felt that the observations that have been made are generally valid, and will hold true under a much broader set of circumstances.

Conclusions

In this work, a prototype shell system has been described which gives single-user applications a multi-user interface by linking together multiple workstations. When the shell is in operation, the linked workstations operate identically, thus providing a uniform computer interface for a group of collaborators.

To overcome the limitations of existing techniques, a parallel processing approach is used, in which each workstation runs the application program independently. The RESCOL shell distributes the user input at each system globally, and regulates parallel execution among workstations. This approach enables the system to function over low-bandwidth commununications links, regardless of the amount of graphical output or database changes produced by the application.

While the groundwork for shell-based systems has been laid, there is much work to be done. First, there remain many technical issues which need to be resolved. The actual amount of communications reduction achieved by use of a parallel processing approach needs to be quantified for different types of programs, in order to determine the suitability of the technique in different situations [15]. Furthermore, implementation of an execution shell system has only been investigated for one operating system environment, and even there the complete power of the environment has not been fully exploited. Refinements of the shell technique under DOS, and investigations of its characteristics under other operating systems are needed to better determine its potential as a general approach.

Another remaining challenge is to more fully evaluate and characterize the benefits and disadvantages of shell-based systems as collaborative tools. The preliminary evaluation was carried out under very controlled circumstances, and did not account for many factors which could impact the effectiveness of a shell-based system in the real world. For example, the long term (more than one interactive session) structure of cooperative activity has been completely neglected as a factor in evaluating the prototype system, which clearly limits the applicability of the results [16]. Furthermore, even the technical utility of such a collaboration tool is not sufficient to guarantee its viability as a commercial product [14], [17]. Thus, a more rigorous evaluation of shell-based systems is needed

to establish their potential for commercial development.

Finally, the use of a shell technique to support collaborative work has not been fully exploited. For example, [4] and [5] conclude that the process of creating artifacts in a collaborative environment can be as important as the resulting artifacts themselves. Because a shell intercepts all user input into a program environment, it would be possible to use the shell to record collaborative interactions for review at a later date. It would likely be worthwhile to investigate the use of a shell for such purposes. Thus, while the groundwork has been laid, there are many opportunities for continuing research into the design and use of shell-based collaborative systems.

Acknowledgements

The authors wish to acknowlege the Leaders for Manufacturing Program and the IBM corporation for their support of this work.

References

[1] J. Galegher and R. Kraut, "Computer-Mediated Communication for Intellectual Teamwork: A Field Experiment in Group Writing", *Proceedings of the Conference on Computer-Supported Cooperative Work*, Los Angeles, CA, October 1990, pp. 65-78.

[2] J. Jakiela and W. Orlikowski, "Back To The Drawing Board? Computer-Mediated Communication Tools for Engineers", Working Paper, Alfred P. Sloan School of Management, Massachusetts Institute of Technology, June 1990.

[3] J. Tang amd L. Leifer, "A Framework for Understanding the Workspace Activity of Design Teams", *Proceedings of the Conference on Computer-Supported Cooperative Work*, Portland, OR, September 1988, pp. 244-249.

[4] S. Bly, "A Use of Drawing Surfaces in Different Collaborative Settings", *Proceedings of the Conference on Computer-Supported Cooperative Work*, Portland, OR, September 1988, pp. 250-256

[5] M. Stefik, D. Bobrow, G. Foster, S. Lanning and D. Tatar, "WYSIWIS Revised: Early Experiences with Multiuser Interfaces", *ACM Trans. Office Inf. Syst*, Vol. 5, No. 2, April 1987, pp 147-167.

[6] M. Stefik, G. Foster, D. Bobrow, K. Kahn, S. Lanning and L. Suchman, "Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings", *Commun. ACM*, Vol. 30, No. 1, January 1987, pp 32-47.

[7] R. Kraut, C. Egido, and J. Galegher, "Patterns of Contact and Communication in Scientific Research Collaboration", *Proceedings of the Conference on Computer-Supported Cooperative Work*, Portland, OR, September 1988, pp. 1-12.

[8] C. Bullen, R. Johansen, "Groupware: A Key to Managing Business Teams?", Working Paper, Alfred P. Sloan School of Management, Massachusetts Institute of Technology, May 1988.

[9] A. Goscinski and K. Beaton, "A Simple Distributed Computer System for Supporting Collaboration in Distant and Synchronous Meetings", *Computers in Industry* 12 (1989), pp 95-106.

[10] K. Kraemer and J. King, "Computer Based Systems for Cooperative Work", *ACM Computing Surveys*, Vol. 20, No. 2: June 1988, pp 115-146.

[11] L. Jordan and B. Churchill, Communications and Networking for the IBM PC and Compatibles, Third Ed., Brady Books, New York NY, 1990.

[12] R. Manning, "Long-Distance Calls: remote control software lets PC users feel free to be in the wrong place at the right time", *PC / Computing*, April 1989, pp 205-206.

[13] D. Perreault, "Interactive Computer Support for Remote Design Teams: A New Approach", *S.M. Thesis*, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, September 1991.

[14] J. Grudin, "Why CSCW Applications Fail: Problems in the Design and Evaluation of Organizational Interfaces", *Proceedings of the Conference on Computer-Supported Cooperative Work*, Portland, OR, September 1988, pp. 85-93.

[15] E. Antonidakis, "Communications Reduction by Simultaneous Program Execution", *Ph.D. Thesis Proposal*, Dept. of Electrical, Computer, and Systems Engineering, Boston University, Boston MA, August 1991

[16] S. Reder and R. Schwab, "The Temporal Structure of Cooperative Activity", *Proceedings of the Conference on Computer-Supported Cooperative Work*, Los Angeles, CA, October 1990, pp. 303-316.

[17] M. Markus and T. Connolly, "Why CSCW Applications Fail: Problems in the Adoption of Interdependent Work Tools", *Proceedings of the Conference on Computer-Supported Cooperative Work*, Los Angeles, CA, October 1990, pp. 371-377.