WHY CSCW APPLICATIONS FAIL: PROBLEMS IN THE DESIGN AND EVALUATION OF ORGANIZATIONAL INTERFACES

Jonathan Grudin

MCC
3500 West Balcones Center Drive
Austin, Texas 78759

Abstract.

Many systems, applications, and features that support cooperative work share two characteristics: A significant investment has been made in their development, and their successes have consistently fallen far short of expectations. Examination of several application areas reveals a common dynamic: 1) A factor contributing to the application's failure is the disparity between those who will benefit from an application and those who must do additional work to support it. 2) A factor contributing to the decision-making failure that leads to ill-fated development efforts is the unique lack of management intuition for CSCW applications. 3) A factor contributing to the failure to learn from experience is the extreme difficulty of evaluating these applications. These three problem areas escape adequate notice due to two natural but ultimately misleading analogies: the analogy between multi-user application programs and multi-user computer systems, and the analogy between multi-user applications and single-user applications. These analogies influence the way we think about cooperative work applications and designers and decision-makers fail to recognize their limits. Several CSCW application areas are examined in some detail.

Introduction. An illustrative example: automatic meeting scheduling.

Where electronic calendars are in use on a large or networked system, an automatic meeting scheduling feature is often provided (e.g., Ehrlich, 1987a, 1987b). The concept that underlies automatic meeting scheduling is simple: The person scheduling the meeting specifies a distribution list and the system checks the calendar for each person, finding a time convenient for all. The system then notifies all involved of the tentative schedule.

For automatic meeting scheduling to work efficiently, everyone involved must maintain a personal calendar and be willing to let the computer schedule their free time more often than not. Data reported by Ehrlich (1987a, 1987b) suggest that neither of these requirements is generally satisfied.

Electronic calendars are *not* electronic versions of paper calendars. They serve communication functions, primarily for managers and executives with personal secretaries who maintain the calendars. An electronic calendar may be used simultaneously by the secretary for scheduling, the manager for reviewing, and other group members for locating or planning. Ehrlich describes the successful use of the electronic calendar in detail; a key point is that "the secretary's role is critical"; those who do not have a secretary are much less likely to maintain an electronic calendar. Another relevant finding is that for managers, "free time is never really free." Unauthorized scheduling of a manager's apparently open time "can be sufficient motivation for total rejection of the system by the manager."

Thus, electronic calendars are voluntarily maintained primarily by managers and executives (or their secretaries). This has dire consequences for automatic meeting scheduling. If a manager wants to meet with non-management subordinates, few of the latter are likely to maintain electronic calendars. The scheduling program will find all times open, schedule a meeting, and conflicts will ensue. "In order to take full advantage of an electronic calendar, all members of a group must commit to using this medium," (Ehrlich, 1987b). If managers or executives keeping on-line calendars wish to meet among themselves, automatic scheduling could work. But as noted above, free time is often not truly free for such managers; it would be wise to consult with their secretaries anyway. Thus, automatic meeting scheduling may rarely be used in this situation, either.

The simple meeting scheduling feature previews the pattern that emerges from the major applications discussed later. Who would benefit from automatic meeting scheduling? The person who calls the meeting: in general, a manager would benefit. But who would have to do *additional* work to make the application succeed? The subordinates, who would have to maintain electronic calendars that they would not otherwise use. The

application might be made to work through persuading or ordering employees to maintain calendars, and replacing people who won't, but automatic meeting scheduling is not perceived to be of great enough collective benefit to warrant such measures.

Why design and implement a feature that is unlikely to be used? The managers who make the final design decisions may see the personal benefit of automatic meeting scheduling to managers such as themselves, without noticing that those users who would be forced to do extra work to support the feature would not benefit from it. (Other reasons for adding this feature might be a potential marketing benefit or simply the ease of implementing it.)

As with the more complex cases described later, the conclusion is not entirely negative. Automatic meeting scheduling could be targeted to environments or groups making the most uniform use of electronic calendars. Their value can be enhanced by adding conference room and equipment scheduling. Individual use of calendars to support the feature might increase if the perceived collective benefit were higher; that is, if organizations recognized how much they may be losing through inefficient meeting scheduling (Ehrlich, 1987b).

Problems in the design and evaluation of organizational interfaces.

Several major CSCW application areas have attracted significant investments of capital and labor over many years, with results that have uniformly fallen far short of expectations. These include the areas of digitized voice, group decision support, natural language interface to shared databases, and project management. The preceding example, automatic meeting scheduling, is a simple and relatively inexpensive feature. Its problems are easily identified, however, and can then be seen as common to many CSCW applications and as key factors in the their disappointing performance. The problems are:

- The application fails because it requires that some people do additional work, while those people are *not* the ones who perceive a direct benefit from the use of the application.
- The design process fails because our intuitions are poor for multi-user applications -- decision-makers see the potential benefits for people similar to themselves, but don't see the implications of the fact that extra work will be required of others.
- We fail to learn from experience because these complex applications introduce almost insurmountable obstacles to meaningful, generalizable analysis and evaluation.

These problems have received altogether inadequate attention given their impact on CSCW applications. This may be due to two powerful, natural analogues to CSCW applications for which these problems are in general less severe: multi-user *systems* (such as management information systems, computer-integrated manufacturing, order-and-inventory-control systems), and *single-user* applications. Our possible unconscious use of the analogies and the danger of failing to identify where they break down are discussed below and explored in more detail in two appendices.

The paper concludes with detailed "case studies" of four CSCW application areas. While the focus is on their problems, all of these areas are important and may lead to significant advances. It sometimes appears that in striving toward very ambitious goals we are taking turns beating our heads against the same wall, but pointing to the wall is not intended to devalue the goals. There are ways to get over the wall -- to build successful CSCW applications. Investing resources adequate to the solution of the problems, developing the appropriate research and development methodologies, finding niches where the problems don't arise or where applications will succeed in spite of them, and adequately preparing users for the introduction of the applications are all approaches that may lead to success. The first step is to see the problems clearly.

Problem 1. The disparity between who does the work and who gets the benefit.

The immediate beneficiary of the automatic meeting scheduler is the manager (or secretary) who initiates a typical meeting. Successful use of the feature in a typical environment would require additional work for other group members, most of whom would have to maintain electronic calendars when they would not otherwise do so. Not all CSCW applications introduce such a disparity -- with electronic mail, for example, everyone generally shares the benefits and burdens equally. But electronic mail may turn out to be more the exception than the rule, unless greater care is taken to distribute the benefit in other CSCW applications.

Can a CSCW application succeed if doing the extra work is left to individual discretion? Unfortunately, probably not. Communication, at the heart of most CSCW applications, will break down without relatively uniform use. If a substantial number of people do not maintain their calendars, the meeting scheduler is pointless. In this respect, the single-user application is a misleading model. In many environments, there is no harm if different users choose to use different editors, for example; but an application designed to support the entire group must be used by everyone in the group. "A critical mass of users is essential for the success of any communication system" (Ehrlich, 1987b).

Can a CSCW application be made to succeed by mandating that those who need to do the extra work do so? Even setting aside the implications of coercion (which are discussed in the final appendix), this is complicated. This traditional approach, changing existing job descriptions or inventing new ones, is widely used when CSCW *systems* are introduced (Cherns, 1980; Rowe, 1985). Word processing skills become a job requirement for secretaries, the new job of database administrator is created, and so forth. However, the multi-user system -- multi-user application analogy breaks down. An organization invests a large amount in introducing a *system* and is usually willing to reorganize to make it succeed, retraining, transferring, or dismissing people as deemed necessary. This will not be done for each CSCW

application that arrives -- the cost will outweigh the benefit. Maintaining a personal calendar in order to support automatic meeting scheduling is unlikely to become a job requirement. In general, *the organization may adapt to the computer system, but an application program must adapt to the organization.*

What if the application really might provide a collective benefit to the group or organization? Of course, measuring a "collective benefit" may be hard. If maintaining the application requires an hour per week from each group member, and its benefit is to save just one person an hour per week, is it worth it? What if the one person is the group manager? What if it is the Vice President of Research & Development? But assuming that a collective benefit has been determined, education and leadership may be critical. If it is demonstrated that inefficiency in scheduling meetings is costly to the group and that maintaining calendars to support the scheduling feature is the best solution, people may be willing to do the extra work.

However, the best solution is to try to insure that everyone benefits directly from using the application. This may mean building in additional features. It certainly means eliminating or minimizing the extra work required of anyone, or rewarding them for doing it. (This includes minimizing the training needed; Carasik and Grantham, 1988, attribute a rejection of The Coordinator, a CSCW application, in large part to the effort required to learn it.) User interfaces must be provided that vary appropriately with a user's background, job, and preferences. This is a substantial undertaking, but there may be no other option.

Problem 2. The breakdown of intuitive decision-making.

Why was the problem with the automatic meeting scheduler not anticipated? More generally, we need to understand why thousands of developer-years and hundreds of millions of dollars have been committed to various CSCW application areas despite little or no return. In most instances of failure, a substantial and timely return on investment was certainly anticipated; the decision-makers were not in business to throw away money.

Decision-makers in a position to commit the resources to application development projects rely heavily on intuition (see e.g. Butler, Bennett, & Whiteside, 1987). The experience, and very likely the track record, of a development manager considering a CSCW application is generally based on single-user applications. Intuition may be a far more reliable guide to single-user applications -- a manager with good intuition may quickly get a feel for the user's experience with a word processor, spreadsheet, or so forth. But a typical CSCW application will be used by a range of user types -- people with different backgrounds and job descriptions, *all of whom* may have to participate in one way or another for the application to succeed. The decision-maker's intuition will fail when an appreciation of the intricate dynamics of such a situation is missing.

Not surprisingly, the decision-maker is drawn to applications that selectively benefit one subset of the user population: managers. Intuitions about what will be useful to people similar to ourselves are generally good. Managers tend to overlook or underestimate the down side, the extra work that might be required of other users to maintain the application; extra work that might not be forthcoming in most environments, subjecting the application to neglect or sabotage. The decision-makers may also fail to appreciate the difficulty of producing and evaluating this new type of application, as described in the next section. The converse possibility also exists: the decision-maker may not see the value in applications or features that will primarily benefit other categories of user, even where they would provide a collective benefit to the organization. This would be particularly true for features that might undercut or create additional work for the manager.

Intuition may be less unreliable for applications directed at smaller or more homogenous groups. In particular, there may be less bias when only peer-peer communication is involved than when the communication moves vertically through the organizational hierarchy. Beyond that, education seems to be called for -- general education about groupware, the risks involved, and the resources and approaches required to minimize the risk; specific research on the application area at hand. Education is needed, and vigilance.

Problem 3. The underestimated difficulty of evaluating CSCW applications.

Task analysis, design, and evaluation are never easy, but they are considerably more difficult for CSCW applications than for single-user applications. An individual's success with a particular spreadsheet or word processor is not likely to be affected by the backgrounds of other group members or by administrative or personality dynamics within the group. These factors are, however, quite likely to affect applications intended to support an entire group, where motivational, economic, and political factors come to the fore (Malone, 1985).

Evaluation of CSCW applications requires a very different approach, based on the methodologies of social psychology and anthropology. This may not be news to those who have been monitoring the field of CSCW very closely, but the skills are largely absent in development and many research environments, where human factors engineers and cognitive psychologists are only starting to be accepted. And the required methods are generally more expensive, more time-consuming, and less precise.

It is relatively easy to bring a single user into a lab to be tested on the perceptual, cognitive, and motor variables that have been the focus for single-user applications. But it is difficult or impossible to create a group in the lab that will reflect the social, motivational, economic, and political factors that are central to group performance (Malone, 1985). In addition, group observation must extend over a longer period of time. Much of a person's use of a spreadsheet might be observed in a single hour, for example, but group interactions typically unfold over days or weeks.

Evaluation of groupware "in the field" is remarkably complex due to the number of people to observe at each site, the wide variability that may be found in group composition, and the range of environmental factors that

play a role in determining acceptance, such as user training, management buy-in, and vendor follow-through (e.g., Lucas, 1976; Gaffney, 1985; White, 1985; Ehrlich, 1987b). Establishing success or failure will be easier than establishing the underlying factors that brought it about.

Finally, the difficulty of evaluation is increased dramatically by the importance of providing features and interfaces that vary according to a user's job, background, and preferences, as mentioned above. A single-user application may get away with appealing to a kind of "lowest common denominator." CSCW applications will often have to appeal to every possible denominator.

As with the other problems, evaluation may be less difficult if the application supports a smaller or more homogeneous group than if the target user population involves individuals distributed across an organization. But it will still be substantial, and management must be aware from the start of the skills and the time that will be required.

Case 1. Digitized voice applications.

At a conference panel titled "Voice: Technology searching for communication needs" it was noted that after 25 years of research, no company specializing in voice technology has become profitable, and that projected sales of voice products have recently been revised downward sharply (Aucella, 1987). Eventual success of voice technology may require an understanding of the exaggerated forecasts and the relative failures to date. Here only the use of voice in computer-mediated communication is considered, as in computer-based voice messaging or voice annotation to documents. (Voice is also used for input only -- speech recognition -- and for output only -- e.g., speech synthesis.)

The advantages of digitized voice as a computer-based communication medium. Almost everyone can speak, while many people cannot type fluently. Moderately paced speech is much faster than even the fastest typing. Speech can readily convey emotion and subtle nuance. Voice messages can be sent or received by telephone when away from the computer. Voice annotation can be added without cluttering or overloading a visual display.

The disadvantages of digitized voice. It can be more difficult to understand than "live" speech, because stereophony and lip movements are absent and the speaker cannot be asked to clarify inaudible or unclear passages. Speaking may be faster than typing, but reading is faster than listening to speech. A digitized voice message cannot be scanned (by computer or human) as written text can -- the only way to be sure there is nothing of interest in a message is to listen to the whole thing. Similarly, the receiver cannot review a voice message later as easily as a written message. If suggested document changes or additions are contained in a voice annotation, the receiver must type those changes into the document. For the speaker, reviewing and correcting a spoken message is more difficult; hence, voice messages may be more likely to contain errors. A recipient cannot edit a voice message and must forward the entire message or none at all, which can be inconvenient (or even embarrassing to the originator; Ehrlich, 1987a, 1987b). Voice mail with no accompanying visual display has only the transient auditory channel for presenting and explaining options, leading to serious user interface challenges (see Aucella and Ehrlich, 1986). Finally, digitized voice requires a lot of disk space. (See Newell, 1984, for a broader discussion.)

The pattern. The advantages of digitized voice over typed input are almost all advantages for the speaker: Speech is faster to produce, conveys emotion and nuance easily, and may be available without access to a computer terminal. The disadvantages to digitized voice, however, are overwhelmingly problems for the listener. It is harder to understand, slower to take in, not easily scanned or reviewed, more likely to contain errors, and more difficult to manipulate.

To succeed, voice systems require that everyone in an environment use them. If some people do not use voice mail, time is wasted trying to reach them and group distribution lists won't work. If some people do not listen to their voice mail frequently enough, calls won't be returned promptly and use of the system may dwindle and die.

The speaker benefits from voice applications, and the listener does additional work. When will it be acceptable for speakers to thus burden listeners? One such time is when all users are both speakers and listeners in equal measure, thus sharing the benefits and costs, which is generally true of voice mail systems. In some cases, there may be no alternative -- a sales force on the road may have no electronic mail option, and for such users voice mail has proven particularly successful (Ehrlich, 1987a). Similarly, voice may be the best recourse for a user whose hands are necessarily busy. A disparity may also be acceptable when the speaker is of higher status than the listener, although this may be unpredictable.

Some past failures of voice technology are no doubt due to technical problems and storage requirements. Digitized voice messaging has proven successful given the right environment and implementation (Ehrlich, 1987a, 1987b). Voice may succeed more generally where its potential collective benefit is conveyed through high-level support and action (Ehrlich, 1987a, 1987b). In one case, a voice messaging system that failed initially succeeded when the alternative, telephone receptionists, was removed.

But, in general, the disparity between who is inconvenienced and who gets direct benefit from digitized voice may work against its adoption in situations where sender and receiver are of comparable status -- the imposition it makes upon the receiver may be unacceptable. Voice annotation may be unacceptable in most environments, where the authorial and editorial roles are rarely evenly shared. Voice annotation particularly benefits those who don't type or who are more likely to act in an editorial than in an authorial role. These are characteristics of managers and executives. Because of their status, they may not be concerned by the inconvenience of voice for others -- dictaphones are used. But the manager-secretary gulf is a particularly wide one; the danger is that decision-makers will support the development of voice applications that appeal to them but that will fail because their use will be onerous to other categories of user.

Case 2. Project management applications.

A project management application running on a distributed system might be the best demonstration of the potential of computer-supported cooperative work -- a major advance over the currently available single-user work management applications. A distributed project management application would cover the scheduling and chronicling of activities, the creation and evaluation of plans and schedules, the management of product versions and changes, and the monitoring of resources and responsibilities (Sathi, Morton, and Roth, 1986). Milestone completions might be signaled, documents routed to appropriate recipients, problems identified early and communicated to those who can help solve them, delays in critical path activities flagged, costs calculated, and so forth. Some people have felt that such an application will be the next major commercial success, "the next spreadsheet."

Cooperative work management applications are being developed. "Callisto: an intelligent project management system" (Sathi et al., 1986; reprinted in Greif, 1988), is a thorough description of a project begun in 1981. It is clear that in this area, it is crucial to ask who is the immediate beneficiary, who will be asked to take on additional work to make the application succeed, and what will be the incentive to do this extra work. The principal beneficiaries are clear: project managers. It is also clear that the success of such an application will be contingent on *all* group members keeping the information base current. This includes updating significant developments that occur around, rather than through, the system: in meetings, telephone conversations, and so forth. The project management application may also require that critical information that is usually unstated be made electronically accessible, such as a secretary's awareness of a manager's priorities (Ehrlich, 1987b).

The greatest user interface challenges will be on the side of information input -- reducing the additional effort to a bare minimum, allowing information to be entered in a manner comfortable to each worker, providing compensatory benefits to those who must take on the additional effort of maintaining the on-line database or knowledge base. But that is not where attention is being directed. It is being directed toward information display, toward the user interface for the principle beneficiary, the manager. "Managers must know what information is needed, where to locate it, and how to interpret and use it. Equally important is that they be able to do so without great effort" (Sathi et al., 1986). This is not unimportant, but exclusive focus on improving the system for the person already its principal beneficiary seems ill-advised, although it might appeal to the manager sponsoring such a project.

This is reflected in experience with management information systems. In one example, a ten year development project culminated in a "computer-assisted management system" installed on an aircraft carrier, "its primary purpose to help the Commanding Officer and his department heads administer the ship" (McCracken and Akscyn, 1984). While numerous factors contributed to its eventual replacement by a system that lacked management features, one reported reason for the failure of the management system was the difficulty of getting everyone to use it (Kling, 1987).

Worse fates than neglect may confront a project management application if monitoring and reporting are not carefully handled. In one implemented system, an employee who reported identifying a priority problem began receiving system-generated requests for progress reports to be forwarded to the Chief Executive Officer! This quickly led to the end of priority problem reporting. The vigilant system noted that employees had stopped using the system, and alerted the administrator. The employees dealt with the resulting complaints by writing programs that periodically opened files and changed dates, which satisfied the watchful, automatic monitor. Thus "sabotaged," the work management application was of little use, and was eventually quietly withdrawn. (Carroll Hall, personal communication.)

Case 3. Natural language interfaces to shared databases.

Natural language is not usually included in treatments of CSCW, but it is typically described as an interface style that by virtue of familiarity will appeal to subsets of users -- novice and "casual" or intermittent users -- with other interfaces available for heavy users (Rich, 1984; Shneiderman, 1987). Thus, it is in fact portrayed as useful in group work settings, and will seem more attractive as we address the problems of designing interfaces that must appeal to almost all users. As computer systems offer more capability, casual use of a given feature will increase, perhaps become the norm. Within a group, frequency of use of a CSCW application will inevitably be uneven; natural language could make it easier for some users to enter and retrieve information.

Database access seems a logical target application: the domain is circumscribed, much of the necessary vocabulary is explicitly set down in the database field and record labels, and the interaction -- user query followed by system response -- is predictable and limited. Natural language interfaces to databases have been available for several years.

Over the last ten years, most major developers of office systems have undertaken natural language projects and over fifty software companies have entered the field (Foley, 1986). While absorbing 1000 developers and hundreds of millions of dollars, none of these ventures had been profitable by 1985 (Johnson, 1985). (While one or two companies marketing databases with natural language interface options have since reported profitability, surveys have shown that the natural language feature was not responsible; Paul Martin, personal communication, 1988.) A survey of the natural language industry concluded "its story is not the stuff of which venture capitalists' dreams are made," (Johnson, 1985).

We need to understand two things: why has natural language failed to meet expectations and how has it attracted such high levels of support?

Problems of natural language interfaces to databases. Natural language understanding is incredibly complex: there is not yet a complete theory of syntax and semantics and pragmatics may be even more difficult. While a database interface based even on primitive linguistic approaches can correctly respond to a high percentage of the limited range of queries it encounters, it is not clear how an occasional error will affect the user's overall confidence in the system. If you ask for the average secretary's salary at the U.N. and are told $50K because it has averaged in the General Secretary of the U.N., you may cease to trust the system (Paul Martin, personal communication).

Another potential problem is coverage. People rely on a huge, structured knowledge acquired over years in order to understand simple things, more than existing systems can hope to incorporate (see e.g., Bobrow et al., 1977). A related problem is that users may expect an application that handles English to exhibit broad human intelligence and be disappointed when it does not (Rich, 1984). Rich also notes that the natural language user must do a lot of typing, although users can and do develop truncated "pidgin languages" that may end up more concise than complex queries in a formal query language. And one must also consider the conservatism of the database market and the need to develop appropriate marketing strategies as contributing factors to the poor reception for natural language interfaces.

Finally and more speculatively, natural language may not be matched to the tasks for which computers are used. In human interactions, we gravitate toward more formal language when we are uncertain of our audience, when we will get minimal feedback and opportunity to correct misinterpretations, and when we desire precise responses by the listener. All of these are characteristic of human-computer interaction. Perhaps if neural net or connectionist systems succeed in giving computers a more "fuzzy," human-like intelligence, natural language will be a good match.

The attraction of a natural language interface to databases. Perhaps more important than the circumscribed domain of a database and the explicit, built-in terminology are the problems outlined in this paper. The casual database user is the beneficiary of the natural language interface. The heavy user pays the price of additional keystrokes and reduced precision. The truly heavy user may work primarily by creating and modifying command files for frequently-issued complex queries in the formal query language. The heavy user always retains the option of using the formal query language that accompanies the natural language interface, but if that query language is not the best available formal interface, the heavy user would pay a price to use the system.

The manager and executive can envision themselves as casual users of a shared database, with others delegated to enter the data and carry out routine queries. Thus, natural language interfaces may appeal to decision-makers. But once again, decision-making intuition has failed if frequent users, the principal users of databases, prefer not to do the extra work that choosing such a system may entail.

Case 4. Group decision support systems.

The many efforts to develop computer support for group decision-making have generally produced systems, but it is clear that group decision support will benefit considerably by integrating with the systems people use for other aspects of their work and will thus become a CSCW application area. Such applications are already under development. At CSCW'86, Kraemer and King reviewed a large number of group decision support systems and concluded that their current reality is "far less than might be expected given their need and promise," and that "although some for-profit companies have undertaken to build (group decision support systems), they are not yet making much money," (Kraemer and King, 1986).

While they vary considerably in character, group decision support systems are highly susceptible to the problems outlined in this paper. They are expressly designed to be of principal benefit to decision-makers, insofar as one person is primarily responsible for the outcome of a meeting or a group decision process (undoubtedly the norm in our culture). The amount of work required of others to learn and use the system may vary. If use of the system requires significant learning, requires putting information on-line to make it publicly available, records information that a participant would prefer not to leave the meeting, blocks other means to influence decision-making (such as private lobbying), or undermines management authority, then the system may encounter resistance.

Conclusion.

Computer support for the activities of individuals in their group and organizational contexts will unquestionably change the way people live in significant ways. It is difficult to imagine anything more important or fascinating than trying to understand and guide that change. The analyses in this paper suggest that we are just beginning. Progress has been technology-driven to a surprising degree -- technologies searching for needs, as one panel organizer described it (Aucella, 1987). Many of us are aware of this general problem, pointed out by Engelbart (1982, 1985) and others, but its specific manifestations may continue to elude analysis, much less solution.

We need to have a better understanding of how groups and organizations function and evolve than is reflected in most of the systems that have been developed. At the same time, we also need to know more about individual differences in responding to technology if we are to develop systems that can support entire groups. One approach may be the contextual research of John Whiteside and his colleagues (Whiteside, Bennett, and Holtzblatt, 1987). Another is that used at Aarhus University in Denmark: "The Aarhus people start out with a problem situation defined by workers, and work beside them a long time in order to develop a new system that is "owned" by the workers... This is very different from traditional systems development, as you can imagine, and you can't simply package a set of techniques to do the job...see Ehn and King (1987)." (Liam Bannon, personal communication).

We must also develop a better behavioral understanding of our own decision-making processes as researchers and developers. The intuitions that have guided us in the past are breaking down. If we are going to support groups that include any diversity at all, we will have to learn much more about how different kinds of people work. Very frequently we see researchers studying other researchers, developers building systems because the technology exists, and managers supporting the development of systems that will appeal to other managers. We must make a strong effort to broaden our intuitions because experiments in the cooperative work area are so expensive and time-consuming.

Appendices.

Analogy 1. Multi-user systems and multi-user applications.

Most of our experience with computer support for group activity is based on the introduction of entire *systems* into an organization. I do not intend "multi-user system" to include a central, timesharing computer that essentially supports several individuals using individual applications, but rather a system that includes hardware and software developed to support group activity, such as a management information system, a computer-integrated manufacturing system, or an inventory control system. Multi-user *application* refers to software (and possibly minor hardware) acquired with the intent of integrating it into an existing computer system, such as a co-authoring program.

Computer support of group activity has typically required the acquisition or development of an entire system because the prerequisites -- multi-tasking, networking, interactive interfaces, and computer literacy -- were not in place. But as more advanced environments become widespread and people are comfortable with the terminal, PC, or workstation on their desk, systems will have to give way to applications. Today, an entire work management *system* might be installed, replacing or absorbing existing technology, but tomorrow a work or project management *application* will be sought, to run on an existing system. Cooperative applications that are appearing include co-authoring aids, sophisticated mail and time management, voice applications, shared databases, shared financial analysis packages, etc.

Our experience with multi-user systems, whether direct or through the literature, may influence our approach to multi-user applications. They have similarities -- they may serve the same purpose and behave much the same. But there are critical differences, particularly at the time of introduction: the system has a much higher cost, greater visibility, and stronger commitment of upper management. As a result, a new system brings with it the expectation of organizational change. While an organization will also adapt or evolve following the introduction of a CSCW application, the far less expensive application will not carry the same visibility, commitment, and expectation of change. From the perspective of the user, the introduction of the application must be smoother. This makes the job of the designer and implementer more difficult (see Pew, 1986).

The strong management commitment to ensuring the success of a new system means that a) the *collective* benefit of the system is recognized to be high; b) the organization may create new jobs to achieve success, if necessary; c) if a few important individuals will not or cannot use the system (the manager who won't use a terminal, for example), ways to work around them may be found; d) pressure from management to try the system may be high (whether through leadership and positive example or through more coercive approaches). Even with these forces working to the advantage of the system, we know that successful implementation is difficult. Introducing CSCW applications without this backing, all else being equal, will be *more* difficult. Better design and implementation are ways to ensure that all else isn't equal. (The application may have the advantage of finding a higher level of computer literacy, since a system is already in place.)

The much less expensive application program is likely to provide a smaller or uncertain collective benefit and won't have the same degree of management commitment. The organization cannot restructure itself around each new application, nor will management be likely to work as hard to ensure full participation. To a greater degree, the application must fit into existing work patterns and appeal to all the people needed to support it. For many of these communication-centered applications, this may be everyone: The application program may require full group participation without the advantage that a system often has of choosing or defining its users.

Analogy 2. Single-user applications and multi-user applications.

Whether we are researchers, designers, implementers, users, evaluators, or managers, most of our computer experience has almost certainly been with single-user applications. This experience has inevitably influenced the skills we have acquired, the intuitions we have developed, and the way we view our work. When we find ourselves thinking about or working with a CSCW application, it is useful to examine our approach carefully with this in mind, as many of our skills, intuitions, and outlooks will not help us in this different domain.

One effect of working with single-user applications is that we do not train ourselves to think extensively in terms of the disparity between the benefit obtained by and the work required of different user categories. We do of course give some consideration to the novice, casual user, heavy user distinctions, but in general, we can rely on feedback from a few "typical users." This experience may lead us to be unaware that we are only viewing a CSCW application from the perspective of the primary intended user, the user who obtains the most direct benefit. For

managers, this may have the effect of biasing their judgment regarding the CSCW application. A manager with good intuition might look at the design of an editor and correctly surmise "I would like these features and I think most users would." Looking at a CSCW application, the manager might surmise "I would like these features and I think most users would," but only be correct insofar as the other users are also managers. The single-user application does not train us to consider users of the same product who have a crucial but entirely different engagement with it.

Another effect of working with single-user applications is that we do not acquire the very different evaluative skills that CSCW applications will require. Most human factors engineers and other user interface specialists are versed primarily in applying techniques from perceptual, cognitive, and motor psychology to study phenomena of relatively brief duration. The one-hour experiment is still typical, and a study involving even a few sessions over several days is rare. But the group processes that will influence and be influenced by the use of CSCW applications bring social, motivational, economic, and political factors into prominence, and the temporal granularity required to understand such dynamics is much larger.

When is job redesign justifiable?

Central to this paper is the point that many CSCW applications will directly benefit certain users, often managers, while requiring additional work from others. A traditional method of coping with such a problem is to create new jobs or "redesign" existing jobs -- in short, to require people to do the additional work. Technology and organizational change is covered in depth elsewhere (e.g., Kraut, 1987a, 1987b; Crowston and Malone, 1987). This paper comments more on how things are than on how they might be, so I will limit myself to a few observations.

First, as noted in the paper, CSCW applications will not have recourse to changing job requirements to the degree that often occurs when entire systems are installed. The investment and commitment are smaller and the organization won't tolerate significant disruption for each new application acquired. CSCW applications will have to be more "group-friendly" than systems have been. They will change the organization, but more gradually. For this reason, the focus of CSCW will shift to user interface issues to minimize the disruption and additional work required of *any* user of the application.

Second, there may be a shift toward greater egalitarianism in the workplace (see e.g. Cherns, 1980), some of it surface and some of it perhaps a deeper emphasis on managing by building consensus. Therefore, it may be more difficult for management to mandate participation in new applications unless the collective benefit is very evident.

Third, when the collective benefit of using an application *does* appear great enough to warrant requiring some people to accept new or different tasks -- and measurements of collective benefit are of course difficult -- management has several options. Educating all users to the collective benefit may create a willingness to do the work. Inspiring through example or positive leadership is another approach. And, of course, improving the user interface to minimize the work or providing compensatory benefits in another area will help.

Finally, in some cases the work *will* be made part of the job. Setting aside tasks that most people would agree *no one* should be asked to do, the discomfort from job redesign is often transitory: Those hired with an understanding of the new requirements will be less uncomfortable with them than those living through the change.

Consider the example of programmer documentation of software code. Twenty years ago programmers writing entirely undocumented code might have been unhappy if forced to change for the collective benefit to the company of having maintainable software. But today more programmers are educated and socialized to accept this as part of their work; it is written into job descriptions, those taking the job are reasonably content to do it.

This is a cursory treatment of a difficult ethical topic, but anything more is, as they say, beyond the scope of this paper.

Acknowledgment.

This paper owes a lot to published and personal communications of Susan F. Ehrlich and to Liam Bannon's insightful comments. Clarence Ellis, Don Gentner, Donald A. Norman, Gail Rein, and Elaine Rich also contributed useful comments and encouragement. I am especially grateful for conversations on specific issues with Carroll Hall, Paul Martin, and Steven Roth. Clarence Ellis, Simon Gibbs, Bill Kuhlman, Steve Poltrock, Gail Rein, and I explored the significance of a group's position on the continuum from a small, homogeneous team to a large, heterogeneous organization using GROVE, a CSCW application developed by the MCC Software Technology Program to support brainstorming, leading to my greater appreciation for the importance of this factor. Many of the ideas in this paper were developed from a paper delivered at Interact'87 (Grudin, 1987).

References.

Aucella, A.F. (moderator), 1987. Voice: Technology searching for communication needs. In *Proc. CHI+GI '87 Human Factors in Computing Systems (Toronto, April 5-9, 1987)*, pp. 41-44.

Aucella, A.F. and Ehrlich, S.F., 1986. Voice messaging: Enhancing the user interface based on field performance. In *Proc. CHI '86 Human Factors in Computing Systems (Boston, April 13-17, 1986)*, pp. 156-161.

Bobrow, D.G., Kaplan, R.M., Kay, M., Norman, D.A., Thompson, H., and Winograd, T., 1987. Gus, a frame-driven dialog system, *Artificial Intelligence, 8*, pp. 155-173.

Butler, K., Bennett, J., and Whiteside, J., 1987. Engineering objectives for usability. Tutorial presented at CHI+GI '87 Human Factors in Computing Systems (Toronto, April 5-9, 1987).

Carasik, R.P. and Grantham, C.E., 1988. A case study of computer-supported cooperative work in a dispersed organization. In *Proc. CHI '88 Human Factors in Computing Systems (Washington D.C., May 15-19, 1988)*, pp. 61-66.

Cherns, A.B., 1980. Speculations on the social effects of new microelectronics technology. *International Labour Review*, 119, 6, pp. 705-721.

Crowston, K. and Malone, T.W., 1987. Information technology and work organization. CISR WP No. 165. Cambridge, MA: MIT Sloan School of Management.

Ehn, P., and Kyng, M., 1987. The collective resource approach to systems design. In Bjerknes, G., Ehn, P., and Kyng, M. (Eds.) Computers and democracy - a Scandinavian challenge. Aldershot, UK: Gower.

Ehrlich, S.F., 1987a. Social and psychological factors influencing the design of office communication systems. In *Proc. CHI+GI '87 Human Factors in Computing Systems (Toronto, April 5-9, 1987)*, pp. 323-329.

Ehrlich, S.F., 1987b. Strategies for encouraging successful adoption of office communication systems. *ACM TOOIS, 5*, pp. 340-357.

Engelbart, D.C., 1982. Towards high-performance knowledge workers. OAC 82. Reprinted in Greif, 1988.

Engelbart, D.C., 1985. Plenary address, CHI '85 Human Factors in Computing Systems (San Francisco, April 18, 1985).

Foley, M.J., 1986. Teaching computers plain English. *High Technology*, May, 1986.

Gaffney, C.T., 1985. Avoiding the "seven deadly sins" of OA implementation. In *Proc. Syntopican XIII Making Business Systems Effective* (Washington, D.C., June 17-20, 1985), pp. 241-254.

Greif, I. (Ed.), 1988. *Computer-supported cooperative work: a book of readings*. San Mateo: Morgan Kaufmann.

Grudin, J., 1986. Designing in the dark: Logics that compete with the user. In *Proc. CHI '86 Human Factors in Computing Systems (Boston, April 13-17, 1986)*, pp. 281-284.

Grudin, J., 1987. Social evaluation of the user interface: Who does the work and who gets the benefit? In *Proc. INTERACT'87 (Stuttgart, September 1-4, 1987)*, pp. 805-811.

Johnson, T., 1985. *Natural language computing: the commercial applications*. London: Ovum Ltd.

Kling, R., 1987. The social dimensions of computerization. Plenary address given at CHI+GI '87 Human Factors in Computing Systems (Toronto, April 5-9, 1987).

Kraemer, K. and King, J., 1986. Computer-based systems for group decision support: Status of use and problems of development. In *Proc. CSCW Conference on Computer-Supported Cooperative Work*, (Austin, December 3-5, 1986), pp. 353-375.

Kraut, R.E. (Ed.), 1987a. *Technology and the transformation of white-collar work*. Hillsdale: Lawrence Erlbaum Associates.

Kraut, R.E., 1987b. Social issues and white-collar technology: an overview. In Kraut (1987a), pp. 1-21.

Lucas, H.C., Jr., 1976. *The analysis, design and implementation of information systems*. New York: McGraw-Hill.

Malone, T.W., 1985. Designing organizational interfaces. In *Proc. CHI '85 Human Factors in Computing Systems (San Francisco, April 14-18, 1985)*, pp. 66-71.

McCracken, D.L. and Akscyn, R.M., 1984. Experience with the ZOG human-computer interface system. *Int. J. Man-Machine Studies, 21*, pp. 293-310.

Newell, A.F., 1984. Speech -- the natural modality for man-machine interaction? In *Proc. INTERACT '84 IFIP Conference on Human-Computer Interaction*, (London, September 4-7, 1984), pp. 231-235.

Pew, R. (moderator), 1986. Socio-tech: What is it (and why should we care)? In *Proc. CHI '86 Human Factors in Computing Systems (Boston, April 13-17, 1986)*, pp. 129-130.

Rich, E., 1984. Natural-language interfaces. *Computer*, September, 1984, pp. 39-47.

Rowe, C.J., 1985. Identifying causes of failure: a case study in computerized stock control. *Behaviour and Information Technology, 4*, pp. 63-72.

Sathi, A., Morton, T.E., and Roth, S.F., 1986. Callisto: An intelligent project management system. AI Magazine, Winter, 1986, pp. 34-52. Reprinted in Greif (1988), pp. 269-309.

Shneiderman, B., 1987. *Designing the user interface*. Reading: Addison-Wesley.

White, K.B., 1985. Socio-technical task team design. In *Proc. Syntopican XIII Making Business Systems Effective* (Washington, D.C., June 17-20, 1985), pp. 32-35.

Whiteside, J., Bennett, J., and Holtzblatt, K., 1988. Usability engineering: our experience and evolution. In M. Helander (Ed.), Handbook of human-computer interaction. Amsterdam: North-Holland, *in press*.