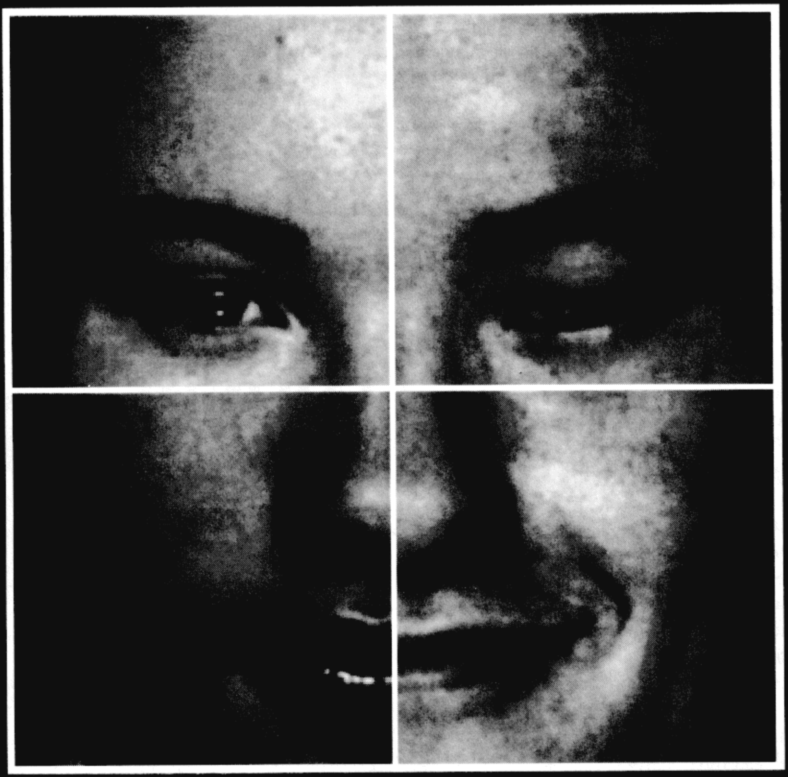
GROUPWARE AND SOCIAL DYNAMICS:



EIGHT CHALLENGES FOR DEVELOPERS

By JONATHAN GRUDIN Computer support has focused on organizations and individuals. Groups are different. Repeated, expensive groupware failures result from not meeting the challenges in design and evaluation that arise from these differences. • Many expensive failures in developing and marketing software that is designed to support groups are not due to technical problems. They result from not understanding the unique demands this class of software imposes on developers and users. This article briefly outlines the origins of groupware, describes eight specific problem areas, and finally examines groupware successes in search of better approaches to supporting work in group settings. • Desktop conferencing, videoconferencing, coauthoring features and applications, email and bulletin boards (b-boards), meeting support systems, voice applications, workflow systems, and group calendars are key examples of groupware. Labels vary: groupware, collaborative computing, workgroup computing, multiuser applications, computer-supported cooperative work (CSCW) applications. What is included? Not everyone agrees. Begin by asking, "Was this software designed to support groups? Is it being used to support groups?" • Email and b-boards are well known, but few other groupware prototypes and products have done as well despite considerable effort. Successes exist, but progress is slow and can lead in unanticipated directions.

Primarily Off-the-Shelf Products

The three rings of Figure 1 place groupware in the software universe somewhere between single-user applications and information systems that support organizations. Each software development area emerged independently and produced the research and development literature identified on the left.

Systems designed to support organizations achieved prominence first, because the expense of early computers required that they address major organizational goals. These include large mainframe (and, later, minicomputer) applications. "Organizational goals" are major goals typically defined by upper management. Such goals are not always fully agreed on, even among management. If they were, the course of internal systems development and acceptance would proceed more smoothly than it does. These research and development activities have variously been labeled data processing (DP), information systems (IS), management information systems (MIS), and information technology (IT).

By the early 1980s, the spread of interactive and personal computing created large markets for applications designed for individual users, such as spreadsheets and word processors. Research and development activities drew on existing human factors (HF) approaches to design and evaluation prior to the emergence in the early 1980s of conferences and journals under such banners as Computer and Human Interaction (CHI).

In the mid-1980s, the terms groupware and CSCW were coined and conference series and literature appeared. Conditions that emerged in workplaces to encourage this included (a) computation inexpensive enough to be available to all members of some groups; (b) a technological infrastructure supporting communication and coordination, notably networks and associated software; (c) a widening familiarity with computers, yielding groups willing to try the software; (d) maturing single-user application domains that pushed developers to seek new ways to enhance and differentiate products.

On the right in Figure 1 are the principal software development contexts involved in each area. Most systems addressing organizational goals are developed in-house or contracted out. Most single-user applications are commercial products, with development costs amortized over many customers. Groupware is largely a new market for product developers, along with telecommunications companies that have a focused interest in multiuser applications such as live video. Attendance at the first three CSCW conferences was primarily from software product development companies (approximately 40%) and universities (30%) with a steady telecommunications presence (5% to 10%).

To understand the problems encountered by groupware applications, it is essential to realize that most interest in groupware development is found among the developers and users of commercial off-the-shelf products who previously focused exclusively on single-user applications. The huge software markets created by standalone personal computers were once restricted to single-user applications, but as networks link the computers, groups represent large potential markets. As developers shift from supporting individual users to supporting groups, many encounter for the first time the challenges described in this article.

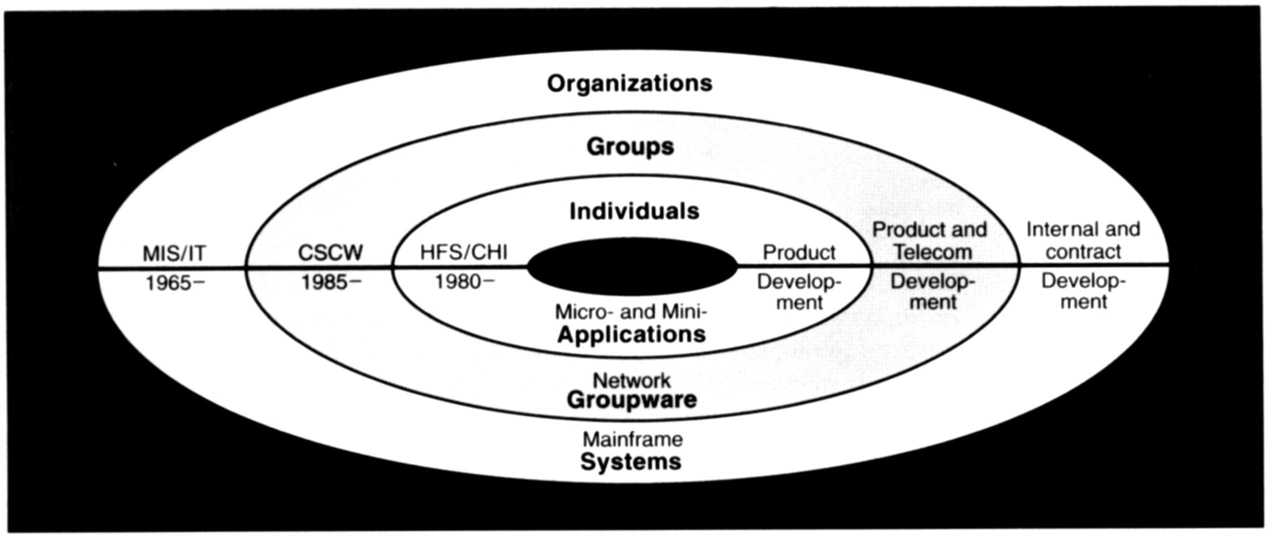


Figure 1. Development and research contexts

IS in Organizations: A Contrast to Product Development

The purchasers of a highly visible, expensive mainframe system or application anticipate a substantial benefit. They know organizational change is likely. Upper management is thus likely to commit to helping the system succeed, through (a) job redesign and creation (e.g., word processing skills are required of new secretaries and a database administrator position is created), (b) providing training in order to increase system acceptance, (c) restructuring to work around important individuals who will not use the system (e.g., a terminal-shy manager), and (d) positive leadership through inspiration or example. Even with such support, success is not assured. The system may not be salvageable or management may be divided. For example, the management of the IS group and other corporate managers may have conflicting goals.

These social and political factors that affect the introduction of large mainframe and minicomputer systems are little known to developers of single-user applications, including those moving on to groupware development. Yet similar forces affect groupware and must be considered by groupware developers. To the extent that groups share characteristics of organizations, groupware developers can learn something from the experiences of IS developers. But groups are not organizations and groupware is different from large systems.

Groupware targets smaller groups than systems serving organizational goals. Management is less committed to the less expensive groupware applications or features. An organization will not restructure itself for each new application the way it does around a major new system. In general, an organization may adapt to a large computer system, but a small application program must adapt to the organization, fitting into existing work patterns and appealing to everyone who must support it. On the other hand, groupware often benefits from user familiarity with the computer system already in place and from the relative homogeneity and shared goals of many groups.

Groupware is marketed as a product, whereas most MIS development is internal or contracted. Products are designed and evaluated to obtain a broad, competitive appeal, whereas internal IS staff have a specific set of users and must orchestrate their acceptance of a system. Each development context has its own objectives, constraints, approaches. There is little communication between product development, located in computer and software companies, and IS development, located in large companies engaged in other businesses or in companies that develop software on contract. Different research communities have grown up around each with different conferences, journals, and even languages [12, 14].

However, the MIS/IT community is interested in groupware—primarily in support for larger groups, such as workflow management systems and electronic meeting rooms. Declining technology costs bring large system software within the economic reach of groups, and thus into the product market. For example, a *group decision support system* or electronic meeting room developed by the University of Arizona and IBM was as recently as 1988 intended only for internal use by IBM. But in 1989 it was rechristened a *group support system* and in 1990 was marketed as a product, TeamFocus. The word "decision" was dropped not because of a sudden discovery that meetings serve many purposes, but rather because with declining system costs, the meeting rooms need not be used by decision makers to justify their expense.

The scope of these systems is shared by software engineering support for concurrent engineering, process programming, and other project-level activities. Thus, the middle ring could be subdivided into small-group and large-group support. CSCW research is defined to be inclusive, encompassing these and every activity at the organizational level.

Seven challenges in designing and evaluating groupware products are described. Because of the social and political factors at work in group settings, achieving groupware acceptance is much trickier than single-user product acceptance. It is difficult for "off-the-shelf" product developers to jump over the counter and help out with product acceptance, but they may have to: an eighth challenge for groupware developers.

New Problems

In addition to technical challenges, groupware poses this fundamental problem for product developers: Because individuals interact with a groupware application, it has all the interface design challenges of single-user applications, supplemented by a host of new challenges arising from its direct involvement in group processes.

Consider two relatively well-worked application areas: A review of group decision support systems concluded that after decades of effort, "their use is far below what could be expected given their need and promise," and "although some for-profit companies have built (group decision support systems), they are not yet making much money" [17]. A 1987 report stated that after 25 years of research, no company specializing in voice technology had become profitable and that projected sales of voice products were being revised sharply downward. More generally, for a panel of leading researchers titled "How can we make groupware practical?" Robert Kraut wrote "the only successful CSCW application has been email" and Lee Sproull wrote "groupware will never be practical and widely used in organizations if it follows its current trajectory." [9]

These gloomy assessments deserve an explanation, given the obvious potential in supporting something as widespread as group activity. Table 1 lists eight major problems that stem from the social dynamics of groups, drawn from developer experiences, descriptions of short-lived products and research prototypes, and experimental and modeling studies in the literature.

Overall, they call for better understanding of work environments and for corresponding adjustments by developers. Progress on the first five requires better knowledge of the intended users' workplace. The final three require changes in the development process. The final challenge in particular, addressing the sensitivity of groupware to aspects of its introduction in workplaces, demands that product developers expand their conception of the development process and product to include concerns that have been outside their sphere of activity.

As these challenges are examined in detail and illustrated with examples from various groupware areas, keep in mind that applications and use situations differ. Success and failure cannot be reliably predicted. Despite past problems and gloomy assessments, we find evidence of progress and ideas for working more effectively.

A Disparity: Work vs. Benefit

A groupware application never provides precisely the same benefit to every group member. Costs and benefits depend on preferences, prior experience, roles, and assignments. Although a groupware application is expected to provide a collective benefit, some people must adjust more than others. Ideally, each individual benefits, even if they do not benefit equally. However, this ideal is seldom realized. Most groupware requires some people to do additional work to enter or process information required or produced by the application.

Consider the automatic meeting scheduling feature that accompanies many electronic calendar systems. The underlying concept is simple: the person scheduling a meeting identifies the participants and the system checks each person's calendar, finding a time that is convenient for everyone. The direct beneficiary is the meeting convener, typically a manager or secretary, but for the feature to work efficiently, everyone in the group must maintain a personal calendar. Otherwise, the scheduling program will create conflicts by scheduling meetings in time that only seems to be open. However, studies have found that electronic calendars are typically used as communication devices by managers and are often not maintained by individual contributors [8]. Thus, successful use of automatic meeting scheduling requires additional work for those group members who would not otherwise maintain electronic calendars. Consequently, this groupware feature is rarely used.

Similarly, consider voice annotation to documents, a frequently developed feature. For speakers, digitized voice has advantages over handwritten or typed input. Speaking is faster than writing or typing, conveys emotion and nuance easily, and may be transmitted by telephone. Unfortunately, digitized voice creates problems for listeners. It is slower to take in, not easily scanned or reviewed, and more difficult to manipulate—for example, proposed edits will have to be typed in. When is it acceptable for speakers to burden listeners this way? Possibly when users speak and listen in equal measure, as in telephone conversations, or when the use of hands or a keyboard is impossible. A disparity may also be accepted when the speaker is of higher status than the listener, as with dictaphone machines, where saving one person time or effort can justify an arduous transcription. But in general, the disparity in effort and benefit works against acceptance in many situations and helps explain the failure of voice products to meet expectations.

As a third example, consider a distributed project management application that covers 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 (e.g., [27]). Its primary beneficiary is a project manager, but for it to succeed, other group members must enter information that is not typically kept on-line. This can lead to resistance. For example, a "computer-assisted management system" for a naval vessel, "its primary purpose to help the commanding officer and his department heads administer the ship," was developed over 10 years [22]. Due in part to the difficulty of getting everyone to use it, it was eventually replaced by a system that lacked management features.

Comparison: single-user applications. The problem does not arise. If a group picks one vendor for a single-user application because of the economy of purchasing a site license or in order to easily share its output, costs and benefits to group members may vary. But this is not an issue that developers can address.

Comparison: organizational IS. An expensive system is perceived to promise a substantial collective benefit. Therefore, management is more committed to take steps to ensure its success, such as hiring administrators and rewriting job descriptions. Thus, doing the additional work becomes someone's explicit job. This is much less likely for groupware. For example, it is unlikely that engineers will be required to maintain on-line calendars in order to support meeting scheduling.

Addressing the problem. Demonstrating an application's collective and indirect benefits can help. Reducing the work required of nonbeneficiaries seems to be an obvious priority, but it is very difficult to do in practice, because pleasing the principal beneficiary is critically important and the natural focus of attention. One promising approach is to design, along with the technology, processes for using it that create benefits for all group members. This has been stressed in several new meeting management applications. For example, a key element of the process in one is a specific commitment delivered by the meeting convener to act on the contributions of the participants.

Critical Mass and Prisoner's Dilemma Problems

Most groupware is only useful if a high percentage of group members use it. Different individuals may choose to use different word processors but two coauthors must agree to use the same coauthoring tool! Achieving a "critical mass" of users is essential for communication systems [8]. Even one or two defections may cause problems for meeting scheduling, decision support, or project management applications. Even in an idealized situation in which every individual will benefit once critical mass is achieved, the early adopters may well abandon it before the critical mass of users is reached.

Markus and Connolly [21] use an elegant model to demonstrate the possibility of "prisoner's dilemma" situations. In this model if *everyone* acts to further his or her personal best interest, the result is worse not only for the group but also for each individual. With some discretionary databases, as long as someone updates them, one's optimal strategy is to "freeload," but of course if everyone tries to freeload, the system is not used at all.

These analyses compound the

problem raised in the first challenge by showing that even a net benefit with equal costs and benefits for all users will not guarantee groupware success.

Comparison: single-user applications. The problem does not arise.

Comparison: organizational IS. One organizationwide voice-messaging system initially failed to obtain a critical mass of users: Those who tried to leave messages were discouraged when recipients did not use the system. This system succeeded, and even came to be appreciated by initial detractors, only after top management forced a critical mass of use by removing the alternative (message-taking receptionists). This kind of solution is available to expensive corporate systems. A less expensive groupware application or feature, such as voice annotation in word processing, is unlikely to get a forceful management shove past the critical point. Similarly, an organization can hire data entry personnel to support a large database, a solution to the prisoner's dilemma problem that most groups cannot afford.

Addressing the problem. Designers can reduce the work required of all users, build in incentives for use, and suggest a process of use that provides or emphasizes individual and collective benefits.

Table 1. Eight challenges for groupware developers

1. **Disparity in work and benefit.** Groupware applications often require additional work from individuals who do not perceive a direct benefit from the use of the application.

2. **Critical mass and Prisoner's dilemma problems.** Groupware may not enlist the "critical mass" of users required to be useful, or can fail because it is never to any one individual's advantage to use it.

3. **Disruption of social processes.** Groupware can lead to activity that violates social taboos, threatens existing political structures, or otherwise demotivates users crucial to its success.

4. **Exception handling.** Groupware may not accommodate the wide range of exception handling and improvisation that characterizes much group activity.

5. **Unobtrusive accessibility.** Features that support group processes are used relatively infrequently, requiring unobtrusive accessibility and integration with more heavily used features.

6. **Difficulty of evaluation.** The almost insurmountable obstacles to meaningful, generalizable analysis and evaluation of groupware prevent us from learning from experience.

7. **Failure of intuition.** Intuitions in product development environments are especially poor for multiuser applications, resulting in bad management decisions and an error-prone design process.

8. **The adoption process.** Groupware requires more careful implementation (introduction) in the workplace than product developers have confronted.

Social, Political and Motivational Factors

Groupware may be resisted if it interferes with the subtle and complex social dynamics that are common to groups. The computer is happiest in a world of explicit, concrete information. Central to group activity, however, are social, motivational, political and economic factors that are rarely explicit or stable. Often unconsciously, our actions are guided by social conventions and by our awareness of the personalities and priorities of people around us, knowledge not available to the computer.

Tacitly understood personal priorities are tactfully left unspoken, yet unless such information is made explicit, groupware will be insensitive to it. For example, secretaries know that managers' unscheduled time is seldom really free; unauthorized scheduling of a manager's apparently open time can lead to rejection of automatic meeting scheduling [8]. Similarly, a priority-based meeting scheduler foundered because participants were reluctant to acknowledge publicly that some of their meetings were low priority.

With one work management system, any employee who reported a "priority problem" received system-generated requests to forward progress reports to the chief executive officer—an extreme example of a design that ignores the sensitivity of certain communications. Employees stopped reporting problems. The vigilant system noted this and alerted the administrator. The employees dealt with the resulting complaint by writing a program that periodically opened files and changed dates, which satisfied the watchful, automatic monitor. Thus sabotaged, the system was of little use and was eventually removed.

As noted earlier, meeting management systems have not met expectations despite the appeal of improving the efficiency of meetings. Decision making is often complex and subtle, with participants holding partially hidden agendas, relying on knowledge of the others involved, and showing sensitivity to social customs and motivational concerns. Because such factors are not represented explicitly, the computer participates at a disadvantage. Kraemer and King [17] wrote "Most efforts have focused on the relatively narrow, rational view of the decision process . . . But as experience shows, this is limited in its utility because it specifically excludes the baffling nonrational or quasirational behaviors individuals often exhibit." In one case, a management group considered using an issue-based IS in which issues, arguments, counterarguments, and decisions are entered, creating a record of decision making that could be used to communicate, review, and explore alternatives. The plan to use the system was abandoned because the manager wanted the group to project a strong sense of consensus. The explicit record of opposing positions that the application would immortalize was politically unacceptable.

Conflicts of interest can become

major obstacles to success when group members have very different occupations or roles. Ehn [7] described such issues arising in the development of a newspaper page layout application to be used by typographers, journalists, and administrative staff.

Comparison: single-user applications. Applications that affect an individual's performance have broader effects. For example, desktop publishing software that enables anyone to produce professional quality documents can disrupt the power balance in an organization. However, these social effects are too indirect and context-specific to be addressed by single-user application developers, whose effort is more naturally directed to perceptual or cognitive interface factors that most users experience in the same way.

Comparison: organizational IS. This problem has been extensively explored in organizational settings [e.g., 6, 26]. Mainframe software developers have some advantages and some disadvantages in contrast to small-systems developers. They have well-defined target environments: Product developers must anticipate a range of sensitivities across customer sites. On the other hand, large systems inevitably affect workers whose goals conflict, whereas groupware focuses on low-conflict collaborations. A cohesive group is more likely to agree to purchase and use a piece of software. Groupware developers may risk overlooking conflict that occurs even in small groups, but organizational systems researchers may overestimate group conflict based on the higher levels of conflict found across larger organizational units.

Addressing the problem. Recognizing the magnitude of the problem and avoiding the common assumption of a "rational" work environment are first steps. Developers need sophisticated understandings of prospective users' workplaces. Working with representative users whenever possible is standard advice for developing interactive systems. It is particularly good advice for groupware developers.

Exception Handling in Workgroups

Work processes can usually be described in two ways: the way things are supposed to work and the way they do work. Software designed to support standard procedures can be too brittle. A passive strike tactic is to bring production to a halt by "working to rule" or "doing things by the book"; groupware that enforces "standard procedures" may also bring work to a halt [1]. A wide range of error handling, exception handling, and improvisation are characteristic of human activity [30]. People know when the "spirit of the law" takes precedence over the "letter of the law." Unfortunately, it is tempting to base design on available work specifications.

Ishii and Ohkubo [15] described the range of problems and consequences for designing groupware to support office procedures. "The main sources of information were an *office work handbook* made by the general affairs department and interviews with clerical workers. While collecting information, we found that the office workers made many shortcuts and modifications to the standard procedures defined in the handbook. Therefore, it was no easy task to determine the *actual standard* procedure, even when it was defined clearly in the handbook." The developers used this insight in designing the system, but it was not enough. "Unfortunately, we experienced problems in handling exceptional cases. This groupware executes predefined office procedures. However, it often happens that the standard procedure cannot be completed because of unpredicted situations." The authors concluded that AI techniques beyond the state of the art would be required to make the system useful.

A case study [26] illustrates the problem at the organizational level. Computerized stock control and sales order processing systems were introduced at a chocolate factory that is part of a large food company. Severe problems arose when the computer services division of the food company installed the systems in the chocolate factory: "[People in] computer services refer to a 'production mentality,' where [chocolate factory] staff respond to problems as and when they arise and are loathe to indulge in long-term planning and adopt specific procedures. Most important, they expect others to adjust to them, and resist the discipline the computer imposes . . . Moreover, not only did management fail to impose set procedures, but further *ad hoc* arrangements were positively encouraged by the sales department, as in the case of one customer who was assured they could amend their Friday order up to 1 P.M. on a Monday . . . No doubt they believed it was working in the best interests of the company, but its actions created considerable problems for those trying to operate the computer." In some areas the manual system continued to be used out of necessity. At one point, the general manager decided someone was sabotaging the system.

By recognizing the large amount of *ad hoc* problem solving in human activity and realizing that descriptions of "standard process" are often *post hoc* rationalizations, we can see the behavior that upset the computer services division as characteristic of efficient performance. After all, catering to the needs of specific customers is often considered a virtue, not a vice. In the case study, the general manager recommended that the system be withdrawn, but "he was overruled by group head office who were not prepared to lose face over the installation." By hiring new personnel and taking other expensive measures, the computer system was made to work. Upper management wanted this large, expensive system to succeed. A typical groupware application or feature, such as meeting scheduling, voice annotation or even meeting support, will rarely have the same degree of cost, visibility, and backing, and thus would fail under similar circumstances.

The strong interest of many organizations in supporting work flow management ensures that this complex issue will remain active. The outcome is difficult to predict: Some fear that the computer will become an enforcer of rigid procedures; others hope that greater explicitness will enable users to learn about their organization, leading to what Kari Kuutti has called "expansively mastered work."

Comparison: single-user applications. The preferences and work habits of an individual are more constant over time than those of groups, so flexibility is a greater consideration in supporting groups. Group activity is difficult to study and characterize. Even establishing the range within which group activity will vary is difficult. Single-user applications support flexible problem solving by providing a range of atomic actions and imposing few constraints on their sequencing, allowing users to construct and evolve work patterns through rapid trial and error. This approach does not work as well for group activity, because trial-and-error testing of options is slower and more public, and adjusting or evolving a group's practices requires negotiation.

Comparison: organizational IS. Groups are often more transitory and less well defined than organizations, so flexibility requirements may be greater. (Some scholars have suggested that when organizations are examined closely, groups seem to cease to exist.) The influential Cohen, March and Olsen [4] model describes an organization as "collections of choices looking for problems, issues and feelings looking for decision situations in which they might be aired, solutions looking for issues to which they might be the answer, and decision makers looking for work." A university, for example, exhibits continuity of purpose and activity at the organizational level that can persevere through dramatic shifts at the group level. An empirical study of 16 hospitals found "the predictability of the tasks confronting individual nurses was more closely associated with the characteristics of the nursing personnel on that unit than with the characteristics of the control system of the ward," according to Comstock (quoted in [24]). That is, ward-level policies did not predict behavior as well as group characteristics.

This volatility should warn groupware developers not to build software that imposes organizational controls on groups. Pfeffer [24] describes a study by Meyer and Rowan entitled "Institutionalized Organizations: Formal Structure as Myth and Ceremony": "To maintain ceremonial conformity, organizations import [views about what they should like and how they should work] and incorporate them in their structure, rules, and reporting requirements. However, since such rules and structures may have little to do with how the work can or should get performed, in fact there is little impact on task performance. . . This decoupling, Meyer and Rowan argued, is actually useful to the organization. It permits the work to get done according to the localized judgments of those doing the work, while presenting to the outside world the appearance of legitimated, rational organization of work." Myths and ceremonies can endure even as the real work processes change. In such environments, it would not be useful to impose at the group level the procedures dictated in the "myth and ceremony." (Of course, organization-level computer support can help perpetuate myths and enact ceremonies.)

Addressing the problem. To avoid the pitfall of supporting rational "myths," learn how work is actually done. Using tailorable systems is a good step to providing flexibility, but to tailor effectively is a challenge, because people are not conscious of detailed organizational functioning and how changes will affect other people. Groupware cannot count on the kind of management push that saved the large chocolate factory system. Carasik and Grantham [3] described the use of The Coordinator, a structured mail system. Users complained "this doesn't fit the way we work," but "management urging was the motivation for continued use." However, one frustrated user threw the software and documentation out of his office and after six weeks, use was discontinued.

Designing for Infrequently Used Features

If "to a hammer, everything looks like a nail," then to a groupware designer, every work situation calls out for communication or coordination support. We exaggerate the importance and frequency of the objects and events on which we focus. But many organizations are structured and responsibilities are divided in order to *minimize* the overall communication requirements and social interdependencies. As is well known, an increase in size can lead to a decrease in efficiency by increasing the communication and coordination overhead. Work has important social elements that can use support, but groupware features will be used less frequently than many features supporting individual activity. This has two important implications:

- First, groupware features will fare better if integrated with features that support individual activity. Consider coauthorship applications. Anyone who has written collaboratively can visualize the potential benefit of features to support annotation, version tracking, and effortless distribution of drafts. But most writing is done alone, whether single-authored or on a section of a jointly written document. Who would abandon their favorite word processor to use a coauthorship application? Features to support coauthorship must be integrated with those supporting authorship. Additionally, standalone groupware applications may not justify high purchase costs or may be perceived to fail if used appropriately but relatively infrequently. How often do most of us manage meetings that a group decision support system could facilitate, or embark on coauthorship projects?
- This leads to the second point: Design to be unobtrusive yet accessible. Infrequently used groupware features must not obstruct more frequently used features, yet they must be known and accessible to users. This is a difficult balancing act.

Comparison: single-user applications. Unlike groupware, the most important features are frequently used, so the problem of dealing with infrequent features is less pressing. However, avoiding clutter while ensuring awareness and access is a general and very serious challenge faced by the designers of all infrequently used features.

Comparison: organizational IS. To justify their cost, many organizational systems focus on high-frequency transaction processing, *Developers need sophisticated understandings of prospective users' workplaces.* **Working with representative users whenever possible is standard advice** *for developing interactive systems.*

reducing this problem. Also, at the organizational level, more than at the group level, there are opportunities to support people who actually do spend a lot of time communicating and coordinating activity.

Addressing the problem. If possible, add groupware features to an already successful application rather than launch a new application with a fanfare that creates expectations of heavy use. Ultimately, creating awareness of and access to infrequently used features could require systems that take the initiative to educate users over time. Work in this area, mostly in AI, has proceeded slowly. Yet the need grows, as computer capability exceeds by ever greater amounts our actual use of them.

The Difficulty of Evaluation

Task analysis, design, and evaluation are much more difficult for multiuser applications than for single-user applications. An individual's success with a particular word processor is not affected by the backgrounds or personalities of other group members. Groupware is affected by such factors, and often must interface simultaneously to users with different and sometimes shifting roles, preferences, and backgrounds. Users can be tested in a laboratory on the perceptual, motor, and cognitive aspects of human-computer interaction that are central to single-user applications. But lab situations and partial prototypes cannot reliably capture complex but important social, motivational, economic, and political dynamics. Even when a full implementation is available, scheduling a test is a logistical challenge.

Evaluation takes longer. Much of a person's use of a graphics program can be observed in a single hour, for example, but group interactions unfold over days or weeks. Groupware that supports limited-duration activities such as meetings has only a modest advantage, because awareness of the preparation and consequences are critical to understanding such events. Additionally, groupware evaluation methods are less precise. Field observations are complicated by the number of people involved over time at each site, the variability in group composition, and the range of environmental factors that affect the use of the technology. The pertinent skills of social psychology and anthropology are absent in most development environments, where human factors engineers and cognitive psychologists are only slowly being accepted.

Finally, generalizing from experience is risky. Establishing success or failure is easier than identifying the factors that brought it about. A highly motivated group can find a way to use a seriously flawed product and a badly managed installation can cripple a good product, so one generally finds some successes and some failures.

Consider this example: More than 10 members of a research laboratory took part in usability tests of a coauthoring application. The data were analyzed to find interface problems. This lab produces many coauthored articles and although it required only a few minutes to bring up the application, several months later it was not being used outside the experimental setting. Why? Some people were not using the right type of computer and others did not want to give up features of their favorite word processor. They were fine-tuning the interface to an application that would not be used.

The absence of definitive studies ensures that other researchers and developers will repeat costly mistakes. More often than not, CSCW and groupware conferences include papers on automatic meeting schedulers that were developed in ignorance of the fate of a decade of commercially available products. Predictable problems were encountered: insufficiently frequent user access, an unwillingness to place true priorities on a public system, incomplete adoption of the system by group members, and so forth. Respondents to a recent Internet poll identified meeting scheduling as the most widely available and the least useful groupware application. Similarly, voice editors for nonspecialists have been marketed for a decade with little success, but they continue to appear. A typical scenario of use devised to sell an application: A sends B a voice message containing directions for driving to a party, which B edits and forwards electronically to C. Because C must transcribe the directions, probably requiring a few passes through the voice message, substantial work is required of the editor-intermediary B, of C, and of any other recipients. In a real situation, anyone in the chain could greatly reduce the overall effort by typing in the directions.

Comparison: single-user applications. As noted, most are easier to evaluate than groupware.

Comparison: organizational IS. The success or failure of a system built for one organization is generally more obvious, although proving that its benefits outweigh its costs (or vice versa) can be difficult or impossible.

Addressing the problem. Development managers must enlist the appropriate skills, provide the resources, and disseminate the results.

The Breakdown of Intuitive Decision Making

Decisions to develop unworkable applications are widespread. The problem often lies not in the detailed design but in the conception, in the nature of decision making in development environments.

Decision makers rely heavily on informed intuition. Most product development experience is based on single-user applications, for which intuition can be a more reliable guide. A manager with good intuition who quickly gets a feel for the use of a word processor or spreadsheet can fail to appreciate the intricate demands on a groupware application that requires participation by a range of users.

In particular, decision makers are drawn to applications that selectively benefit one subset of the user population: managers. Project management applications primarily benefit project managers. Meeting schedulers and meeting management systems benefit those who convene meetings. Decision support systems primarily benefit decision makers. Digitized voice products appeal to those who rely on speech (remember the dictaphone). Similarly, managers envision their own use of features such as a natural language interface and support development efforts without recognizing the drawbacks and costs.

This bias is understandable—each of us has ideas about what will help us do our job. But in the case of groupware, managers often underestimate the down side, the unwelcome extra work that an application will require of other users, resulting in neglect or resistance. For example, a group decision support or work management application can require many people to learn to enter data, it can record information that participants prefer not to have disseminated, and it can block other means to influence decision making, such as private lobbying. Intuition fails when the intricate dynamics of such situations are not appreciated. Managers can also fail to appreciate the difficulty of developing and evaluating groupware, and not recognize that users will not be required to do the work to ensure success. Finally, their interactions with customers are with customer management, who share their biases. My observation, as a product developer, was that development managers whose intuitions were generally superb could fail spectacularly with groupware. Perhaps, with more confidence in their intuition, they pushed such projects more strongly than more cautious managers would.

Good intuition for multiuser applications is unlikely to be found anywhere in a product development environment. Experience as designers, implementers, users, evaluators, or managers is heavily based on single-user applications. This has shaped the skills and outlooks that are present. For example, human factors engineers are trained to apply techniques based on perceptual, motor and cognitive psychology to study phenomena of brief duration. They are unfamiliar with the techniques needed to study group dynamics over time.

Once a project is underway, most researchers or developers rely on feedback from a few potential users, often those expected to benefit the most. For example, the greatest interface challenge for an intelligent project management application is to minimize the information entry effort required of each subordinate (or provide compensatory benefits), but attention is instead directed toward information visualization: the interface for the project manager. "Managers must know what information is needed, where to locate it, and how to interpret and use it. Equally important, they must be able to do so without great effort" [27]. This appeals to the manager sponsoring the project, but it is not wise to focus exclusively on designing for the principal beneficiary, who should already be relatively highly motivated to use the product.

The converse intuition failure also occurs: A decision maker does not recognize the value of an application that primarily benefits nonmanagers, even when it would provide a collective benefit to the group or organization. This is particularly true for applications that create additional work for managers. This point is addressed later in the context of email.

Comparison: single-user applications. Early interactive applications (e.g., line editors) were developed by and for programmers, so intuition was particularly reliable. Intuition is generally better for single-user applications than for groupware, although it is relied on too heavily.

Comparison: organizational information systems. The problem can be less severe and stronger remedies are available. An internally developed system is to support the familiar business at hand, not external customers. While personnel can be hired or retrained, customers usually cannot.

Addressing the problem. Recognition of this problem was a factor in the emphasis on user involvement in the sociotechnical and Scandinavian collective resource approaches to IS development. Product developers face obstacles in involving users that could be particularly detrimental to groupware development [13]. If development management recognizes the risks, complexities, and fallibility of intuition, we could see fewer groupware projects, but those few might have realistic design goals and the resources to meet them.

Managing Acceptance: A New Challenge

Much research in organizational information systems (IS) has addressed system acceptance (e.g., [18, 19]).[1] Product developers are usually shielded from such concerns by marketing, customer support, documentation developers, training developers, and others who stand between them and the users. Customers also accept some responsibility for their choice and may have consultants, internal developers, and other groups to tailor, supplement, or oversee the introduction of a product.

Unfortunately, groupware can be so sensitive to aspects of its introduction that these strategies fail: If sold off the shelf in the usual fashion, it can be doomed. A word processor that is immediately liked by one in five prospective customers and disliked by the rest could be a big success. A groupware application to support teams of five nurses that initially appeals to only one nurse in

[1]The word 'implementation' is generally used. Unfortunately, product developers use 'implementation' as a synonym for coding, one of many terminological differences that hinder communication.

five is a big disaster. Groupware must be introduced very carefully, leaving little to chance.

Not surprisingly, the first research articles to consider adoption from a product developer's perspective focused on groupware. Product developers have been isolated from user environments and have little awareness that factors other than utility and usability govern a product's acceptance or rejection. The following strategy for encouraging successful adoption of groupware products, drawn from the work of Ehrlich, Francik, and their colleagues [5, 8], involves cooperation between developers and marketers.

Identify a group's problems and match the computer solution to it. For example, geographic proximity of group members guides choices between voice or email, or synchronous or asynchronous decision support. Identify appropriate work processes: Our tendency to focus on structured processes can be inappropriate for communication technologies that best support important (but often unrecognized) unstructured processes. Select appropriate pilot groups and individuals: Systems can fail if placed on executive desks when secretaries are more appropriate, if restricted to secretaries when professionals should be included, and so on. Work processes can cut across an organization chart (complicating purchasing decisions). Allocate equipment properly: The positioning of peripheral equipment such as printers and scanners can be critical.

Give the adopting group a clear understanding of the mature use of the application, perhaps through a site visit, to overcome uncertainty. In particular, provide education that demonstrates a positive impact on the work day. Step-by-step training on unfamiliar features can reduce anxiety even when insufficient for complete learning. Management attitude is critical to acceptance, a common observation of special significance for applications that represent a smaller organizational investment. Finally, someone should be prepared to prevent premature rejection by anticipating and dealing quickly with early problems, and follow-through support should be in place to handle the posthoneymoon period, when the group's curiosity wanes and work returns to center stage.

These strategies, familiar to those concerned with organizational systems, have been beyond the scope of product developers. Consultation is not packaged with shrinkwrap software. But if customers walk off with a groupware product the way they do with a spreadsheet program, these steps will not be taken and the product will probably fail. Through involvement with the adoption process developers can contribute to it and learn to build support for adoption into the product itself. Recognition of this is evident in the successful marketing of the Lotus Notes groupware application: A product development company shifted to an IS approach based on direct sales of software bundled with consulting support. The same approach was used by IBM with TeamFocus. The innovative but unsuccessful developers of Wang Freestyle reached the same conclusion [5]. But most groupware has been marketed with a traditional off-the-shelf approach—and failed.

Comparison: single-user applications. Developers have not dealt with individual users. Adding consulting services to a groupware package increases the cost and shifts the transaction away from the packaged software model held by vendors of single-user applications.

Comparison: organizational IS. Groupware developers can learn from IS experience. They face a daunting challenge: They must pay more attention to system acceptance problems than product developers have in the past, yet they face more difficult acceptance problems than large systems developers have in the past, due to lower visibility and consequently less management support.

Addressing the problem. By adding groupware features to existing applications, this problem is sidestepped. Standalone groupware must first be designed to meet the real needs of group members. Developers who understand the work environment well enough to design successfully will be in a good position to help design strategies for supporting adoption as well.

Email and Other Successes

Products such as email, databases, and code management systems are used successfully in group contexts. How do they avoid the pitfalls? Are they potential models? First, consider email.

1) Who does the work and who benefits? Email provides an equitable balance for sender and recipient. The person with a message to communicate must type it, while the receiver can read it easily at a convenient time. Thus, the primary beneficiary typically does a little more work. 2) Critical mass problems: These can have an effect, although with only one other user or a path to an external b-board, email can be useful. 3) Compatibility with social practices: At times almost conversational, at times almost epistolary, email allows us to apply existing social conventions. However, differences lead to problems such as "flaming," "junk email," "smileys,"[2] and to more subtle but significant problems described later. 4) Exception-handling: The asynchronous, informal nature of most email makes it flexible. Applications that impose more structure can suffer accordingly [2, 3]. 5) Frequency of use: Email is often relatively heavily used for groupware and basic use involves few features to learn and recall. 6) Difficulty of evaluation: Organizational costs and benefits are difficult to assess, but the heavy discretionary use by individuals is a sign of success. 7) Poor intuitions for groupware: Not all email applications succeed. There has been trial and error and intuitions have improved. 8) Acceptance: An interesting anomaly is that use has spread from academic and public sources more than through product development and marketing processes, a point returned to later.

Many of the applications successful in group settings share several properties with email. As is often true of email, the primary beneficiaries of databases and code management systems are not managers or

[2]Flaming refers to the very angry messages that email seems to elicit; junk email results from the ease of adding people to distribution lists (removing individuals is often more difficult than including them); a smiley signals humorous intent or emotional context using an image of a face rotated 90 degrees, such as :-)

decision makers, but people who use computer systems more routinely. These object management applications, like email, focus on organizing and handling information without incorporating notions of role, process, and social interaction. For this reason some do not consider them groupware. Also for this reason they largely avoid being overly rigid and disrupting social processes, challenges 3 and 4.

Shifting to a Work Perspective

Email demonstrates how important it is to adopt a workplace perspective rather than a technology perspective, and how difficult it is.

As developers, we see the distinction between sender and receiver as the key role distinction in the use of email. But, as the anthropologist Constance Perin noted in [5], the key distinction in email use in many organizations is that of manager and subordinate. The technology does not recognize the supervisor-subordinate distinction, but it is critical *in the workplace*. Whatever distinctions are designed into the technology, its reception is determined by distinctions that exist in the organization.

One groupware anomaly of email is its success. Another is that its use does not selectively benefit managers or decision makers. In fact, Perin documents that the contrary can be true. The ability for anyone to disseminate information rapidly can create problems for managers whose jobs involve filtering and routing information. In a classic bureaucracy, lateral communication is minimized—information flows up and down through the hierarchy. Email, even more than a telephone on each worker's desk, supports efficient lateral communication. This may provide greater flexibility and efficiency—but also create difficulties for managers in organizations built on the hierarchical model.

Similarly, the informality of email makes it easier, less imposing, and more private to bypass hierarchical levels. People who would not think of scheduling a meeting with their manager's manager will raise an issue by email, which can provide a level of informality approaching that of a chance conversation in the hall. Being bypassed can complicate managers' jobs. Rice [25] notes a study in which 7% of the messages spanned more than one level. This number may not seem high, but many employees *never* have face-to-face skip-level meetings. A few such messages, or even the possibility of making them, could subtly shift the managerial function.

One managerial responsibility is to absorb information from higher levels and tailor its presentation to subordinates to maximize their understanding or obtain a desired response. Correspondingly, information obtained from subordinates is filtered and recast to higher management. But information received electronically is more easily forwarded without tailoring. In fact, editing such messages can be problematic: If the original electronic version is forwarded by another path, the tampering is revealed. This places managers in a no-win situation. Olson and Lucas [23] suggest that it could lead to more "rational" environments by eliminating "distortion" introduced by bad managers, but good management involves translating and adding context to messages, taking time to prepare others to receive information, and other tasks that email can make more difficult.

And, of course, the ability of anyone to send a rumor or piece of news instantly to everyone in an organization creates a volatility with which management must cope. The asynchronous quality of email, often seen as a virtue, can bother managers whose time is tightly budgeted: "Mostly, a lot of times, I won't respond. I'll print the message and stick it in their file and wait until their weekly meeting," said one manager in an interview. In support of this view, Eveland and Bikson [10] found that professionals used email steadily through the day, but managers used it primarily in the early morning or late afternoon.

Perin [in 5] analyzed field studies and suggested that "these electronic social formations represent new sources of industrial conflict . . . they are seen as subverting legitimated organizational structures." While noting the collective value of electronic communication to large organizations, she describes how it can conflict with traditional organizational practices. For example, "the very 'invisibility' of electronic social fields, which may be cultivated bureaucratically because they are believed to enhance productivity, also delegitimates them and becomes the source of managerial negativism and suspicion." A study by Fanning and Raphael, cited by Perin, concluded that email "is simply not a management tool, if by management we mean those above the level of project leader . . . a medium which allows widely separated people to aggregate their needs is, in fact, quite frightening. Some managers correctly foresee that such a system can be most upsetting to the current established order, and do not participate in it as a result."

Email can be introduced under conditions that lead to different patterns of use, perhaps at times even strengthening hierarchic control But consider the implications if the general pattern outlined earlier proves true. Some managers can discourage or terminate email use, but many organizations have introduced it. Many students and professionals are accustomed to it. Thus, the forces Perin described are likely to play themselves out over time, forcing organizations designed on outmoded notions of efficiency and control to evolve. Finding new organizational forms and minimizing the cost of shifting to them are the challenges ahead.

Can we as technology developers change our perspective or must we rely on anthropologists and others? Visionary writers have stressed the need for designers to understand the functioning and evolution of groups and organizations. But recognizing the problem is easier than escaping the technology orientation reflected in the term "groupware." And intuition-governed, technology-driven, trial-and-error approaches are proving particularly expensive and failure-prone in this area.

Through the 1970s and 1980s, the British socio-technical and Scandinavian participatory design approaches experimented with meaningful engagement of users in systems development, a slow process of mutual education. Recently these have attracted wider attention through conference presentations and published overviews [11, 28]. The June 1993 issue of *Communications* was devoted to efforts to apply these methods in Europe and the U.S.

The following methods can help overcome the behavioral and social challenges facing groupware development and use:

• Extend the use of single-user applications in group settings by adding groupware features—collaborative writing features to an existing word processor, group support features to spreadsheets, and so forth. The economic barrier to acquiring a new system to support relatively infrequent activity is bypassed and rapid adoption is replaced by incremental adoption.
• Find niches where existing groupware succeeds, either in spite of the problems described in this article or because they do not arise. Voice applications help a traveling sales force that relies on the telephone, structured email applications may succeed in autocratic organizations, and so on.
• Build on object management or shared IS that have fared better than those that incorporate elements of organizational structure and work process. Object Lens [20] and Lotus Notes combine email and databases. Electronic b-boards are used to guide research, development and marketing [e.g., 29]. Modeling group process has proved more difficult, but workflow software is reportedly successful in supporting structured activity such as processing insurance forms. The variability of much group activity is a brake to much wider application.
• Find ways to provide direct benefits for all group members. In particular, supplement the technology with a design for the process of its use. Design and evaluation are easier and intuition better if relatively homogeneous groups are involved.

Be wary of applications that will selectively benefit managers or decision makers who are typically not heavy computer users.
• Educate managers and developers about groupware, the risks involved, and the resources and approaches that are required. Successful products such as Lotus Notes were longer in development than most applications. Working with users, extensive prototyping, and iterative design can be more cost-effective, but they are expensive.
• We need a better understanding of decision-making processes in development. Too often researchers study other researchers, developers build systems because the technology exists, and managers support the development of systems that appeal to other managers. We need a more empirical approach to broaden our intuitions. Trial-and-error learning has become too slow and costly.

When you examine research prototypes and available products, keep in mind that projects have purposes other than producing something useful. Other goals include exploring an interesting technical problem or matching a competitor.

Consider adoption issues from the outset. A groupware application may lead to organizational evolution, but its introduction must be smooth. Groupware must be more "group-friendly" than mainframe systems have been. To minimize the disruption requires interfaces adapted to users' backgrounds, roles, and preferences.

Anticipate organizational change. Some technology will replace or deskill workers; groupware that handles communication and coordination—management functions—can erode authority structures. Decentralized control could in turn further dim the prospect for groupware that selectively benefits management, a description of most groupware that has been developed.

Groupware may follow the pattern of other network technologies such as the telephone and the interstate highway. They spanned existing organizational boundaries, were designed for purposes unrelated to their ultimate use, and led slowly to a wide range of indirect effects. Our tentative exploration of a new technology is a step toward organizational and societal change that is not easily predicted or hurried.

Acknowledgment
My interest in these issues arose from successes and more often failures of applications that as a developer or user I embraced enthusiastically. The analyses emerged from interactions over many years with close colleagues and others. The significance of different contexts of system development arose in discussions with colleagues at Aarhus University. Discussions with Clarence Ellis, Gerhard Fischer, Joan Greenbaum, Rob Kling, Lynne Markus, Steve Poltrock, Susan Ehrlich Rudman, and Suzanne Watzman were particularly helpful. Comments by Bob Kraut and anonymous reviewers led to improvements. C

References
1. Bannon, L.J. and Schmidt, K. CSCW, four characters in search of a context. In *Studies in Computer-Supported Cooperative Work*, J.M. Bowers and S.D. Benford Eds., North-Holland, Amsterdam, 1991, pp. 3–16.
2. Bullen, C.V. and Bennett, J.L. Learning from user experience with groupware. In *Proceedings of CSCW '90* (Los Angeles, Calif., Oct. 7–10, 1990).
3. Carasik, R.P. and Grantham, C.E. A case study of computer-supported cooperative work in a dispersed organization. In *Proceedings of CHI '88* (Washington D.C., May 15–19, 1988), pp. 61–66. Some specific details were given only in the verbal presentation.
4. Cohen, M.D., March, J.G., and Olsen, J.P. A garbage can model of organizational choice. *Admin. Sci. Q. 17* (1972), 1–25.
5. Commun. ACM 34, 12. Perin, C. Electronic Social Fields in Bureaucracies.
6. Curtis, B., Krasner, H. and Iscoe, N. A field study of the software design process for large systems. *Commun. ACM, 31*, 11 (1988), 1268–1287.
7. Ehn, P. *Work-Oriented Design of Computer Artifacts*. Erlbaum, Hillsdale, N.J., 1989.
8. Ehrlich, S.F. Strategies for encouraging successful adoption of office communication systems. *ACM Trans. Off. Inf. Sys. 5* (1987), 340–357.
9. Ensor, R. Moderator. How can we make groupware practical? In *Proceedings of CHI '90* (Seattle, Wash. Apr. 1–5, 1990).
10. Eveland, J.D. and Bikson, T.K. Evolving electronic communication networks: An empirical assessment. *Off. Tech People 3* (1987), 103–128.

11. Greenbaum, J. and Kyng, M. *Design at Work: Cooperative Design of Computer Systems.* Erlbaum, Hillsdale, N.J., 1991.

12. Grudin, J. Interactive systems: Bridging the gaps btween developers and users. *IEEE Comput. 24*, 4 (1991), 59–69.

13. Grudin, J. Obstacles to user involvement in software product development, with implications for CSCW. *Int. J. Man-Machine Studies, 34*, 3 (1991), 435–452.

14. Grudin, J. Interface: An evolving concept. *Commun. ACM 36*, 4 (1993), 110–119.

15. Ishii, H. and Ohkubo, M. Message-driven groupware design based on an office procedure model, OM-1. *J. Inf. Process. 14*, 2 (1990), 184–191. Information Processing Society of Japan.

16. Kling, R. Social analyses of computing: Theoretical perspectives in recent empirical research. *ACM Comput. Surv. 12*, 1 (1980), 61–110.

17. Kraemer, K. and King, J. Computer-based systems for cooperative work and group decision making. *ACM Comput. Surv. 20*, (1988), 115–146.

18. Lucas, H.C., Jr. *Why Information Systems Fail.* Columbia University, New York, 1975.

19. Lyytinen, K. and Lehtinen, E. Seven mortal sins of systems work. In *System Design for Human Development and Productivity: Participation and Beyond,* P. Docherty, K. Fuchs-Kittowski, P. Kolm, and L. Mathiassen Eds., North-Holland, Amsterdam, 1987.

20. Malone, T.W. and Lai, K-Y. Toward intelligent tools for information sharing and collaboration. In COMPUTER AUGMENTED TEAMWORK: A GUIDED TOUR, R.P. Bostrom, R.T. Watson and S.T. Kinney, Eds. Van Nostrand Reinhold, New York, 1992.

21. Markus, M.L. and Connolly, T. Why CSCW applications fail: Problems in the adoption of interdependent work tools. In *Proceedings of CSCW '90* (Los Angeles, Calif., Oct. 7–10, 1990).

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

23. Olson, M.H. and Lucas, H.C., Jr. The impact of office automation on the organization: Some implications for research and practice. *Commun. ACM 25*, 11 (1982), 838–847.

24. Pfeffer, J. Organizations and organization theory. In *Handbook of Social Psychology,* G. Lindzey and E. Aronson, Eds., Random House, New York, 1985.

25. Rice, R.E. Computer-mediated communication systems network data: Theoretical concerns and empirical examples. *Int. J. Man-Machine Studies, 32* (1990), 627–647.

26. Rowe, C.J. Introducing a sales order processing system: The importance of human, organizational and ergonomic factors. *Behav. Inf. Tech. 6*, (1987), 455–465.

27. Sathi, A., Morton, T.E., and Roth, S.F. Callisto: An intelligent project management system. In *Computer-supported cooperative work: a book of readings,* I. Greif, Ed., Morgan Kaufmann, San Mateo, Calif., 269–309.

28. Schuler, D. and Namioka, A., Eds., 1993. *Participatory Design: Principles and Practices.* Erlbaum, Hillsdale, N.J., 1993.

29. Sproull, L. Using electronic mail for data collection in organizational research. *Acad. Manage. J. 29*, 1 (1986), 159–169.

30. Suchman, L. Office procedures as practical action: Models of work and system design. *ACM Trans. Off. Inf. Syst. 1* (1983), 320–328.

CR Categories and Subject Descriptors: D.2.9 [**Software Engineering**]: Management—*productivity*. H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces—*theory and models*; K.4.3 [**Computers and Society**]: Organizational Impacts

General Terms: Management

Additional Key Words and Phrases: Computer-supported cooperative work groupware

About the Author:

JONATHAN GRUDIN taught in the Aarhus University Computer Science department prior to joining the faculty at Irvine. He also worked in the now defunct Human Interface Laboratory at MCC and as a developer at Wang Laboratories. His interests include human-computer interaction and the history of interactive systems development. **Author's Present Address:** Department of ICS, University of California, Irvine, CA, 92717; email: grudin@ics.uci.edu

CONTINUED FROM PAGE 63

27. Schuler, D. Community Networks. *Commun. ACM 37*, 1 (Jan. 1994) (this issue).

28. Schuler, D. and Namiok, A. *Participatory Design: Principles and Practices.* Erlbaum, Hillsdale, N.J., 1993.

29. Shneiderman, B. *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* Addison-Wesley, Reading, Mass., 1987.

30. Tapscott, D. and Caston, A. *Paradigm Shift: The New Promise of Information Technology.* McGraw-Hill, New York, 1992.

31. Vehviläinen, M. Social construction of information systems: An office workers' standpoint. Department of Computer Science, University of Tampere, 1991.

32. Zuboff, S. *In the Age of the Smart Machine: The Future of Work and Power.* Basic Books, New York, 1988.

CR Categories and Subject Descriptors: K.4.3 [**Computer and Society**]: Organizational Impacts; K.6.1 [**Management of Computing and Information Systems**]: Project and People Management—*systems development*

General Terms: Design, Human Factors, Management

Additional Key Words and Phrases: Case studies, participatory design (PD), user empowerment

About the Author:

ANDREW CLEMENT is an associate professor in the Faculty of Library and Information Science at the University of Toronto. His current research interests focus on the workplace implications of computerization, particularly privacy issues and the participatory development of information systems. **Author's Present Address:** Faculty of Library and Information Science, University of Toronto, 140 St. George Street, Toronto, Ontario, Canada M5S 1A1; email: clement @flis.utoronto.ca