ISSN 0105-8517

Two Analyses of CSCW and Groupware

Jonathan Grudin

DAIMI PB – 323
July 1990

TWO ANALYSES OF CSCW AND GROUPWARE

Jonathan Grudin

Department of Computer Science
Aarhus University, Denmark
(on leave from MCC, Austin, Texas, USA)

1. CSCW: The convergence of two development contexts

2. Seven plus one challenges for groupware developers

ABSTRACT

This report consists of two papers on the topics of computer supported cooperative work and groupware. The first examines the emergence of CSCW in the mid-1980s, finding that its timing and composition reflect changes within one context of systems development, *product* development. Many of the issues -- and some of the participants -- have a history in *internal* systems development; some confusion of identity in the field may be traced to the merging of these development paradigms. The second paper examines eight challenges in designing and evaluating groupware or CSCW applications, challenges that are new for *product* or application developers. One that is particularly problematic, due to the way software products are marketed, is the need to address a range of organizational factors on a site-by-site basis to obtain acceptance of systems developed to support groups.

CSCW: THE CONVERGENCE OF TWO DEVELOPMENT CONTEXTS

Jonathan Grudin

CSCW research and groupware development represent converging interests from two contexts of interactive systems development. Issues of group dynamics and organizational impact have primarily been explored in the context of the internal or in-house development of systems for organizations -- systems that support organizational goals. Many of the same or similar issues are now being encountered by the researchers and developers with a product development orientation who are seeking to support small groups. To integrate effectively the interests, experiences, and approaches arising in these two contexts, we have to go beyond what is shared and explore the differences.

INTRODUCTION: THE EMERGENCE OF CSCW

If Computer Supported Cooperative Work is a discipline, it can only be characterized as an unruly discipline. Some authors have lauded the lack of definition, observing that it contributes to a healthy cross-fertilization by attracting researchers from different backgrounds (Bannon, Bjørn-Andersen and Due-Thomsen, 1988). A similar confusion accompanies the often-associated technology description "groupware." For example, at a CHI'90 panel on groupware, one panelist argued that network file servers are groupware, another panelist argued that the only groupware success is electronic mail, and a third argued that there are no groupware successes (Ensor, 1990).

Uncertainty and disagreement are evident in conference programs, published papers, reviews, and informal comments. The thirty papers of the CSCW'88 Conference included four centered on video and three studies of work practices involving little or no computer use. Six papers drew from the Scandinavian participatory or collaborative development tradition: the collaboration did not center on computer use, only three of the six described the development of computer systems, and even those were systems supporting organizations, not "groupware." Strong reactions to this composition[1] may partially account for the quite different mix at CSCW'90. In another demonstration of the unsettled nature of the field, Bannon and Schmidt (1990) argue against considering electronic mail a "CSCW application" and for including Computer Integrated Manufacturing, both relatively iconoclastic positions.

However, the field is not incoherent. The participants have histories, the issues have histories, and both are situated in a broader context of systems development that is becoming much better understood (e.g., Friedman, 1989). By taking into account the roots of CSCW research and groupware development, the purposes of those participating, and the trajectory (or trajectories) of systems development as a whole, we can position our efforts and avoid cutting against a strong historical grain.

The field has a strong technology interest, so it is worth noting explicitly that its foremost concern is with aspects of the behavior of people and organizations. This is implicit in the terms "groupware" and "CSCW" and is accepted by those whose daily work may be strictly technical, such as working out algorithms to support concurrent activity. Some are more focused on utility and some on usability, as reflected by the joint sponsorship of CSCW by SIGOIS and SIGCHI. But the line between CSCW and its technological substrate, however uncertain its precise placement, is drawn at a relatively high level for a field of computer science.

---

[1] These have been expressed verbally and noted in written conference reviews; they also appear in anonymous paper reviews, even two years after CSCW'88.

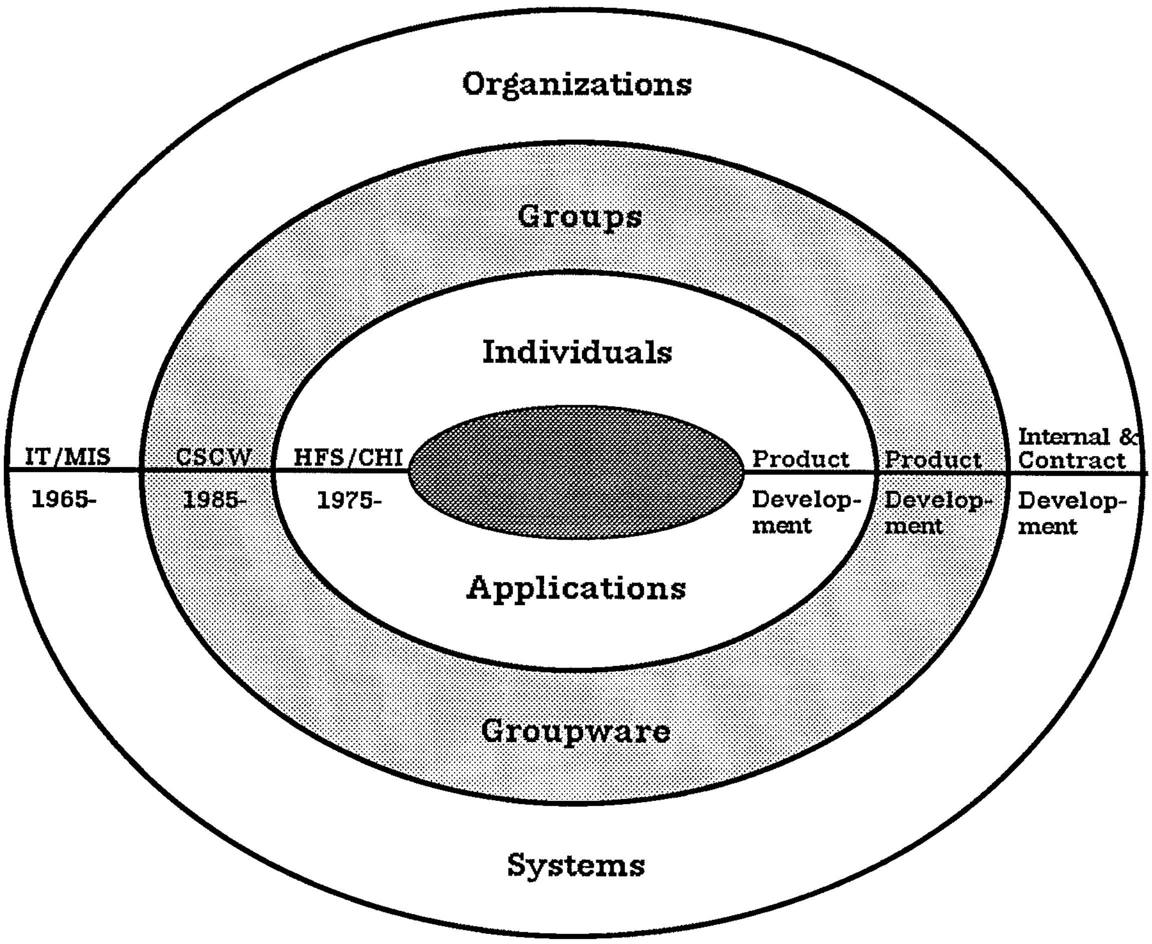

Figure 1. CSCW and Groupware in the development and research contexts.

Each ring in Figure 1 represents one focus of computer development and the principal "customer" or "user" of the resulting product. The outer ring represents entire systems designed to serve organizational goals -- large transaction processing systems, management information systems, computer integrated manufacturing systems, order and inventory control systems, etc. The inner ring represents applications designed for individual users, such as word processors, graphics programs, spreadsheets, games, etc. The middle ring represents groupware, designed with groups in mind. There are of course complications, but before delving into them, consider the remaining elements of Figure 1.

On the right are the software development contexts that dominate system, application, and groupware development. Software systems that support entire organizations are not usually bought at the local computer store or even at the local IBM sales office. The hardware and some software components might be acquired that way, but a great deal of the development is unique to the organization, done by internal developers or through a contract to another party. Single-user applications, on the other hand, *are* the province of the product development companies. The developers rely on the large potential market to provide sufficient sales and do little or no customization for individual customers. Groupware logically lies between and does in fact draw on both development contexts, but for several reasons discussed below, its emergence in the 1980s represents above all else the movement of product developers into this middle ground.

On the left is the research context associated with each development context, along with its approximate date of emergence. The literature associated with systems in organizations arrived with the "third generation" computer systems built with integrated circuits, in the mid-1960s. Friedman (1989) summarizes, "There is very little on the subject up to the mid-1960s. Then the volume of literature on (computers and the) organization of work explodes. Issues of personnel selection, division of labour, monitoring, control and productivity all subsequently receive considerable attention." Single-user applications drew substantial research interest following the spread of interactive systems in the mid-1970s. The Human Factors Society provided one showcase in the United States for such work prior to the emergence of the CHI organization. Finally, the field of CSCW emerged in the mid-1980s, with a small workshop in 1984 and the CSCW '86 Conference in Austin, Texas. The term "groupware" was coined at around the same time.

CSCW has drawn developers and researchers from each of the pre-existing development cultures. It represents a merging of issues, approaches, and languages. With two boundaries to contend with, CSCW and groupware are more difficult to define and no definition satisfies all the people who identify themselves as working in the field. And as Bannon et al. (1988) noted, it is the differences and the potential for cross-fertilization or synthesis that gives the field much of its excitement.

THE TIMING OF THE EMERGENCE OF CSCW

Why has the focus on computer support for groups formed last? In this section, the timing of the emergence of each development focus is taken up in turn.

Computer support for organizational goals attracted developers and researchers first due to the tremendous expense of developing, purchasing, and maintaining early computers. Systems were expensive, required hiring new personnel, and generally involved a long-term commitment and significant organizational change. Through the 1960s, these expenses could be justified if the system promised to help meet organizational goals. Although the organizational goals could hinge on prestige as much as real economic return (Greenbaum, 1979), acquiring a system had substantial economic and organizational consequences. Following the introduction of the IBM 360 series in 1965, the impact of computers on organizations became substantial. As noted by Friedman in the passage quoted above, this led to the development of the corresponding research field. Internal and contract development of large systems, and research into their use and impact in organizations, remains active.

Until time-sharing systems began edging out batch processing and computing costs dropped, single-user applications were primarily a semi-illicit pastime enjoyed by a few programmers during slow moments in computing centers. Stephen Levy's book Hackers captures the mid-1970s transition of software applications from hobby to business. Even in the early 1970s, text editing was too costly to be more than a programming tool (Irons and Djorup, 1972). This changed with the arrival of home computers and interactive word processing systems in the mid-1970s. While the 1960s saw some research on computer "console operators" and a lot on computer programming, studies of computer use by individuals took off in the late 1970s.

Computer support for groups, as embodied in the use of the terms CSCW and groupware, required conditions that were not present when these earlier foci were established. In some sense, each previous advance in technology use *did* support certain groups within an organization. Management Information Systems provide information to the managerial layer, whether or not they participate directly in the use of the system, and single-user applications indirectly affect the groups to which their users belong. What has changed is that groupware is *designed* with the explicit intention of supporting groups and with the assumption that all or most group members will use it. The conditions required for serious research and development in this area were: i) a low enough cost of computation that virtually all members of groups can afford to use computers; ii) the development of the technological infrastructure for supporting communication and coordination, notably networks linking computers; iii) widespread familiarity with computers, creating sufficient numbers of groups that are willing and able to try the software. These conditions are only now emerging, and still only in certain workplaces. Two forces *motivating* research and development are: i) declining system costs mean that smaller organizational units can afford powerful systems; ii) the maturing of some single-user applications areas pushes developers to look in new places for product enhancement or differentiation.

Activity in ACM itself underlines the sequential emergence of the outer ring, the inner ring, and then the middle ring. In 1976, CACM introduced the department "Social Impacts of Computing," stating "During the last decade, computer based technologies have become common in complex organizations in the industrial and public sectors." In 1980, the department "Human Aspects of Computing" was introduced with "Recently, there has been emerging interest by both industry and universities (focused on) the *human user*," and two years later SIGCHI formed. More recently, ACM partially sponsored CSCW'86, and SIGCSCW is forming this year. Similarly, when *Computing Reviews* revised its classification scheme in 1982, the social and organization categories were consolidated and expanded, whereas the topic of human-computer interaction, hitherto not represented at all, received a few scattered entries.[2] Another major revision, expected in 1990, will consolidate and expand the human-computer interaction categories, and may introduce a few scattered entries in the CSCW/groupware area.

By examining the systems development contexts that diverged, evolved, and are partially reconverging under the CSCW umbrella, we can better understand the issues in the field, the different approaches being suggested for addressing them, and the opportunities and obstacles to applying these approaches successfully.

---

[2] "User interfaces" appears as a subject under Software Engineering, Tools and Techniques; "User/Machine Systems" appears under Information Systems, Models and Principles; and "Human Factors" is a General Term.

THE EMERGENCE OF CSCW IN THE PRODUCT DEVELOPMENT CONTEXT

Early systems were expensive and geared to large organizations, whose specific requirements were addressed through in-house or contracted systems development. Beyond the hardware and core system software, these systems were not amenable to the "off the shelf" or even aggressively marketed product development that marks single-user applications. Internal systems development and to a lesser extent contract development dominated the industry at the outset,[3] and historical surveys and studies of organizational impact have focused on them (e.g., Friedman 1989).

However, with the emergence of interactive systems, product development has attracted far more attention, particularly in the United States. Established product vendors -- IBM, Digital, Hewlett Packard, etc. -- also sell hardware and system software for internal development and may have "Federal Systems Divisions" that compete for contracts. But as hardware profit margins decline and the industry settles on a few software system platforms, applications are increasingly central. Many of the successful new companies of the 1980s -- Lotus, Microsoft, Ashton-Tate, etc. -- primarily sell applications. Product development dominates usability research in the United States, as reflected in the CHI Conference programs and attendance.[4]

The varied pedigree of CSCW research issues is discussed in the next two sections. Nevertheless, we can trace the emergence of the field in the 1980s to the interest of product developers. The term "CSCW" was invented by employees of two product development companies. The first CSCW Conference was co-sponsored by a consortium of product development companies. And about 50% of CSCW'86 and CSCW'88

[3] For example, twenty years ago, industry-wide expenditures on in-house programming and in-house data preparation were estimated at $6 billion *each*, while only $0.5 billion was spent on external software (CACM, 1970). In much of Europe, product development still has a much smaller role than in the United States.

[4] A rank ordering of CHI Conference attendance since 1985 attendance lists these five companies first: IBM, AT&T, Xerox, Hewlett-Packard, and Digital, alone accounting for over 15% of the attendance. Organizations represented on the CHI'90 technical program were about 40% academic, 30% computer product development, 10% telecommunications, 10% consulting, 5% government and 5% internal software development.

registrants were from product development companies,[5] with approximately 25% from academic institutions and 25% from government, consulting companies, or the internal systems groups. A product orientation is suggested by the terms "CSCW applications" and "multi-user applications," and of course "groupware" has the ring of a product marketing concept.

The tie between *groupware* and product development is natural: Justifying a project to develop software for one group may be difficult, but the allure of the multitude of "groups" in the working places of the world as potential users is understandable. For *CSCW*, with its more exploratory approach to technology and workplaces, the identification with product development is not so tight. The result is a convergence of workers from product and internal development contexts, bringing some similar and some different issues and approaches.

THE INFLUENCE OF LARGE SYSTEM DEVELOPMENT ON CSCW

Groupware development highlights several issues rarely encountered in single-user application development. Social, political, and motivation aspects of computer use environments that can be ignored in designing a word processor, programming language, or business graphics application suddenly become important. Individual impact in terms of productivity or preference may be measured relatively easily; organizational impact is more complex. Software acceptance is highly dependent on a host of factors in the workplace, including management attitude, training and support, and group composition.

These issues were occasionally anticipated in the product development literature (e.g., Ehrlich, 1987), but have received steady attention in the internal systems development context for a decade (Friedman, 1989). In-house development targets specific groups of users -- it does not have the luxury of putting its product on the market to find customers. Researchers and developers working in this context have had to deal with the complexities of group dynamics, organizational change, and acceptance problems. Researchers have experimented with observational and quasi-experimental approaches to the issues. New systems development approaches have also evolved, such as the sociotechnical

[5] This includes 5% from the difficult-to-categorize telephone operating companies.

systems approach in England and participatory or collaborative design in Scandinavian countries. Some work in this field has been able to ride the declining cost curve down from organizations to groups: expensive Group Decision Support Systems that could once only be justified for upper management have evolved into less expensive applications that could support almost any team, for example.

Researchers from the organizational system / internal development context may have been surprised and pleased in equal measures when some product developers turned to issues that they had worked on for years. But the match is not perfect. Many of the CSCW researchers who come from the internal development context are committed to developing systems for organizations -- hence calls for including Computer Integrated Manufacturing in CSCW, for example. This may not appeal to product developers. Also, development approaches, such as the Scandinavian approach, evolved in the context of internal development. They may not easily be understood by researchers or developers from the product development context and "porting them" may not be easy (Grudin, in press). To summarize, tension within the CSCW field that has been attributed to conflicting "cooperative work" and "computer support" camps is at times due more to conflicting preferences for organizational issues vs. smaller group issues, for systems vs. applications.

THE INFLUENCE OF PRODUCT DEVELOPMENT ON CSCW

If the "outer ring" contributes familiarity with the social issues and possible approaches to dealing with them, the "inner ring" contributes new challenges, most of the workers (in the United States), and a new set of constraints on the solution space. The interest in "CSCW application areas" such as meeting management, decision support, co-authorship, project management, and electronic mail comes mainly from product development (and academic research). Product development provides opportunities unfamiliar to large systems developers, such as development costs amortized over many sales and no need to satisfy a particular customer. These opportunities are accompanied by new constraints, including less well defined or accessible user populations, more tightly controlled development schedules, and a reduced ability to influence organizational change in user environments.

Researchers and developers from product organizations also bring *motivation* to the CSCW melting pot, born of a less than impressive groupware success rate and a growing awareness that the issues are complex and require new approaches. For example, the Scandinavian researchers have been working for over a decade, but the interest in their work is suddenly intense (books and reviews recently published in the United States include Bjerknes, Ehn, and King, 1987; Suchman, 1988; Ehn, 1989; Floyd, Mehl, Reisin, Schmidt and Wolf, 1989; Bødker, 1989; Greenbaum and Kyng, 1990; Schuler and Namioka, in press). The interest is not uniform, of course, and there is a hesitation on *both* sides as people push beyond the shared interests to discover the differences, and confront the degree of adjustment needed to communicate clearly and to profit from one another's experiences.

DEFINING GROUPWARE

Much of the time it may not matter precisely what is and is not considered "groupware" or "a CSCW application." Each software object has its own set of system contexts and work contexts and needs to be considered accordingly. However, when Kraut (1990) writes "the only successful CSCW application has been electronic mail," and Bannon and Schmidt (1990) declare that electronic mail is an "enabling technology" and *not* a CSCW application, the terms are being used differently and establishing a common usage would help. In principle, any agreement would be workable; in practice, there is concern that software not categorized as a CSCW application (or groupware) will receive less attention from researchers (or popularizers and consumers).

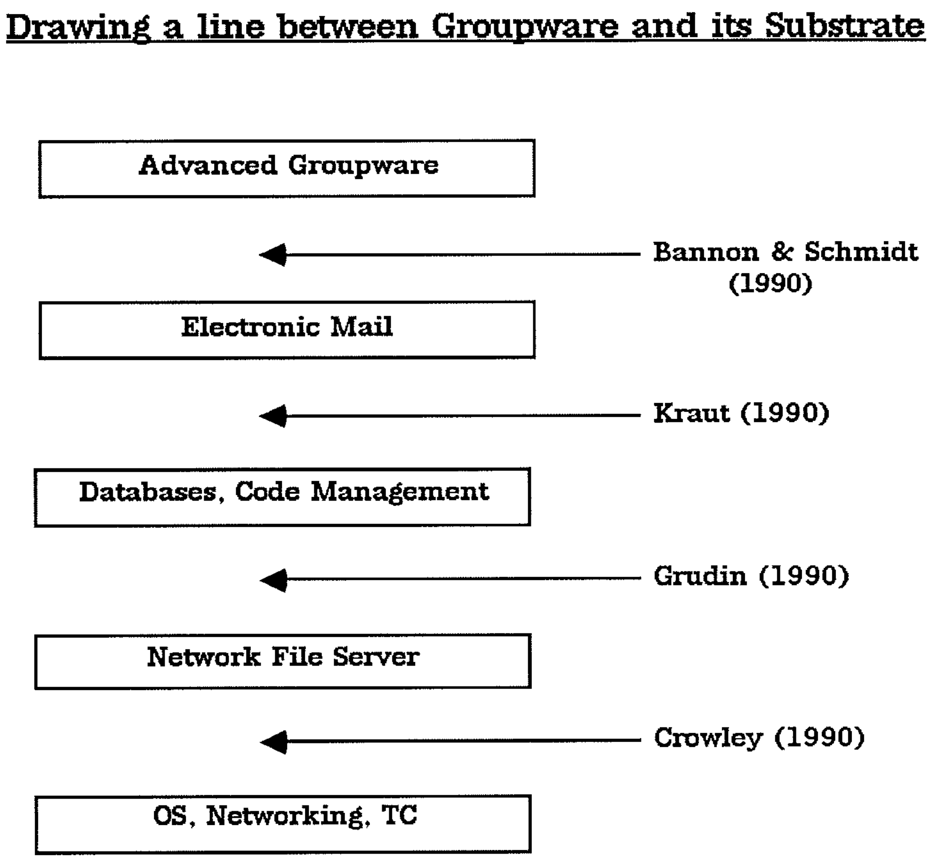

Figure 2. Everything above each arrow is considered groupware or "CSCW applications" by the author(s) to its right; elements below an arrow are considered a foundation. (Grudin, 1990, is agnostic but discusses databases and other object management systems.)

The range of opinion to be found is illustrated in Figure 2. At each point, software above the line can be meaningfully distinguished from that below. Crowley writes that in the PC and workstation worlds, "...the lack of a shared file system was the biggest impediment to cooperative work." Moving up a step, I have found it useful to discuss "object management" software along with more standard groupware applications because these products, used in group settings, have been more successful and offer interesting contrasts. But a database is not "group-aware," in Rein's (1990) words -- apart from password protection, it has no sense of different individuals or their roles. Only in the initial inspiration of these systems did the designer consider the needs and dynamics of groups. Thus, we might join Kraut and consider them to be the *foundations* on which to build CSCW applications[6] -- for example, a database that considers people's roles in alerting them to relevant changes. Finally, Bannon and Schmidt apply the same argument to electronic mail, which recognizes no role beyond the sender-receiver distinction but is a foundation for advanced applications such as The Coordinator or Object Lens.

The issues raised by these authors is more important than choosing one arrow in Figure 2 to make thicker than the others, but I will use this opportunity to suggest that we try to adopt the computer *users'* perspectives and understand the distinctions *they* make.

At the lower end, engineering distinctions may be transparent to users -- users will notice something built on a network file server, but not the file server per se. On the higher end, electronic mail users define their own groups through practice (and perhaps with aliases and distribution lists), imbuing electronic mail with clear qualities of CSCW or groupware -- in effect, users extend the design they are given. In the middle, the perception of applications such as databases and code management systems may vary with the context. If so, we cannot avoid considering the context in our discussions, and more may be lost than gained by categorizing the application in advance.

CONCLUSION

Whether or not we are involved in CSCW research or groupware development ourselves, we can learn from this dynamic field. We can see the influences of the different contexts in which interactive systems are developed. Over the past decade or two, these development contexts have faced different problems and developed different approaches. By watching them reconverge and relearn a common language, as is happening in the CSCW field today, we may get a better sense of our own position in the rapidly-changing computer systems world. We may discover similar opportunities to benefit from experiences that are very unlike our own.

---

[6] However, some anonymous reviewers are upset by a failure to define databases, version control systems, etc. to be successful groupware.

REFERENCES

Bannon, L., Bjørn-Andersen, N. and and Due-Thomsen, B., 1988. Computer support for cooperative work: An appraisal and critique. In Bullinger et al. (Eds.), *EURINFO'88: Information systems for organizational effectiveness.* Amsterdam: North-Holland.

Bannon, L. and Schmidt, K., 1990. CSCW, or What's in a name? Manuscript.

Bjerknes, G., Ehn, P. and Kyng, M. (Eds.) (1987). *Computers and democracy: A Scandinavian challenge.* Brookfield, VT: Gower Press.

Bødker, S., 1990. *Through the interface: A human activity approach to user interface design.* Hillsdale, NJ: Lawrence Erlbaum Associates.

CACM, 1970. Data processing industry assessed by consultant. *Communications of the ACM, 13,* 11, 704.

Crowley, T., 1990. In Ensor (1990).

Ehn, P., 1989. *Work oriented design of computer artifacts.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Ehrlich, S.F., 1987. Strategies for encouraging successful adoption of office communication systems. *ACM Trans. on Office Information Systems, 5,* 340-357.

Ensor, R. (Moderator), 1990. How can we make groupware practical? In *Proc. CHI'90 Human Factors in Computing Systems* (Seattle, April 1-5), 87-89.

Floyd, C., Mehl, W.-M., Reisin, F.-M., Schmidt, G. and Wolf, G. (1989). Out of Scandinavia: Alternative approaches to software design and system development. *Human-Computer Interaction, 4,* 4, 253-349.

Friedman, A.L., 1989. *Computer systems development: History, organization and implementation.* Chichester, UK: Wiley.

Greenbaum, J., 1979. *In the name of efficiency.* Philadelphia: Temple University Press.

Greenbaum, J. and Kyng, M. (Eds.), 1990. *Design at work: Cooperative design of computer systems.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Grudin, J., 1990. Seven plus one challenges for groupware developers. Manuscript submitted for publication.

Grudin, J., in press. Obstacles to participatory design in large product development organizations. In Schuler and Namioka (in press).

Irons, E.T. and Djorup, F.M., 1972. A CRT editing system. *Communications of the ACM,* 15, 1, 16-20.

Kraut, R., 1990. In Ensor (1990).

Levy, S., 1985. *Hackers.* Garden City, NY: Anchor Press/Doubleday.

Rein, G., 1990. In Ensor, 1990.

Schuler, D. and Namioka, A. (Eds.), in press. *Participatory design.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Suchman, L., 1988. Designing with the user. *ACM Trans. on Office Information Systems, 6,* 173-183.

SEVEN PLUS ONE CHALLENGES FOR GROUPWARE DEVELOPERS

Jonathan Grudin

Design and evaluating applications to support groups introduces challenges that do not affect single-user applications or information systems that support organizational goals. Repeated, expensive failures have resulted from not recognizing and meeting these challenges.

INTRODUCTION

Computer applications that are designed to support groups are commonly known as groupware. The 'groups' are relatively small or are narrowly focused, as in the case of a 'computer conference' interest group. Although electronic mail and bulletin boards or conferences are the best known examples, considerable effort has gone into developing other groupware applications. As networks spread, we will see more voice applications, coauthoring tools, intelligent databases, group decision support systems, and other applications designed to support work in its social context. However, progress may be slow and may go in unanticipated directions. This paper first describes problems that have led to expensive failures of groupware development efforts, then examines groupware successes to find better approaches to computer support of work in group settings.

Groupware, also called work-group computing or multi-user applications, fits into the software universe somewhere between single-user applications and information systems that support organizations. A field is emerging that looks at how the following fit together, or might fit together: individual computer users, groups, and organizations; applications designed for individual users but often used in group and organizational contexts, groupware designed to support multiple users while interacting with each person individually and also adjusting to organizational contexts, and systems developed to support organizational goals that must of course act through individuals and groups. This field is called computer-supported cooperative work, or CSCW.

The terms groupware and CSCW were coined in the mid-1980s. Many of the issues encountered in groupware development are discussed in the "Information Technology" literature, but there are important differences. The IT focus is on systems designed to support organizational goals, such as order-and-inventory-control, computer integrated manufacturing or large office automation systems. These systems are often developed in-house or tailored for a specific customer. Groupware comprises applications supporting more restricted goals and is typically developed by product development companies.[1] Internal development and product development are very different contexts, with different objectives, constraints, and modes of operation. Problems may occur in one context that do not occur in the other, or may occur in different ways with different weights.

One important distinction is that product developers focus on design and evaluation in producing an application with broad appeal, whereas systems development efforts place more emphasis on system acceptance by a specific set of computer users. Product developers generally have very limited involvement in or control over the installation of their software. Internal developers have a greater investment and involvement in installation. Therefore, the IT literature has emphasized issues affecting the acceptance of organizational systems, whereas the groupware literature has focused more on design and evaluation. This paper follows the latter course, presenting seven challenges to groupware design and evaluation. However, acceptance is trickier and more important for groupware than for other application product development. Difficult as it may be for "off-the-shelf" product developers to jump over the counter and help out with product acceptance, they may have to try: the eighth challenge for groupware developers.

The term 'groupware' focuses on technology, the technology supporting cooperative work. Although most work is carried out in a social or group context, 'groupware' designers explicitly consider the multi-person aspect, in which individuals take on different roles. Almost any useful technology can claim to support group activity; at issue is whether this is reflected in the design or is incidental to its use. Word processing does

[1] Electronic mail may be available throughout an organization, but electronic mail transactions overwhelmingly support group functions.

not become groupware if different users access and edit the same document sequentially. A co-authorship application might support document preparation as a group activity by allowing co-authors to work simultaneously, by supporting their communication, by identifying edits according to the user making them, or by interacting with users according to their role -- principal author, co-author, editor. Similarly, if a database provides multi-user access but does not distinguish among users beyond password recognition, is it groupware or is it a foundation upon which to build groupware? This question is revisited below.

A note on sources: My interest in these issues arose from living through the failure of applications that as a developer or user I had embraced enthusiastically. I worked on, or belonged to a small group that worked on, time management, project management, group decision support, voice, and other groupware applications or features. My own work tools have included time management (including meeting and resource scheduling) and voice applications, a group outline editor, bulletin boards, and electronic mail. Some observations are supported by a survey of over 200 designers (Grudin and Poltrock, 1989). The published literature in this area is growing rapidly -- 17 of the 40 references were published in the past two years. And conversations with many other developers and researchers have provided insight into experiences that all too often remain undocumented.

PROBLEMS IN GROUPWARE DEVELOPMENT

Why is groupware difficult to design and evaluate? Since individuals interact with a groupware application, it has all the usual interface design problems. In addition, group members with different backgrounds, experiences, and preferences may all have to use the same groupware application. For example, different individuals may choose to use different word processors -- or some may choose to write by hand -- but two co-authors must agree to use the same co-authoring tool! Each of the several people to be supported as they work together on a task has a different and potentially shifting role. Various social, motivational, political, and economic factors that can be ignored in designing single-user applications (e.g., word processors or spreadsheets) are central in the design of groupware (e.g., a system to support meeting management).

Group processes are difficult to study. They are often variable, context-sensitive, and unfold over a longer interval than individual activities. Key activities may occur relatively infrequently. Weeks may be needed to observe the pattern of use of a group calendar or even electronic mail system, with the further complication that the activity takes place in different locations. Organizational change that results from introducing the technology may take even longer to observe. And finally, it is hard to generalize -- each group is different and their experiences with an application are highly influenced by the conditions under which it is introduced.

Thus, the complexity means that no one person's intuitions are likely to provide the necessary insight, yet acquiring understanding is particularly difficult. Further analysis of these problems enables us to identify seven major factors contributing to failure of groupware:

1. Groupware applications often require that some people do additional work, while those people are not the ones who perceive a direct benefit from the use of the application.

2. Groupware may lead to activity that violates social taboos, threatens existing political structures, or otherwise demotivates computer users who are crucial to its success.

3. Groupware may not allow for the wide range of exception handling and improvisation that characterizes much group activity.

4. Groupware may not reach the "critical mass" of users required to be useful, or may not be used because while it would be beneficial, it is never to any one individual's advantage to use it (the "prisoner's dilemma" paradox).

5. Features that support group processes may be used relatively infrequently, presenting challenges of unobtrusive accessibility and integration with more heavily used features.

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

7. Our intuitions are especially poor for multi-user applications, resulting in bad management decisions and a more error-prone design process.

Confronting each of these challenges requires both a greater understanding of the intended use environment and a corresponding adjustment in the development environment. However, progress on the first four requires primarily a greater understanding of the intended users' work environments, whereas the latter two primarily argue for changes in the development environment to remedy known problems. The fifth challenge requires a balanced effort in both design and use environments.

Application developers traditionally provide relatively little support for their customers: documentation, on-line help (perhaps), and training (sometimes). Such is the nature of selling products. Unfortunately, groupware can be extremely sensitive to aspects of its introduction -- so much so that software that is more or less thrown over the transom in the usual fashion may have negligible chance of being accepted. Therefore, the eighth challenge differs in that it may require application developers to change their basic conception of their product:

8. Groupware requires extreme care regarding the circumstances of its implementation (introduction) in the workplace.

Not all of these challenges confront every effort. Every application and use situation is unique. The developer must carefully decide which challenges apply.

1. THE DISPARITY BETWEEN WHO DOES THE WORK AND WHO GETS THE BENEFIT

Given the different preferences, experience, roles, and tasks of members of a group, a new groupware application never affords every member precisely the same benefit. When it is introduced, a collective benefit is expected, but some people have to adjust more than others. Ideally, everyone will benefit individually, even if some benefit more; however, this ideal is rarely realized. Most groupware requires some people to do additional work to enter or process information that the application requires or produces.

Consider the automatic meeting scheduling feature that accompanies many electronic calendar systems. The underlying concept is simple: the person scheduling the meeting specifies a distribution list and the system checks each person's calendar, finding a time that is convenient for everyone. The immediate beneficiary is typically a manager or secretary who convenes a meeting, but for the feature to work efficiently, everyone in the group must maintain a personal calendar and be willing to let the computer schedule their free time. Otherwise, the scheduling program will create conflicts by scheduling meetings in time that only seems to be open. However, electronic calendars are typically used as communication devices by managers and are often not maintained by individual contributors (Ehrlich, 1987a). Thus, successful use of automatic meeting scheduling requires additional work for those group members who would not otherwise maintain electronic calendars. As a result, this groupware feature is rarely used.

Similarly, consider voice products, such as voice annotation to documents. The advantages of digitized voice over handwritten or typed input are almost all advantages for the speaker: speaking is faster than writing or typing, conveys emotion and nuance easily, and may be transmitted by telephone. The disadvantages to digitized voice, however, are overwhelmingly problems for the listener. It is harder to understand than typed or written material, slower to take in, not easily scanned or reviewed, more likely to contain errors, and more difficult to manipulate -- for example, proposed edits to a document will have to be typed in by the listener anyway. When is it acceptable for speakers to burden listeners thus? Possibly when users speak and listen in equal measure, or when there is no alternative, as when 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 an executive's time or convenience is considered valuable enough to warrant the arduous transcription effort. But in general, the disparity in effort and benefit may work against acceptance in many situations and help explain the failure of voice products to meet expectations (Aucella, 1987).

As a third example, consider a project management application on a distributed system, covering 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., Sathi et al., 1986). The primary beneficiary is the project leader or manager, but to be used successfully, other group members must enter considerable information that is not typically kept on-line. This may be resisted. For example, McCracken and Akscyn (1984) describe a ten year project to develop a "computer-assisted management system" for an aircraft carrier, "its primary purpose to help the Commanding Officer and his department heads administer the ship." One factor contributing to its eventual replacement by a system that lacked management features was the difficulty of getting everyone to use it (Kling, 1987).

2. 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 the 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 our awareness of the personalities and priorities of people around us, knowledge not easily made available to the computer. For example, secretaries know that managers' unscheduled time is rarely really free; unauthorized scheduling of a manager's apparently open time can lead to total rejection of automatic meeting scheduling (Ehrlich, 1987a). Even trying to make tacitly understood personal agendas and priorities explicit may be a problem -- we often tactfully leave such matters unspoken. Yet unless such information is made explicit, groupware will be insensitive to it.

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 programs that periodically opened files and changed dates, which satisfied the watchful, automatic monitor. Thus "sabotaged," the work management system was of little use and was eventually quietly withdrawn.

Meeting management or group decision support systems have also failed to meet expectations (Kraemer and King, 1988). The appeal of improving the efficiency of meetings is clear, but the decision-making process is often complex and subtle, with participants holding partially hidden agendas, relying on knowledge of the personalities of the others involved, and showing sensitivity to social customs and motivational concerns. Since such factors are never represented explicitly in a support system, the computer participates at a great disadvantage. In one case, a leadership group considered using an issue-based information system in which arguments, counter-arguments, and decisions are entered by participants, creating a record of the decision-making process that can be used for subsequent review and exploration of alternatives. The plan to use the system was abandoned because the manager wanted the group to project a strong impression of consensus; the explicit record of opposing positions that the system would immortalize was politically unacceptable.

3. EXCEPTION HANDLING IN WORKGROUPS

Software may be designed to support group activities or procedures as they are "supposed to" happen, but descriptions of "typical" procedures can be misleading. Suchman (1983) argues persuasively that a wide range of error, exception handling, and improvisation are characteristic of human activity. Group activity may be particularly variable -- strict adherence to a standard procedure may be more the exception than the rule. But given the overall difficulty of developing software to support group activity, the desire to build the design around specific work procedures may be especially strong.

The problems that can result are illustrated in a case study by Rowe (1985, 1987). 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, are eager to keep production operating 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 that they could amend their Friday order up to 1:00 pm on a Monday... No doubt it 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 "became convinced that someone was sabotaging the system."

Here Suchman's observation is critical: if more human activity is *ad hoc* problem solving then we realize, if descriptions of "standard process" are often *post hoc* rationalizations, then the workers' behavior that seemed pathological to the computer services division may well have been an optimal response to the work environment. After all, catering to the needs of specific customers is often considered a virtue, not a vice.

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 system was made to work. This anticipates our discussion of groupware acceptance. The system described by Rowe was a large, expensive system: upper management was prepared to do a lot to make it succeed. But a typical groupware application or feature, such as meeting scheduling or voice annotation or even meeting support, will rarely have the same degree of visibility and backing -- and thus would fail under similar circumstances.

4. CRITICAL MASS AND PRISONER'S DILEMMA PROBLEMS

Most groupware is only useful if a certain percentage of group members use it. The percentage may be high -- a co-authorship system, for example, may not help much unless all authors use it. Ehrlich (1987b) noted the importance of a "critical mass" of users for communication systems. Even one or two defections may cause problems for meeting scheduling, decision support, or project management applications. Markus and Connolly (1990) point out that even in an idealized situation in which every individual would benefit from the use of a groupware application, they may never use the application long enough to discover this. The early adopters will not find it useful and may abandon it before the critical mass is reached. Grudin (1989) describes an expensive voice messaging system that failed initially to reach a critical mass of users: those who did leave messages became discouraged because all too often the receiver did not use the system. However, this system became highly successful, even in the eyes of some initial detractors, when top management forced its use by removing the alternative (receptionists taking messages). But that was an expensive organization-wide system; a less expensive groupware application is unlikely to be given such a forceful shove past the critical point. A less visible voice annotation feature will not be made obligatory, employees with discretionary control over their own time may not be forced to keep on-line calendars, and so forth.

Markus and Connolly (1990) also point out the possibility of "prisoner's dilemma" situations, in which *everyone* acting in their own best interest leads to a worse situation not only for the group but for each individual. They suggest that with some discretionary databases, as long as *anyone* updates them, your optimal strategy is to "freeload," but of course if *everyone* tries to "freeload," the system is not used at all. A critical point also exists here: the number of non-freeloaders who must use it before each user gets more out of it than he or she puts in.

These analyses show that even equalizing costs and benefits for all groupware users (and insuring a net benefit) would not assure success. Markus and Connolly suggest management solutions, such as mandating system use. But this particular approach seems unpromising for low-visibility groupware applications. It seems preferable, where possible, for designers to build in features that do provide strong enough incentives to users. If this is impossible, product developers might consider directing their efforts to projects with more chance to succeed! This may seem obvious, but there is an almost lemming-like drive to embrace certain ill-starred applications, which is the subject of the seventh challenge.

5. RECOGNIZING AND DESIGNING FOR INFREQUENTLY USED FEATURES

We have a natural tendency to exaggerate the importance or frequency of the objects or events that we are attending to. If "to a hammer, everything looks like a nail," then perhaps to a groupware designer, every work situation calls out for communicative or social support. To maintain perspective, recall that many organizations are structured and responsibilities divided *in order to minimize* the overall communication requirements and social interdependencies. The decrease in efficiency that accompanies increase in size, much of it due to increased communication overhead, is well known. Work does have important social elements that call out for support, but groupware features will generally be used much less frequently than many features supporting individual activity. This has two important implications.

First, it may be necessary or desirable to integrate groupware features with those supporting frequent individual activity. Consider co-authorship applications. Anyone who has written collaboratively may visualize the potential benefit, but most writing is done alone, whether on single-authored efforts or on a section of a jointly written document. As a result, most people would be unwilling to abandon their favorite word processor in order to use a co-authorship application. Features to support co-authorship need to be integrated with those already supporting authorship. A corollary of this is that stand-alone groupware applications may not justify high purchase costs and be perceived to fail simply because they are relatively infrequently used. How often, really, do you call a meeting, manage a meeting of the size that a group decision support system could facilitate, etc.? Rather than as applications launched with fanfare, groupware may succeed as features slipped into already successful applications.

This brings us to the second implication. If groupware features are relatively infrequently used elements of frequently-used applications, developers must keep them in the background so they do not impede more frequently-used individual-use features, yet insure that they are known to and accessible by users when it *is* time to use them. These conflicting requirements are faced by the designers of any relatively infrequently-used feature; it is perhaps the greatest interface challenge of the future.

6. THE UNDERESTIMATED DIFFICULTY OF EVALUATION

Task analysis, design, and evaluation are never easy, but they are far more difficult for multi-user applications than for single-user applications. An individual's success with a particular spreadsheet or word processor is unlikely to be affected by the differing backgrounds or by the personalities of other group members. Single users can be tested in a laboratory on the perceptual, motor, and cognitive aspects of human-computer interaction that are central to many single-user applications, but lab situations cannot capture the social, motivational, economic, and political dynamics that often strongly affect the use of groupware. In addition to being used in this more complex setting, groupware itself has more complex requirements, needing to interface simultaneously to users with different roles, backgrounds, and preferences. And due to the ways these factors interact, evaluation of a partial prototype may not capture crucial influences; a full implementation may be required.

The evaluation process is far more time-consuming. 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. Furthermore, the evaluation methods used are less precise. Evaluating groupware in the field is remarkably complex due to the number of people to observe at each site, the wide variability of group composition, and the range of environmental factors that play roles in determining how it is used. The evaluation of groupware requires an approach based on the methodologies of social psychology and anthropology. These skills are absent in most development environments, where human factors engineers and cognitive psychologists are only starting to be accepted.

Finally, the greatest problem in generalizing from experience is that establishing success or failure is much easier than identifying the underlying factors that brought it about. A highly-motivated group may find a way to use a seriously flawed product, and a badly-managed installation may cripple a good product, with the result that one generally finds both successes and failures.

7. THE BREAKDOWN OF INTUITIVE DECISION-MAKING

Often the problem lies not in the detailed design of an application but in its very conception: decisions are frequently made to develop unworkable systems. To understand why, note that decision-makers in product development environments rely heavily on intuition. The experience and track record of today's development managers is likely to be based on single-user applications. Intuition can be a far more reliable guide to single-user applications than to multi-user applications. A manager with good intuition may quickly get a feel for the user's experience with a word processor or spreadsheet, for example, but fail to appreciate the intricate requirements of a groupware application that requires participation by a range of user types.

Decision-makers are often drawn to applications that selectively benefit one subset of the user population: managers. Consider these active groupware development areas: 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 may see themselves as prospective casual users of features such as a natural language interface and support its development, without recognizing their limited utility or development cost (Grudin, 1989).

This bias is understandable -- each of us has ideas about what will help us do our job. But managers may underestimate the down side, the unwelcome extra work required of other users to maintain such an application, a burden that often leads to neglect or resistance. For example, a group decision support or work management application may require many people to learn the system and enter data, it may record information that participants prefer not to have disseminated, and it may block other means to influence decision-making, such as private lobbying. The decision-maker's intuition will fail when an appreciation of the intricate dynamics of such a situation is missing. Also, as described above, managers may fail to appreciate the difficulty of developing and evaluating good groupware. Finally, they may not recognize that as systems and applications become cheaper and thus less visible to management, users will less often be forced to do additional work to insure their success.

No one has good intuition for multi-user applications. As researchers, designers, implementers, users, evaluators, or managers, our computer experience is generally based on single-user applications. This history determined the skills we acquired, the intuitions we developed, and the way we view our work. For example, human factors engineers are trained to apply techniques based on perceptual, motor, and cognitive psychology to study phenomena of brief duration but are unfamiliar with the techniques of social psychology or anthropology needed to study group dynamics over time.

In particular, we are not trained to consider many of the issues discussed here. Once a project is set in motion, researchers or developers may rely on feedback from a few potential users. Often these are the expected beneficiaries of the application, and often that means managers. For example, while the greatest interface challenge for an intelligent project management application may be to minimize the information entry effort required of each worker or provide compensatory benefits for doing this work, attention may instead be 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 is that they be able to do so without great effort," (Sathi et al., 1986). This may appeal to the manager sponsoring the project, but it is not wise to focus exclusively on designing groupware for the principal beneficiary, whose satisfaction may be critical but who may be relatively highly motivated from the start.

The converse intuition failure also occurs: a decision-maker may not recognize the value of an application that primarily benefits non-managers, even when it would provide a collective benefit to groups and organizations. This is particularly true for applications that might create additional work for managers. This point is addressed below in the context of electronic mail: a groupware success story.

REVIEWING THE RECORD

The same mistakes are made over and over again. The proponents of a product call attention to the successes and quietly move on if, as often happens, the product fails. Thousands of developer-years and hundreds of millions of dollars have been committed to various application areas that could be termed groupware despite little or no return. Consider two of the oldest areas of research and development: A review of group decision support systems concluded that after decades of work, "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," Kraemer and King (1988). At a panel titled "Voice: Technology searching for communication needs," it was stated that after 25 years of research, no company specializing in voice technology had become profitable and that projected sales of voice products had been revised sharply downward (Aucella, 1987). In most instances of failure, of course, a substantial and timely return on investment had been anticipated.

Nor is the pattern confined to the commercial world -- each generation of researchers may rediscover these problems in technologically flashier forms. Two examples: Zellweger (1990) shows a fictional enactment of the use of a new voice editor. Someone receives a voice message containing directions for driving to a party, which they carefully edit and forward to a third person. Since the final recipients must transcribe the directions, probably requiring several passes through the voice message, a substantial burden has been given to the editor-intermediary and recipients. Considerable reduction in effort would result if anyone in the chain simply typed in the directions; in a real situation this would hopefully happen. Similarly, Beard et al. (1990) report on an automatic meeting scheduler developed in apparent ignorance of the fate of almost a decade of commercially available meeting schedulers. The system encountered interesting if predictable problems: unwillingness to place true priorities on a public system, incomplete adoption of the system by group members, etc., but the developers concluded only that if the system is rewritten in C to run on a more convenient computer, these problems may disappear. Evaluating groupware is difficult.

Others have noted the lack of progress. For a CHI'90 panel titled "How can we make groupware practical?" Bob Kraut wrote "the only successful CSCW application has been electronic mail," (Ensor, 1990, p. 88). Lee Sproull wrote "Groupware will never be practical and widely used in organizations if it follows its current trajectory," (p. 89). She argued for a focus on either applications for individuals or systems to support entire organizations -- essentially denying the possibility for groupware as typically defined: marketable products developed with an expanded, group focus.

A SOURCE OF POOR INTUITIONS: TWO MISLEADING ANALOGIES

Groupware lies between two worlds. It is a new class of products attracting interest in organizations heretofore focused on single-user applications and it encounters problems in organizational dynamics heretofore experienced primarily by internal or contract-based IT projects. These provide the two natural but ultimately misleading analogies: 1) between multi-user applications (groupware) and multi-user systems (supporting organizational goals), and 2) between multi-user applications and single-user applications. Designers and decision-makers fail to recognize the limits to the analogies, to intuitions based on these different experiences.

I have so far stressed the differences between groupware and single-user applications. Now consider the second potential source of confusion: similarities and differences of groupware and more familiar systems introduced to support organizational goals. Here we do not refer to central, timesharing computers that primarily support individuals and single-user applications, but rather to hardware and software systems developed to support organizational goals, such as management information systems, computer-integrated manufacturing systems, or inventory control systems.

These highly visible, expensive systems arrive with an anticipation of substantial collective benefit and an expectation of some organizational change. Upper management may express its commitment through a) job redesign or creation: e.g., word processing skills are made a job requirement for new secretaries and a new database administrator job is created; b) support for training and education of the users to the collective benefit, which may create a willingness to support the use of the system; c) reorganization to work around important individuals who do not use the system (e.g., a manager who will not use a terminal); d) positive leadership through inspiration or example. These forces may work to the advantage of the system, but even so, successful implementation is not assured.

In contrast, less expensive groupware applications will rarely have the same level of management commitment. It has some advantages -- most users may be familiar with the computer system already in place; the smaller group may be relatively homogeneous and share many goals. But the organization cannot restructure itself around each new application, nor will management work as hard to ensure full participation. For example, 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 a computer system, but an application program must adapt to the organization, fitting into existing work patterns and appealing to all the people needed to support it. A system may provide group members with several (single-user) text editors to choose among, for example, but all must agree to use the same group decision support system.

Thus, our final challenge is a merger of two converging problems: as product developers, groupware designers must pay greater attention to the problems of system acceptance than they have before, just when these problems are becoming a greater challenge!

8. THE DYNAMICS OF ACCEPTANCE: A NEW CHALLENGE FOR PRODUCT DEVELOPERS

The Information Technology field has been substantially concerned with system acceptance or adoption -- often called "implementation," (a term used synonymously with "development" in many product organizations); e.g., Lucas (1976), Lyytinen and Lehtinen (1987). In application development environments, however, designers and developers are shielded from acceptance or implementation concerns. This is partially due to the nature of "off-the-shelf" product marketing, where the buyer exchanges the right to choose among alternatives for the responsibility for finding an acceptable one, but mediators also play a role in bringing about acceptance. Within the development organization, they include groups such as marketing and customer support or field service. Training materials or programs are often developed by groups located outside software development and even documentation may be produced by writers who have minimal contact with application designers and developers. User organizations may rely on consultants, internal developers, or other groups to help tailor or supplement the product and oversee implementation or acceptance.

The mediation that works for individual applications and general-purpose systems may be unreliable for groupware applications that are more sensitive to social dynamics. Designers and developers may not learn enough about user environments to develop useful systems, and conversely I have seen marketing representatives ignore carefully researched recommendations for implementation. While these mediators may be a fact of life, product designers and developers should be aware of those factors independent of the quality, utility, and even usability of a product that can determine its acceptance. Discussion of application or small system acceptance has entered the research literature (e.g., Gaffney, 1985; Ehrlich, 1987b). Ehrlich's strategies for encouraging successful adoption of "office communication" systems, broadly defined to cover several groupware categories, are summarized here.

First, a group's problems must be identified and the computer solution matched to it. For example, geographic proximity of group members may indicate whether voice or electronic mail, or synchronous or asynchronous decision support might be appropriate. Next, appropriate pilot groups and individuals must be selected. Systems may fail if placed on executive desks when their secretaries would be more appropriate, or if restricted to secretaries when professionals should be included, for example. The adopting group should be given a clear understanding (perhaps through a site visit) of what the mature use of the application might be, to overcome uncertainty; providing education that demonstrates the positive impact on the work day may help here. Also, step-by-step training on unfamiliar features, even if not enough to insure later recall of the details of use, may reduce anxiety. Many observers have commented that the attitude of management is critical to acceptance. For less expensive applications, requiring less organizational investment, this may require particular attention. Finally, someone must be educated to anticipate early problems so that they can be dealt with immediately to prevent premature rejection, and follow-through support must be in place to handle the post-"honeymoon" period, when the group's curiosity wanes and need to get on with work moves back to center stage.

These considerations, crucial for systems introduced in group settings, are traditionally beyond the control of application designers and developers. Groupware developers may have to address them to succeed. They must insure that their product meets real needs (for instance, by involving prospective users in the process), support some of these requirements through the design itself, or by working more closely with the mediating groups (e.g., marketing) than they have in the past.

MEETING THE CHALLENGES

SHIFTING TO A WORK PERSPECTIVE

The problems described above suggest that the intuition-governed, technology-driven approach to progress that works relatively well with single-user applications is failing with multi-user targets. Some visionary writers have stressed the need for designers to understand more about how groups and organizations function and evolve, as well as more about individual differences. But it is easier to recognize the problem than to truly escape the technology orientation that is reflected in the term "groupware" itself.

In the world of large systems development, methodologies that truly focus on users' work and workplaces have been developed. Scandinavian researchers stress the importance of "workplace democracy" -- engaging the users or workers meaningfully in the design process, a slow mutual education process that results in users becoming true members of the design team. Such projects have brought systems developers into meaningful contact with labor unions, historians, and economic theorists, groups with a history of concern for social and motivational issues in group situations. These approaches are reaching application developers through books (e.g., Bjerknes, Ehn, and Kyng, 1987; Docherty et al, 1987; Ehn, 1989; Bødker, 1990; Greenbaum and Kyng, 1990) and conferences (e.g. the CSCW'88, EC-CSCW'89, and 1990 Participatory

Design Conferences). However, adapting these time-consuming, labor-intensive approaches to product development environments may not be easy, due to the very nature of product development (Grudin, 1990b). Methodologies for accomplishing this are only starting to appear (e.g., Whiteside et al., 1988).

Although many researchers and developers from product development companies have adopted the term "computer supported cooperative work," with its explicit focus on work as the central concern, a substantial technology-driven element remains. But the technology-driven, trial-and-error approach is proving to be too expensive and failure-prone in this area, suggesting that work-centered approaches must be examined. Yet, there have been successes in groupware development; their examination may offer some guidance.

VIEWING ELECTRONIC MAIL AND OTHER GROUPWARE SUCCESSES FROM AN ORGANIZATIONAL PERSPECTIVE

What are groupware successes? First, we must review the question raised in the previous paper: what is groupware and what is its substrate? The most inclusive interpretation argues that any application used successfully in a group setting qualifies. For example, Nardi and Miller (1990) find that spreadsheets may usually be developed and used by teams, even though they lack features designed to support this aspect of cooperative use. Crowley (in Ensor, 1990) includes network file servers as groupware, which is relatively low (see Figure 2, p. 12). In contrast, "object management systems" such as databases and code management systems, are often categorized as (potential) groupware *substrates* because they do not distinguish among users, apart from some password protection -- the design does not support different roles, preferences, etc. Kraut, in Ensor (1990), termed email the only groupware success, thereby drawing the line above object management systems. Most extreme are Bannon and Schmidt (1990), who consider even electronic mail to be a substrate for groupware, rather than being groupware itself. Without worrying too much about whether we are discussing successful groupware or successful groupware foundations, we will draw the line between Crowley and Kraut, although we will focus on electronic mail, broadly defined to include bulletin boards or computer conferencing.

The potential for electronic mail to augment group activity was foreseen thirty years ago; today, a variety of related computer-mediated communication forms have succeeded. How were the pitfalls avoided? Is electronic mail a potential model for groupware?

1) Who does the work and who benefits? Electronic mail provides an equitable balance insofar as sender and recipient are concerned. The person with a message to communicate does a little more work to type it, while the receiver can read it easily and whenever convenient; thus, the primary beneficiary typically does a little more work. 2) Compatibility with social practices: The essentially conversational format of electronic mail allows us to apply existing social conventions. However, there are differences, which lead to clearly identified problems such as "flaming," "junk email," "smileys," e.g., :-) and to more subtle but significant problems described below. 3) Exception-handling: the asynchronous, informal nature of most electronic mail makes it flexible, although mail applications have been developed that impose more structure -- and may suffer accordingly (e.g., Carasik and Grantham, 1988; Erickson, 1989; Bullen and Bennett, 1990). 4) Critical mass problems: these can affect the utility of email, although either one other user or a connection to any external bulletin board or conference may be enough to insure its utility. 5) Frequency of use: for many users email is quite heavily used and requires relatively little learning and recall. 6) Difficulty of evaluation: As with all groupware, the overall costs and benefits of electronic mail are difficult to assess. 7) Poor intuitions for groupware: Not all mail applications succeeded, but our intuitions on the topic may be improving as email use spreads.

Electronic mail has avoided most of these problems. That leaves our additional challenge: 8) Acceptance. An interesting anomaly is that email has spread less through the normal product development and marketing processes than by spreading from academic and public sources. Understanding this may be the key to understanding the future of groupware and the impact of information technology on organizations.

The key user distinction for electronic mail in many organizations is not that of sender and receiver, it is that of manager and subordinate. This is not because the *technology* recognizes the supervisor-subordinate distinction, but because that distinction is so critical *in the workplace.* We may have a bias to focus on the distinctions designed into the technology, but its reception in an organization is determined by the distinctions that exist there. In hierarchic organizations, the work and the rewards going to managers and other workers are quite different. We may downplay the differences and seek cooperation between labor and management, but some tension exists in most organizations large enough to profit from electronic mail. Even where tension is not evident, the roles of manager and subordinate differ.

Unlike most groupware applications, electronic mail does not selectively benefit managers or decision-makers. In fact, the contrary is probably true. Asynchronous interrupts may 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. The ability for anyone to disseminate information rapidly can create new and not always welcome challenges for managers whose jobs involve filtering and routing information. In a classic hierarchic bureaucracy, lateral communication is minimized -- information flows up and down through the hierarchy. The resulting inflexibility can lead to inefficiency; rigid bureaucracies, from the Soviet Union to the U.S. Navy, spawn tales of undercover exchange systems devised to cope with it. Electronic mail, even more than a telephone on each worker's desk, supports efficient lateral communication -- but may create difficulties for managers in organizations built on the hierarchical model.

One managerial responsibility may be to absorb information from higher levels and tailor its presentation to subordinates to maximize their understanding or obtain a desired response. But if such information is received electronically, it is easier to forward it without such tailoring. In fact, editing may be counterproductive, since other electronic versions of the message may be forwarded laterally into the group, revealing the tampering. This may place a manager in a no-win situation. Similarly, the ability of anyone to send a message instantly to everyone in an organization creates a volatility that management must cope with.

The anthropologist Constance Perin (1989) 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, Perin 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." One case study concluded that electronic mail "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," (Fanning and Raphael, 1986).

What are the implications for electronic mail and groupware built on it? While there are cases of individual managers discouraging or terminating its use, many organizations have assumed an overall benefit and successfully introduced it. Many students and professionals are now accustomed to it. Thus, the forces Perin describes may play themselves out over time. Organizations designed around notions of efficiency and control that become outmoded may evolve; finding new organizational forms and minimizing the cost of shifting to them are challenges for the near future.

"Object management" applications used by groups, such as databases and code management systems, share some properties with email. In particular, they primarily benefit not managers or decision-makers, but people who use computer systems more routinely in their work.

POSSIBLE DIRECTIONS FOR GROUPWARE DEVELOPMENT

This picture requires balancing. Technical obstacles that are in the process of being addressed may also be factors, such as the processing and storage requirements of voice technologies or lack of sufficient networking. Development is sometimes indirectly motivated; for example, one company may build an automatic meeting scheduler simply because it is technically easy -- after which its rivals may feel forced to do so to maintain competitive positions. On the positive side, the problems may be mitigated in some contexts. If an application targets relatively homogeneous groups, intuition may be a better guide, facilitating both evaluation and design. Also, most of the application areas described have enjoyed some successes and are potentially important. Investing resources adequate to the solution of the problems, developing the appropriate research and development methodologies, finding niches where the problems do not arise or where applications will succeed in spite of them, and adequately preparing users for the introduction of groupware are all approaches that may lead to success.

Beyond that, managers and developers must be educated -- about groupware, the risks involved, and the resources and approaches required to minimize the risks. We will have to learn more about how different kinds of people work together. Given the expense of developing groupware and the current prevalence of failure, time-consuming "front-end" activities -- careful needs-finding, prototyping, and iterative design -- may be cost-effective, leading to changes in software development practices.

While a groupware application may lead to gradual organizational evolution, its introduction must be smoother. This makes the job of the designer and implementer more difficult. Groupware applications will have to be more "group-friendly" than systems have been. The focus will shift to user interface issues to minimize the disruption. Approaching the ideal in which everyone benefits directly will require minimizing the extra work required of anyone or providing a compensatory benefit. It may require interfaces that vary according to a user's background, job, and preferences. This is a substantial undertaking.

We must also develop a better understanding of our own decision-making processes as researchers and developers. Too often 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.

Since executives and managers are typically not heavy computer users, it may not be surprising that the most successful groupware applications have been the few whose primary beneficiaries were non-managerial. This has several implications for groupware developers.

First, if an application will primarily benefit managers or decision-makers, focus strongly on finding ways to provide benefits for other group members. One approach is to supplement the technology with a design for the *process of its use*. The importance of carefully managing technology use has been reported in two successful implementations of group decision support systems (Whiteside and Wixon, 1988; McGoff et al., 1990). In the latter case, the process of use ensured a strong, visible buy-in by management and clear benefits to all participants.

Second, look to existing computer use as a starting point. Extend or build on single-user applications used in group settings. The group use of spreadsheets reported by Nardi and Miller (1990) might be facilitated through the addition of specific features. Similarly, features supporting co-authorship could be built into a successful word-processor. Co-authors who exchange drafts by email would likely use a facility that automatically converts a document to an interchange format and posts it to co-authors, perhaps assigning a version number -- *if* it did not require learning a new word processor.

Third, focus on object management applications, which have so far fared better than process management applications. Building on electronic mail offers particular promise. One ambitious development effort is Object Lens (Lai and Malone, 1988), whose users can fill in message templates and other forms to filter or share information in varied, flexible ways. Networks are already being used to guide research, development, and even marketing (Perlman, 1985). Grudin (1990a) describes real and speculative extensions of "electronic markets" (Malone et al., 1987).

An email system may be likened to a highway through an organization, an information-bearing interstate highway that may cross organizational boundaries. Pursuing the metaphor for a moment, the U.S. interstate highway system was designed for a military purpose that is not reflected in its subsequent use, which led slowly to changes in transportation systems (e.g., freight shifted from railroads to trucks), demographics, and patterns of automobile use, sales, and design. Technology can produce organizational and societal change that may not be predictable or easily hurried. The effects of groupware will surely follow that pattern. By enhancing communication, groupware may erode the authority structure of some hierarchic organizations. This decentralization of control may further reduce the promise of groupware that selectively benefits management -- which includes most groupware being developed today. Because these failure-prone applications seem to emerge from a product development process that is heavily driven by technology and managed by intuition, we may shift to a development process based on obtaining a deeper understanding of the structure of workplace activity.

REFERENCES

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

Bannon, L.J. and Schmidt, K., 1990. CSCW, or What's in a name? Manuscript.

Beard, D., Palaniappan, M., Humm, A., Banks, D., and Nair, A., 1990. A visual calendar for scheduling group meetings. In *Proc. CSCW'90 Conference on Computer-Supported Cooperative Work,* (Los Angeles, October 7-10), in press.

Bjerknes, G., Ehn, P., and Kyng, M. (Eds.), 1987. *Computers and democracy - a Scandinavian challenge.* Aldershot, UK: Gower.

Bullen, C.V. and Bennett, J.L., 1990. Learning from user experience with groupware. In *Proc. CSCW'90 Conference on Computer-Supported Cooperative Work,* (Los Angeles, October 7-10), in press.

Bødker, S., 1990. *Through the interface: A human activity approach to user interface design.* Hillsdale, NJ: Lawrence Erlbaum Associates.

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), 61-66.

Docherty, P., Fuchs-Kittowski, K., Kolm, P., and Mathiassen, L. (Eds.), 1987. *System design for human development and productivity: Participation and beyond.* Amsterdam: North-Holland.

Ehn, P., 1989. *Work-oriented design of computer artifacts.* Hillsdale, NJ: Lawrence Erlbaum Associates.

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), 323-329.

Ehrlich, S.F., 1987b. Strategies for encouraging successful adoption of office communication systems. *ACM Trans. on Office Information Systems, 5,* 340-357.

Ensor, R., 1990 (Moderator). How can we make groupware practical? In *Proc. CHI'90 Human Factors in Computing Systems,* (Seattle, April 1-5).

Erickson, T., 1989. Interfaces for cooperative work: An eclectic look at CSCW'88. *SIGCHI Bulletin, 21,* 1, 56-64.

Fanning, T. and Raphael, B., 1986. Computer teleconferencing: experience at Hewlett Packard. In *Proc. CSCW'86 Conference on Computer-Supported Cooperative Work,* (Austin, December 3-5).

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), 241-254.

Greenbaum, J. and Kyng, M., 1990. *Design at work: Cooperative design of computer systems.* Hillsdale, NJ: Lawrence Erlbaum Associates.

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

Grudin, J., 1989. Why groupware applications fail: Problems in design and evaluation. In *Office Technology and People, 4,* 3, 245-264.

Grudin, J. and Poltrock, S., (1989). User interface design in large corporations: Communication and coordination across disciplines. In *Proceedings CHI'89 Human Factors in Computing Systems,* (Austin, April 30-May 4).

Grudin, J., 1990a. Groupware and cooperative work: Problems and prospects. In B. Laurel (Ed.) *The art of human-computer interface design.* Reading, MA: Addison-Wesley.

Grudin, J., 1990b. Obstacles to user involvement in interface design in large product development organizations. In *Proc. INTERACT'90 Conference on Human-Computer Interaction,* in press.

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

Kraemer, K. and King, J., 1988. Computer-based systems for cooperative work and group decision making. *ACM Computing Surveys, 20,* 115-146.

Lai, K.Y. and Malone, T.W., 1988. Object Lens: A "spreadsheet" for cooperative work. In *Proc. CSCW'88 Conference on Computer-Supported Cooperative Work* (Portland, September 26-28, 1988).

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

Lyytinen, K. and Lehtinen, E., 1987. Seven mortal sins of systems work. In Docherty, P., Fuchs-Kittowski, K., Kolm, P. and Mathiassen, L. (Eds.) *System design for human development and productivity: Participation and beyond.* Amsterdam: North-Holland.

Malone, T.W., Yates, J., and Benjamin, R.I. 1987. Electronic markets and electronic hierarchies. *Communications of the ACM, 30,* 6, pp. 484-497.

Markus, M.L. and Connolly, T., 1990. Why CSCW applications fail: Problems in the adoption of interdependent work tools. In *Proc. CSCW'90 Conference on Computer-Supported Cooperative Work,* (Los Angeles, October 7-10), in press.

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

McGoff, C., Vogel, D., and Nunamaker, J., 1990. IBM experiences with GroupSystems. In *Proc. DSS-90,* May.

Nardi, B.A. and Miller, J.R., 1990. Twinkling lights and nested loops: Distributed problem solving and spreadsheet development. In *Proc. CSCW'90 Conference on Computer-Supported Cooperative Work,* (Los Angeles, October 7-10).

Perin, C., 1989. Electronic social fields in bureaucracies. American Anthropological Association, Organized Session, "Egalitarian Ideologies and Class Contradictions in American Society." Washington, D.C., November.

Perlman, G., 1985. USENET: doing research on the network. *UNIX/World,* December, 75-81.

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

Rowe, C.J., 1987. Introducing a sales order processing system: the importance of human, organizational and ergonomic factors. *Behaviour and Information Technology, 6,* pp. 455-465.

Sathi, A., Morton, T.E., and Roth, S.F., 1986. Callisto: An intelligent project management system. In Greif (1988), 269-309.

Suchman, L., 1983. Office procedures as practical action: Models of work and system design. *ACM Transactions on Office Information Systems, 1,* 320-328.

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.

Whiteside, J., and Wixon, D., 1988. Contextualism as a world view for the reformation of meetings. Presentation at *CSCW'88 Conference on Computer-Supported Cooperative Work* (Portland, September 26-28, 1988).

Zellweger, P., 1990. More Voice Applications in Cedar. *SIGGRAPH Video Review, 59,* Entry 4.