COMPUTER SYSTEMS SUPPORTING COOPERATIVE WORK: A CSCW '90 TRIP REPORT

SCOTT HENNINGER

Computer Supported Cooperative Work (CSCW) is a relatively young research field that concerns itself with issues of using computers to support working groups. The third meeting of this biannual conference, CSCW '90, held last October in Los Angeles, brought together a mix of researchers, primarily from social sciences, computer science, business, and psychology. Over 500 researchers and developers attended the conference. There were 30 papers presented in non-overlapping sessions. The papers were mostly either descriptions of groupware systems or empirical studies of human working groups. In what follows, I will give my personal highlights of the conference. My overall assessment is that while the systems presented weren't that exciting, the studies were well done and illuminated problems with current CSCW applications, giving some direction for future systems.

E-MAIL

Work on e-mail systems was not well represented in the conference, but Allan Shepherd (with Niels Mayer and Allan Kuchinsky) of HP Labs gave a talk on Strudel, a system that structures e-mail conversations. The approach reminds me of Winograd's Coordinator where users add structure to their messages based on conversational moves and action items, such as "request" and "notification". It also has facilities for conversation threading. Strudel differs from some previous attempts at conversation structuring by allowing users to define their own conversation types and action items, thereby making the message structure "more task specific".

For the most part, the current emphasis on improving e-mail systems has focused on adding features to simple e-mail. Christine Bullen of MIT presented a study which looked at a number of groupware systems based on structured e-mail, such as Coordinator, PROFS, and All-In-1. They found that the "wonderful" functions often cited by users were not actually used by those users. People tended to use the simple e-mail functions of sending and reading mail. For example, in the Coordinator, people would send the message type "request" for almost all messages, regardless of message content. This occurred because "request" was the default menu choice. It may be a matter of having the wrong features, but Bullen also observed that people weren't interested in learning new techniques and features. For example, if people were not already doing project tracking, great innovations for project tracking will do them no good.

The conclusion of the study reverberated many times throughout the conference: "rather than looking at 'fancy,' innovative functions for groupware systems, designers should be focusing on how to better solve the basic need of office workers, i.e. managing large volumes of information.'' (p. 294). She went on to say that technology is "simultaneously a social and technical intervention''. Therefore, throwing technology at perceived problems focuses on at best half of the problem.

Jolene Galegher of the University of Arizona presented a paper (with Bob Kraut of Bellcore) which compared group communications using e-mail with other communication mediums. They studied people preparing reports in three experimental groups: one communicated face-to-face, the second could only use the computer e-mail facilities, and a third could use e-mail plus the telephone. The results were not that surprising, but instructional nonetheless. The e-mail group experienced the most coordination problems, the highest frustration level, and social relationships and interactions suffered the most. Although the reports did not differ significantly in quality in the end, e-mail users had the most trouble getting started.

Again in this study, it is not clear that the problem lies inherently in e-mail, or just the impoverished tools subjects were given. I think this type of study is most useful when thought of as a method to study e-mail to see where and in what ways it should be enhanced. It was stated in discussion that people have to be taught the new social context of CSCW applications to be effective. Doug Engelbart later stated that users must adapt to new work practices. This may be true, but one must be careful of "postulating a new man", and these studies provide the means to avoid this pitfall.

VIDEO

There were a few presentations that focused on video interactions, mostly from the standpoint of putting a camera and screen in remotely located coffee and conference rooms. There was an interesting person-to-person video system presented by Hiroshi Ishii of NTT Human Interface Labs that allowed one to superimpose the computer screen and a camera shot of a desktop working area. Its not clear what advantage this gives you but it was a rather cute trick. Their goal was to allow users the freedom to choose either the computer or a desktop as a collaborative working media.

The bottom line of the empirical studies of video is that the technology is still a bit flawed from a human factors perspective. Robert Fish (with Bob Kraut and Barbara Chalfonte) of Bellcore gave an informative paper on how video differs from face to face communication. Their study involved putting a camera in a coffee room an analyzing how often and when people engaged in conversation. The problems they found have a social and perceptual basis. Since the technology is crude and not very sensitive to standard human communication needs, people generally had problems adapting to the video media. Simple things, like feedback, were missing. This caused rather comical interchanges like "can you see my head?". There were also problems with standing in the right place, so that people could be heard but not seen or vice-versa. Because of these problems, and because its easier to ignore another's presence across a video channel, they found that the number of interactions between people were fewer with the mere presence of video equipment when compared with face to face encounters. Another issue worth considering is whether the small but expensive gains that can be obtained with video communication are truly superior to current remote communication techniques like phones or air travel, especially given that people particularly enjoy the social interaction that only the physical presence of another person can offer.

SYNCHRONOUS GROUPWARE

Real-time CSCW applications were represented by three papers describing toolkits for making single-user applications into groupware. Each paper pointed out the inherent difficulties of creating groupware above and beyond single-user applications. While each had similar concerns such as what should be shared and when, each also addressed a set of more specific issues. Patterson et al.'s Rendezvous is a general purpose architecture designed to make groupware as easy (ore equally difficult) to write as single-user applications. Beyond the standard issues, it also deals with issues of session joining, where users join a groupware session already in progress. Crowley et al. described MMConf, a tool that supports both single-user and shared applications. It provides basic mechanisms to implement desktop conferencing applications, including voice and video. Knister and Prakash presented DistEdit, a toolkit for converting editors into interactive group editors for distributed environments. This approach is unique in that it is tailored to the specific domain of editors.

While these systems may seem a bit low-level to some, they have an advantage over groupware designed from scratch in that existing applications can be adapted to the shared context. With this approach, new systems do not need to be learned, allowing groups to use their favorite applications. It also introduces the possibility of using different types of hardware, eliminating the sometimes expensive requirement for uniform hardware to run groupware.

What was missing were studies that examine how such groupware fits into the social fabric of working groups. It is unclear to me how effective these systems are in supporting group work. Some informal retrospectives exist from people who have used synchronous groupware, but good empirical work would be a plus for evaluating these systems. There is still a burden of proof on these systems to show performance advantages that justify their expense.

DESIGN RATIONALE

Design rationale is a method of recording design decisions in development projects. Most are based on the IBIS methodology, which is an informal recording method that structures issues and their associated questions, arguments, and answers in a hierarchy that can be viewed by designers as a documentation of the design process. Jinate Lee of MIT presented a paper on SIBYL, a groupware system which uses a representation language for design rationale built on Object Lens. In contrast with the IBIS methodology, SIBYL tries to represent and structure information for automated processing. They therefore make attempts at managing dependencies among the objects represented to dynamically maintain consistency between related claims. They also provided facilities for explicitly representing goals. The authors' goal is to position SIBYL somewhere in the knowledge-based continuum between a syntactic representation and a formally represented knowledge base.

K.C. Burgess Yakemovic of NCR (with Jeff Conklin of MCC) presented a study of a software development project that used some IBIS techniques. Although it was unclear what their techniques were compared to, they found that using an issue-based information system allowed them to detect errors earlier in the development cycle, and improved inter-organizational communication (i.e. not within the development group, but between the group and outside organizations). They also made the conjecture that the strength of the approach was in pointing out where you missed a possible solution. The issues were captured by appointing a scribe at design meetings, which was identified as requiring a large effort on the scribe's behalf. The most important limitation identified was that it was humanly impossible to capture everything the occurred at the meetings, causing some things to be glossed over or omitted by the scribes. The problem was that people didn't know beforehand which issues were important, often causing information that was later deemed important to be lost.

DEMO NIGHT

A new addition to the CSCW conference was a demo night. This is significant in that the field has matured to the point that there are a number of groupware systems to compare and evaluate. Unfortunately I didn't find many of them to be very exciting. Most systems were able to display nodes connected by links (ho-hum). But harsh criticism is a little unfair to the developers since it is inherently difficult to demo groupware. The power of the system largely lies in supporting inter-personal interactions, which must be staged or otherwise simulated, and this is difficult to convey in a short amount of time. Nonetheless, two demos in particular caught my attention.

The first was ShrEdit from the University of Michigan (Gary and Judith Olson et al.). ShrEdit allows people to simultaneously edit a single file. Their rationale (given in an accompanying video) was that meetings "restrict" human to human communication because only one person can speak at a time. The system therefore circumvented this shortcoming by allowing people to "speak" simultaneously. As I hinted before, this approach doesn't have a lot of intuitive appeal for me. When using the system, the text you're working on can be changing in three or four places at once. It would be difficult to keep up with what's happening, demonstrating Simon's principle that the critical resource is not the amount of information generated, but limits on the capacity of human attention (H. Simon, The Sciences of the Artificial). To their credit, some empirical work has been done, and they plan to use the results to analyze where such systems pose an opportunity for enhanced group work and the kinds of skills that are developed after becoming proficient with this form of groupware.

My favorite demo was CSILE (Computer-Supported Intentional Learning Environments) from the University of Toronto that worked on a collaborative environment for children. Children told stories in a group setting by intermixing pictures and text on a computer. To me, the interesting aspect is the emphasis of having kids work together, an invaluable resource they can use later in life. There were also some interesting results from studying the kids that have implications for instructional techniques. For example, the students were encouraged to explore different learning strategies, such as re-using words from previous stories when telling their own. Curiously, on subsequent individual spelling tests, the kids who used the system did better those that learned spelling in the traditional rehearsal manner.

The state of the demos probably reflects a normal evolution of an emerging technology where first theories are identified, then systems that partially instantiate or are otherwise affected by the theories, are built. The next step therefore seems to be to evaluate these systems and their corresponding theories to set up the next cycle. Perhaps future demo sessions will be marked by more creative demo techniques that really bring out the collaborative aspects of the systems.

EMPIRICAL STUDIES

For me, the high points of the conference were empirical studies that took a hard look at collaborative work settings. The studies of video and e-mail were one source that I've mentioned earlier. Another source were a few studies on computer applications in their social context. Bonnie Nardi (with Jim Miller) of HP Labs investigated how spreadsheets are developed in practice. They interviewed a few groups that used spreadsheets as part of their everyday work environment and found that they are often developed and evaluated cooperatively. They observed a kind of cooperative programming environment where domain knowledge is exchanged and shared within a cultural standard: the spreadsheet. Perhaps the most interesting aspect of the study is that this collaborative process works very well without any explicit mechanisms for coordination or collaboration. One aspect of spreadsheets that facilitates cooperation is that novice spreadsheet users can do a significant part of the programming, calling on expert programmers only when absolutely needed. In contrast to traditional computing environments where specialists are necessary to deal with any programming task, "the problem solving needed to produce a spreadsheet is distributed across a person who knows the domain well and can build most of the model, and more sophisticated users whose advanced knowledge is used to enhance the spreadsheet model, or to help the less experienced user improve spreadsheet skills.'' (p. 202).

Wendy Mackay of MIT did some empirical work on how people share customization files. She found that customizing applications is a social process. The most common method of getting information was to talk to others. People would see what others were doing and if it interested them, they would learn it or borrow the customizations. Interestingly, the customizations were often employed to make a new version of the application look like an old one. This creates problems because customizations often occur before people know how they will eventually use the system, decreasing any performance advantages that may have otherwise been gained with the new program. Mackay concluded that "Software manufacturers should consider designing software to be reflective, allowing users to become more aware of their own patterns of use and providing methods for evaluating effectiveness.'' and that "Reflective software should increase the user's awareness of how they actually use the software.'' (p. 219). Mackay also showed that while most users took advantage of their social contacts for learning how to use the applications, system programmers were more likely to look at documentation or source code[1]. This mismatch between designer and user attitudes shows the weakness of designer introspection that especially affects groupware design, and underscores the need for user evaluation and participation in application design.

Andrew Clement of the University of Toronto gave a talk on how an organizational support structure is needed in addition to tools to cope with mastering computer technology. His study focused on secretaries and clerical workers, who are often not paid adequate attention by systems developers. He found that computer systems and applications were often mandated without user involvement, causing a mismatch between the system and the needs of the actual users. In addition, the training was often inadequate. Users were able to learn more about the system on their own or with the help of colleagues than with formal training. The problem seems to lie in the fact that the systems were predicated on the idea of adapting work practices to the system, which simply didn't happen. Instead, the users constructed an informal "web of support" to make the machines work for them. The moral of this story is especially applicable to user interface design. While intrinsically useful products need to provide more power with lower learning curves, the social context must be accounted for to create systems that are more oriented toward actual work practices and evolving needs of users.

Lynne Markus of UCLA (with Terry Connolly of the University of Arizona) presented an analysis of how CSCW applications can fail even when there are no asymmetries between beneficiaries and no confusion between personal and collective benefit, thus extending Jonathan Grudin's influential analysis at CSCW '88. The analysis is applied to discretionary databases, where the collective benefit is best when users both enter new data and use that data. But presuming that someone enters information, then everyone's optimal strategy is to use the information, but not participate in the additional work needed to update the information. A prisoners dilemma situation follows where the result of individuals following the optimal strategy is collectively non-optimal and in fact detrimental. The analysis is backed up with empirical work that supports the theory. A similar effect occurs in e-mail, where the benefit of group communication is not realized until a "critical mass" of users adopt the system. The authors state that policy interventions that mandate system usage are needed to reach critical mass. Once this is accomplished, adoption and use are self-reinforcing because both the collective and individual benefit is realized.

While the above studies all involved computer applications in one way or another, Stephen Reder and Robert Schwab gave a paper which analyzed the relationship between individual and group work activity from a temporal perspective independent of computers. They showed how three types of workers (Senior Management, Sales Development, and Marketing Groups) balanced their time between having time to work alone and being accessible to others for communication. They followed people around in their work environment and recorded the number of distinct tasks they engaged in and the number of distinct individuals they interacted with per day. They found marked differences between the groups and much similarity within groups, showing that the constraints of the job determined the temporal structure. This means that groupware systems must be sensitive to different job types even within the same application. The study also implied the need to support multiple, intertwined, tasks that included synchronous as well as asynchronous work. There is also the need to support multimedia and transposing communication from one media to another because a collaborative task was often suspended and picked up later on a different communication media. For example a face to face meeting could end with the need for more information, which is later relayed by phone. This study points out the danger of using one's own work practices as a model for groupware to be used by others, while giving some hope that classes of workers can be successfully characterized by their work type.

OVERALL ASSESSMENT: CREEPING FEATURISM CONSIDERED HARMFUL TO CSCW APPLICATIONS

As I see it, the main problem with what I saw at the conference was a lack of analysis of work environments to motivate the need for systems. In the systems work, there seems to be a focus on technology for the sake of technology, without much thought about what people actually need. Tom Moran of Xerox EuroPARC concurred with this assessment with a presentation on how technology can be implanted into a complex social fabric, without undue attention to the technology. More so than any other computer discipline, CSCW has to evaluate why a system solution is needed and integrate the computer into the work setting. As was stated by a few of the speakers who did experiments, the entire social setting of group work must be taken into account; there are many tools, such as face to face communication, that are an integral part of group work that perhaps shouldn't be computerized. Hiroshi Ishii stated that users should be free to choose a computer or desktop in collaborative environments. This should be augmented with an attitude on the part of system designers that paper and pencil, social structures, and etc. all play an important part in everyday working environments, and that accounting for these factors in groupware design is crucial to the eventual success of the CSCW field.

ACKNOWLEDGEMENTS

Special thanks to Jonathan Grudin for insightful comments on an earlier draft.

[1] MIT programmer slogan: "use the force, read the source."