An Ethnographic Study of Distributed Problem Solving in Spreadsheet Development

Bonnie A. Nardi
James R. Miller

Hewlett-Packard Laboratories
Human-Computer Interaction Department
1501 Page Mill Road
Palo Alto, CA 94304

ABSTRACT

In contrast to the common view of spreadsheets as "single-user" programs, we have found that spreadsheets offer surprisingly strong support for cooperative development of a wide variety of applications. Ethnographic interviews with spreadsheet users showed that nearly all of the spreadsheets used in the work environments studied were the result of collaborative work by people with different levels of programming and domain expertise. Cooperation among spreadsheet users was spontaneous and casual; users activated existing informal social networks to initiate collaboration.

INTRODUCTION

People organize themselves and their work so that problems can be solved collectively (Vygotsky, 1979; Bosk, 1980; Lave, 1988; Newman, 1989; Seifert & Hutchins, 1989). We are interested in the artifacts that support and encourage this collective problem solving. A spreadsheet is a "cognitive artifact" (Chandrasekaran, 1981; Holland & Valsiner, 1988; Norman & Hutchins, 1988; Norman, 1989) that can be understood and shared by a group of people, providing a point of cognitive contact that mediates cooperative work. In this paper we examine the shared development of spreadsheet applications. We report the results of our ethnographic study of spreadsheet use in which we found that users with different levels of programming skill and domain knowledge collaborate informally to produce spreadsheet applications. This paper is a descriptive, empirical report of collaborative work practices, meant to document the kinds of cooperation found among spreadsheet users, and the ways in which problem solving is distributed across users with different skills and interests. In other papers (Nardi & Miller, 1990a, 1990b) we analyze how the spreadsheet itself provides a basis for collaborative work, arguing that the division of the spreadsheet into two distinct programming layers permits effective distribution of computational tasks across different kinds of users.

In contrast to studies of computer-supported cooperative work (CSCW) that focus on software systems specifically designed to support cooperative work within an organization (Grudin, 1988), we address how a certain class of traditional personal computer applications — spreadsheets — function as *de facto* cooperative work environments. We describe how spreadsheet users work together, even though spreadsheets lack "designed-in" technological support for cooperative work.

We use the term "cooperative work" in the general sense of "multiple persons working together to produce a product or service" (Bannon & Schmidt, 1989). In this paper we

want to draw attention to a form of cooperative computing already well established in office environments. As we will describe, spreadsheets emerge as the product of several people working together, though not in formally designated teams, task forces, or committees. On the contrary, spreadsheet work flows across different users in fluid, informal ways, and cooperation among spreadsheet users has a spontaneous, self-directed character.

Our research highlights two forms of cooperative work that are central to computer-based work and that have received little attention in the CSCW community: the sharing of programming expertise and the sharing of domain knowledge. Because of the CSCW emphasis on computer systems that enhance interpersonal communication (e.g. e-mail, remote conferencing, shared whiteboards), the importance of collaboration in programming itself has been overlooked. The current interest in "empowering users" through participatory design methods (Bjerknes, Ehn & Kyng, 1987) and end user programming systems (Panko, 1988) will, we believe, begin to draw attention to collaborative programming practices of the kind we describe in this paper. The sharing of domain knowledge has been only implicitly recognized in CSCW research; studies tend to focus on communication techniques themselves, rather than on what is being communicated. In this paper we describe how spreadsheet users exchange domain knowledge in the process of spreadsheet development.

Since 1986 about five million spreadsheet programs have been sold to personal computer users, second in number only to text editors, and far ahead of any other kind of software (Alsop, 1989). Spreadsheets deserve our interest as the only widely used end user programming environment; text editing and drawing packages are used by many, but involve no programming. With spreadsheets, even unsophisticated users can write programs in the form of formulas that establish numerical relations between data values. Users who show no particular interest in computers *per se* voluntarily write their own spreadsheet programs, motivated by interests beyond or completely unrelated to job requirements — a claim that cannot be made for any other kind of software that we know of. In large part this is because the spreadsheet's "twinkling lights"[1]— the automatically updating cell values — prove irresistible. Spreadsheet users experience a real sense of computational power as their modifications to data values and formulas appear instantly and visibly in the spreadsheet.

Spreadsheets have had an impact on personal computing well beyond the financial uses envisioned by the creators of VisiCalc, the first personal computer spreadsheet program. Spreadsheets are now used for mathematical modeling (Arganbright, 1986), simple databases, managing small businesses, forecasting trends (Janowski, 1987), analyzing scientific and engineering data, and of course the financial applications for which they were first intended, to name but a few examples. In the research community, investigators have used the spreadsheet interface as a model for other types of systems, such as the logic programming spreadsheet of Spenke and Beilken (1989). Lewis and Olson (1987-1988), Piersol (1986), and Van Emden, Ohki, and Takeuchi (1985) have also built other kinds of prototypes that leverage off the spreadsheet model.

Despite the prevalence of spreadsheets in the personal computing world, spreadsheets have not been widely studied. Kay (1984), Hutchins, Hollan, and Norman (1986), and Lewis and Olson (1987) enumerated some of the benefits of spreadsheets, including a concrete, visible representation of data values, immediate feedback to the user, and the ability to apply formulas to blocks of cells. There are some experimental studies of spreadsheet use that focused on small aspects of the user interface; for example, Olson and Nilsen (1987) contrasted the methods by which subjects entered formulas in two different spreadsheet products. (See also Brown & Gould, 1987, and Napier, Lane,

[1]We are indebted to Ralph Kimball of Application Design Incorporated of Los Gatos, California for this turn of phrase.

Batsell, & Guadango, 1989.) In another type of study, Doyle (1990) reported his experiences teaching students to use Lotus 1-2-3,[2] although most of his observations could apply to any kind of software (e.g., inconsistencies in file naming conventions).We know of no literature on collaborative work practices among spreadsheet users, or among users of any kind of end user programming environment,[3] and we are still far from understanding spreadsheets and the ways in which they are used.

Our study began with the traditional "single-user application" perspective. We were (and still are) interested in spreadsheets as computational devices, and wanted to learn more about how spreadsheet users take the basic structure of a spreadsheet and mold it into an application that addresses some specific need. In particular, we were interested in the success *non-programmers* have had in building spreadsheet applications. We saw no reason to dispute Grudin's (1988) comments that spreadsheets are "single-user applications" in which "an individual's success . . . is not likely to be affected by the backgrounds of other group members," and that "motivational and political factors" are unimportant for spreadsheet users. However, as the study progressed, we were struck by two things:

- **Spreadsheet co-development is the rule, not the exception.** In the office environments we studied, most spreadsheets come about through the efforts of more than one person. The feeling of co-development is very strong; people regularly spoke of how "we" built a spreadsheet, and were very aware of the cooperative nature of the development process.

- **Spreadsheets support the sharing of both programming and domain expertise.** Because of our focus on end-user programming, we soon noticed that one reason spreadsheet users are so productive is that they successfully enlist the help of other, more knowledgeable users in constructing their spreadsheets. In the same way, experienced co-workers share domain knowledge with less experienced colleagues, using the spreadsheet as a medium of communication.

We do not mean to suggest that spreadsheets are never developed by individual users working completely independently. But presupposing that spreadsheets are "single-user" applications blinds us to seeing the cooperative use of spreadsheets, of which we found much evidence in our study. Our data show that spreadsheet users share programming expertise as they work together building spreadsheets, and train each other in new techniques. Users transfer domain knowledge via spreadsheet templates and the direct editing of spreadsheets. We will describe these activities in some detail via ethnographic examples from the research.

METHODS AND INFORMANTS

The ideas presented in this paper are based on our ethnographic research including extensive interviewing of spreadsheet users, and analysis of some of their spreadsheets which we collected during the course of interviewing. We have chosen to study a small number of people in some depth to learn how they construct, debug, and use spreadsheets. We are interested in the kinds of problems for which people use spreadsheets and how they themselves structure the problem solving process — topics

---

[2]Lotus and 1-2-3 are registered trademarks of Lotus Development Corporation.

[3]In his book on end user programming, Panko (1988) briefly mentioned a study in which users in a training class helped one another, but there is no other discussion of cooperative work practices in the book.

that by their very nature cannot be studied under the controlled conditions of the laboratory. We have also examined and worked with several different spreadsheet products including VisiCalc (the original personal computer spreadsheet), Lotus 1-2-3 and Microsoft Excel.[4]

For the field research we interviewed and tape recorded conversations with spreadsheet users in their offices and homes.[5] Our informants were found through an informal process of referral. We told them that we were interested in software for users with little formal programming education and that we wanted to talk to people actively using spreadsheets. The interviews were conversational in style, intended to capture users' experiences in their own words. A fixed set of open-ended questions was asked of each user, though the questions were asked as they arose naturally in the context of the conversation, not in any particular order. During the interview sessions we viewed users' spreadsheets on-line, and sometimes in paper form, and discussed the uses and construction of the spreadsheets. The material in this paper is based on about 350 pages of transcribed interviews with 11 users, though we focus on a smaller subset here to provide ethnographic detail.

Informants in the study were college-educated people employed in diverse companies, from small start-ups to large corporations of several thousand employees. Informants had varying degrees of computer experience ranging from someone who only recently learned to use a computer to professional programmers. Most were non-programmers with 3-5 years experience with spreadsheets. Informant names used here are fictitious. Four sets of spreadsheet users illustrate the cooperative nature of spreadsheet development:

- *Betty and Buzz* run a start-up company with eight employees. Betty is the chief financial officer of the company and Buzz a developer of the product the company produces. Betty does not have a technical background though she has acquired substantial computer knowledge on her own, largely through using spreadsheets. Buzz is a professional programmer. They use spreadsheets for their customer lists, prospective customer lists, product sales, evaluation units, tradeshow activity, and accounts receivable.

- *Ray* manages a finance department for a large corporation and has a large staff. He has an engineering degree and an MBA, and some limited programming experience. He uses spreadsheets to plan budget allocations across several different departments, to track departmental expenses and headcounts, and to forecast future budgetary needs.

- *Louis*, in his seventies, is semi-retired and works as an engineering consultant about two hours a day for a large manufacturing corporation. He has been working with Lotus 1-2-3[6] for about a year, and has no other computer experience of any kind (he uses Lotus as his word processor). Louis's main application is analyzing test data from his engineering simulations of radar designs. He learned Lotus with the help of his son Peter, an architectural engineer.

---

[4]Microsoft and Excel are registered trademarks of Microsoft Corporation.

[5]The interviews were conducted by the first author. We use the plural "we" here for expository ease.

[6]All those in our study use either Lotus 1-2-3 or Microsoft Excel.

- *Laura and Jeremy* work for a medium size high tech equipment manufacturer. Laura is an accountant, the controller of the company. She directs a staff of eight, all of whom use spreadsheets. Laura is knowledgeable about spreadsheets but has no programming experience. Jeremy, Laura's manager, is the chief financial officer of the company. He is skilled at spreadsheet macro and template development.

Segments from the interviews will be presented at some length as we feel it is most convincing to let users speak for themselves. The segments are verbatim transcriptions.

COOPERATIVE DEVELOPMENT OF SPREADSHEETS

Bridging differences in programming expertise

Spreadsheets support cooperative work among people with different levels of programming skill. We have found it useful to break the continuum of skill level into three groups: non-programmers, local developers, and programmers. Non-programmers have little or no formal training or experience in programming. Local developers have substantial experience with some applications, and often, much more willingness to read manuals. Programmers have a thorough grasp of at least one general programming language and a broad, general understanding of computing. Local developers typically serve as consultants for non-programmers in their work environments. Local developers may in turn seek assistance from programmers.

It is also important to note that the three kinds of users vary along another related dimension: *interest in computing*. In some cases non-programmers may be budding hackers, but many are simply neutral towards computers, regarding them as a means to an end rather than objects of intrinsic interest. A key to understanding non-programmers' interaction with computers is to recognize that they are not simply under-skilled programmers who need assistance learning the complexities of programming. Rather, they are not programmers at all. They are business professionals or scientists or other kinds of domain specialists whose jobs involve computational tasks. In contrast, local developers show a direct interest in computing, though their skills may be limited in comparison to programmers as a result of other demands on their time.

Betty and Buzz's work on spreadsheets for their company's finances offers a good example of cooperation among spreadsheet users with different levels of programming skill. As individuals, Betty and Buzz are quite different. Betty has a strong focus on her work as chief financial officer, and claims few programming skills. She has limited knowledge of the more sophisticated capabilities of the spreadsheet product she uses, knows little about the features of competing spreadsheets, and relies on Buzz and other more experienced users for assistance with difficult programming tasks, training, and consulting. In contrast, Buzz has a clear technical focus and strong programming skills. He is well-informed about the capabilities of the spreadsheet product in use in the company and of other competing products, and provides Betty with the technical expertise she needs.

From this perspective, then, Betty and Buzz seem to be the stereotypical end user/developer pair, and it is easy to imagine their development of a spreadsheet to be equally stereotypical: Betty specifies what the spreadsheet should do based on her knowledge of the domain, and Buzz implements it. *This is not the case.* Their cooperative spreadsheet development departs from this scenario in two important ways:

- Betty constructs her basic spreadsheets *without assistance from Buzz.* She programs the parameters, data values and formulas into her models. In addition, Betty is completely responsible for the design and implementation of the user interface. She makes effective use of

color, shading, fonts, outlines, and blank cells to structure and highlight the information in her spreadsheets.

- When Buzz helps Betty with a complex part of the spreadsheet such as graphing or a complex formula, his work is expressed in terms of Betty's original work. He adds small, more advanced pieces of code to Betty's basic spreadsheet; Betty is the main developer and he plays an adjunct role as consultant.

This is an important shift in the responsibility of system design and implementation. Non-programmers can be responsible for most of the development of a spreadsheet, implementing large applications that they would not undertake if they had to use conventional programming techniques. Non-programmers may never learn to program recursive functions and nested loops, but they can be extremely productive with spreadsheets. Because less experienced spreadsheet users become engaged and involved with their spreadsheets, they are motivated to reach out to more experienced users when they find themselves approaching the limits of their understanding of, or interest in, more sophisticated programming techniques.

Non-programming spreadsheet users benefit from the knowledge of local developers and programmers in two ways:

- Local developers and programmers *contribute code* to the spreadsheets of less experienced users. Their contributions may include: macros; the development of sophisticated graphs and charts; custom presentation formats, such as a new format for displaying cell values; formulas with advanced spreadsheet functions such as date-time operations; and complex formulas, such as a formula with many levels of nested conditionals.

- Experienced users *teach less experienced users* about advanced spreadsheet features. This teaching occurs informally, not in training classes. Often a user will see a feature in someone else's spreadsheet that they would like to have, and he or she simply asks how to use it.

As shown in the way Betty and Buzz divide up spreadsheet tasks, 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. Compare this division of labor to traditional computing which requires the services of a data processing department, or expert system development in which knowledge engineers are necessary. In these cases, the domain specialist has no role as a developer, and domain knowledge must first be filtered through a systems analyst, programmer, or knowledge engineer before it is formulated into a program.

Our interview with Ray offers another example of co-development. Ray is a local developer who makes use of programmers for some aspects of spreadsheet development. As with Betty and Buzz, the chief difference between the spreadsheet environment and traditional programming is that more experienced users develop only specific pieces of the spreadsheet program, working directly off the basic work done by the original user. For example, Ray recently commissioned a set of Lotus macros for custom menus to guide data input for the spreadsheets used by his staff. He prefers to concentrate on using spreadsheets for forecasting future trends and allocating money among the departments he serves — his real work. Ray is not interested in becoming an expert macro writer, even though he has taken an advanced Lotus 1-2-3 class where macros were covered. In the following exchange we are looking at the custom menus:

Interviewer: . . . [these menus] look like they'd be pretty useful. And who developed those for you?

Ray: A programmer down in Customer Support.

Interviewer: Okay, not somebody in your group. You just sent out the work, and . . .

Ray: Yeah, well, essentially, you know, I came at it conceptually, this is what I'd like to see, and they developed it. So [the programmer] made [the menus] interactive, set up the customized use.

Ray has reached the limits of his interest in programming advanced spreadsheet features himself. But he is not limited to spreadsheets without these features; he distributes the work to someone who has more interest in such things. This task distribution is similar to traditional software development in that a user provides a specification to a developer for implementation. The difference, however, is that here the user has constructed the program into which the contributed code fits. In some sense, the roles of user and "chief programmer" (Brooks, 1975) have been merged.

Spreadsheets also support cooperation between users with different programming expertise via tutoring and consulting exchanges. For example, Louis has learned almost everything he knows about Lotus 1-2-3 from his son Peter. He avoids the manual, finding it easier to be tutored by Peter. Louis's spreadsheet use highlights an important feature of the cooperative development of spreadsheets: because the initial effort to build something really useful is relatively small, less experienced users, having had the reward of actually developing a real application, are motivated to continue to learn more, at least up to a point. Louis is starting to have Peter teach him about controlling the presentation format; for the first several months of use he concentrated only on creating basic models of parameters, data values and formulas. In general, users like Louis successfully engage other, more experienced users in the development of their spreadsheet models. They make use of problem-solving resources — i.e. more experienced users — in a very productive manner, building on their existing knowledge in a self-paced way, as they feel ready to advance.

Distributing tasks across different users and sharing programming expertise are characteristic of many programming environments — programming in Pascal or Lisp or C would almost certainly involve such collaboration. However, with spreadsheets the collaboration is specified quite differently: the end user, usually relegated to "naive user" status in traditional software development, comes center stage, appearing in the role of main developer. Spreadsheets have been successful because they give real computational power to non-programmers. Accountants and biologists and engineers who may never have taken a computer science course build useful, often complex spreadsheet applications (see Arganbright, 1986). Spreadsheet users are not "naive users" or "novices"; they command knowledge of both their domain of interest and the programming techniques necessary to analyze problems in their domain. With spreadsheets, problem solving is distributed such that end users do not rely on programmers as the indispensable implementers of a set of specifications; instead end users are *assisted by* programmers who supply them with small pieces of complex code, or with training in advanced features, as they build their own applications.

Bridging differences in domain expertise

An important aspect of cooperative work is the sharing of domain knowledge. Because spreadsheet users build their own applications, spreadsheets allow the direct transfer of domain expertise between co-workers, obviating the need to include a programmer or other outside specialist in the development cycle. Domain knowledge flows from manager to staff since managers tend to be more experienced than those they supervise,

but also from staff to manager, as staff members often have specialized local knowledge needed by managers. This direct transfer of domain expertise provides efficient knowledge sharing and helps co-workers to learn from one another. Instead of transferring domain expertise to a programmer or systems analyst or knowledge engineer who may never need it again, less experienced workers directly benefit from the knowledge of co-workers.

Spreadsheets mediate collaborative work by providing a physical medium in which users share domain knowledge. Spreadsheet users distribute domain expertise by directly editing each other's spreadsheets, and by sharing templates.

For example, Laura works very closely with Jeremy, her manager, in developing spreadsheets. Jeremy happens to be a skilled spreadsheet user who provides macros and tutoring that Laura and her staff use. However, the more interesting distinction to be drawn here is centered around Jeremy's greater experience with their company, its manufacturing and marketing procedures, and its managerial and budgeting practices. Spreadsheets provide a foundation for thinking about different aspects of the budgeting process and for controlling budgeting activity. In the annual "Budget Estimates" spreadsheet that Laura is responsible for, many critical data values are based on assumptions about product sales, costs of production, headcounts, and other variables that must be estimated accurately for the spreadsheet to produce valid results. Through a series of direct edits to the spreadsheet, Laura and Jeremy fine-tune the structure and data values in "Budget Estimates." Laura describes this process:

> **Interviewer**: Now when you say you and your boss work on this thing [the spreadsheet] together, what does that mean? Does he take piece A and you take piece B - how do you divide up [the work]?
>
> **Laura**: How did we divide it up? It wasn't quite like that. I think more . . . not so much that we divided things up and said, "OK, you do this page and you do this section of the spreadsheet and I'll do that section," it was more . . . I did the majority of the input and first round of looking at things for reasonableness. Reasonableness means, "What does the bottom line look like?" When you look at the 12 months in the year, do you have some funny swings that you could smooth out? Because you want it to be a little bit smoother. So what can you do for that? Or, if you do have some funny spikes or troughs, can you explain them? For example, there's one really big trade show that everybody in the industry goes to . . . So our sales that month are typically low and our expenses are high. This trade show is very, very expensive . . .
>
> **Interviewer**: So there's a spike in your [expenses and a trough in sales] . . .
>
> **Laura**: Yeah. So as long as you can *explain* it, then that's OK. So what my boss did was, I would do the first round of things and then I would give him the floppy or the print-outs and I'd say, "Well this looks funny to me. I don't know, is that OK, is it normal? Should we try to do something about it?" And so what he did was he took the spreadsheets and then he would just make minor adjustments.
>
> Interviewer: Now was he adjusting formulas or data or . . .?
>
> **Laura**: Data.
>
> . . .

> **Interviewer:** . . . So it was a process of fine tuning the basic model that you had developed. And then you of course had to get his changes back, and look at them and understand them.
>
> **Laura:** Yes. And one thing he did do, was, he added another section to the model, just another higher level of analysis where he compared it to our estimate for this year. He basically just created another page in the model - he added that on.

In preparing a budget that involves guesswork about critical variables, Laura is able to benefit from her manager's experience. They communicate via the spreadsheet as he literally takes her spreadsheet and makes changes directly to the model. She has laid the groundwork, provided the first line of defense in the "reasonableness" checking; Jeremy then adjusts values to conform to his more experienced view of what a good estimate looks like. Jeremy also made a major structural change to the spreadsheet, adding another level of analysis that he felt would provide a useful comparison. The spreadsheet was cooperatively constructed, though not in a simple division of tasks; instead the model emerged in successive approximations as Laura and Jeremy passed it back and forth for incremental refinement.

Spreadsheet users often exchange templates as a way of distributing domain expertise. Jeremy, for example, prepares budget templates used by Laura and her staff. They contain formulas and a basic structure for data that he works out because of his greater knowledge of the business. Laura and her staff fill in the templates according to their knowledge of their individual areas. Laura and her staff are doing more than "data entry"; as in the "Budget Estimates" spreadsheet, estimates requiring an understanding of many factors often make up a significant aspect of a spreadsheet, and deriving these estimates demands thought. Users such as Laura may also specialize a template if their particular area requires additional information, such as another budget line item. The use of templates takes advantage of domain expertise at local levels, such as that of Laura and her staff, and higher levels, such as Jeremy's.

Ray's work with spreadsheets provides another example of how users share spreadsheet templates. Ray prepared "targeting templates" for his staff in order to standardize the process of targeting expenses. Because of his wider perspective looking across several departments, Ray is in the best position to develop a standard. The templates also contain the custom menus that facilitate data input. Each staff member builds the spreadsheet for his or her area on top of the template, insuring that minimum requirements for data collection and analysis are met, and insuring that the best possible information at the local level goes into the spreadsheets. Ray links them together. In these spreadsheets, problem solving is distributed over users who vary in both level of programming skill and domain knowledge: Ray, a local developer with domain expertise, provided the basic template; a programmer created the menus constructed of macros; and Ray's staff members, domain experts in their departments, supply data values for their respective areas.

SUMMARY

Spreadsheet development entails the distribution of cognitive tasks in many different ways: users design and implement spreadsheets cooperatively, share templates, edit each other's spreadsheets, and learn from users with more advanced programming expertise and domain knowledge. The image of an isolated "single user" laboring in privacy is not borne out by the cases in our study. On the contrary, spreadsheet models emerge as the product of the closely interwoven efforts of several individuals working cooperatively.

In contrast to traditional computing environments, spreadsheets distribute problem solving in a different and arguably better way. Spreadsheets enable non-programmers to build basic spreadsheet models unassisted, giving them control over the design and basic development of their applications. Spreadsheet products also include macros, graphs and

charts, advanced functions, simple control constructs and various presentation techniques which non-programming users take advantage of via collaborative computing. Local developers and programmers aid less experienced users by writing small pieces of code for them, and training them in advanced spreadsheet techniques. Spreadsheets thus shift development tasks away from programmers, but use programmers to their best advantage, to supply small but technically advanced assistance to developers. Given that there never seem to be enough programmers to go around for all the applications non-programmers want, spreadsheets appear to have hit upon a happy solution, distributing tasks such that non-programmers can get on with their work, yet not be limited by their lack of programming sophistication.

Because spreadsheet users have direct computational power, they are in a good position to directly share domain knowledge with one another. There is no need to give specialized domain knowledge to a programmer to code into a program — end users do that themselves. When data values based on local knowledge are needed, as with the spreadsheet templates we discussed, or data values need to be adjusted by more knowledgeable domain experts, such work is handled among co-workers themselves, without outside technical help.

Spreadsheets support an informal but effective interchange of programming expertise and domain knowledge. Spreadsheets accomplish the distribution of cognitive tasks across different kinds of users in a highly congenial way; sojourners of the twinkling lights mix it up with crafters of nested loops — and all with software for which no explicit design attention was given to "cooperative use."

ACKNOWLEDGEMENTS

Thanks to Dave Duis, Danielle Fafchamps, Nancy Kendzierski, Robin Jeffries, Jasmina Pavlin, and Craig Zarmer for helpful discussions and comments on earlier drafts of this paper. We are grateful to our informants for finding time in crowded schedules to talk to us, and for the careful, reflective explanations they provided about their use of spreadsheets.

REFERENCES

Alsop, S. Spreadsheet users seem satisfied with what they already have. *InfoWorld*, September 11, 1989, 102-103.

Arganbright, D. Mathematical modeling with spreadsheets. *Abacus*, 1986, *3*(4), 18-31.

Bannon, L., & Schmidt, K. CSCW: Four characters in search of a context. *Proceedings of the First European Conference on Computer Supported Cooperative Work EC-CSCW'89*, Gatwick, England, 358-372.

Bjerknes, G., Ehn, P., & Kyng, M. *Computers and democracy: A Scandinavian challenge*. Brookfield, Vermont: Gower Publishing Company, 1987

Bosk, C. Occupational rituals in patient management. *New England Journal of Medicine*, 1980, *303*(2), 71-76.

Brooks, F. *The mythical man month: Essays on software engineering*. Reading, Mass.: Addison-Wesley Publishing Company, 1975

Brown, P., & Gould, J. D. How people create spreadsheets. *ACM Transactions on Office Information Systems*, 1987,*5*(3), 258-272.

Chandrasekaran, B. Natural and social system metaphors for distributed problem solving: Introduction to the issue. *IEEE Transactions on Systems, Man and Cybernetics*, 1981, Vol. SMC-11(1), 1-5.

Doyle, J. R. Naive users and the Lotus interface: A field study. *Behavior and Information Technology*, 1990,*9*(1), 81-89.

Grudin, J. Why CSCW applications fail: Problems in the design and evaluation of organizational interfaces. *CSCW'88: Proceedings of the Conference on Computer Supported Cooperative Work*, 1988, Portland, Oregon, 85-93.

Holland, D., & Valsiner, J. Cognition, symbols and Vygotskty's developmental psychology. *Ethos*, 1988,*16*(3), 247-272.

Hutchins, E., Hollan, J., & Norman, D. Direct manipulation interfaces. In D. Norman & S. Draper (Eds.), *User-centered system design*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1986.

Janowski, R. Spreadsheets: An initial investigation. Internal technical report, Hewlett-Packard Laboratories, Bristol, England, 1987.

Kay, A. Computer software. *Scientific American*, 1984,*5*(3), 53-59.

Lave, J. *Cognition in practice: Mind, mathematics and culture in everyday life.* Cambridge: Cambridge University Press, 1988.

Lewis, C., & Olson, G. Can principles of cognition lower the barriers to programming? *Empirical studies of programmers: Second workshop*. Norwood, New Jersey: Ablex,1987.

Napier, H., Lane, D., Batsell, R., & Guadango, N. Impact of a restricted natural language interface on ease of learning and productivity. *Communications of the ACM*, 1989, *32*(10), 1190-1198.

Nardi, B., & Miller, J. R. Twinkling lights and nested loops: Distributed problem solving and spreadsheet development. *International Journal of Man-Machine Studies*, 1990(a), in press.

Nardi, B., & Miller, J. R. The spreadsheet interface: A basis for end user programming. *Proceedings of Interact'90*, Cambridge, England, 1990(b).

Newman, D. Apprenticeship or tutorial: Models for interaction with an intelligent instructional system. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*, Ann Arbor, Michigan, pp. 781-788.

Norman, D. Cognitive artifacts. Presentation at the Workshop on Cognitive Theory and Design in Human-Computer Interaction, Kittle House Inn, Chappaqua, New York, 1989.

Norman, D., & Hutchins, E. Computation via direct manipulation. Final Report to Office of Naval Research, Contract No. N00014-85-C-0133. University of California, San Diego, 1988.

Olson, J., & Nilsen, E. Analysis of the cognition involved in spreadsheet software interaction. *Human-Computer Interaction*, 1987-1988, *3*, 309-349.

Panko, R. *End user computing: Management, applications, and technology.* New York: John Wiley and Sons, 1988.

Piersol, K. Object-oriented spreadsheets: The analytic spreadsheet package. In *OOPSLA '86 Proceedings*, 1986, 385-390.

Seifert, C., & Hutchins, E. Learning from error. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*, Ann Arbor, Michigan, 42-49.

Spenke, M., & Beilken, C. A spreadsheet interface for logic programming. In K. Bice & C. Lewis (Eds.), *Proceedings of the CHI'89 Conference on Human Factors in Computing Systems*, Austin: ACM Press, 1989.

Van Emden, M., Ohki, M., & Takeuchi, A. Spreadsheets with incremental queries as a user interface for logic programming. ICOT Technical Report TR-144, 1985

Vygotsky, L. S. *Thought and language.* Cambridge, Massachusetts: MIT Press, 1979. (First published 1934.)