Design at Work:

Cooperative Design of Computer Systems

Edited by

JOAN GREENBAUM, City University of New York
MORTEN KYNG, Aarhus University, Denmark

LEA
1991 LAWRENCE ERLBAUM ASSOCIATES, PUBLISHERS
Hillsdale, New Jersey Hove and London

1

Introduction: Situated Design

Joan Greenbaum and Morten Kyng

Well, I don't really know how to tell you what I don't like about the system. I guess one of the things is that it makes me think and work differently, like for example, when I want to make separate columns, I need to type it and then rearrange it. That's not the way I see it in my mind.

Word processing user

If a lion could speak would we understand her?

Paraphrase of Wittgenstein, 1953, p. 223

As the dust of the 1980s settles, we who design computer systems for the workplace know a little more about working with the people who will use them. But just a little. Our intention in writing this book is to document some of the things we know, and to point to a growing body of projects where the people who use computers and the people who design them have learned more about each other's work and about what happens in their workplaces.

In 1985 in Aarhus, Denmark, several of the contributors to this book organized a conference on Computers and Democracy (Bjerknes, Ehn, & Kyng, 1987). That conference, a follow-up to over a decade of involvement in Scandinavian user centered design projects, brought out a kaleidoscope of visions evolving from the original idea of workers-as-user participant. At the risk of confining the wide range of visions, the authors in this book share, as general background, a set of design ideals which have guided our work. These are:

- Computer systems that are created for the workplace need to be designed with *full participation* from the users. Full participation, of course, requires training and active cooperation, not just token representation in meetings or on committees.
- When computer systems are brought into a workplace, they should *enhance* workplace skills rather than degrade or rationalize

them. Enhancing skills means paying attention to things that are often left out of the formal specifications, like respect for tacit knowledge, building on shared knowledge and, most importantly, communication. Computer systems are a lot more than the simple flow of information represented in the flowcharts that systems analysts present to their clients.

Computers systems are *tools*, and need to be designed to be under the control of the people using them. They should support work activities, not make them more rigid or rationalized.

Although computer systems are generally acquired to increase productivity, they also need to be looked at as a means to increase the *quality* of the results. More output, such as the reams of printed pages emerging from many management information systems, doesn't mean better output. The double emphasis on productivity and quality raises new questions in the design process (see Daressa, 1986).

The design process is a political one and includes *conflicts* at almost every step of the way. Managers who order the system may be at odds with workers who are going to use it. Different groups of users will need different things from the system, and system designers often represent their own interests. Conflicts are inherent in the process. If they are pushed to the side or ignored in the rush to come up with an immediately workable solution, that system may be dramatically less useful and continue to create problems.

And finally, the design process highlights the issue of how computers are used in the context of work organization. We see this question of focusing on how computers are used, which we call the *use situation*, as a fundamental starting point for the design process. In fact, it is here that we put our attention in the first part of the book by introducing ideas about how people work.

hese ideas were a base for the authors as we began new projects in ıe late 1980s. Our experiences in these projects have shaped an nerging approach to cooperative or participatory design that focus- s on workplace activities. Part I, Reflecting on Work Practice, cetches our understanding of how people work together and what e need to do when we look at the intricate fabric of workplace ctivity. Part II, Designing for Work Practice, tells a number of ifferent stories from the on-going "dialogue" between a wide vari- y of our empirical design projects and our reflections on the rocess of design.

Users as Competent Practitioners

To system designers, the people who use computers are awkwardly called "users," a muddy term that unfortunately tends to focus on the people sitting in front of a screen rather than on the actual work people are doing. The word lumps all kinds of workplace activity together implicitly putting the computer in focus and treating people as a blurred background. Like Wittgenstein's riddle about the lion, these users are all too often understood by system developers in "system terms." Just as the human observer misleadingly assigns meaning to what lions are doing based on the human's own world view, system developers tend to make sense out of the work of the users by applying their own system development concepts, often missing the understanding of the users which stems from a knowledge of and experience with the work being done. Wittgenstein's point in the lion riddle is that understanding between humans and lions is not possible because they don't share a common practice. Fortunately, we believe our possibilities for mutual understanding with users are much better. However, it is not something which is there, a priori, but something that has to be carefully developed.

Although the identification of user issues now dominates the computer management and system development literature, the majority of books, articles, and seminars addresses the issue of how best to "integrate the user" into the system development process (see *Communications of the ACM*, Jan. 1990). We set ourselves apart from this, for the intent of this book is to show, through examples, how the diverse groups of people called users can actively learn, participate, and cooperate with system designers. Our interest is not in fitting users into an already existing system development process, but in creating new ways of working together. For us, user participation does not mean interviewing a sample of potential users or getting them to rubber stamp a set of system specifications. It is, rather, the active involvement of users in the creative process we call design.

In this book you will meet a wide range of users—people whose jobs include baggage handler, dental assistant, clerical worker, journalist, librarian, and typographer. They are some of the people with whom we, as authors, computer scientists, systems designers, linguists, and social scientists have worked. We see users not as one homogeneous group, but, rather, as diverse groups of people who have competence in their work practices. Our perspective focuses explicitly on all the different groups of people using computers in their work, and not on the managers. There has been a great deal written about management objectives in the system

evelopment process. We shift that focus and work directly with ıe people whom system developers typically call "end-users."

By viewing the people who use computers as competent in their ield of work, we find that the workplace takes on the appearance of rich tapestry, deeply woven with much intricacy and skill. The irst part of the book, Reflecting on Work Practice, examines this ıpestry and finds four underlying patterns. These are:

the need for designers to *take work practice seriously* ;

the fact that we are dealing with *human actors*, not cut-and-dried human factors;

the idea that work tasks must be seen within their context, and are therefore *situated actions;* and perhaps the most important of all, as it links the rest together,

that *work is fundamentally social*, involving extensive cooperation and communication.

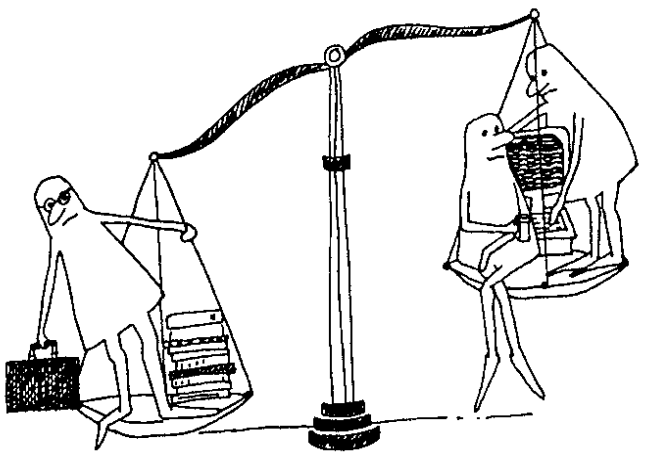
'his last pattern, the focus on the social nature of work practice, ıakes us acutely aware of the *cooperative nature* of workplace ısks. Our interest in cooperative activities does not mean that we lorify the way people work with each other, but rather it forces us) look at the ways people in an organization create, use, and change ıformation, knowledge, and tasks. Few work tasks are done in olation, and fewer still are easy to describe. While traditional sys-:m development methods treat specific work tasks as formalizable ata processing done by individuals in isolation, communicating via ata channels, the premise of our approach shifts to looking at roups interacting in multifarious ways within complex organiza-onal contexts. And, by the same token, although traditional meth-ds require describing these work tasks, our approach is based on ıe belief that the complex pattern of workplace life *is not easily escribable*. Therefore, we need new tools and techniques to cap-ıre this complexity, and to develop a more detailed understanding f its depth.

The second part of the book, Designing for Work Practice, takes a ook at the kinds of techniques designers and users can employ to ring this seemingly indescribable rich tapestry to bear on design. ere we begin with the idea that each of us understands the work-lace from within our own experience. Thus the computer system esigner, peering into a workplace from the outside, can't expect to ıpture the same meanings that someone involved in day to day ac-vity could. And the designers cannot automatically expect the sers to participate creatively in design activities, which may be ompletely new to them. Again bringing the lion analogy back into ıe picture, since neither designer nor user groups can *fully* under-

stand each others' practices or meanings, we need to build a bridge that brings these experiences closer together. Part II addresses ways to bridge this gulf, presenting examples from case studies in order to provide an approach based on *cooperative action* rather than formal description. The underlying ideas in this section involve:

- *mutual learning* between users and designers about their respective fields (see also Bjerknes & Bratteteig, 1984);
- use of tools in the design process that are *familiar* to the users;
- *envisionment* of future work situations to allow the users to experience how emerging designs may affect the work practice rather than relying on the seemingly esoteric language of system developers; and
- the importance of starting the design process *in the practice of the users.*

The traditional scales of system development have placed far more weight on getting users to understand the language of system designers. The authors in this part present their experiences as a way of creating room for users to act, and thus we hope to balance the scales.



This book, then, is an attempt to look at the development of computer systems as a process, where people as living, acting beings, are put back into the center of the picture. A picture of the workplace, where the situations people find themselves in, with all of its conflict-laden social and political tensions, comes under close scrutiny. The authors have been invited to write chapters on the

asis of their experiences in undertaking projects that attempt to do his. They are not in agreement on the question of how computer ystem designers *should* do things, but, as we mentioned in the reface, this is not another how-to book. They are in agreement, owever, on the fact that design needs to take place with users as ull and active participants, and that the tools and techniques for oing this are dependent on the situations within the workplace.

Our focus on starting design with our eyes firmly fixed on the nteractions taking place within the workplace was inspired, in part, y Lucy Suchman's book, *Plans and Situated Actions* (1987). Her ork takes as its starting point the idea that human actions are not so uch guided by concrete plans as based on situations. Thus, as the ircumstances in which we find ourselves change, so do our ac-ons. Most books on system design begin with the need for system esigners to carefully define the "problem." This book doesn't. nstead we ask ourselves and our readers to consider design issues om the broader context of the *situated actions* of the people with hom they are working. This chapter, which serves as an intro-uction to the book, takes a look at the changing circumstances that d us to this perspective.

As the book title states, our approaches are based on cooperation etween system developers and those people we call users. But it lso implies that most work is cooperative and that the process of utting this book together, like any collaborative venture, involved a reat deal of interaction among people of different disciplines. The neme of cooperation or respect for *mutual competencies*, whether ey be between designers and users, or authors in this book, is a entral one for us. Just as we see users as diverse groups of compe-nt practitioners, we have had to look at ourselves, as authors, as a iverse assortment of academic practitioners who speak different rofessional languages and use different approaches. We are lucky be writing this at a time when walls between academic fields are eginning to collapse. In fact, by virtue of having undertaken pro-cts where we looked at workplaces from a variety of perspectives nd designed systems with people who use them, we have con-ibuted to collapsing these boundaries. We hope to do more.

At the end of this chapter we will return to a discussion of the ackground of the authors and their contributions to the book. It is me now to take a look at how we got here, by examining some of ie history of things that have gone wrong in developing computer ystems, and then, of course, by looking at some of the more op-mistic trends that have led us to a better understanding of Design at ork. We had originally planned to call the book "Design by oing," highlighting our focus on "doing" work cooperatively with sers, and on using methods that emphasize action more than formal

description. But in the process of finding out what we were doing individually and by sharing our experiences, we found that before we could actually do "design by doing," we had to develop a more in-depth picture of what takes place at work and what practices constitute cooperative design. Thus, *Design at Work* brings to the foreground our enthusiasm for the work of design and for the importance of designing to support what actually takes place at work.

What's Gone Wrong?

Writing in 1965, Robert Boguslaw attempted to challenge computer system developers, whom he called "the new utopians," by saying:

> And so it is that the new utopians retain their aloofness from human and social problems presented by the fact or threat of machined systems and automation. They are concerned with neither souls nor stomachs. People problems are left to the after-the-fact efforts of social scientists. (p. 3)

Twenty-five years after Boguslaw's warning, and the warnings of many others (see Ackoff, 1974; Hoos, 1961; Nygaard & Bergo, 1975; Weizenbaum, 1976), we are repeatedly "surprised" by computer systems that don't work as intended. In almost every workplace there are stories that pass from department to department about computer systems that simply do not work. Often the stories take the form of legend, as workers tell the tale of how they "got around some dumb" computer system. As system developers, we like to think that we have learned a lot from our overzealous 'mistakes' of the past. But from both computer system literature and workplace folklore, the fact remains that many computer systems don't fit the work activities of the people who use them.

Thinking of the past as some random pattern of mistakes will not help get us away from being stuck in some blind alleys. If we were to look, instead, at the history of western scientific thought, we would find that the objectives and methods of computer systems development clearly grow out of several centuries of beliefs and practices in the natural sciences. According to Terry Winograd and Fernando Flores (1986), these deeply rooted practices, referred to as the rationalistic tradition, do the following:

1. Characterize the situation in terms of identifiable objects with well-defined properties.
2. Find general rules that apply to situations in terms of those objects and properties.
3. Apply the rules logically to the situation of concern, drawing conclusions about what should be done. (p. 15)

These step-by-step, rather linear processes force our thinking into narrow pathways, where emphasis is placed on isolating single problems and searching for the one "right" solution. While these ideals may be suited to certain types of scientific problem-solving, we feel that they are ill-suited to the dynamic and generally chaotic conditions of developing computer systems for the workplace. As Boguslaw pointed out, they channel our thinking away from "souls and stomachs."

This rationalistic tradition has its roots in Cartesian philosophy, which tends to split the world into the inner world of our minds and the outer world of things, thus dividing mind and body and drawing a line between our emotional interior and the objective, thing-oriented environment. In *Work-Oriented Design of Computer Artifacts*, Pelle Ehn (1989) tells us how this applies to system development:

> The prototypical Cartesian scientist or system designer is an observer. He does not participate in the world he is studying, but goes home to find the truth about it by deduction from the objective facts that he has gathered. (p. 52)

Although some might think that Ehn's example seems like a caricature of the way we work, a look at system development literature will verify his point. The dominant threads in systems literature today, reflected in the well-read works of DeMarco (1978), Yourdon (1986), and Jackson (1983), illustrate the call to objectivity and problem isolation. Edward Yourdon, in one of his many books on the system approach, defines his design strategy as one "that breaks large complex problems into smaller less complex problems and then decomposes each of these smaller problems into even smaller problems, until the original problem has been expressed as some combination of many small solvable problems" (p. 61).

Of course day-to-day systems practice is different from the formal methodology that traditional systems people, like Yourdon, advocate; yet the belief in this way of doing things is so strongly ingrained in our thinking that it pervades both the way we ask questions and the questions we ask. We believe that these rationalistic traditions have led us away from expanding our horizons and asking new questions about the design process. We don't expect the reader, or ourselves, to suspend disbelief and simply discard current system methodology, but our intent, rather, is to show that by looking at the assumptions behind this rationalistic tradition we can better understand how we get trapped and limited in our thinking—limits that may appear as "mistakes" in our practice, but are, in fact, embedded parts of the rationalistic world view and the accompanying system approach. As Pelle Ehn and Morten Kyng (1984) point out, many systems that are designed to reduce the need for skilled and experienced workers do so because the designs are based on this rationalistic world view which tends to make designers:

- view the application from the top of the organization,
- view the organization as a structure, whose important aspects may—and should—be formally described,
- reduce the jobs of the workers to algorithmic procedures, and thus
- view men and computers as information processing systems, on which the described data-processing has to be distributed. (p. 215)

Russell Ackoff, a well-known proponent of the systems approach, was well aware of the problems embedded in the dominating world view in the field. Trying to get system designers to focus on this issue, he said:

> We fail more often because we solve the wrong problem than because we get the wrong solution to the right problem....The problems we select for solution and the way we formulate them depends more on our philosophy and world view than on our science and technology. (p. 8)

This focus on world view reminds us that the issue of selecting problems within the workplace is heavily laden with cultural, political, and economic values. As computer system designers we can no more jettison our past than we can ignore the traditions of computer system users. Our approach in this book is to try to realistically appraise the way we work with a clearer understanding of the politics of the workplace and a careful eye on the assumptions we bring with us. The case studies in the following chapters are, like most workplace situations, potentially explosive in terms of contrasting management's objectives for the system with users' skills and interests. We won't attempt to sidestep this issue, but as we mentioned earlier, we take seriously the idea that systems should be concerned with the quality of work, not just its quantitative output. Thus we are asking different questions and perhaps looking for "solutions" to other problems.

In this book we contrast traditional system development with our own evolving cooperative or participatory approach. We know that we can't make a clean break with the Cartesian dualism that has dominated rationalistic thinking in the past. As Kuhn (1970) highlights in *The Structure of Scientific Revolutions*, paradigm shifts evolve through contradiction over time. But we will certainly try to

highlight ways in which our approach differs from the rationalistic world view and to point to emerging contradictions. In doing this we pay particular attention to the complex social relations of the workplace, and the need to use techniques that support involvement, rather than the detached reflection of the Cartesian scientist.

How Did We Get Here?

Like most projects, the writing of this book was in part serendipitous. Some of what we learned we learned through experience, and, as the chapters indicate, not all of it can be counted among the success stories. Some of our learning came through groping our way through the literature, as we attempted to find theories and examples that made sense of our experiences. Here, we present some of the strands that came from our earlier work and our reading, in order to provide an introduction to ideas presented in the book.

An obvious starting point for many authors in this book was in trade union-oriented projects in Scandinavia. Employee influence through unions and collaboration with management is a well-known part of the practice of social democracy in the Scandinavian countries. In the early 1970s, when new legislation increased the possibilities for worker influence, this strategy, called co-determination, was supplemented with a series of projects set up by central and local unions independent of employer organizations and management. In these projects, workers aided by consultants and researchers struggled to develop a better, more coherent platform for worker influence on the use of new technology in the workplace. The need for a new platform, based on the workers' own perspective, was described by Kristen Nygaard (1979) in the following way:

> It became clear that (the new possibilities for) industrial democracy would only give useful results if .. new knowledge (was) built up and a broad activity base established among the members of the trade unions.... Also, it was felt that workers would risk brainwashing themselves if they tried to assimilate the existing knowledge about the effects of (computer) systems as a starting platform. (p. 95)

New work practices, focusing on group work and the development of local resources for action, were being shaped, tried out in practice, and reshaped in the projects. Some of the work groups produced criteria for better working environments and suggestions for systems to support groups of workers planning their own work. As a result of the first of these projects, the existing legislation on worker influence was supplemented by Technology Agreements that gave workers a direct say in the development and use of technology in their workplaces. This also led to an extensive series of union education programs (Ehn & Kyng, 1987).

In the 1970s these early projects introduced the notion of worker participation in decisions about technology, but they ran into a series of problems. Writing about some of these projects, Pelle Ehn and Morten Kyng in *Computers and Democracy* (1987) noted:

> From a union perspective, important aspects like opportunity to further develop skill and increase influence on work organization were limited. Societal constraints, especially concerning power and resources, had been *underestimated*, and in addition the existing technology constituted significant limits to the feasibility of finding alternative local solutions which were desirable from a trade union perspective. (p. 29, italics added)

In short, whereas workers had a legal say in workplace technology, the laws did little to shift the balance of power from a managerial perspective. And, as discussed in the last section, the rationalistic tradition embedded in computer system development did little to give workers a voice in putting forth their own ideas when trying to agree on the introduction of new technology. As in the U.S., this was reflected in the tools of system development, which emphasized developing technical specifications rather than seeing the system from the perspective of the users.

By the early 1980s, a "second generation" of design projects was initiated in Scandinavia. These projects focused on using skill as a wedge to push computer system design more towards a users' perspective. They had as their theoretical starting point ideas put forth in the book *Labor and Monopoly Capital* by Harry Braverman (1974), which documented how capitalism increasingly took skill away from workers and brought it more and more within the hands of management. This process, called deskilling, intensified division of labor and resulted in work processes that were extensively routinized. Indeed, in the computer field in the 1960s and 1970s, this process of routinization resulted in the kinds of rationalized, cut-and-dried systems with which mainframe systems became synonymous (Greenbaum, 1979); systems that separated, for example, data entry from data verification, or customer relations from record keeping.

Braverman's point was that the act of dividing labor and deskilling workers was dehumanizing. The second generation of Scandinavian design projects took the issue of dehumanization and put it on the table as a central problem in the design and use of computer systems. Thus, to put some muscle on the bones of the Technology Agreements, the issues of *quality of work* and *worker skill* were put

ito the foreground of the computer system design projects. An cample of this was the UTOPIA project, named both for its ideals id as an acronym for its use (Bødker, Ehn, Kammersgaard, Kyng, Sundblad, 1987; see also Chapters 7 and 9). In this project com- ıter system developers and researchers worked with a group of pographers to help them formulate the ways that computer techno- gy could be used to enhance their skill and better the typographic ıality of newspapers.

In their quest to make workplace skill a more central aspect of mputer system design, these project organizers ran into severe fficulties in trying to apply the tools and techniques of traditional stem development. They experienced how limited these tools and chniques were in their ability to actually allow workers, as users, express their ideas about skill. In particular, they pointed out w the notion of tacit skill—those essential, yet not easily artic- ated qualities that we use in daily work—was difficult for com- ıter system designers to grasp using formal system specifications.

From the introduction of Technology Agreements in the 1970s, rough the focus on skill in the early 1980s, computer system velopment in Scandinavia struggled with the elusive concept of er participation. In *Work-Oriented Design of Computer Artifacts* 989), Pelle Ehn outlines the story of these changes and delves into me of the theoretical work that has helped some of the authors of is book reflect on their work practice and that of users. In general, ese theories can be grouped under the philosophical heading of cial construction, which sees our understanding of the world as nerated by people (through their social interactions) rather than as set of fixed, immutable facts (see Bruffe, 1986; Rorty, 1979). In ntrast with the rationalistic tradition of computer science, social nstructionist theory veers away from rigid poles like "objective- bjective," and steers toward understanding different, pluralistic rspectives of how we think and act. Seriously, system developers ve little room to hide behind a mask of objectivity, for developers, e users, need to get involved in day to day activities and learn to are perspectives.

In order for this book to move from user participation to active operation between users and system designers, several more ues had to fall into place. Among these, in both the United States d Europe, was the influence of feminist thinking. In *Reflections Gender and Science*, Evelyn Fox Keller (1985) points out that traditions of science have historically been rooted in a language it places value on words like "objectivity," "reason," and npersonal judgement." These terms are associated both with the y that "good" science is done, and with those characteristics that usually associated with men. Thus, historically within our

culture, to be emotional or personal has been considered "female," while rational and impersonal are more closely linked with male attributes. It is not surprising to find this gender-biased thinking within system development; indeed the rationalistic tradition of computer system development with its step-by-step impersonal and supposedly objective procedures is considered as "good" rigorous practice (Greenbaum, 1987). In searching for ways to overcome these biases some of the authors in this book have borrowed from activities that grew out of the women's movement in the 1970s, such as small group "consciousness raising" sessions that focused on getting women to "speak in their own voice," and encouraged participation by all (Bødker & Greenbaum, 1988).

Within the computer field, we were also helped by the thinking of Terry Winograd and Fernando Flores in their book *Understanding Computers and Cognition* (1985). The book, subtitled *A New Foundation for Design*, critiqued the rationalistic tradition and laid the theoretical groundwork for helping us understand that "in designing tools we are designing ways of being" (p. xi). Indeed, their emphasis on the importance of action and on the difficulty of articulating assumptions grounded our work in the importance of the action-based techniques in design. Hubert and Stuart Dreyfus in *Mind over Machine* (1986) added to our theoretical understanding that skill and expert performance cannot be captured as a set of formal rules. Their work also helped us to focus on how skills are performed within the context of specific situations.

In *User Centered System Design*, Donald Norman and Stephen Draper (1986) brought together a range of articles that address the issue of design from what they call pluralistic perspectives—looking at design with the user's point of view in mind. As one of the first major American books to place users in the foreground, they take Human-Computer Interaction (HCI) as a focal point for designing for people, not for technology. In Chapter 2 of this volume, Liam Bannon, also a contributor to that book, investigates some of the ways that we can now go beyond conventional HCI studies in order to move from user-centered design to more cooperative design.

In 1986 in Austin, Texas, the first biannual Conference on Computer Supported Cooperative Work was held (see CSCW Proceedings, 1986, as well as CSCW Proceedings, 1988, from the second conference). While the conference was not about cooperative design, it did bring Americans and Europeans together to talk about design of systems that support the cooperative and interactive nature of workplace activity. In 1989 the first European Conference on this issue was held in England (EC-CSCW Proceedings, 1989), and in 1990 the first Participatory Design Conference was organized in Seattle, Washington (PDC Proceedings, 1990). To us, it is clear

hat the time is ripe for mixing and matching the Scandinavian raditions discussed in this book with American projects that focus on user involvement. Some American system designers have said hat although they liked the Scandinavian approach, they feared that t wasn't applicable in the United States because of the weak trade-nion movement.[1] Scandinavia's high degree of union member-ship, however, may only be a partial blessing for participatory lesign, for in those countries, as in the U.S., established unions ometimes tend to be stuck in their ways. On the other hand, American discussions about cooperative work and team approaches o work tasks, while perhaps overstated in the business press, nevertheless point out some fertile ground for planting seeds for cooperative design.

What Have We Learned?

> About 20 years ago, a sign in the "in-room" of a large computer center in Denmark read: "The impossible takes an hour—miracles take two." In the output room another sign stated: "You are lucky to get anything at all!"

Traditionally, people in the computer field have been quite optimistic about what computers can do. To many users and managers, system developers, like salespeople, promise them everything but seem to deliver very little. Even during the 1980s, when system developers began to involve users in the design process, users continued to grumble about the "stupid mistakes" of the designers. Certainly the complaints made by designers about users are equally loud and numerous.

In this book we have no easy way out of this situation. As we found out, switching the focus from users as passive participants in systems design to active user-designer cooperation is not easy. We hope that the readers will not think that we are promising miracles. Yet from our actual project experiences, from our readings, and from our own theoretical work, we think that we have learned a good deal about what can be done. Here is a summary of the approach advocated in the following chapters. It grows out of our

[1] Obviously a number of other differences exist between design in the U.S. and n Scandinavia, differences which are important in relation to user/designer co-peration. A number of these, including the emphasis on "product-development" rojects in the U.S. vs. the predominance of "in-house-development" projects in candinavia, are discussed in Grudin (1990) and Grønbæk, Grudin, Bødker, & annon (1990).

experience in applying the design ideals mentioned in the beginning of the chapter.

- The design process needs to start with an understanding of the use situation. Traditional system development advocates beginning with the identification of "the problem," yet problems out of context have little meaning. That is why we believe that examining the context and paying close attention to the situations in which computers will be used is an appropriate starting point. To do this effectively, designers and users will find themselves stumbling along some untravelled paths as they try to learn about each others' basic assumptions.
- When computer systems are introduced within an organization they change the organization. Likewise, computer systems are not static entities, but rather systems that adapt as they are used. This dynamic process of ongoing change means that as designers we need to better understand the organization, and design for ongoing change. In the past, changes in work organization have often been looked at as "unintended consequences" of a new computer system. We don't think so. By designing with an understanding of work practice and its organizational setting, we think that we can, at least, be less naive about future changes.
- The design process is firmly rooted in experience, not just rules. Most computer system methodologies recommend step-by-step procedures for carrying out the design and implementation of computer systems. We don't think that these rule-based approaches should be thrown out, but we do believe that relying more on the experiences of designers and users can lead us toward systems that are more suitable for the workers involved.
- Users are competent practitioners. With this in mind we need to design for their skill, knowledge, problems and fears. Rather than planning "idiot proof" systems, we think that skill and quality of work should be given priority. Of course, not all users (or designers for that matter) are equally skilled, but diversity of skill is among the things we can learn by more carefully reflecting on work practice.

Early in this century, the famous American pragmatist and educator John Dewey, made a number of comments that we feel can speak directly to design at work. In his critique of education, he warned that students were seen as passive receptacles into which static bits of knowledge were to be placed. For us as system designers we are sharply aware of the need for active learning during the development process in order to avoid the trap of seeing users as passive receptacles. In *Experience and Education*, Dewey (1938) argued

that rote learning limits the "power of judgment and the capacity to act intelligently in new situations." (p. 27). Dewey, sometimes called one of the first social constructionists (Rorty, 1979), advocated making the learning process more active and firmly rooting it in the experiences of the teachers and students. As we reflect on the problems of computer system development today, Dewey's work makes lively reading. For example, he warns that "the central problem of an education based on experience is to select the kind of present experiences that live fruitfully and creatively in subsequent experiences" (p. 28). As the authors in this book set out to work with users in understanding each others' experiences, we are reminded that selecting and interpreting relevant experiences can be as difficult as selecting problems in the traditional approach.

For readers with a background in system development, we think it's worthwhile to present an overview of some of the major differences between our approach and the combination of approaches traditionally put forth in literature and practice today. Like all overviews it is rather simplistic, but we believe it can serve as a guide for understanding the book.

TRADITIONAL APPROACH	COOPERATIVE APPROACH
focus is on	*focus is on*
problems	situations and breakdowns
information flow	social relationships
tasks	knowledge
describable skills	tacit skills
expert rules	mutual competencies
individuals	group interaction
rule-based procedures	experience-based work

Figure 1. Contrasting Approaches.

Reframing from Within and Without

This book attempts to reframe our thinking about design at work with experiences from both inside and outside of the computer field. This process of reframing, or seeing things in new ways, is useful for presenting our ideas about experiencing the present and envisioning the future. Part I, Reflecting on Work Practice, is largely

based on ideas from the social sciences and humanities and their contributions toward helping us understand current work practices, especially the use of language and artifacts and their relationship to design. Part II, Designing for Work Practice, takes a look at ongoing computer system design projects and focuses on ways that designers and users can work together to envision future use situations as well as adapt systems in on-going use. The browsing readers can pick and choose the chapters that apply to their interests, but we suggest that Chapter 7 be read as background to Part II, before browsing in that section.

In the first two chapters of Part I the authors present a broad, conceptual overview of ideas that influence how we look at work and the use of computers.

Liam Bannon, with a background in computer science and cognitive psychology, builds a bridge between more conventional Human-Computer Interaction studies and cooperative design. His essay travels down a road, which, as he points out, may lead traditional research in HCI to address the ideas of cooperative design developed in this book. Specifically, he is critical of traditions in both psychology and human factors that place too much reliance on controlled laboratory studies and not enough on the actual setting of work.

Eleanor Wynn, a linguistic anthropologist, focuses on the power of perspective and the involvement that anthropologists use in their field work. Her work, applied to an understanding of conversation among clerical workers, brings to the foreground the complexity and problem-solving capacity found in daily work. Using language as a handle on assumptions, she points to ways that we can better understand what takes place at work.

The next three chapters present research from projects in anthropology, linguistics, and organizational behavior, suggesting strategies that we could apply to better understand how people work, talk, and use artifacts, this includes hints on how such an understanding may be used in the design of computer systems.

Lucy Suchman and Randall Trigg tell us about a project which uses videotapes to help researchers reflect on work practices. Lucy Suchman, an anthropologist, and Randall Trigg, a computer scientist, give us suggestions for how interaction analysis in general, and videotaping in particular, can be used in developing an understanding of the situated use of artifacts. They also discuss how their techniques may be developed and applied in a participatory approach to design.

In Chapter 5, Peter Bøgh Andersen and Berit Holmqvist, both linguists, take us on a tour of their field, explaining concepts and methods that could be used as a basis for system design. They

analyze work language in specific cases to help us understand how perspectives differ within organizational roles and with the task at hand. They continue by showing how these differences give rise to different demands which a computer system will have to fulfil in order to suit the work.

The last chapter in this part of the book takes a look at workplace cultures and lets us see how artifacts at work can be clues for understanding basic assumptions within the organization. Jesper Strandgaard Pedersen, with a background in organizational behavior, and Keld Bødker, educated in computer science, team up to give us examples of how shared values and beliefs can be looked at in small working environments. Their approach is different from traditional studies of organizational culture, which look at the values of a whole organization. By zeroing in on workplace cultures *within* organizations they give us ideas about how computers, as artifacts, could better reflect workplace values.

Part II opens with a chapter entitled "Setting the Stage for Design as Action" by Susanne Bødker and Morten Kyng, both computer scientists, and Joan Greenbaum, who has a background in system design and economics. This chapter gives an overview of the ideas and theories developed and used in Part II, and also looks at some of the common questions that people ask when trying cooperative design. The theoretical perspectives presented here build on the social constructionist views we outlined earlier. It begins with the notion that we only understand what we have already understood. One of the constant problems in system design, of course, is the issue of how users can imagine the future, and how designers and users can transform this imagination into practical computer systems. "Setting the Stage" gives some clues for what to expect when embarking on this process.

The subsequent chapters in Part II are laid out in a pattern which resembles the process that designers and users go through in designing, using, and modifying a computer system.

Chapter 8, by Finn Kensing and Kim Halskov Madsen, both computer scientists, suggests the use of Future Workshops and metaphorical design in order to help users generate visions about their organization and future computer use that transcend their current work practices. The Future Workshop technique uses specific methods to help people brainstorm about their current practices and its shortcomings and to find possibilities for future alternatives.

Chapter 9, by Pelle Ehn, educated in both sociology and computer science, and Morten Kyng, tells the story of how mock-ups were used in the early stages of design to help typographers envision the kind of hardware and software support they would need to enhance the quality of their work as well as of their products. The chapter gives examples of the ways that mock-ups and and even computers can be used to experience and envision different future possibilities before large amounts of money are invested.

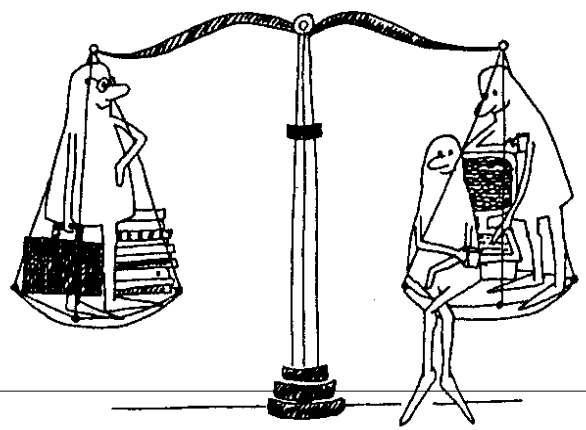
In Chapter 10, Susanne Bødker and Kaj Grønbæk, also educated in computer science, present ideas and examples for developing prototypes in cooperation with users. Prototyping as a tool for trying out future systems has been gaining popularity. The authors' focus, however, goes beyond presenting prototypes as demonstrations *to* users, suggesting instead ways that users and designers can change prototypes cooperatively.

The next chapter is not on implementation, as one might expect in a traditional book on system development. The gap between the tools and techniques of implementation and those used in the different application areas is still so big that we have not yet attempted to bridge it together with the users, although current work on object-oriented programming may improve this situation. Therefore, in Chapter 11 we turn our attention to the process which, in recent years has become known as "tailoring," or adapting and modifying systems once they are in use. Austin Henderson, educated in mathematics and computer science, and Morten Kyng present arguments for the importance of continuing design in use. They go on to give examples of how users are to take an active role in continuing design in use.

The last chapter in Part II, by Pelle Ehn and Dan Sjögren, educated in economics, takes us back to the problems of developing new work practices when technology changes. Supplementing the preceding chapter, this chapter focuses on how users can enhance their work practices as new systems are adopted.

These chapters in Part II look at design from the perspective of users and designers working together. Although the perspective of managers and organizations is an important one, much has already been written about it. Our approach is to flip the pyramid and look at the problems from what is traditionally viewed as the bottom or the end user perspective.

The epilogue, Chapter 13, introduces many of the ideas we would have liked to put into the book had we had more time and experience. It tells the tale of a Future Workshop in which all the authors participated, in order to "envision" the book that we would like to have written. Entitled "Design by Doing," it takes us from our current experiences in cooperative design to an action-based approach that we would like to imagine in widespread use in the near future. It summarizes the cooperative approach discussed here, but it also presents our frustrations about the present and our hopes for the future.



As a note on terminology to the people from different backgrounds who will be reading this book, we mainly look at the process of developing multi-user computer systems in workplaces and organizations. We are writing primarily for the designers, researchers, students, and curious users who are involved in that dynamic intersection where technical support and actual use situations meet.

When we use the term computer system, we use it in the broadest possible sense. Just as a building is more than the sum of the wood, nuts, bolts, and plans used in its construction, a computer system is both the entity itself and the way it is used. It includes the use of the hardware, such as processors, printers, monitors, and keyboards; it encompasses the design of software, whether it is custom designed programs for large-scale projects or the selection of off-the-shelf packages; and, of course, it focuses on the people who are going to use the new system.

The specialists who design and plan the construction of a building are conveniently called architects, yet the legions of technical and analytical specialists who put together computer systems go by many names. Here, we refer to the technical people as *designers* in order to keep our eyes on the *process* of developing computer systems. In traditional terminology these system designers may be programmer/analysts, system analysts, system engineers, or consultants, and their tasks may range from conducting feasibility studies through design and implementation of systems. But as all system analysts know, regardless of their titles or range of experience, the process of taking a system from a set of vague ideas to the actual use of computer tools is not a straightforward, linear one. Noting this, we use the terms *design* and *development* interchangeably, because both are on-going creative activities.

Our use of the term *user* may create less immediate confusion. However, among the authors, our inability to find a suitable substitute for the word created quite a lot of frustration, because this word groups all the different kinds of competent practitioners together under one label. Thus, when talking about "designers and users," the specific words do not emphasize the participation of specific workers who, as users, are active in a design process. This being said, we hope that the message gets through despite the shortcomings in the language.

In many instances throughout the book, the words "computer system" could be replaced with words covering other kinds of technology. For the reader interested in a broad understanding of workplace and technology issues, we offer numerous examples that could be creatively applied to different kinds of workplace issues. Additionally, although most of the examples deal with developing custom-designed systems for networked workstations or personal computers, the knowledge gained from these examples is, we believe, applicable to situations where off-the-shelf software is developed or adapted.

Each of the chapters in the book applies some ideas about cooperative design *depending on the situations* in which the designers and users find themselves. This emphasis on situated design means that we can offer the reader no straightforward method or universally applicable set of tools. We invite you to read, reflect, and try out the ideas presented here, based on the workplace situations in which you find yourself. We hope that our experiences spark your curiosity about cooperative design, leading you to the suggested readings and to try your own applications.

References

Ackoff, R. (1974). *Redesigning the future. A system approach to societal programs*. New York: Wiley.

Bjerknes, G. & Bratteteig, T. (1984). The application perspective—Another way of conceiving systems development and edp-based systems. In M. Sääksjärvi (Ed.), *Proceedings from the Seventh Scandinavian Research Seminar on Systemeering, Part II* (pp. 204-225). Helsinki, Finland: Helsinki School of Economics.

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

Boguslaw, R. (1965). *The new utopians—A study of system design and social change*. Englewood Cliffs, NJ: Prentice-Hall.

Braverman, H. (1974). *Labor and monopoly capital—The degradation of work in the twentieth century.* New York: Monthly Review Press.

Bruffe, K. A. (1986). Social construction, language and the authority of knowledge. In *Journal of College English, 48,* 773-790.

Bødker, S. & Greenbaum, J. (1988). A feeling for system development work—Design of the ROSA project. In K. Tijden, M. Jennings, & U. Wagner (Eds.), *Women, work and computerization: Forming new alliances, Proceedings of the IFIP TC 9/WG 9.1.* Amsterdam: North-Holland.

Bødker, S., Ehn, P., Kammersgaard, J., Kyng, M., & Sundblad, Y. (1987). A utopian experience. In G. Bjerknes, P. Ehn, & M. Kyng (Eds.), *Computers and democracy—a Scandinavian challenge* (pp. 251-278). Aldershot, UK: Avebury.

CSCW (1986). *Proceedings of the Conference on Computer-Supported Cooperative Work, Dec. 3-5, 1986.* Austin, Texas. New York: ACM.

CSCW (1988). *Proceedings of the Conference on Computer-Supported Cooperative Work, Sept. 26-28, 1988.* Portland, Oregon. New York: ACM.

Daressa, L. (Producer). (1986). Computers in context. [Film]. San Francisco: California Newsreel.

DeMarco, T. (1978). *Structured analysis and system specification,* Englewood Cliffs, NJ: Prentice-Hall.

Dewey, J. (1963). *Experience and education.* New York: MacMillan. (Original work published 1938.)

Dreyfus, H. & Dreyfus, S. (1986). *Mind over machine—the power of human intuition and expertise in the era of the computer.* Glasgow: Basil Blackwell.

EC-CSCW (1989). *Proceedings of the first European Conference on Computer-Supported Cooperative Work, Sept. 13-15, 1989.* London. New York: ACM.

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

Ehn, P. & Kyng, M. (1984). A tool perspective on design of interactive computer support for skilled workers. In M. Sääksjärvi (Ed.), *Proceedings from the Seventh Scandinavian Research Seminar on Systemeering, Part I* (pp. 211-242). Helsinki, Finland: Helsinki School of Economics.

Ehn, P. & Kyng, M. (1987). The collective resource approach to system design. In G. Bjerknes, P. Ehn, & M. Kyng (Eds.), *Computers and democracy—a Scandinavian challenge* (pp. 17-57). Aldershot, UK: Avebury.

Greenbaum, J. (1979). *In the name of efficiency—Management theory and shopfloor practice in data processing work.* Philadelphia: Temple University Press.

Greenbaum, J. (1990). The head and the heart. In *Computers and Society, 20* (2), 9-16.

Grudin, J. (in press). *The development of interactive systems: Bridging the gaps between developers and users.* IEEE Computer.

Grønbæk, K., Grudin, J., Bødker, S., & Bannon, L. (forthcoming). Improving conditions for cooperative design—shifting from a product to a process focus. In D. Schuler & A. Namioka (Eds.), *Participatory Design.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Hoos, I. (1961). *Automation and the office.* Washington, DC: Public Affairs Press.

Jackson, M. (1983). *System development.* Englewood Cliffs, NJ: Prentice-Hall.

Keller, E. F. (1985). *Reflections on Gender and Science.* New Haven, CT: Yale Univ. Press.

Kuhn, T. S. (1970). *The structure of scientific revolutions.* (2nd ed.) Chicago: University of Chicago Press.

Norman, D. A. & Draper, S. W. (Eds.). (1986). *User centered system design—New perspectives on human computer interaction.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Nygaard, K. & Bergo, O. T. (1975). The trade unions—New users of research. *Personnel Review, 4* (2), 5-10.

PDC'90 Proceedings (1990). *Proceedings, Participatory design conference.* April, 1990, Seattle, Washington. Palo Alto, CA: Computer Professionals for Social Responsibility.

Rorty, R. (1979). *Philosophy and the mirror of nature.* Princeton: Princeton University Press.

Suchman, L. A. (1987). *Plans and situated actions—The problem of human-machine communication.* New York: Cambridge University Press.

Weizenbaum, J. (1976). *Computer power and human reason.* New York: W. H. Freeman.

Winograd, T. & Flores, F. (1986). *Understanding computers and cognition.* Norwood, NJ: Ablex.

Wittgenstein, L. (1953, 1963). *Philosophical investigations.* Oxford, UK: Oxford University Press.

Yourdon, E. (1986). *Managing the structured techniques.* New York: Yourdon Press.