Integrating the User into Research on Engineering Design Systems

Peter Piela, Barbara Katzenberg, Roy McKelvey

EDRC 06-117-92

Integrating the User into Research on Engineering Design Systems

Peter Piela
Carnegie Mellon University

Barbara Katzenberg,
Stanford University

Roy McKclvcyt
Carnegie Mellon University

Table of Contents

[+] Peter Piela (pp0sOedrccmu.edu) is a research scientist at the Engineering Design Research Center at Carnegie Mellon; Barbara Katzenberg (barbk9ieland.stanford.edu) is a Ph.D. candidate at the Stanford University School of Education and; Roy McKelvey (rm0nOandrew.cmu.edu) is an associate professor in the Department of Design at Carnegie Mellon.

Forthcoming in *Research in Engineering Design*

Abstract

Engineering design research has historically been evaluated in terms of its computational performance. However, in many cases this research implies hypotheses about human behavior which are ignored. In this paper, we argue that the systems emerging from design research will benefit from investigating how people use them to accomplish work, and from the incorporation of principles of Participatory Design (Floyd, et al. 1989) at the earliest stages of the development process. Using a case study, we present the evolution of a CAD system that supports complex mathematical modeling. This design effort is examined according to principles outlined in another well-documented effort in Participatory Design. Finally, despite the common misconception that such considerations impede basic research, we argue that continuous user involvement can guide research by validating experimental hypotheses and pointing to areas for future inquiry.

1. Introduction

Research in engineering design has historically focused on the development of algorithms, formulations, and representations that can be shown mathematically to be feasible and superior in performance. There are, however, issues of interest to design researchers that cannot be evaluated solely by formal proof. Of particular interest to us is the performance of systems to support designers in solving non-routine design problems—problems that are not conducive to a primarily automated solution. We characterize non-routine design problems as "ill-defined" (Hayes, 1981) because they involve incomplete task descriptions and non-deterministic solution paths. Hayes says that "to solve an ill-defined problem, you may be required either to make decisions based on your own knowledge and values (gap-filling decisions) or to discover new information through your own active exploration of the problem (jumping in), or both."

Because non-routine design requires significant, on-going human intervention, facilitating designers in this work will depend crucially on how well systems support use. Given the difficulty of obtaining a measured characterization of this aspect of a system, it becomes difficult to establish the superiority of one system over another in any absolute sense. The systems

researcher is left searching for ways in which a convincing argument in favor of a technology can be made

Our work is based on die following beliefs: (1) that a rich source of understanding about a technology can be found by observing people using it to solve their own problems in their own workplace, and (2) that the traditional dichotomy between system development (with its emphasis on *technology* and the *system-designer)* and system evaluation (with its emphasis on *technology* and the *user)* is inappropriate if the goal is to explore a technology that requires a designer to significantly reconceptualize his or her understanding of a task. Further, we would argue that not only is it a mistake to delay consideration of how people work with the technology until the underlying theory has stabilized, but that there are advantages in moving such considerations to the start of the research process.

This paper describes a research project in which a new technology for equational simulation is being developed and evaluated by a collaborative team of developers and users. Our objective in this paper is to show through this example the following three points: first, that an experimental approach based on observation of, and involvement in people's actual problem solving efforts can allow for better evaluation of the underlying hypotheses of a technology; second, that a system can be viewed as an experimental apparatus for examining a new technology; and lastly, that a methodology of continuous user involvement in combination with analytic observation can create principled support for decisions in systems design.

The paper is arranged as follows: Section 2 describes the motivation for investigating systems in terms of their use as a part of design research. Section 3 is a description of the engineering system called ASCEND (Piela, Epperly, Westerberg, & Westerberg, 1991) that forms the basis of our case study. The description emphasizes aspects of the system that could only be studied through active participation by users. Section 4 provides details of the system development effort and the community who became engaged in various aspects of this effort. Section 5 is a closer look at the methods employed in learning from users with specific, illustrative examples, and Section 6 is an analysis of the ASCEND development process in terms of Participatory Design principles.

2. Motivation lor the Development Effort

In response to the need for improved industrial competitiveness in the United States, the National Science Foundation (NSF) has initiated the Engineering Research Center program which has located centers in universities around the country. Each center has been chartered to study particular aspects of engineering research. The Engineering Design Research Center (EDRC) at Carnegie Mellon has directed its efforts to the interdisciplinary study of design theory and to the development of computer-aided design systems. Specifically, the EDRCs mission has been to develop advanced design methodologies that drastically reduce the design-to-product cycle, and to build flexible, domain-independent, design environments that integrate quantitative and qualitative methods for design optimization. Projects that are directed towards these goals should ideally focus on forward-looking research issues yet remain grounded in actual engineering practice.

The desired balance between theory and application has been difficult to achieve. All too often system users are treated as noise in the design process with little thought being given to how the emerging technology might be integrated into a workplace of existing practices, and whether there are in fact social or cognitive barriers to the technology's acceptance.

We believe that if the dominant issue is practical problem solving then omission of considerations of how people use different technologies can seriously skew debates about the merits of one over another. For example, in the area of mathematical modeling as practiced by chemical engineers, the established style of reporting work is to discuss the theoretical properties of the model formulation and the computational results on a set of example problems, omitting any mention of the human effort required to achieve the computational results. Indeed, if a given problem requires a hundred initial starting points to generate the reported solution, it is not uncommon for this information to be omitted from the discussion. Based on our experience, and those of other academic and industrial colleagues, we believe this to be a critical omission, and believe that these factors need to be incorporated into the more general debate.

For example, there is an on-going discussion about the use of mathematical programming versus rule-based expert systems. Since production rules (Horn clauses) can be posed as mathematical constraints, there are those who argue that mathematical programming has subsumed rule-based technology because it will guarantee an optimal solution for large classes of problems whereas rule-based technology gives only satisfactory solutions. However, Dhar and Ranganathan (1990) compared two approaches for a teacher assignment problem, and judged the rule-based system to be superior for the following reasons: (1) the problem could be described in more natural terms; (2) it was possible to ask for explanations about the nature of a solution; and (3) it was possible to make useful interventions during the solution process. This debate raises a general set of issues associated with how mathematical modeling is accomplished in practice which we are studying in the ASCEND project.

3. A Description of the Technology

The work we describe in this paper is taking place in the Design Systems Laboratory of the EDRC and focuses on the evaluation of a new approach to Equation-Oriented Simulation (EOS) (Gupta, Lavoie, & Raddiffe, 1984; Perkins, 1984; Shacham, Macchietto, Stutzman, & Babcock, 1982; Westerberg & Benjamin, 1985; Westerberg, Hutchinson, Motard, & Winter, 1979). EOS is used to analyze design alternatives in situations where a design can be described as an explicit set of mathematical relations. In short, EOS can be divided into two major tasks:

(1) *formulation* in which the designer constructs the system of equations, and

(2) *solution* in which the designer fixes the values of some of the variables in the equations and then uses an independent numerical solving algorithm to compute the others.

This method has been shown to have applicability in many areas of engineering design and affords the designer a degree of flexibility that is missing in most current methods; however, it has not been widely adopted in engineering drcles. The limited use of EOS is hardly surprising because it requires that the designer contend with many inherent complexities. Generated simulations are typically very large (several thousand equations); problem formulation is intolerant of error; and solving algorithms do not succeed under all conditions.

To date, most research in EOS has focused on developing better solving algorithms with a view of the simulation system as a black box into which designers place equations at one end and collect answers at the other. Our assessment of EOS is radically different We believe that good results can be obtained only with significant human intervention throughout an often iterative process, and that the success of this intervention will be a direct function of the way in which people formulate their problems.

Presently, two basic methods exist for formulating equational simulations. The first provides a library of predefined mathematical procedures, but requires the designer to write equations in a conventional imperative programming language. The second is typified by the GAMS system (Kendrick & Meeraus, 1985), which significantly improves modeling capability by providing a special-purpose declarative language in which equations can be written as if on a piece of paper. However, GAMS only permits the writing of unstructured sets of relations, and has been described as too low-level and too inflexible for solving real-world design problems (Dhar & Ranganathan, 1990).

Our work has been directed toward the development of a high-level specification language that is both declarative and structured. The addition of structure has been primarily motivated by a desire to encourage people to use a "divide and conquer" approach to simulation. It is our intent that simulations written in this language will be correct[1] when submitted to the solver, and allow for efficient debugging in case of solver failure. To support this activity, we are currently developing a set of highly interactive tools for working with simulations created by the language.

[1] Correctness in this context includes the syntactical and semantic correctness of the problem descriptions, and the dimensional correctness of the equational simulation.

However, the use of structured languages such as die one envisioned above is completely outside the experience of most people working with EOS systems. Therefore, (me of the primary goals of our research has been to investigate what effect the specification language has on the character of the overall process. That is, will designers who are given a special-purpose language be able to take advantage of it as hoped by the developers? We also expected that the language could radically alter the way in which people conceptualize the task of EOS, but what form would this new conceptualization take? How willing would users be to work with a system that expected their active participation *during* the solving process and how might their participation be made efficient and relatively pain-free?

It has been argued that new technology inevitably enables and constrains the way people think about problems, and researchers have stressed the need for studies that track the relationship between users and modes of technology (Ong, 1982; Penzias, 1989; Winograd & Flores, 1986). Given our questions about the viability of this mode of technology, we sought research methods that would allow us to document the experiences people had with it—at a level of detail that could directly inform development

4. The Development of ASCEND

A commitment to studying people's work with a technology implies building a system that can function as an adequate testbed for generating and testing hypotheses about use; and finding people who can test this system through their own work. In this section we describe some of the constraints we placed on the development of our system, the growth of a community of developers and users who shared an interest in its application, and a brief chronology of how work proceeded. In the section that follows, we will provide a detailed discussion of the methodology employed for learning from users and their work-

4.1 Constraints on the Design of the System

We wanted a system that supported analyses of use early in the development process. This required that it be easily modified in response to user feedback (because so much was open to change) and careful attention to interface design (so users could attend to fundamental issues, not system-level deficiencies). We imposed the following constraints on our system development work: First, because we hoped to solicit feedback from multiple disciplines, we chose to keep the system domain-independent. Second, because of the newness of the modeling paradigm (e.g., object-oriented modeling, frequent and in-depth debugging, parametric manipulation of existing models) we assumed no preliminary knowledge of which tasks the system should support. Third, rather than exploit such mechanisms as metaphor to "protect" users from the newness of this paradigm, we chose not to hide the technology, and instead to communicate its (evolving) nature as clearly as possible. Lastly with the anticipation of rapid iteration, we sought a design framework that would allow for modification with minimal upheaval to both users and developers.

4.2 Users and Developers in ASCEND

The basic concepts in ASCEND grew out of a collaboration between Arthur Westerberg and Peter Piela as a thesis project in chemical engineering, beginning in early 1986. The aim was to address the problems both had encountered in industrial experience with existing simulation technologies, and a structured modeling language was hypothesized as a possible solution to these problems. As these discussions became more concrete, conjectures were made about what kind of system would be required to realize these ideas further. Participation in the shaping of a system was sought from both inside and outside the chemical engineering department, attracting a number of contributors, some of whom became developers and some who became users.[2] The resulting development group included Westerberg and Piela (cf.,

[2]**For this article, we define a *developer* as any contributor to the system that has "final say" on how functionality is implemented. *Users* are defined as all others involved with the system-no distinctions are made here based on the amount or perceived relevance of use.**

Westerberg, 1990); two representatives of the Design Department conversant in human factors, graphic design and user-interface issues; and two undergraduate programmers. The early users were engineering graduate students and a faculty member from the Department of Architecture. Some of these users had facility with existing mathematical modeling approaches, and others experience with structured languages. Each influenced the development in different ways. For instance, die architect focused on representational aspects of ASCEND and helped in the refinement of certain features of the still evolving language, while the chemical engineers looked for stronger support in solving, which motivated a sharper definition of the support needed by users in interacting with simulations.

At the beginning, all that existed was a mock-up of the interface with a mechanism for creating simulations from a subset of the language and for browsing their structure. There was no capability for solving systems of equations. However, it was possible to provide a sketchy demonstration of how people might interact with structured models. Using this as a starting point, group members were able to volunteer some thoughts about what they might look for in a prototype system. We then moved toward a functioning prototype through an evolving series of mock-ups, each of which was presented and discussed within the group. By the winter of 1988, we had developed a baseline system—kernel and interface—which could actually be used by members to solve problems.

During the summer of 1989, the user community was expanded to include a group of off-campus participants, primarily engineers from industrial sites. These participants were introduced to ASCEND in a one-week workshop course held at the EDRC, and afterwards, there was a continuing cycle in which the hardware and software for the current version of ASCEND would be sent to an individual worksite for an average of about three months. During this time the participants would communicate via phone and electronic mail about the work they were attempting and specific difficulties they were encountering. At the end of the time they also provided feedback in the form of an oral or written report.

What we learned from the experiences of both on and off-campus users was incorporated into new versions of the system, and the evaluation work continued. In this respect, our development team employed what Floyd et. al. (1989) refer to as *evolutionary prototyping* techniques: They describe this as a development strategy that encourages people to view the product as "a series of versions."

> In this way each version can be looked upon as the "prototype" of the subsequent version. The system development process is seen as a sequence of development cycles, not as a series of development steps to be executed only once in a linear manner. Changed and new requirements can, in each case, be incorporated into the next cycle (313).

This process of generating new versions of ASCEND and evaluating them through use has continued for the past three years. During this time there have been changes in the group's make-up. A social science graduate student has begun contributing to the development effort by performing detailed observational analyses, and there have been changes among those implementing the system. As for users, each year one or two new people at the EDRC decide to try to accomplish work using ASCEND, and there has been a continuation of the off-site industrial program.

5. Learning from Users and Their Work

In this section we provide details of how we have gathered and analyzed data from users. We began with the assumption—since the technology represented a major shift in approach—that neither we nor the users would know in advance how to articulate relevant questions about system use. Because of mis, we reasoned that the approach of asking users "what they do" or "what they want" (via interview, questionnaire, or conversation) would have only limited value. This reluctance to rely too much on self-reports was also drawn from research that shows systematic bias in how people answer such questions. Self-reports are not filtered neutrally; they reflect the cognitive categories people have used to represent the world, rather than being accurate reflections on the world itself (cf., tfAndrade, 1974; Jordan, 1983).

We have also found that people rarely formulate their opinions in terms of the system as an artifact if not directed that way explicitly. Instead they tend to talk in terms of their own problems, which might or might not map to general system issues. When we have directed the discussion toward system attributes —in discussions about a hypothetical change, for example—we have found an unpredictable relationship between what people articulate, and how they respond to an actual change. [3] In sum, although users' views were welcomed, the common understanding was that opinions (including our own) were a starting point for a closer look, not reliable data by themselves.

To gain a fuller image of the users' experiences we have looked for data sources that depend less on individual memory and which allow—by virtue of their availability to more than one person's view—for the development of a shared interpretation within the group. We have relied on three major categories of data, which we will describe in turn.

5.1 Conversations during Problem Solving

First, are the conversations (in person, phone, or electronic mail) that are generated during actual problem solving efforts. The key feature of these conversations which makes them valuable is that they are problem-driven, so system issues discussed arise from actual work situations, not imagined ones. Conversation via electronic mail, which is extensively used in this community, has the added benefit of being easily distributable among many parties, and leaving a historical trace for later review.

[3]Another difficulty pointed out by Rosenbrock (1989) is that when users are asked whether they would like to see a certain option included, it is the nature of such an interchange to have a default answer of "yes," since the penalties associated with an accumulation of such answers—in terms of overly complicating the system or costly development time—are hidden (11).

Vignette a **conversation within** a **problem** solving situation

This is an example of an email conversation between a user (a chemical engineering graduate student) and a developer (Peter Piela) which informed both the individual's problem solving and the development effort The student was using ASCEND in solving an optimization problem and, having formulated the problem, attempted to solve it with one of the algorithms attached to ASCEND known as MINOS (Stanford University, 1977). It failed, reporting through the system that there was too little memory allocated. He wrote Piela-who was then working in another city-about increasing the allocation, but also asked that ASCEND be modified, so he could "have more freedom in setting the solving parameters for MINOS, just as you would in an options file for GAMS" (as he later wrote). GAMS (Kendrick & Meeraus, 1985) is an environment he knew well, and tweaking the parameters of the algorithm is a standard approach for obtaining solutions within it. As a first step, Piela immediately increased the allocation since this was easy to do, but when the student attempted to solve the problem again, it failed to converge. Piela, who had also used MINOS for optimization, knew that it was sensitive to variable scaling, i.e., it worked best when values of variables were not orders of magnitude apart from one another. So, he investigated whether the formulation could be improved with better initial values. He had been experimenting with an analysis tool for ASCEND that would enable users to easily check the scaling of their variables. Using it, he found that some of the student's variables were in fact badly scaled and wrote to him suggesting he check these. Based on this advice, the student was able to assign better initial values to the badly scaled variables and the problem converged. This circumstance led him to generalize a formulation strategy; he developed a procedure for computing good initial values for this class of problems.

For the developers, this interchange provided direct evidence that improved tools for debugging during solving were needed. It also underlined the difference between the existing emphasis on the solution phase of EOS (i.e., the practice of tweaking the algorithm) and the emphasis in ASCEND on better formulation—which had implications for the kind of support people might need. Note that this evidence became available only by developers cooperating with users in the context of actual problem solving. Another outcome of the interaction was that the student became sensitive to the scaling problem in MINOS; months later he was able to offer very similar advice about scaling when an industrial participant wrote him because he had reached an impasse in a comparable optimization problem.

The debugging problem which is documented in this vignette is a specific one that emerged from a novel combining of the ASCEND approach, a particular problem type, and a user's approach to solving it. Thus, we argue that the modifications to both system and work practice implied by it (and similar instances), could not have been planned in advance, no matter how careful the preliminary analysis.

5.2 Observation

The second data source we have used is observation of people working with the system, particularly using videotape. Videotape analysis is most fruitful when at least two people can watch together, ideally including the person whose working has been recorded. In addition to everyday activity, we try to observe people who are new to the system, because many difficulties inherent in a technology are masked once a person becomes proficient (Suchman, 1987). The data collected through observation may provide evidence to support opinions that have been voiced, but may also contradict the expectations that both users and developers have about how their work proceeds.

What follows is a vignette that demonstrates the use of videotape in ASCEND evaluation work. The subject is an apparently trivial one: a user's response to the mechanism for cancelling pop-up windows, which is the type of detail that is typically dealt with later in the development process. We believe, however, that such details can make the difference between whether a technology is eventually accepted or written off as "too foreign/' and have attempted whenever possible to attend to them.

Vignette: video observation of a user and ASCEND pop-ups:

Pop-ups are used in ASCEND when a user needs to enter data such as a file name, and to inform users of certain system states and errors. Our initial design followed the conventions of the hardware platform on which ASCEND was implemented. A pop-up was made to cancel when the user hit <return> after typing an entry, which is typical, or whenever the cursor was moved outside the pop-up borders. This latter differs from the more common implementation (e.g., on the Macintosh), which requires an explicit "OK" or "Cancel" in order to make the pop-up disappear. However, consistency with a platform in common use in the community was considered worth preserving, and this implementation also followed the maxim (frequently dted by both developers and users in the project) of minimizing keystrokes. Later, a new team member watching videotapes of a user at work flagged a few sequences as potentially problematic, and brought them to the attention of the group. In attempting to perform an operation within a pop-up, this user would accidentally move the cursor outside of its borders thereby causing a cancellation. This required that he start again many steps back, and from his occasional mutters, she inferred his frustration. Once we became sensitized by watching the tape, we saw that even the most skilled ASCEND users occasionally had the same difficulty.

No users called this to our attention even though we assume it to have been undermining of a their overall efforts. This is typical of a class of apparently superficial implementation problems which tend not to get talked about, but which can be illuminated in videotape review.

The videotaped sequences did not constitute a proof of bad design, and in a sense did not even strike the other team members as news. (They had considered revisiting pop-up design, but other issues seemed more pressing.) Instead, the videotape allowed one participant to argue for a point of view in such a way that others could also make judgments. The eventual outcome of the discussions was the redesign of pop-ups to require explicit cancelling, a decision which, like all ASCEND changes, is still subject to recall. However, to date, there has been no evidence against the change.

5.3 Solved Problems

The third data source used is the products of people's work with die system—the partial or complete solutions to their problems. Because we have hypotheses about how ASCEND will change practice, having the tangible products of people's efforts allows us to refine our ideas and to investigate ways in which the solved problems are distinctive or similar to those produced with existing technologies. In this section we will examine three characterizations of solved problems that have emerged during development as revealing ones: the degree of nestedness, the degree of structuring, and the degree of evolutionary style demonstrated in people's model building.

Degree of nestedness: The area of chemical engineering process analysis was founded on the view that processes could be described in terms of a set of standard unit operations and a set of streams which connect them. Most process modeling languages recognize this convention, and support the construction of complex models in terms of streams and unit operations. However, these languages also restrict the user to making statements at one of three levels: process, unit operation, or stream. This approach although providing a good framework for describing typical characteristics has in most systems been augmented with other facilities for describing non-standard aspects of the process. In our work we were interested to see whether given a modeling environment that did not impose the traditional framework whether engineers might employ different decompositions based on individual problem situations. The ASCEND language was therefore designed to support any number of levels of specification (nesting) through the creation of "glass box" models whose entire internal structure could be referenced by the user. We were concerned, however, that there would be a point at which deeply nested models would become cognitively overwhelming. Because of this difference, both we and the companies who had developed these more restrictive languages were interested in the degree to which users developed deeply nested models, and the kinds of decomposition strategies they employed. We have kept track of this dimension over example problems contributed by users, and have found a variety of decomposition strategies and routine use of level of nesting that are greater than three. In the area of chemical engineering in most cases we have seen the reconstruction of familiar unit operations; however, we have also seen different internal decompositions attributable to different styles of modeling.

Degree of structuring: Another statistic we have tracked is the number of equations in the entire model divided by the number of submodels. This ratio is employed as a crude indicator of the amount of decomposition and structuring people have used.

Degree of evolutionary style: Lastly, we have sought ways of characterizing an evolutionary "style" of modeling. The way we have adopted is to count during compilation the number of acts of model *refinement,* and compare this to the number of acts of model *creation.* The larger the ratio, the greater the indication that people are using an evolutionary approach.

In Section 5, we have described three types of data: conversations during problem solving, observation of users at work, and solved problems. Our goal in this data gathering and the subsequent analytic work is to fold back what we have learned directly into the development effort. If a change seems warranted, we present the ideas of a redesign to the active users who can register their support or misgivings. Since users don't always agree with developers, the process is one of negotiating points of view. This often leads to false starts when factors which seem important fade from attention while others, initially unnoticed, come to the fore.

Our method is closely aligned with the concept of Participatory Design, which is an approach to the development of workplace tools originally developed in Scandinavia, aimed at finding strategies that can allow continuous interaction between developers and user groups (Bjerknes, Ehn, & Kyng, 1987; Floyd, 1987). This approach acknowledges that all the information needed to develop new technology cannot be known at the beginning of a project. Instead, design must evolve through a process of mutual learning in both developer and user communities (Thoresen, 1990,34). The crux of the participatory method is the development of a *shared understanding* of the goals, roadblocks, and accomplishments possible with a new technology.

6. Participatory Design in ASCEND

Most of the Participatory Design literature coming from Scandinavia illustrates the method via collections of case studies. True to the spirit of the Scandinavian approach—i.e., that system design is aimed primarily at how it affects end-users—most results are discussed with respect to particular successes in particular situations. Formal procedures for engaging in a Participatory Design effort are not usually offered; nor are clear-cut measures for evaluating success. A design effort is characterized not only by the quality of the finished product, but also by that product's effect on existing practices and the degree to which the development process was collaboratively achieved.

In analyzing one particular design effort at the Xerox Corporation,however, Blomberg and Henderson (1990) did develop a set of principles that constitutes an idealized model of Participatory Design. They then used these principles to judge the extent to which the system design effort they were studying conformed to this ideal. In the section that follows, we follow Blomberg and Henderson's lead and discuss our design process using the framework they employed.

In some **categories the match is closer than others. For example,** ASCEND is a research **project and is not** directed toward a specific end-user community. **(For ASCEND-like technology to become accepted** in **end-user** communities, we envision **that further, domain-specific participatory** efforts would be required.) Most **strikingly, the project's goals are technology-centered,** which is perceived **by** some to be in direct conflict **with the whole** spirit of participatory design. We contend that **by applying these methods** where they are rarely used—at the earliest **stage** of **the introduction of a new** technology— we have extended their **range** in **a way that** could have a salutary effect on downstream efforts. However, as such, a certain degree of technology-centeredness is unavoidable.

What follows in an examination of the relationship between the ASCEND project and general participatory methods according to the principles outlined by Blomberg and Henderson, which will be examined in turn. These are:

- Establishing common criteria for evaluating success by all members of the group.
- Strengthening the character of user-developer interaction.
- Aiming to improve the quality of work life.
- Creating organizational structures that support collaboration.
- Supporting mutual appreciation of each group member's various competencies.
- Emphasizing design as an iterative process.

Establishing Common Criteria for Success. In a true Participatory Design approach, the focus and goals of system development are actively negotiated between the developers and users—they are not imposed by one group upon the other. An important feature of this negotiation is that there is equal respect present for both technical and work-related expertise. This requirement might imply that developers sacrifice part of the system design which seems particularly elegant to them, or that users agree that they would benefit in the long run from adopting new work practices **made** possible by a new technology.

In ASCEND our goals have been aligned with the users in the sense that we wanted people to be able to work with the system, because this work was needed grist for our continued research. On the other hand, ASCEND was developed to test a set of assertions about modeling practice, particularly that large, complex models would ideally be built from smaller ones in a structured fashion, and that careful formulation using a structured language would pay off in more successful solving. These prescriptions are at the core of die research effort, and as such were not considered negotiable.

For example, ASCEND development began with the assumption that the formulating language would allow users to specify their problems in such a way that solver failure would be very rare. So, early versions of ASCEND did not have a solver debugging facility; however a makeshift capability was added to support the developers in debugging the software. Although users knew of the existence of the debugger, the developers suggested they not use it, and rather focus on getting their problem formulations correct.

One user, however, found he could save time by minimizing his formulation efforts and by taking full advantage of the debugger during iterative attempts at solving—thus manipulating solver failure to his own advantage. His success (as evidenced by the size and difficulty of the problems he tackled) awakened the group to the legitimacy of this alternate approach. At the same time, the developers' hopes that solver failure could be avoided entirely with careful formulation were not borne out. The need for a good, debugging facility in the solver became increasingly apparent, resulting in the prominence of the current, frequently used tool.

In another situation/ a request was made by a user for the addition of a special mode in which lists of equations could be rapidly entered in an unstructured manner. Although the technology did not actually prevent this behavior, we had hoped to discourage it The user agreed that structure was important for large problems, but complained that simple lists are adequate for small problems. The question became one of determining "how big is small?" We were able to dte a specific case in which someone had spent hours trying to understand why a problem with four equations was giving unexpected answers. This user had finally concluded that there was a bug in the system; however, a closer inspection of the situation revealed that his formulation was inaccurate. We were able to show that the problem was avoidable with a structured formulation, and in this case no system change was made.

These situations are two sides to the coin of developer's prescriptions. In the latter case, the outcome of the mismatch was an informal dialogue which still continues. In the former case, the ASCEND technology was altered to better support the user practice.

Strengthening User-Developer Interaction. An important feature of the Scandinavian method is that the development and design of systems takes place predominantly in the workplace of the user. The goal is increased contact between developers and users in situations where actual work is taking place. This enables " developers to see people working with the technology, and gives users access to support when and where they need it. This level of interaction is difficult to achieve if even relatively small geographic or temporal distances stand between the two groups.

During the first year of ASCEND development, both users and developers were located on the same campus and, for the most part worked in the same building. More recently there have been extended visits to campus by off-campus participants and trips to

industrial sites by members of the development team. Another point of communality was that some members of the ASCEND development team, like the users, had competency in the domains for which the system was constructed, often very specific This combination of geographic proximity and shared expertise brought the groups closer together.

Most importantly, however, was a willingness on boih sides to undertake extensive learning efforts-the users to assimilate a new language and method of working; and the developers to support them by getting involved in actual problem solving efforts and by getting acquainted, when necessary, with new domains of application. [4]

Improving the Quality of Work Life. In the Scandinavian model, the primary focus of systems design is not on technology but, rather, on the quality of life that the technology implies for future users. This objective is agreed to by both the developer and user groups, and care is taken to balance traditional work practices against those which would be required by the new technology. Blomberg and Henderson (1990) point out that more is required of the two groups than having "the intent to cooperate" (356). Rather, it is how that intent is manifested in the actual interaction between the two groups, and to what degree each group is willing to compromise. In particular, it must be understood that a system may fail in practice for reasons that are not easily responded to by modification of the technology alone. Our interpretation of the Scandinavian literature is that the quality of work life issue has been a particular focus when a workplace is being transformed from one that is largely manual, to one in which computer technologies will play a major role. As such, there is a great concern about what might be lost through the transformation, and it is considered essential that

[4] Curtis et al. focused on the importance of developers' knowledge of the application domain in their review of 17 large software development projects. In their interviews a lack of this knowledge was seen as a major reason for difficulties, whereas developers with superior domain knowledge stood out as exceptional and central to a project's success (1988, p 1271).

those people who will be affected have a say in governing the transformation process.

We see the ASCEND situation as fundamentally different. Although a change of technological paradigms is being proposed, the community for whom it is being developed are already technology-centered in their work. More importantly, those who use ASCEND have not had.it thrust upon them, they have chosen it because it suits their own goals. In moving toward it, we assume these people have taken their own quality of work life into account in making the choice.

Providing Organizational Structures to Support Collaboration. In the Blomberg and Henderson study the developers and users of the system belonged to completely different organizational units, and reported to different areas of management. Such a situation introduces an aura of risk to the negotiation process, with each group having a vested interest in protecting their own areas of expertise and authority. Also, collaboration between groups requires people to interact with one another in situations where they are, by definition, less-than-expert. If the circumstances are such that people are being constantly evaluated for what they have and haven't achieved, there will be less willingness to be placed in such a position.

The EDRC provided an environment with fewer organizational constraints than many businesses. This looser structure made the needed collaboration easier to build and maintain. It may also be that existing cross-disciplinary ties in the center made people somewhat more likely to take risks in working outside their domain of expertise. This relaxed atmosphere did not extend to the industrial participants, however, who were generally less likely to want on-site support and less likely to want to be videotaped at work. While not entirely stifling collaboration, this difference affected its character.

Supporting Mutual Appreciation of Differing Competencies. In addition to considering the effects of environment and organizational structures on use, Participatory Design also takes into account the effects of social interactions between developers and users. For the Participatory method to succeed, these groups must acknowledge that each possesses information which is critical to the ultimate success of the system. Further, it is expected that, through the course of the iterative design process, there will be significant transfer of work-related constraints and goals to the developers and, reciprocally, of technological constraints and goals to the user community. In the ASCEND project, this flow of information was made easier when the users and developers had shared skills.

However, as a separate issue, we want to take up the point of how competencies can be communicated through the community, particularly as the group becomes more dispersed. For example, we would ideally like to expand the scope of our work to a include a larger group of off-campus users. However, earlier industrial users have described difficulties in working with the system, some attributing this to their lack of familiarity with object-oriented concepts as well as the structured approach to formulating equational models. A frequently cited problem has been the absence of adequate support structures (e.g., documentation and examples) to help in crossing the barrier inherent in the newness of the ASCEND approach.

One style of working users have chosen is to peruse existing examples for those that most closely resemble their own, to get ideas or to find parts which can be reused. Unfortunately this strategy has been poorly supported. Although our examples are valid solutions to problems, they do not explain how they were derived, and, therefore, by themselves have minimal instructional value. We are currently investigating mechanisms for building "active exemplars" which could increase the availability of missing information. These exemplars would

integrate process and product information as well as providing **advice on** modeling strategies. Ideally, such a mechanism could **be used by** people who have developed and solved problems **relevant** to **a** certain domain and want to communicate their ideas to others within that area.

Emphasizing Design as an Iterative Process. At the heart of the Participatory Design method is an emphasis on iterative design. It is expected that the systems will undergo substantial re-definition as data is gathered from people's working. This process is intentionally focused on the discovery of issues and the ability to rapidly pose responses to those issues for further evaluation.

In the development of ASCEND, alterations to the system are often made on a day-to-day basis, and major revisions have come almost monthly. Blomberg and Henderson (1990) note that there are two views one might take of iteration: as a sign of failure, in that it indicates something wrong with the current system; or, as an indication of progress, assuming that change is generally indicative of improvement. We take the latter view. As well, in the practice of iteration we sent a dear message to the user group that their feedback would be integrated into the development process.

Frequent iteration has, however, some disadvantages. One infrequent user looked ambushed when he attempted to demonstrate the system and found that a key tool had been changed. Another user stated that he looked forward to a time **when** the system became more stable. Communication about modifications is also an issue; our attempts to keep people informed via an electronic "change log" has been only partially effective, because people tend not to read it. Another unwanted side-effect of the rapid iteration process is that small tutorials people used to create in order to help newcomers became quickly out-of-date. Now, the tendency is not to bother, so there is one less support structure for those who are trying to get started. From the developers' standpoint there are also frustrations. The care being put into the constant building of detailed prototypes— of a new tool, for example— is sometimes revealed as wasted when it is shown that there is something wrong in the underlying conception.

7. Conclusions

In this paper we have aimed to show that there is information vital to a technology's success that can emerge only when people begin to seriously use it in their work. We have argued that opinions about the form and function of a system must be supported with evidence culled from continuous data gathering efforts; and that this requires a set of methods that are observation and problem-centered. At the heart of these methods is the view that the success of a technology depends as much on the work people do to integrate it into their practices as it does on any inherent quality of what developers produce. Given this view, researchers in systems design must find ways to accommodate the social and cognitive environment in which they expect their systems to have a role.

Participatory methods require frequent contact with users in their workplace, a sympathetic organizational structure, and a willingness on the part of all involved to respect one another's viewpoints. Not surprisingly, these methods have not always transplanted well outside of Scandinavia. It may be, however, that university-based research centers provide one kind of environment that can support technology development in which it is possible to seriously consider the user.

Acknowledgments

We thank Christina Allen for her important contributions to early phases of this work and to die ideas presented here. We also thank Tom Epperly, Vincent Lobe, Oliver Smith, Art Westerberg, Karl Westerberg, Rob Woodbury, and Joseph Zaher, for their active participation in the development of ASCEND; and Beth Adelson, Jeanette Blomberg, Susan Irwin, and Brad Mehlenbacher for their detailed critiques of earlier drafts of this paper.

References

Bjerknes, G., Ehn, P., & Kyng, M. (1987). .*Computers and Democracy—A Scandinavian Challenge.* Aldershot, England: Avebury.

Blomberg, J. & Henderson, A (1990). Reflections on Participatory Design: Lessons from the Trillium Experience. *CHI '90 Conference Proceedings: Human Factors in Computing Systems.* Seattle, WA: ACM P, 353-360.

Curtis, B., Krasner, H., & Iscoe, N. (1988). A Field Study of the Software Design Process for Large Systems. *Communications of the ACM,* 31(11), 1268-1287.

Dhar, V. & Ranganathan, N. (1990). Integer Programming versus Expert Systems: An Experimental Comparison. *Communications of the* ACM, 33 (3), 326-336.

D'Andrade, R. (1974). Memory and assessment in behavior. *Measurement in the Social Sciences,* Blalock, H. M. (Ed.), Aldine Publishing Company: Chicago, 159-186.

Floyd, C. (1987). Outline of a Paradigm Change in Software Engineering. *Computers and Democracy—A Scandinavian Challenge.* G. Bjerknes, P. Ehn, & M. Kyng (Eds.). Aldershot, England: Avebury, 191-210.

Floyd, C, MrtL.W-M., Reisin, F-M., Schmidt, G.,& Wolf, G. (1989). Out of Scandinavia: Alternative Approaches to Software Design and System Development. *Human-Computer Interaction,* **4,** 253-350.

Gupta., P. K., Lavoie, R. C, & Raddiffe, R- R. (1984). An Industry Evaluation of SPEEDUP. *Paper Presented at the AIChE Annual Meeting.* San Francisco, CA.

Hayes, John R. (1981) *The Complete Problem Solver.* Philadelphia: The Franklin Institute Press.

Jordan, Brigitte. (1983). *Birth in Four Cultures.* Montreal: Eden Press.

Kendrick, D. & Meeraus, A. (1985). GAMS, an Introductory Technical Report. *Technical Report, Development and Research Department at the World Bank.* Washington, DC.

Ong, W. J. (1982). *Orality and Literacy: The Technologizing of the Word.* NY, NY: Methuen.

Penzias, A. (1989). *Ideas and Information: Managing in a High-Tech World.* New York: Simon & Schuster.

Perkins, J. D. (1984). Equation-Oriented Flowsheeting. In A. W. Westerberg & H. H. Chien (Eds) *Proceedings of the Second International Conference on Foundations of Computer-aided Process Design.* 309-367. United States: CACHE Publications.

Piela, P., Epperly, T., Westerberg, K., & Westerberg, A. (1991). ASCEND: An Object-Oriented Computer Environment for Modeling and Analysis. Part 1—The Modeling Language. *Computers and Chemical Engineering,* 15 (1), 53-72.

Rosenbrock, H, (1989). The Background to the Project. *Designing Human-centered Technology; A Cross-disciplinary Project in Computer-aided Manufacturing.* H. Rosenbrock (Ed.). London: Springer-Verlag.

Shacham, M, Macchietto, S., Stutzman, L. F., & Babcock, P. (1982). **REVIEW—**Equation-Oriented Approach to Process Flowsheeting. ***Computers and Chemical Engineering, 6 (2), 79-95.***

Stanford University. Department of Operations Research, Systems Optimization Laboratory. (1977). MINOS system manual. (Technical Report Collection, SUORSOL-7731) Stanford, California.

Suchman, L. A. (1987). ***Plans and Situated Actions: The Problem of Human-*** *Machine Communication.* Cambridge, England: Cambridge University **Press.**

Thoresen, K. (1990). Experiences with Participatory Design. *PDC90:* ***Participatory Design Conference Proceedings.*** **A. Namioka & D.** Schuler (Eds.). Seattle, WA: 34-35.

Westerberg, A. (1990). ASCEND: A System for the Rapid Building of **Equational Models.** ***CU, The Magazine of the Carnegie Institute of*** *Technology,* 9 (2), 23-26.

Westerberg, A. W. & Benjamin, D. R. (1985). Thoughts on a Future Equation-Oriented Flowsheeting System. *Computers and Chemical* ***Engineering, 9*** **(5), 517-526.**

Westerberg, A. W., Hutchinson, H. P., Motard, R. L., & Winter, P. (1979). *Process Flowsheeting.* Cambridge, England: Cambridge University Press.

Winograd, T. & Flores, F. (1986). ***Understanding Computers and Cognition: A New Foundation for Design.*** **Norwood, NJ: Ablex.**