OBSTACLES TO PARTICIPATORY DESIGN IN LARGE PRODUCT DEVELOPMENT ORGANIZATIONS

Jonathan Grudin

Computer Science Department, Aarhus University
Bygning 540 Ny Munkegade 116
8000 Aarhus C Denmark
jgrudin@daimi.dk

ABSTRACT

Development of an "off-the-shelf" product typically starts with a product idea and only limited knowledge of the eventual users. Since the functionality is partially predefined, the most natural focus for participatory design in product development companies is the human-computer interface. However, these organizations contain inherent obstacles to involving existing or potential users even in interface design and development. Today's large product development companies were formed before the human-computer interface attained its present prominence: their organizational structures and development processes have evolved with minimal consideration of the particular needs of interface development. This paper outlines the problems in achieving and benefiting from user involvement in design that stem from typical divisions of responsibility and development processes. Overcoming such organizational constraints may ultimately require organizational change; in the meantime, those working within the organization must be aware of the problems and constantly seek constructive paths around them.

PRODUCT DEVELOPMENT ORGANIZATIONS

This paper focuses on large companies that develop software systems and applications for wide distribution. A direct sales force, independent sales organizations, or value-added resellers may stand between development and the customer. This paper does not address internal development groups within a large organization, whose software is not developed as a product to be marketed externally. Also not considered are development groups formed to work on specific contracts. These other types of development groups have different advantages and disadvantages in obtaining user participation in design. Of course, a company may straddle categories: a product development company may bid on government contracts, an internal development group may decide to market a system built initially for internal use, and so forth. In addition, small product development companies may not find the problems associated with division of labor and management practices described here, while companies of moderate size may experience some of them but not others.

Thus, the obstacles to user involvement described below affect a fraction of systems developers -- perhaps only a small fraction. However, it is a significant fraction, especially in the United States. These product development companies have hired and trained large numbers of user interface specialists, recruiting heavily from leading research universities. They have recently been prominent in promoting the importance of usability and "look and feel." Their user interface specialists have dominated the conferences and journals in the field of human-computer interaction, as reflected in the strong presence of IBM, Digital, Apple, Hewlett-Packard, Xerox, and other large product development companies on the program and in the audience at recent CHI Conferences, for example.

These companies grew up primarily in the 1960s and 1970s. At that time, profits came from selling or leasing hardware; software functionality was secondary, and the human-computer interface received little attention. Since then, software has come to rival hardware in importance -- many of the successful new product development companies of the 1980's were primarily software companies. The focus has been on functionality and price, not the interface, but the

success of the Macintosh in the late 1980s is changing that. The interface has become important, and its importance will grow (Friedman, 1989; Grudin, 1990).

Unfortunately for those whose concern lies with the interface, these companies organized their business operations and development procedures when hardware and software functionality were the only important considerations. It is therefore not surprising that existing organizational structures and processes do not facilitate interface development. In fact, they often systematically *obstruct* the design and development of good interfaces (Grudin, 1986). One way they do so is by blocking user involvement in the process.

PARTICIPATORY DESIGN AND INTERFACE DEVELOPMENT IN PRODUCT DEVELOPMENT ORGANIZATIONS

Drawing a line between software functionality and its "user interface" is notoriously difficult. Nor is it necessarily a significant distinction -- since both interface and functionality determine how computer systems support work, participatory design would logically span them. In practice, however, a project to develop or upgrade a system begins with general preconceptions about the functionality to be provided. This is particularly true in a product development environment, by its very nature: a project begins with a product idea, defined in terms of hardware and software function. Of course, product ideas do not arise in a vacuum; individuals within a company develop varying degrees of intuition and domain expertise, in part through direct or indirect contact with users of existing products. While one can argue that better product ideas might emerge from a participatory design exercise begun without preconceptions, product development companies often find themselves with existing products in clear need of improvement as well as a wealth of new product proposals to select among.

Thus, user participation in product development environments typically occurs, if it occurs at all, in defining the human-computer interface (assumed here to include low-level functionality). The interface builders are most likely to encourage user involvement, although usually at a level far short of "full participation" in design. The juxtaposition of the Participatory Design Conference and CHI'90 reflects this connection. This paper outlines the obstacles to user participation that are encountered by interface designers, who are the natural advocates for greater user involvement in product development companies. Participatory design that focused on deeper aspects of product direction would encounter many of the same problems, depending on where in the organization it was undertaken.

ORGANIZATIONAL STRUCTURE AND DEVELOPMENT PROCESSES IN PRODUCT DEVELOPMENT ORGANIZATIONS

The division of responsibility within product development companies separates software developers from the users (or their "representatives") of the companies' products. Points of contact are typically in other organizational divisions: sales, marketing, training, field service, customer support, and perhaps upper management. But the people assigned these tasks are not primarily concerned with the interface, their relevant knowledge is not systematically organized, and they are often located far from the development groups. In sum, they probably have a limited sense of what information would be useful or to whom it might be forwarded.

Many standard software development procedures and techniques used in large projects were developed in the context of government contracts. On such projects, which are often sought by primarily product development companies, product definition precedes the designation of the development group. User involvement following the earliest design phase is usually difficult and may be explicitly forbidden. In any case, standard approaches to development were developed with minimal consideration of interface requirements. Also working against interface optimization is the pressure for a product development company to respond quickly to competition by releasing frequent enhancements, which inhibits scheduling adequate time for user participation.

The next several sections describe in more detail specific obstacles to user involvement that result from these structural and procedural characteristics of organizations, as well as indicating some positive consequences of these characteristics -- that is, their reasons for existing. The remainder of the paper describes approaches to removing or working around these obstacles.

CHALLENGES TO IDENTIFYING APPROPRIATE USER-PARTICIPANTS[1]

Obstacles to identifying appropriate users stem from the nature of developing products intended to appeal to a broad range of people, many of whom are not identified in advance. Further obstacles are found in the division of responsibilities within organizations that are set up to develop and market such products. User interface specialists rarely have "the big picture." They typically work with a development team assigned to a single application or even to part of an application. Not even the project manager has a perspective that encompasses the application mix that customers are expected to use, the practices and preferences of the installed customer base, and strategic information about the intended market for a product (e.g., banking or insurance industry; private doctors or those in an HMO or group practice). This broad perspective may be found in Marketing or Sales divisions, but these are often geographically and organizationally distant from the development groups. The projected market -- the identity of the future users --may be closely guarded by high-level management due to its competitive importance.

In large companies, marketing and sales representatives are almost themselves "users" of products emerging from development. They typically consider themselves internal advocates for the customers, who may not actually be "end-users." One or more marketing representative may track a project through written communication or by attending meetings. People in an International Marketing group may review designs to insure that they can be easily translated. But mutual respect between marketing and development is often low (e.g. Kapor, 1987; Poltrock, 1989a), so even this indirect connection to product users is not a strong one.

Another complication in identifying user-participants is that a system is often modified substantially after the development company ships it but before the users see it. This may be done by a software group within the customer's organization, by a "value-added reseller," or by another third party that tailors the product for specific markets. These developers not only are in a real sense "users" of the product, but may be the most important potential external participants in design. It may be *their* job to involve actual end-users. In any case, the initial development team is well-advised to discover which aspects of their design are likely to be "passed on" to users. Third-party intermediaries can represent an opportunity, but may also make it more difficult to select "representative end-users" for participation in design.

CHALLENGES TO OBTAINING ACCESS TO USER-PARTICIPANTS

Once potential user-participants are identified, the next challenge is to make contact with them. Obstacles here may be found within either the users' organization or the development organization, or both.

Perhaps the potential user is *within* the development company. This is convenient, but it is a dangerous special case to rely on. The company is not in business to build products for itself. Involving internal users brings with it a host of unique advantages and disadvantages. On the other hand, if the prospective user-participant works for a potential customer, that company may see little benefit in giving the employee time to work with an outside design group. What's more, contacts with customers are often with managers or information system specialists, rather than with the computer users themselves. Getting past these people may not be easy, as their job is precisely to represent the users.

Within the product development company, *protecting (or isolating) developers from customers is traditionally an important priority*. The company cannot afford to let well-intentioned developers spend all of their time customizing products for individual users -- the priority is on developing generic improvements to benefit scores or hundreds of users. Savvy customers are well aware of

---

[1] ***A note on terminology.*** The term "users" focuses on a limited and increasingly unexceptional part of people's lives: their use of computers. The full term "computer users" is rarely used, although it has the virtue of avoiding a perspective in which the centrality of the computer is silently assumed. In this paper, "user-participants" identifies existing or potential computer users who are possible design team members. "Users" designates existing or potential users of computer systems, as the context mandates.

the value of having the phone number of a genial developer. Of course, barriers erected to keep users from contacting developers also prevent developers from easily connecting with users. The relationships and channels aren't there.

Another problem is that the development company's sales representative may be reluctant to let developers meet with customers. The developer, coming from a different culture, might offend or alarm the customer, or create dissatisfaction with currently available products by describing developments in progress. Similarly, Marketing may consider itself to be the knowledgeable conduit into the development organization for information about customer needs, and may fear the results of random contacts between developers and users. In at least one company, developers, including Human Factors Engineers, were strongly discouraged from attending the company's annual User's Group meeting. The meeting was organized by the Marketing Department, who saw it as a show staged strictly for the customers.

CHALLENGES TO BENEFITING FROM USER CONTACT

Given the typically uncertain identity of future users, and the wide range of possibilities that may exist, assessing experiences one has with a small number of possible users can be difficult. The problems encountered in the often more focused settings in which participatory design has been tried are magnified. The Scylla of over-generalizing from a limited number of contacts is accompanied by the Charybdis of bogging down when users disagree, which is more likely to happen with a diverse set of user-participants. Finally, if user participation succeeds in producing design recommendations, the work has just begun. Design recommendations, whatever their source, must be steered through a software development process that is typically fraught with obstacles to interface optimization. One outcome of participatory design may be to increase the odds of successfully navigating this course, but the journey is never an easy one, for reasons described below.

CHALLENGES TO OBTAINING FEEDBACK FROM USERS

Feedback from users may be collected, often quite haphazardly, either informally or from bug reports and design change requests. These typically focus on hardware and high-level software functionality, rather than on interface features, because the former are still of primary importance in the marketplace. Unfortunately, even the little information that *is* collected rarely gets back to developers. As mentioned above, the organization may consciously shield developers from external contacts. In addition, field service or software support groups typically maintain products and work with customers on specific problems, while the original product developers move on to develop new releases or product replacements, are reassigned to altogether different projects, or even leave the company, in this industry marked by constant change.

The extent of feedback may vary with the pattern of marketing and product use. A company such as Apple, historically relying on purchase decisions initiated by the actual users rather than by management or information systems specialists, may benefit from having a particularly vocal user population. In general, though, the lack of user feedback may be the greatest hindrance to good interface design in product development organizations and among the least recognized problems with standard software development processes. It is true that system developers cannot spend all of their time fielding requests from customers, but their overall lack of experience with feedback is an obstacle both to improving specific products and to building developer sensitivity to the potential value of user participation in design. Developers rarely become aware of the users' pain.

The neglect of on-line help systems is a good example of how division of labor impedes interface design. Developers are not notoriously sympathetic to less experienced users to begin with, so encouragement to provide on-line help may be necessary. A good help system might save the company a substantial amount of money in customer "hand-holding," service calls, printed documentation, and so forth. But the principal beneficiaries would be in Customer Service. The savings would be in their budget, while the effort and expense would come from Development. As a result, help systems typically have very low development priority.

This point deserves emphasis. Engineers are engaged in a continuous process of compromise, of trading off among desirable alternatives. They would give more weight to interface improvements if they were more aware of the far-reaching, lasting consequences of accepting an inferior design. Consider some typical tradeoffs: "This implementation will save 10K bytes but be a little less modular." "This design will run a little faster but take a month longer to complete." "This chip provides two more slots but adds $500 to the sales price." Each requires a decision. Once the decision is made, the price in development time, memory size, or chip expense is paid and the matter is left behind. In this environment of tradeoffs, the interface is just one additional consideration. "This interface would be nicer, but take two months longer to design." Because the developer does not get feedback from users, once this decision is made, it too can be forgotten. The developer remains unaware that the decision, say, adversely affects thousands of users daily for the life of the product. The interface *really is* special, but developers don't recognize that -- once it is built and shipped, they are on to the next job, and other people (including users) must do the sweeping up.

WHERE IS THE DESIGN TEAM?

It would help users participate in design if there were a design or development group for them to join. But the "user interface," broadly defined, is not often the province of one easily-identified team. The hardware is designed by one group, the software by another, the documentation by a third, and the training by a fourth. Representatives from other groups may have continual or periodic involvement -- reviewing design specifications, for example. A product manager with little direct authority over developers may coordinate scheduling. Individuals from several different marketing groups, such as competitive or strategic analysis[2] and international marketing, may occasionally contribute. Members of support groups such as human factors or performance analysis may participate. Several levels of management may monitor the process and comment at different stages. In concert, these people contribute to defining a computer user's experience with the system or application, yet communication among them may be surprisingly sparse. With whom is a user to participate?

The timing of involvement in design and development is a matter of controversy. Typically, members of support groups such as human factors and technical writing are brought into projects later than they feel they should be (Grudin and Poltrock, 1989). This is telling, because these professionals have interface responsibilities and are the most likely to advocate greater user participation. Yet they themselves are unable to participate as fully as they wish in the software design process. With this lack of corporate awareness, how likely is it that management will recognize the value of placing *users* on development teams? In any case, (unplanned as well planned) turnover in personnel at the core of a development project is common and is a further obstacle to sustained user participation in design.

THE SOFTWARE DEVELOPMENT PROCESS

When today's software methodologies were being developed in the 1960s and 1970s, interface requirements were not important. Input and output channels were limited and computing resources were too expensive to devote much to the interface. Furthermore, most computer users were engineers with an understanding of the system. The interface that emerged during development and debugging was often adequate or even appropriate for the very similar "user environment." In general, the interface was ignored or received minor adjustments near the end of development.

Over the past ten to fifteen years, pressure has mounted to abandon "engineering interfaces" and develop interfaces for an increasingly diverse user population. This follows a pattern seen in other maturing technologies (Gentner and Grudin, 1990). Not surprisingly, software development methodologies did not anticipate this change. New approaches to development are emerging in response to the challenge (e.g., Boehm, 1988; Perlman, 1989), but have yet to be proven or

---

[2] Competitive analysis may seem to be a logical ally of a development organization. However, in practice their efforts may be directed far more toward the effective marketing of existing products against competition than toward the planning of future products.

widely adopted. Natural inertia is perhaps abetted by youth; software developers emerging from comparatively homogeneous academic environments may lack the experience with other workplaces and the people in them that would help them appreciate the diversity among computer users.

One aspect of this inertia is the persistence of the belief that the interface can be handled by tidying up at the end of development. As noted above, late involvement in development is a chronic complaint of human factors engineers and documentation writers. Once the software code is frozen, documentation moves into the critical path to product release. Because it is largely the interface to the software that is being documented, the interface is also frozen -- at the very moment that a fully functioning system is finally available for participant-users to try!

Perhaps the most widely accepted principles for designing usable systems are those outlined by Gould and Lewis (1985). In addition to an early and continual focus on users (albeit not as full participants in design), they specify prototyping and iterative design. The latter go hand in hand -- there is little point to prototyping if the design cannot be changed. But the high visibility of the interface works against iterative design in three ways: a) the interface is naturally grouped with aspects of the product that must be "signed off" on early in development; b) support groups, such as those producing documentation, training, and marketing, cannot avoid strong dependencies on the interface; c) iteration that takes place is evident, which can create uneasiness in an environment where early design is stressed.

The emphasis on careful early design and review makes sense for software, with its relatively predictable development course. But it works less well for the interface, where it is uncertainty that motivates iterative and participatory design in the first place. While user participation might in theory occur arbitrarily early, in practice approval is needed prior to extensive exploration. As the interface grows in importance, the desire to see it alongside the proposed functionality in the preliminary design will only grow. And once management has "signed off" on a design, changes typically require approval. Poltrock (1989b) observed the unique problems that high visibility and dependencies create for the interface development process. One developer summed it up: "I think one of the biggest problems with user interface design is that if you do start iterating, it's obvious to people that you're iterating. Then people say, 'How is this going to end up.' They start to get worried as to whether you're actually going to deliver anything, and they get worried about the amount of work it's creating for them. And people like (those doing) documentation are screwed up by iterations. They can't write the books. Whereas software, you can iterate like mad underneath, and nobody will know the difference."

The key here is the distinction between interface development and other software development. Solutions to these problems can be found -- *will* be found -- but because the problems are new and unique to interface development, the solutions will require procedural change. And an innovative process proposal is unlikely to leave management as comfortable as a detailed product design specification.

THE ROUTINIZATION OF DEVELOPMENT[3]

As competition and the pace of change increase in the electronics industry, product development companies feel growing pressure to turn out new products or enhancements in a reliable, predictable fashion. This may be a legitimate concern, as reflected in this analysis: "Ashton-Tate's decline began with what is becoming a well-worn story in the industry: failure to upgrade a market-leading product. Dbase III Plus went for almost three years before being upgraded, while competitor's products were upgraded as often as twice in that time," (Mace, 1990). A similar pattern is found in other maturing markets, from automobiles to stereo systems. The result is pressure for a predictable and controllable software development process: for routinization of

---

[3] Friedman (1989) suggests that a trend toward routinization and deskilling of programming has been exaggerated. This may be true, but his focus is on the internal software development centers of large corporations engaged in the support of business operations, not product development. Some of the competitive and marketing pressures described here as promoting direct control of development are less evident in such environments (see Aarhus, 1990).

development. Parker (1990) describes a perceived solution to the problem described in the previous quotation: "Lyons (an Ashton-Tate executive) responds that he can keep customers by providing predictable if not always exciting upgrades. 'Customers don't want to be embarrassed; they want their investment to be protected. If you are coming out with regular releases, even if they skip a release because a particular feature is missing, they will stay (with the product) because the cost of change is large.'"

This perceived need for controlled development creates difficulties for design elements or approaches that have uncertain duration or outcome. Interface design in general has a relatively high level of uncertainty, and participatory design is likely to increase the time needed for development while also introducing the possibility of changing the direction of development. This is the intent, of course -- to produce a better design -- but it nevertheless works against these powerful pressures within the organization.

ADVANTAGES THAT LARGE PRODUCT DEVELOPMENT ORGANIZATIONS PROVIDE FOR PARTICIPATORY DESIGN

The picture is a little grim, but grounds for optimism exist. Foremost is that these companies are motivated to distinguish their products and to increase their acceptance in a competitive marketplace, some of which is characterized by increasingly discretionary application use. In addition, large product development organizations often have considerable resources to devote to usability -- the cost of development is highly amortized. As noted before, these companies are already major employers of human factors engineers and are heavily represented in conferences focused on interface issues. There is also a positive potential in these companies' relatively frequent upgrades to or replacements of existing products: developers can more easily break out of "single-cycle" development. Evaluation of the use of existing products can feed into the design of a later version, and good ideas that arrive too late for use on a specific development project can be retained for use later. Another advantage is the large supply of potential users for most products. Also, the successful completion and evaluation of the system doesn't ride so precariously on the wide range of tangential factors that operate in any one given site.

OVERCOMING THE OBSTACLES

To someone working within a large product development organization these obstacles sometimes seem insurmountable. But as just noted, the company has a powerful incentive to improve its product interfaces. As software products mature, ease of learning and use becomes a more important marketing edge. Adding a new bell or whistle may not help much if the already available functionality is underutilized. In addition, declining hardware and software costs permit more resources to be directed to the interface. These forces have already pushed large product development companies into the forefront of human factors research and development in the United States. In the long term, organizational structures and development processes may evolve, institutionalizing solutions to the problems described here. This is likely because the forces in development companies that work systematically *against* user involvement undermine product optimization and success. The directions that these companies will take are not obvious. As the focus of development shifts from generic products to systems and applications that meet the needs of different specific markets, companies may have to choose between working closely with independent developers, working with value-added resellers who in turn work with "end-users," or working with the diverse computer users themselves. Each alternative will benefit from "user participation" -- the difference lying in the identity of the participants.

In the meantime, practitioners may find a growing climate for limited experimentation. But even to *begin* working effectively requires a clear awareness of the obstacles, an understanding of why they are there, and a tolerant recognition that their source is in institutional constructs, not in unsympathetic individuals.

WORKING STRATEGICALLY

There are many possible approaches to understanding computer users, not all of which require direct user involvement. The problem is clear: As computer users become less technical and more diverse, there is more need for developers to obtain information about them and their work

environments. Participatory design is a direct solution to this problem. Where its drawbacks are not serious, it may be the most effective. A limited form is seen when people from user organizations are hired to work as "domain experts."[4] In the absence of participatory design, other approaches and channels may funnel information about computer users and their work environments to developers. Whether or not they are adequate for a given situation today or will remain adequate in the future, they must be analyzed and understood, because the introduction of participatory design will affect people engaged in roles that serve some of the same purposes. Of course there is the most common form of direct user involvement -- limited-duration studies by developers of potential users, or individuals presumed to be much like them. Indirect channels include: hiring knowledgeable consultants, who serve as "surrogate user representatives"; organizing "focus groups" of user representatives, who often end up being information system specialists rather than end-users; and obtaining information about customers and users through marketing, field service, management and other contacts, as described above.

While each of these approaches can be criticized, they may relieve some of the pressure for participatory design, and their advocates may resist change. A new approach -- participatory design -- may be better received on a project where the strongest case can be made that these existing alternatives are inadequate.

ACKNOWLEDGMENT

I thank Don Gentner for suggestions and Morten Kyng for useful comments on an earlier draft.

REFERENCES

Boehm, B., 1988. A Spiral Model of Software Development and Enhancement. *IEEE Computer, 21,* 5, 61-72.

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

Gentner, D.R. and Grudin, J. (1990). Why good engineers (sometimes) create bad interfaces. In *Proceedings CHI'90 Human Factors in Computing Systems,* (Seattle, April 1-4).

Gould, J.D. and Lewis, C. (1985). Designing for usability: Key principles and what designers think. *Communications of the ACM, 28,* 3, 300-311.

Grudin, J. (1986). Designing in the dark: Logics that compete with the user. In *Proceedings CHI'86 Human Factors in Computing Systems,* (Boston, April 13-17).

Grudin, J. (1990). The computer reaches out: The historical continuity of interface design. In *Proceedings CHI'90 Human Factors in Computing Systems,* (Seattle, April 1-4).

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

Kapor M. (1987). Interviewed in INC. Magazine, January, 1987.

Mace, S. (1990). Defending the Dbase turf. *InfoWorld, 12,* 2 (January 8), 43-46.

Parker, R. (1990). Bill Lyons' task: Incremental moves to consistency. *InfoWorld, 12,* 2 (January 8), 44.

Perlman, G., 1989. Design/evaluation methods for hypertext technology development. In *Proceedings Hypertext'89,* (Pittsburgh, Nov. 5-8), 61-81.

Poltrock, S.E. (1989a). Participant-observer studies of user interface design and development. MCC Technical Report ACT-HI-125-89.

Poltrock, S.E. (1989b). Innovation in user interface development: Obstacles and opportunities. In *Proceedings CHI'89 Human Factors in Computing Systems,* (Austin, April 30-May 4).

---

[4] The first time such a direct hire is involved in development, it is a form of "user participation." However, as time passes, the domain expert acquires characteristics of the development organization and becomes less representative of the simultaneously evolving user population.