Interactive Systems:

Bridging the Gaps Between Developers and Users

Jonathan Grudin, Aarhus University

Today, user needs are recognized to be important in designing interactive computer systems, but as recently as 1980, they received little emphasis. In most major systems development companies, the basic organizational structures and development processes were defined in an earlier era, when the dialogue between computer systems and computer users did not have to be considered.

This point is of more than historical interest; it raises some basic questions: Can these companies develop effective interactive systems? What changes would bring greater knowledge of users and their work environments into systems development? How long will it take to translate this new understanding into reliable, effective practice?

Until 15 years ago, most computer system users were engineers and programmers. Developers were designing systems for their own use or for other technically proficient users. They felt no need to seek user participation in design. Now, however, computer use has spread to workplaces that are very unlike engineering laboratories. Contact with system users is required, but determining how direct or extensive this contact need be and actually achieving it have been surprisingly difficult.

Three development contexts provide a framework for understanding interactive software development projects: competitively bid, commercial product, and in-house/custom development.

Toward a user focus. Early proponents of greater user involvement included both human factors specialists and systems developers. An IBM usability research group stressed "an early focus on users" in the 1970s, and an influential 1983 paper recommended that "typical users (e.g. bank tellers, as opposed to a 'group of expert' supervisors, industrial engineers, programmers) ... become part of the design team from the very outset when their perspectives can have the most influence, rather than using them to 'review,' 'sign off on,' 'agree' to the design before it is coded."[1] Perhaps coincidentally, this paper appeared when popular books such as *In Search of Excellence* were urging industry to cater to the customer. A user focus was further emphasized in *User Centered System Design*,[2] an influential 1986 collection of research papers. Also in the 1970s and 1980s, Europeans experimented with greater user involvement in systems design. In particular, the Scandinavian "participatory design" approach, in which users collaborate as full development team members, attracted attention (see Suchman's review, "Designing with the User").[3]

The challenge of designing better interactive systems has not gone unnoticed in software engineering. Boehm observes that the dominant waterfall development model "does not work well for many classes of software, particularly interactive end user applications."[4] His proposal, a "spiral model" of software development, incorporates user involvement, prototyping, and iterative design. Yourdon recently wrote, "The first, and by far the most important, player in the systems game is someone known to systems analysts as a user."[5]

The spiral model is not yet widely used, and as Boehm notes, it may be difficult to apply in some contexts. Similarly, Yourdon's observation has not been fully translated into practice. The software methods that are employed widely today were developed before interactive end user applications became important. They do not provide for an early and continual focus on users — quite the contrary. Traditional structured analysis relegates the task of establishing a "man-machine interface" to one subphase of system development.[6] Jackson System Development "excludes

0018-9162/91/0400-0059$01.00 © 1991 IEEE

the whole area of human engineering in such matters as dialog design .... It excludes procedures for system acceptance, installation, and cutover."[7] Because such methods do not specify user involvement in design, project plans do not anticipate it, and development organizations are not structured to facilitate it. In fact, organizational structures and processes often work against user participation.

Many development projects do not include direct user involvement. Such projects indirectly acquire knowledge about system users, finding ways to bridge the gap between developers and users. These indirect methods sometimes work, but they may not meet the rising expectations of computer buyers and users.

Terminology. The authors above employ the term "user" when referring to the people directly engaged with the system, sometimes called "end users." I follow their lead in equating "user" with "end user." Terms such as "customer" and "buyer" refer to other people involved in system acquisition and use. Similarly, "developers" refers to active members of a development project, a relatively narrow definition that excludes those in management and support roles, whose contributions are described separately. Of course, developers are also users of the tools and the development system.

The term "functionality" describes the high-level system or application functions or operations that relate to the work the software supports, while "interface" or "user interface" describes the lower level operations and features that define the way users interact with the system. From the users' perspective, the functionality and the interface are intertwined aspects of usability and utility, but in a given project, the terms can be defined at different times by different people.

Identifying developers and users

Figure 1 presents three paradigms for software project development based on when users and developers are identified. The left edge of each horizontal bar represents the point when many of the project's developers or eventual users are known. Of course, not every project matches one of these time courses precisely. Also, it can be difficult to pin down events to one moment in time.

This framework best applies to projects that begin with a decision to build, modify, or replace a system or application and include a planned completion or delivery date. The three development paradigms describe a large set of projects that have significantly influenced interactive systems development.

In contract development or software acquisition, the user organization is known from the outset, but the development organization is identified after a contract is awarded. The clearest cases involve competitive bidding; for example, a government agency prepares a design specification for a new computer system and awards the contract to the developer with the most responsive bid. Actual practice can involve ambiguity or complications. The user organization may have some idea of who will get the contract; the user population may change before the system is completed, perhaps due to system personnel requirements; and contracts may be awarded in stages. But the essence of contract development is that the users are identified before the developers.

In product development, also called commercial off-the-shelf or shrink-wrap software development, the developers are known from the outset, but the users often remain unknown until the product is marketed. Of course, all product development begins with some idea of the intended customers, whether existing or new. But uncertainty about the eventual user population is an important facet of product development, as the unexpected fates of many products, positive as well as negative, remind us. The IBM PC and the Apple Macintosh are successful systems whose markets were not initially foreseen, whereas the designers of countless failed products anticipated user populations that did not materialize.

Finally, in in-house development, also called internal or information systems development, both the users and the developers are known at the project outset. (For example, a bank's computer services division develops a system for the bank's platform managers.) The user population may evolve or be too large or too dispersed to deal with individually, but the degree of initial identification is very high. This also occurs in custom development, where a specific external developer is engaged from the start to produce a system for a specific customer.

Projects may not be pure expressions of one paradigm. When a contract is negotiated without bidding, the developer has greater early involvement and encounters fewer restrictions on user access, a situation that lies somewhere between competitively bid contract development and in-house/custom development. Another paradigm merge occurs when a development company acquires a contract for a single system with the idea of subsequently developing it into a product. Similarly, an in-house project acquires some characteristics of product development when the organization features a large, diverse, and geographically dispersed user population (for example, when a bank's data processing department develops a system for branches that vary in size and operation).

It is not surprising that conditions for user participation vary across these development contexts. The timing gap between user and developer involvement in a prod-

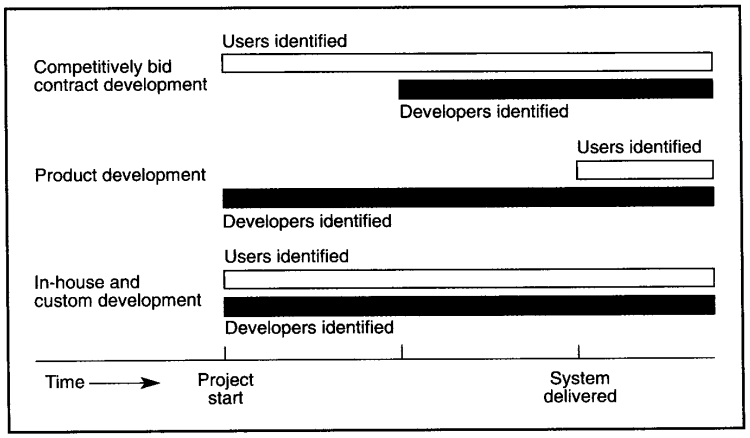

Figure 1. Project time lines with points of user and developer identification.

uct or competitively bid contract development project is an obstacle to collaboration. Many aspects of development practice have evolved to "communicate across time" — to bridge the gaps shown in Figure 1 — as well as to bridge the physical distances that often separate developers and users. These "bridges" or mediators include consultants, independent or third-party software developers and vendors, domain experts hired by development companies, internal market research or development groups, users and standards organizations, and flexible or multistage contracts.

Such mediation works better in some cases than in others. Keep in mind the chorus of recommendations that interactive systems developers establish and maintain continual contact with users, and the widening gulf between development and user environments. Intuition has become a less reliable guide to development, and the effectiveness of replacing it with indirect contact between developers and users must be continually reexamined.

Three development contexts

Each of the paradigms or contexts in Figure 1 has contributed to contemporary practices of developing interactive systems while maintaining its own history, literature, and even language. Contract development has been central to the evolution of software project management methods; product development has focused attention on computer interfaces to individual users; and in-house development has explored social and organizational aspects of system use. Many people work exclusively in one paradigm or another. Understanding the different perspectives can eliminate some of the resulting confusion and allow us to evaluate approaches that are based on unfamiliar research and development experiences.

Contract development and a focus on software methods. From the beginning of commercial computer development, government contracts have been a powerful force in the industry. The United States government has been and remains the largest computer user.[8] Government-initiated large-scale projects focused attention on software development methods.

Major contributions to the stage model of systems development were first described at a 1956 Office of Naval Research Symposium and a 1966 Air Force Exhibit.[4] The waterfall model of the software life cycle became the basis for most government software-acquisition standards. This model describes an unvarying sequence of phases in which feasibility is established, requirements are specified, and preliminary and detailed designs are drawn up prior to coding. Then, these phases are followed by testing, integration, implementation (or installation), operation, and maintenance. This provides minimal opportunity for prototype testing and iterative design, which form the basis of ongoing user involvement.

The reliance on specifications documents imposes a "wall" between users and developers.

The waterfall model restricts prototyping to "build-it-twice" development and restricts further iteration to feedback from adjacent phases. These limitations were recognized early by some software development managers and have been a source of criticism that has gathered force with the spread of interactive systems. But, the waterfall model is a natural fit to competitive contract development, where the user organization determines feasibility, prepares a requirements specification, and then issues one or more contracts for design, development, administration, and maintenance.

The waterfall model and its refinements were suited in other ways to the large, often government-contracted systems that dominated early systems development. The heavy emphasis on early design is more successful for relatively predictable, noninteractive systems, which have less uncertainty about requirements than systems that support substantial user interaction. In addition, the documentation and the distinct phases facilitate creation of an audit trail.

Strongly promoted by IBM, the waterfall model became the foundation for many structures and procedures found in most systems development. However, for today's interactive systems developer, the reliance on specifications documents imposes a "wall" between users and developers. This wall may not be impenetrable, but it clearly impedes user-based iterative design. The other development contexts are slowly freeing themselves from the problematic organizational structures and development practices that originated in contract development.

Product development and a focus on the user interface. As hardware costs fell and the market for computer systems broadened during the 1960s and 1970s, off-the-shelf product development grew in importance, especially in the United States. The discretionary nature of product acquisition means that systems must appeal to people, rather than meet written requirements specifications. However, product developers are buffered from end users by two intermediaries: the market and the customer.

With a large number of possible users, a product can do very well if only a fraction of the market finds it acceptable. In addition, especially in less mature markets, the product generally has to appeal to the customer — the person who makes the purchasing decision, often a manager or information systems specialist — not the end user. These customers share with the product development company the job of assuring system acceptance. The customer can hire systems administrators, provide training, establish internal development groups to tailor the product, supplement the product with third-party software, or even mandate use. Thus, just as the specification requirement in contract development is a wall between developers and users, the market and the customer in product development represent a thick hedge. Information about users' needs gets through, but it takes time and is muffled. Individual voices are not heard.

The delay in responding to users' needs allowed product development companies to focus on functionality, even if they met the usability needs of only a fraction of the potential users. Product development organizations could postpone attending to the human-computer dialogue while slowly finding indirect methods (consultants, market research, user groups, shows, trade press, etc.) to learn about major user needs. Now, as usability expectations increase in more mature software markets, product development companies are entering the phase in which users' needs replace software constraints as the dominant influence on development. We will see that the indirect, mediating organizational structures and processes that product developers

formed to bridge the gap to users often inhibit direct user-developer contact.

Muffling individual users' voices meant that in the 1980s, as computer use spread and usability became an issue, product developers focused on the generic aspects of the human-computer dialogue that are shared by almost all users. Motor control, perception, and lower cognitive processes — the "look and feel" of software — were explored by researchers and developers in the field of human-computer interaction. Since the hardware, the software environment, and product functionality are typically specified before a product development team is formed, only these aspects of the user interface design remain to be defined.

A community focused on interface issues coalesced in 1983 with the formation of the ACM Special Interest Group on Computer and Human Interaction (SIGCHI). Industry representation at the annual CHI conferences and in the related journals (such as *Human-Computer Interaction*, established in 1985) has come predominantly from product development companies.

Developers could safely ignore social or organizational concerns when the spread of multitasking systems and personal computing made single-user systems and applications very profitable. The design and use of word-processor or spreadsheet programs, for example, are relatively independent of the social context in which they are used. The profits enabled American product development companies to form research groups, recruit heavily from leading universities, and influence the direction of academic research.

This field, human-computer interaction, has had less involvement from those working in contract development, where usability is taking even longer to come into focus. In-house development also remains relatively uninvolved, due in large part to differing interests: Internal development must focus on the individual and group differences and social dynamics that product developers can ignore; these are central to the acceptance of a specific in-house or custom-built system. In addition, the narrow "user interface" focus is less useful to in-house developers, who are more likely to consider functionality and its interface together.

In-house and custom development and a focus on user participation. In-house or internal development, in which the developers and users work under the same corporate roof, was the original development context. The early developers were also the users of their own systems. In the United States today, the other contexts have achieved greater visibility through their concentration of resources and association with the major computer manufacturers. But in-house development is a major generator of software (and in much of Europe, it is the most visible). This paradigm affords an obvious potential advantage to user participation in design: The developers and the eventual users are known when the project is initiated.

There is growing competitive pressure for usability in the marketplace and greater focus on the human-computer dialogue in contracts.

Potential obstacles to success do exist. Projects for multiuser systems are more challenging than single-user applications. Also, along with any benefits received by adopting the specifications-based waterfall model, internal developers created the documentation "wall" between themselves and users. This wall has less purpose in this environment, and it comes down immediately when the system is completed.

Many system design challenges for end users were first experienced in in-house development, where success required that a predefined set of users accept the system. Successful contract development requires conformance to a written specification; successful product development does not depend on appealing to a specific individual or group. But an internal development project must be accepted by a set group of users.

By the late 1970s or early 1980s, user needs replaced software constraints as the dominant concern within internal systems development.[8] It is not surprising that this occurred as soon as interactive systems began replacing batch processing. (A similar transition is under way, a decade later, in product development. Its arrival has been delayed by the market and the customer, which stand between off-the-shelf product developers and their users.) Design approaches based on active user involvement gathered strength, notably in British and Scandinavian projects. Many were collaborations between researcher-developers and specific user organizations; they shared the in-house development characteristic of identifying developers and prospective users at the outset.

Cultural and political factors, including a strong trade union movement, are often credited for the Scandinavian interest in collaborative design. While these factors surely contribute, the dominance of the in-house and custom development paradigm in these countries presented precisely the right motivation and conditions for such experiments. Unlike the situation in the United States, research and development resources were not absorbed by large competitively bid government contracts or by product development, contexts in which user needs have been slower to come into focus and in which conditions for engaging users in development are less favorable.

Today, usability is becoming more important to product and contract development organizations. These organizations are still to some extent buffered from the end users by the size of the product market and by the reliance on contract documents. However, resistance to unfriendly systems is growing. There is growing competitive pressure for usability in the marketplace, particularly in mature application domains, and greater focus on the human-computer dialogue in contracts. The success of the Macintosh in the mid-to-late 1980s was a turning point — a good interface drove hardware and software sales.

Product and contract developers who foresaw the importance of usability encountered obstacles to involving users. Few successful cases of user participation in development have been reported. For this reason, some researchers and developers are turning to Europeans who have a track record in such collaborations. For example, at the 1988 ACM-sponsored Conference on Computer Supported Cooperative Work held in Portland, Oregon, six of the 30 presentations were based on the Scandinavian approach. (Recent books and articles on this approach are listed in "Further reading.") Thus, just as contract and product development have influenced interactive systems development, work originating in in-house development contributes techniques and approaches.

Product and contract developers can gain

insight from the European approaches, but must bear in mind that in-house development is a different context. Today, product development projects in particular have acquired the same motivation to involve users that in-house development projects had 10 years ago. However, they experience different conditions within which to engage users. Roadblocks to a strong focus on users include the timing of involvement shown in Figure 1, as well as the organizational structures and processes that were established before the importance of the interaction dialogue. Adapting to the new situation may ultimately require substantial organizational change. We can guide that change, and work effectively in the meantime, by identifying each paradigm's unique advantages and disadvantages for realizing successful user participation in design and by understanding the alternatives to direct user involvement and their limitations.

Factors influencing interactive systems development

Before exploring the opportunities and obstacles that each paradigm offers, let's consider several constraints on development projects that influence the conditions for user contact. One constraint, the time for involving development partners, was used to identify the three development paradigms: a single user organization for which there are many potential developers (contract development), a single development group with many potential users (product development), and a single development group with a single user organization (in-house development). Other factors include the size, charter, and structure of the development organization; the nature of the user population; the degree of design uncertainty; the presence or absence of other partners or contributors to the project; the nature of the commitments and agreements among the parties involved; societal conditions that the partners may be unable to control; and changes encountered over the life of the project.

The size of the development company or organization. This article generally focuses on projects in large organizations. A start-up or a small product development company may have fewer resources, less division of labor, fewer installed customer base concerns, and may succeed with far fewer sales than a larger company. These factors permit greater flexibility, more latitude for (inexpensive) innovation, and, particularly significant for user involvement, the possibility of focusing on a few potential customers (perhaps even finding a customer willing to cosponsor development). In these ways, a small product development company takes on some characteristics of in-house or custom development.

Product development projects in particular have acquired the same motivation to involve users that in-house development projects had 10 years ago.

At the other end of the size continuum, very large companies also present mixed appearances. A large telecommunications company, for example, blends substantial internal development with opportunities for product development provided by its large, identifiable market. Similarly, in a highly divisionalized company, one division may "sell" its product to another or even bid for an internal contract to deliver a system.

The charter of the company or organization. Organizational charters vary, and a company may work in more than one paradigm. For example, large product-development companies pursue government contracts for systems that can be built on or around their products. A separate Federal Systems Division may manage these projects, but influences often cross divisional boundaries. Paradigms also blur when a product development company that considers entry into a new market experiments by custom building a system for one customer to obtain domain expertise. The Scandinavian UTOPIA project (an acronym in the Scandinavian languages for "training, technology, and products from the quality of work perspective") did the reverse experiment: Methods from the in-house/custom development paradigm were adapted to a product development effort.[9] An organizational charter shifts gradually when a company develops a system under contract to one user organization, then decides to market it more broadly. Finally, several influential Scandinavian projects have involved development teams drawn from university research laboratories, small groups with a mixed agenda of research and development interests.

Organizational structures and procedures. Companies that do similar work do not necessarily divide responsibilities and meet obligations in the same way. While certain job descriptions and work procedures are widespread in the industry, companies of similar size and charter organize, distribute authority, and approach system development differently. Marketing divisons drive some companies, while engineering drives others. Some are hierarchical bureaucracies, others are strongly divisionalized into semiautonomous subunits, and some have experimented with almost ad hoc approaches. An in-house project in a large, divisionalized company may be managed through a contract competed for by internal development groups. Even where high-level approaches to organizational structure and process are similar, small variations in structure and process can greatly affect individual development projects. Within a single company, such variation results from internal reorganizations.

The nature of the system user population. The generic term "user" masks a tremendous diversity of computer users and contexts of use. This diversity will continue to increase — even if progress in hardware development stopped today, current technology would take decades to realize its potential. The number and heterogeneity of users is often a particular concern to in-house development. More generally, some users are captive audiences, who use systems acquired for them by others, while others are discretionary users. At the extreme end of discretionary use, those who pay for their own systems become involved through their personal economic stake. An Apple developer told me, "We don't have to ask our users for advice; they volunteer it."

The physical separation of developers from some or all users is often critical, as are barriers of class, culture, or language. The sensitivity of the users' work is another factor: A group designing a system for the CIA may have to accept a limit to their ability to "know the users."

The degree of design novelty or uncertainty. The uncertainty that arises in designing for a new market or in utilizing a new medium not only affects the motivation for working with users but also affects the conditions for doing so. Experienced users are more easily found in a mature application area, and mediators — consultants, trade publications, empirical research — are more reliable sources of information about users or technical alternatives.

Mediators: additional partners in the development project. Although I have primarily considered users and developers, projects generally involve other parties. These parties include other groups within the development and user organizations, as well as external consultants, subcontractors, value-added resellers, independent software vendors, third-party developers, product user organizations, trade unions, and standards organizations. Inanimate mediators include published guidelines, trade publications, and trade shows.

The roles of such mediators are sometimes central, sometimes incidental; they include informing developers of users' needs and informing users of technological opportunities. A critical uncertainty in interactive systems development is the adequacy of these channels for indirect collaboration in development.

Commitments and agreements among the groups involved. Commitments vary in formality and scope, ranging from informal understandings between individuals within one organization to binding contracts among companies. They also vary in content, focus, and flexibility. The content can be restricted to technical aspects of the system or can include such commitments to the users as organizational impact statements, installation, or training. Similarly, an agreement can be restricted to the system to be developed or can specify aspects of the development process, including techniques to facilitate user involvement, such as prototyping or scheduled reviews. Contracts vary in flexibility — even a formal contract might include a level-of-effort provision or specify times at which its terms can be reconsidered, permitting design changes based on user involvement.[4,10]

Societal conditions and change over time. Projects encounter dynamic influences that the development partners cannot directly control.[10] These influences include aspects of the labor market, economic considerations of supply and competition, available technology, formal standards, and legal restrictions on technology use or safeguard requirements. Finally, the factors described in this section are subject to change within the life of a project; it is a rare project that enjoys static conditions from start to finish.

Focusing on users: Opportunities, obstacles, and mediators

Each project has particular advantages and disadvantages for involving users and specific strategies to cope with the gaps between developers and prospective users. These are explored here at the more general level of the three development paradigms.

Contract development. User involvement faces the most formidable obstacles in this context, especially with fixed-cost competitively bid contracts.

Opportunities. Starting with a well-defined user population is an advantage in obtaining user involvement. Of course, the users themselves do not write the specifications, but the opportunity exists to enlist their cooperation. (However, when specifications address only system function, such user involvement stops short of contributing to the user interface.) Also, contract development projects are often relatively large and slow moving (in contrast with an application upgrade, for example), which could provide substantial resources and time for iterative development with user involvement. High-level management is committed to the success of the project, which is an important factor in system acceptance. Finally, within constraints described below, the user organization has the power of authoring the contract and thus can obtain greater user involvement either directly or indirectly (that is, contract provisions may require prototyping or periodic review; interface features, usability targets, training requirements, or an organizational impact statement may be specified).

Obstacles. Consider a composite case described by Gundry[11]: The user organization, a large government agency, prepares a requirements specification. This is followed by a request for proposals (RFP) to develop the system design. A contract is awarded. This work results in a design specification that is the basis for an RFP for system development, which leads to the development contract award.

In the initial requirements specification executed within the large bureaucracy, user involvement is not assured. If a task analysis is done at all, it may rely on a few interviews of users or supervisors. Although interaction can occur before the RFP is issued, communication between designers in the bidding organizations and anyone in the user organization is then sharply curtailed and monitored to prevent a bidder from acquiring an unfair advantage.

When the contract for design is awarded, the designers work to the specification; by avoiding contact with the user organization, they avoid influencing the subsequent bidding on the more lucrative development contract. Since the follow-on development contract can be awarded to another company, the designers do not necessarily know who the developers will be. And, when the same company does obtain both the design and development contracts, as often happens, the development team is usually distinct from the design team, which has moved on to other design proposals. In addition, companies minimize risk by preparing joint bids, which spreads development across two or more organizations.

Contact with users continues to be controlled during development. This is for security or geographic separation reasons or to avoid influencing later bidding (on system administration, for example, or on system maintenance, which is often the most lucrative contract of all).

The designers and developers have another reason to avoid contact with users. Contract compliance is based on conformance to the written specifications. Any deviation from the specification is to be avoided: Programmers have been known to code nonfunctioning procedures described in miswritten specifications to avoid jeopardizing compliance. The development organization might prefer to commit to a relatively comprehensible and static document rather than try to satisfy unpredictable users. Usability requirements in contracts for interactive systems can be as vague as "the system shall present information succinctly" or "the system shall be easy to operate efficiently."[11]

Some contracts require that prototypes be demonstrated to users yet preclude or discourage changes based on feedback from the demonstrations. This only serves to

alert users to the inadequacies of the system they will receive. When changes are possible, the developers may not learn much beyond "the system is unusable."

Mediators. Figure 2 shows several groups that serve as intermediaries, educating system users about contract developers and vice versa. Within a user organization, system analysts or engineers specify requirements, and contract specialists or monitors handle negotiations. Similar professionals in the development organizations write proposals and negotiate contracts, with developers not assigned to a project until contract award.

External consultants help the user organization by playing a surrogate developer role during requirements definition, providing the contracting organization with insight into feasible technologies. Although consultants are more likely to deal with managers or specialists, they may actively participate with users. Contractors working on one phase of the project are in a sense consulting on subsequent phases; for example, those who write the design specification provide guidance to the as-yet-unidentified developers. Similarly, developers who are barred from direct access to real users employ consultants familiar with the target environment to serve as surrogate users or hire domain experts away from a large customer to help staff the project.

User organizations address eventual shortcomings in the system through modifications by in-house or third-party developers. More broadly, contractors communicate their needs by working collectively with vendors to develop formal standards, adherence to which may be mandated in subsequent contracts. (A large enough customer organization, such as the United States Department of Defense, can by itself promote standards development and compliance.) Gundry[11] proposes that the contracting organization supplement the requirements specification with a "concepts of use" document: an extended description of the users, their work practices, and their working organization. That this static view of the users' environment is perceived by Gundry as a major step forward is a dramatic statement of how little contact between developers and users currently exists.

Government contracts impose further challenges to developing more usable systems. Cost-plus contracts reduce the incentive to reuse successful components; in fact, using a similar interface can suggest that the government is being charged twice for the same work. In addition, contractors who are also engaged in product development may be reluctant to place their best work in the public domain by including it in a government-owned system.

On the positive side, concurrent engineering and computer-aided acquisition and logistic support are new approaches that reduce development time by using centralized databases and multidisciplinary teams to reduce the emphasis on phased development. While designed for other purposes, the increased communication and coordination could promote continuous customer and user participation. Also, as noted above, the contract itself may be used to open up communication. Contractors may be required to describe planned human factors activities. Where legal barriers do not intervene, user involvement can be specified. Even a formal contract may address the uncertainties of dialogue design by providing points for renegotiation of contract terms based on prototype testing.[10] Experiments have been tried with design-to-cost contracts, reward-for-effort contracts, and "buy-downs" or "fly-offs": multistage contracts with a "competitive conceptual development phase" in which several developers build and test prototypes before the final development contract competition.

Product development. This context provides a strong incentive to increase usability, but user involvement is a challenge when the potential users are numerous yet faceless.

Opportunities. Because development costs are amortized over many sales, product development organizations typically have considerable resources and many potential users that could be drawn upon to improve usability. Competition in the marketplace provides a motive for doing so — the attention given to "look and feel" reflects the growing awareness of the importance of usability. Product development companies are major employers of human factors engineers, technical writers, and other user interface specialists. Continual

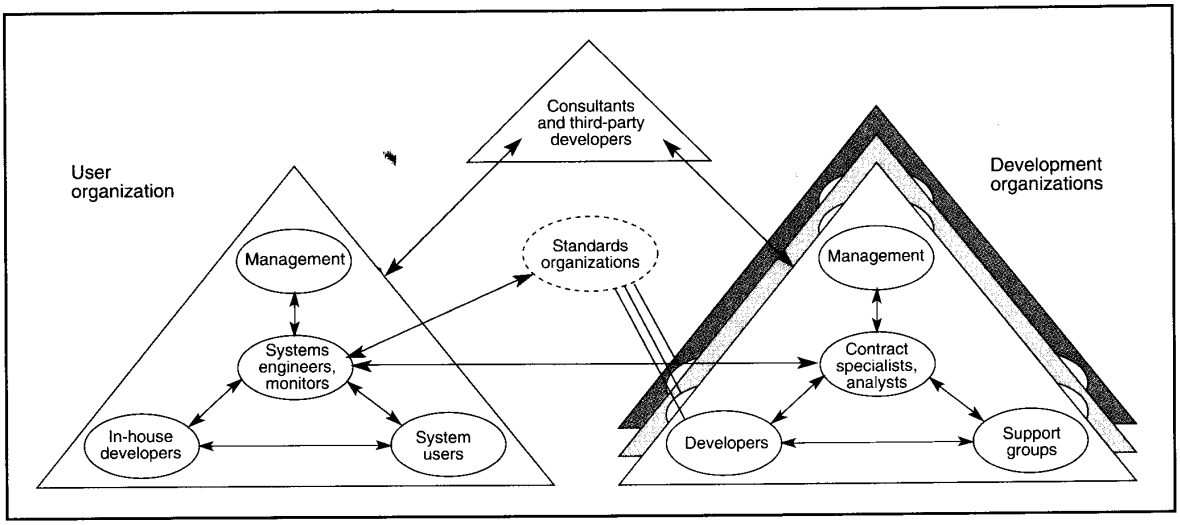

Figure 2. Contract development: Only the users are specified fully at the outset.

product upgrades or new releases free some developers from "single-cycle" development. They evaluate existing practice and feed it into the next version's design, and they may save good ideas that arrive too late for one project for use on another one. Finally, while these companies can succumb to inertia or conservative forces, they were founded on change and at some level recognize that survival requires openness to new ideas and approaches.

Obstacles. First, the development team members must commit to involving users. Developers who are isolated in large engineering laboratories may not empathize or sympathize with users who are inexperienced, nontechnical, or have different values and work styles. Even identifying the development team is difficult: Functionality is defined by management or marketing before the project team is formed, project membership changes over time, and the developers of different user interface components — such as software dialogue, documentation, and training — often communicate very little. In many companies, placing all aspects of a product's usability under the same management would conflict with deeply rooted aspects of organizational structure and process.

Also, the difficulty of identifying future users is a major obstacle to involving them. Strategic marketing decisions are carefully guarded by upper management to prevent the competition from using the information. Development teams often do not know which applications will be marketed as packages. In addition, before reaching an end user, many products are extensively modified or tailored by third-party developers or by the customer's in-house developers. These developers are users of the product, too.

Another obstacle is accessing potential users once they are identified. They work in different organizations, perhaps even different countries. Also, product development companies tend to shield developers from the time-consuming task of responding to individual user requests.

Figure 3 identifies several groups charged with knowing the needs of users: product management, marketing, and customer support in the development organization, and information systems specialists in the user organization. These go-betweens or mediators often discourage direct developer-user contact, either overtly or by partly alleviating the need for it.

These mediators are often ineffective conduits for usability information because the task is difficult and the usability concerns are new. It is particularly difficult in this context to obtain adequate time from potential users, who have little at stake. When contact does occur, product developers risk overgeneralizing from a few contacts, and they must contend with conflicting views of different users. (Careful selection of test sites happens primarily for major products and relatively late in development.) Finally, information that is obtained must be worked into the development process, which is fraught with competing interests and trade-offs. The process was designed to maximize the predictability and reliability of developing noninteractive systems; it does not work as well for developing the less predictable interactive systems.

Another obstacle is the pressure to produce new releases of existing products — the relatively short development cycle favors small enhancements over substantial innovation and does not provide enough time for users and developers to educate one another. Caution is also encouraged by the new fear of incurring an interface copyright infringement suit. Emphasis on rapid development engenders attempts to routinize the process through the early approval of specifications and schedules, which limits flexibility.

One approach that product developers pioneered to overcome their separation from users is providing customized or tailored systems, which is a useful step but not a panacea. Another approach has been to rely on mediators.

Mediators. As shown in Figure 3 and described above, sales and marketing departments, management, customer support, and other groups within large product-development companies mediate between users and developers. Other developers form part of a corporate memory that for usability issues can operate more effectively than formal records. External consultants serve as surrogate users, providing developers with detailed product critiques and information on market direction. Consulting, market research, and competitive

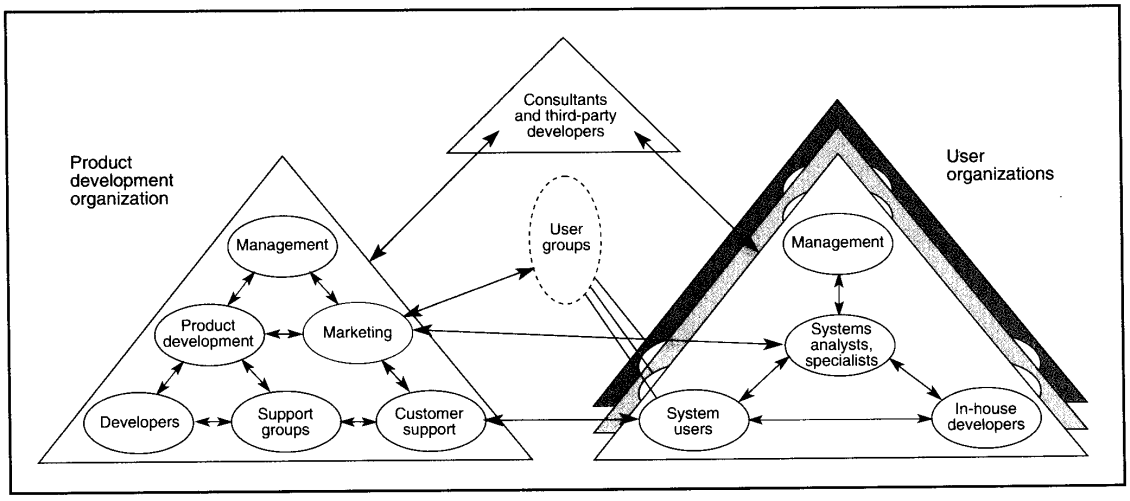

Figure 3. Product development: Only the development group is fully specified at the outset.

analysis, although undertaken to support the marketing of existing products, can also guide developers. Value-added resellers, third-party developers and independent software vendors, who adapt or match products to specific markets, stand between development and user organizations. Customer organizations engage consultants as surrogate developers to advise them about purchasing or installation. Buyers also rely on in-house developers to supplement or tailor products.

Most customers exert little direct influence on product development companies individually, but user groups representing many customers have more influence. However, this communication channel is less effective when meetings are attended by customers or buyers rather than users and by marketers rather than developers.

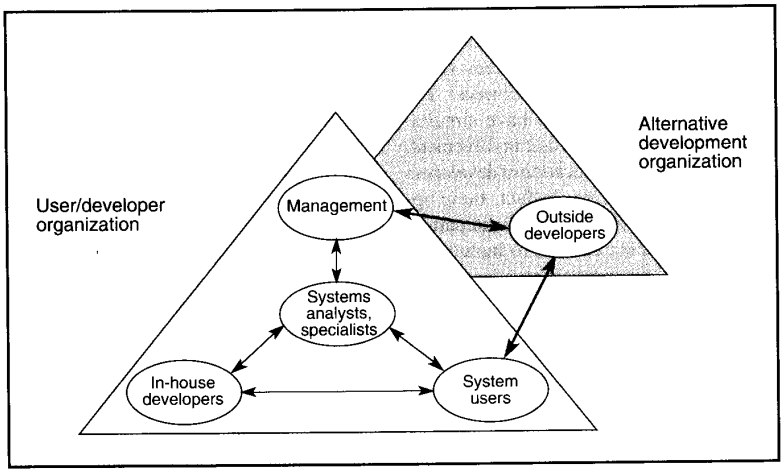

Figure 4. In-house and custom development: Developer and user populations are known at the outset.

In-house development. This development context appears to offer good prospects for collaboration among users and developers, but the challenges are substantial. One challenge is that internal development is often modeled on contract development, adopting methods that work against user involvement.

Opportunities. Collaboration is logically easier when users and developers are known from the beginning (see Figure 4). An early relationship can lead to parallel work on the functionality and the interface and is needed for prototyping and iterative design. User involvement can also help with system acceptance because participating users acquire an interest in the outcome; they may accept system features that they otherwise would have resisted.[8] In in-house development, communication between developers and users can be enhanced by the shared corporate culture (but see below). Also, the transitions across development phases are smoother; in particular, the developers are accessible during product introduction and use. A further advantage is that these projects have strong management support, an important element in obtaining system acceptance. Finally, they often enjoy a less pressured, more flexible development schedule — shifting focus from the product to the development process occurs most naturally in this environment.

Obstacles. In-house projects are generally systems or major applications designed to support organizations, which are more difficult than the single-user applications that dominate software product development. In addition, identifying future users is easier than ensuring collaboration. Large companies may impose geographic as well as organizational separations. Conflicts of interest exist within organizations: Management or information systems staff themselves may be the principal beneficiary of a project; if their interests conflict with the end users', user alienation is more likely than cooperation. Conflicts also occur among different worker groups — Ehn[9] describes jurisdictional disputes among typesetters, journalists, and administrative workers in one project. Friction between developers and users can result from differing codes of values, conduct, and dress, as well as disparities in age and salary.[8]

Selecting representative users is a challenge. Not all potential users have the time or inclination to participate fully, management may wish to participate or to control participant selection, and workers with greater knowledge of technology may be more interested but less representative. Potential participants' political roles in the organization must be balanced against their roles as system users. In addition, some techniques must be used very carefully in internal development — rapid prototyping can unduly raise users' expectations of the system's capabilities or state of completion. (This problem also appears as the rapid-prototyping technique is introduced in contract development; in product development, where disappointing one potential user is not as serious, the risk is that development management will be misled.)

Fewer resources may be available to in-house development projects than to contract or product-development projects. Building a single system to be used for many years provides less opportunity for evaluation, feedback, and catching missed opportunities the next time around than is found in product development. Prototyping and iterative design are not infinitely flexible, so the in-house development team must plan very carefully, considering, for example, the likely organizational impact. Such planning is beneficial, but given the inherent uncertainty of interactive systems development, the need to anticipate correctly is not an advantage.

Mediators. The spread of interactive systems in the early 1980s immediately confronted in-house developers with the needs of end users. The mediators who enabled the other developers to adjust slowly while delaying their entry into this phase are less available to in-house developers. Only upper management or human resources departments typically intervene between internal developers and users.

Friedman[8] explores five possible approaches to bridging the gap between users and developers in internal development: direct user participation in development based on traditional methods, end user computing (effectively, trying to eliminate developers by providing systems that can be tailored or customized by users), decentralizing the information center (to bring developers into closer contact with users), changing the systems development approach (through a process focus, notably an emphasis on prototyping and iterative

design), and relying more heavily on information systems specialists with both domain expertise and development skills. These are not mutually exclusive: The Scandinavian experiments have simultaneously employed direct user involvement, prototyping, and the education of developers about the domain area. In fact, these approaches are all forms of user participation, if participation is extended to include the education that precedes a particular project.

Several convergent forces push interactive systems development toward greater concern for users' needs. First, reaching untapped markets requires learning more about them. Computer versatility enhances the likelihood of finding a way to support a group of people whose concerns are properly understood. Second, users who can choose their tools will be influenced by usability in exercising this discretion. The focus on "look and feel," while reflecting aesthetic as well as utility judgments, arises from the availability of functionally equivalent product alternatives in mature markets. Third, understanding user needs is more central to software support of groups.

Until recently, mainframe systems focused on supporting organizations while most micro- and minicomputer applications focused on supporting individuals. Now, networking and multitasking systems make group-level support feasible. "Groupware" product developers face issues that were previously encountered mainly in in-house development as the focus of application development shifts from individual similarities (with the goal of appealing to a large group of similar people) to individual differences and social dynamics (to attract all members of groups, independent of background, role, or preferences). Here, application developers are at a disadvantage relative to system developers, since there is less corporate commitment to ensuring the acceptance of a groupware product than a large system. Finally, the cost of computation was a major obstacle to providing more usable systems. As processing time, memory, and maintenance costs continue to fall, more computational power is devoted to handling user interaction. Today, the price to obtain a better interface is not much more than the cost of building or buying one.

In summary, computer vendors' interests, computer users' interests, and economic factors work in concert to encourage the development of more usable software. Further progress will result from sharing experiences across the three development contexts. Conferences such as the 1988 Computer Supported Cooperative Work Conference and the 1990 Participatory Design Conference in Seattle bring together Scandinavian and American researchers and developers from in-house and product development.

Boehm[4] outlines recommendations for contract development drawn from in-house development experience. Techniques developed in one context can be modified and applied in others. The process focus and low-cost techniques that developed naturally in in-house development environments are being applied more broadly. Prototyping techniques first used in product development are being adapted to in-house projects. Further contact occurs as in-house projects come to rely more heavily on off-the-shelf software product components and as contracts specify more standard platform products.

The prevalence of specific development contexts may shift with societal changes. For example, the current integration of the European community will promote competitively bid contract development to ensure equal access to large projects. Europeans may profit from American experience and find ways to introduce approaches that achieve higher levels of user involvement.

Such communication requires effort. Many interests are not shared; there are specialized terminologies, journals, trade publications, conferences, and shows. Even word meanings vary across development contexts; for example, "implementation" is a synonym for "development" or "coding" to product developers, while it means "installation" or "adoption" elsewhere. "End user" or "operator" to an in-house developer is just "user" in a product environment.

Procedures take on different appearances: To a contract or in-house developer, a "task analysis" generally addresses repetitive activity carried out in an organizational context by an "operator" hired to perform it, whereas to a product developer, a "task analysis" is more often a cognitive analysis of individual behavior in a less structured, discretionary use situation.

Techniques are viewed differently: Internal developers (and many in the research community) link software "evaluation" to testing that occurs late in development, when only minor design changes are possible; in contrast, product design is more likely to begin with the evaluation of existing products or versions, so "evaluation" has a less negative ring to product developers.

Finally, confusion may result from failing to differentiate among the contexts, as illustrated by the debate over the routinization of software development. In large product development organizations, strong competition and the pressure for frequent releases create a strong desire to control the development process and to render it immune to the loss of any individual — conditions leading to "deskilling," where individual developers are given less responsibility and become more expendable. However, a tendency toward deskilling is not reported in internal development, where competitive pressures are lower and developers increase their value to their organization by acquiring domain knowledge.[8]

References to "the computer industry" disguise the multiplicity of computer industries that have evolved. They share a need to examine current practices and search for new ones to meet the challenge of developing usable, useful interactive software. Only by recognizing the range of existing conditions and their effects can we adapt the hard-earned lessons of one development context and apply them in others. The potential benefits are new techniques, processes, and organizational structures for development. ■

Acknowledgments

Morten Kyng and the Systemarbejde Group at Aarhus University provided the time and inspiration for developing this paper. Kaj Grønbæk, Susanne Bødker, and Liam Bannon participated in its iterative design. Andrew Friedman's book was further inspiration and an excellent resource on internal development. Bob Glushko, Henry Lahore, Jane Mosier, Scott Overmyer, Steve Poltrock, and Craig Will contributed to my understanding of contract development, although their full understanding is not reflected. Elizabeth Dykstra and Tom Erickson provided useful suggestions. Students at Aarhus University commented helpfully on earlier drafts. Also, anonymous *Computer* reviews were invaluable.

References

1. J.D. Gould and C.H. Lewis, "Designing for Usability — Key Principles and What Designers Think," *Proc. CHI83 Human Factors Computing Systems*, 1983, pp. 50-53.

2. D.A. Norman and S.W. Draper, *User Centered System Design: New Perspectives in Human-Computer Interaction*, Lawrence Erlbaum Associates, Hillsdale, N.J., 1986.

3. L. Suchman, "Designing with the User," ACM *Trans. Office Information Systems*, Vol. 6, 1988, pp. 173-183.

4. B. Boehm, "A Spiral Model of Software Development and Enhancement," *Computer*, Vol. 21, No. 5, May 1988, pp. 61-72.

5. E. Yourdon, *Modern Structured Analysis*, Yourdon Press, New York, 1989.

6. T. De Marco, *Structured Analysis and System Specification*, Yourdon Press, New York, 1978.

7. M. Jackson, *System Development*, Prentice Hall, Englewood Cliffs, N.J., 1983.

8. A.L. Friedman, *Computer Systems Development: History, Organization and Implementation*, John Wiley & Sons, Chichester, UK, 1989.

9. P. Ehn, *Work-Oriented Design of Computer Artifacts*, Arbetslivcentrum, Stockholm, 1988.

10. K. Grønbæk et al., "Cooperative System Design: Shifting from Product to Process Focus," to be published in *Participatory Design*, D. Schuler and A. Namioka, eds., Lawrence Erlbaum Associates, Hillsdale, N.J., 1991.

11. A.J. Gundry, "Humans, Computers, and Contracts," *People and Computers IV*, D.M. Jones and R. Winder, eds., Cambridge University Press, Cambridge, United Kingdom, 1988.

Further reading

In-house development: the new interest in Scandinavian approaches. As noted in the text and described in the cited work of Friedman and Suchman, interest is growing in the Scandinavian participatory design approach based on the in-house and custom development paradigm. Examples of recent or forthcoming publications:

Bjerknes, G., P. Ehn, and M. Kyng, eds., *Computers and Democracy — A Scandinavian Challenge*, Gower, Aldershot, United Kingdom, 1987.

Bødker, S., *Through the Interface: A Human Activity Approach to User Interface Design*, Lawrence Erlbaum Associates, Hillsdale, N.J., 1991.

Docherty, P., et al., eds., *System Design for Human Development and Productivity: Participation and Beyond*, North-Holland, Amsterdam, 1987.

Ehn, P., *Work-Oriented Design of Computer Artifacts*, Lawrence Erlbaum Associates, Hillsdale, N.J., 1989.

Floyd, C., et al., "Out of Scandinavia: Alternative Approaches to Software Design and System Development," *Human-Computer Interaction*, Vol. 4, No. 4, 1989, pp. 253-349.

Greenbaum, J. and M. Kyng, eds., *Design at Work: Cooperative Design of Computer Systems*, Lawrence Erlbaum Associates, Hillsdale, N.J., 1991.

Howard, R., "Utopia: Where Workers Craft New Technology," *Technology Review*, April 1985, pp. 43-49.

Schuler, D. and A. Namioka, eds., *Participatory Design*, Lawrence Erlbaum Associates, Hillsdale, N.J., to be published in 1991.

Product development: the focus on usability in the late 1980s. Among the many works published, articles explicitly considering the product development environment include:

Gould, J.D., "How to Design Usable Systems," M. Helander, ed., *Handbook of Human-Computer Interaction*, North-Holland, Amsterdam, 1988.

Grudin, J., "Systematic Sources of Suboptimal Interface Design in Large Product Development Organizations," to be published in *Human-Computer Interaction*, Vol. 6, No. 2, 1991.

Grudin, J., and S. Poltrock, "User Interface Design in Large Corporations: Coordination and Communication Across Disciplines," *Proc. CHI89 Human Factors Computing Systems*, Apr. 1989.

Poltrock, S.E., "Innovation in User Interface Development: Obstacles and Opportunities," *Proc. CHI89 Human Factors Computing Systems*, Apr. 1989.

Contract development: starting to address usability concerns. In addition to the Gundry paper cited in the text there are the following sources:

Overmyer, S.P., "The Impact of DoD-Std-2167A on Iterative Design Methodologies: Help or Hinder?" *ACM SIGSoft Software Engineering Notes*, Vol. 15, No. 5, pp. 50-59, 1990.

Smith, S.L., "User-System Interface Design in System Acquisition," Tech. Report ESD-TR-84-158 Air Force Electronic Systems Division, Hanscom Air Force Base, Mass., 1984.

Jonathan Grudin is a visiting associate professor of computer science at Aarhus University, Denmark. His interests include systems development in large organizations and the effects of systems on organizations.

He received a BA in mathematics/physics from Reed College, an MS in mathematics from Purdue University, and a PhD in psychology from University of California at San Diego.

After spending several years as a systems developer at Wang Laboratories, he worked in the MCC Human Interface Laboratory, studying the development process at MCC shareholder companies. Past publications range from *Byte* to the *Communications of the ACM*.

The author can be reached at Aarhus University, Computer Science Department, Ny Munkegade, Building 540, 8000 Aarhus C, Denmark; e-mail jgrudin@daimi.aau.dk.