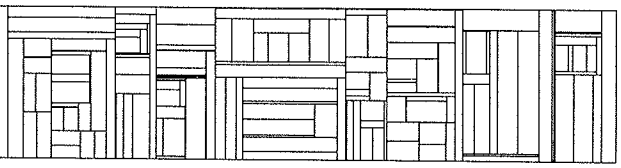
ISSN 0105-8517

The Development of Interactive Systems: Bridging the Gaps Between Developers and Users

Jonathan Grudin

DAIMI PB – 320
July 1990

COMPUTER SCIENCE DEPARTMENT
AARHUS UNIVERSITY
Ny Munkegade, Building 540
DK-8000 Aarhus C, Denmark



The Development of Interactive Systems: Bridging the Gaps Between Developers and Users

Jonathan Grudin

Computer Science Department, Aarhus University
Ny Munkegade Bygning 540 8000 Aarhus C

Abstract

A framework for interactive systems development projects is outlined. Three development paradigms are contrasted: competitively bid contract development, product development, and in-house or custom development. They vary as to when in a project the users and the developers can be identified -- a distinction that is particularly significant for systems with a human-computer interface component. The historical influence of each of these paradigms on interactive systems development is reviewed. Other factors affecting the flow of information between users and developers are outlined. For each development paradigm, I identify the opportunities for adopting an effective "user focus," the obstacles, and the mediators who contribute to bridging the gaps between developers and users.

Introduction: Early Calls for More Focus on Users

The importance of understanding the intended user when designing interactive computer systems is widely enough acknowledged in 1990 that it is easy to forget how little emphasis it received as recently as 1980. This is of more than historical interest, because system development companies will need time to translate this new understanding into reliable, effective practice. Most major companies defined their current organizational structures and development processes without having to consider the particular needs of designing the dialogue between computer systems and computer users. This raises some questions: Can these companies develop effective interactive systems? What changes are required to bring greater knowledge of users and their work environments into systems development?

Until 15 years ago, most computer system users were engineers and programmers. The system use environment was very similar to the system development environment. Developers felt no need to seek "user participation" in design -- developers themselves were good user representatives. Now, computer use has spread to workplaces very unlike engineering laboratories. Bridging the widening gulf between developer and user environments requires greater effort. Contact with system users is required, but determining how direct or extensive this contact need be is more difficult, and actually achieving such contact has often proven surprisingly difficult.

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 1970's and in 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," (Gould and Lewis, 1983). Perhaps coincidentally, this appeared when popular books such as *In Search of Excellence* (Peters and Waterman, 1982) were urging industry to cater to "the customer." A user focus was further emphasized in Norman and Draper's (1986) book *User Centered System Design.* Also in the 1970s and 1980s, European projects experimented with greater user participation in systems design (Bjerknes, Kyng, and Ehn, 1987). Perhaps because it demonstrates how the principle of user involvement might actually be applied, the Scandinavian approach in particular has recently drawn attention (see e.g., "Designing with the user," Suchman's (1988) review of Bjerknes et al., 1987).

The challenge of designing interactive systems has not gone unnoticed in software engineering. Boehm (1988) stated that the dominant waterfall model of development "does not work well for many classes of software, particularly interactive end-user applications." His proposal, a "spiral model" of software development, incorporates user involvement, prototyping, and iterative design, steps recommended by Gould and Lewis (1983).

However, the spiral model is not yet widely used, and Boehm notes that it may be difficult to apply in some contexts. Software methods employed more widely today, developed prior to the importance of "interactive end-user applications," do not provide for "early and continual focus on the users," Gould's (1988) central recommendation. Quite the contrary. De Marco's (1978) structured analysis approach relegates the task "establish man-machine interface" to one sub-phase of system development. Jackson (1983) states that "(Jackson System Development) excludes the whole area of human engineering in such matters as dialog design... it excludes procedures for system acceptance, installation, and cutover." Because such methods do not specify user involvement in design, project plans do not anticipate it. Development organizations are not structured to facilitate it. In fact, organizational structures and processes often work strongly against user participation. Many development projects never have a significant possibility for direct user involvement, as described in the next section. Such projects may acquire knowledge about system users indirectly, finding some way to bridge the gap between developers and users. These indirect methods sometimes work, but may not succeed in meeting the rising expectations of computer users.

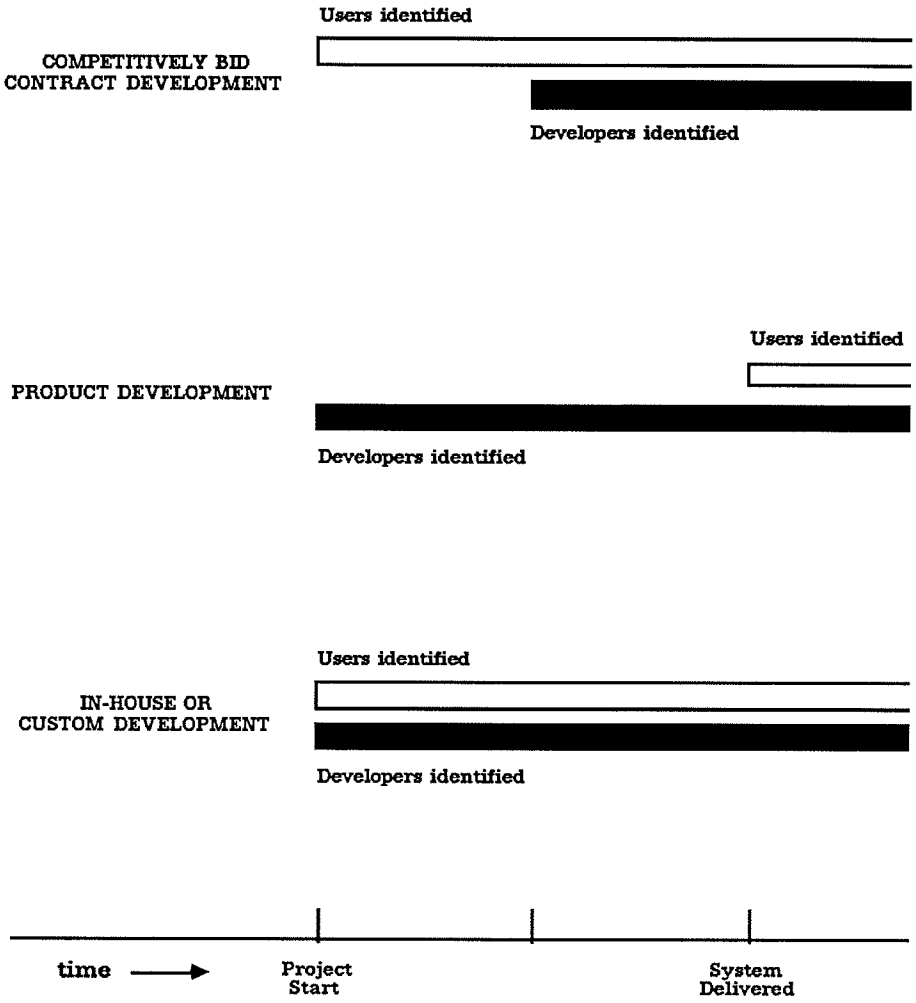


Figure 1. Project time lines with points of user and developer identification.

Projects Vary as to When Developers and Users Are Identified

Figure 1 presents three paradigms for software project development based on when users and developers can be identified. The left edge of each horizontal bar in Figure 1 represents the point at which a substantial subset of the project's developers or eventual users are known. Not every project matches one of these time courses precisely. It may be difficult to pin events down to one moment in time, for example. Projects that begin with a decision to build, modify, or replace a system or application and that include a planned completion or delivery date are most directly addressed here (although ongoing operation and maintenance occur, of course). However, each of the three paradigms describes a large set of projects which has had a major influence on interactive systems development.

In *competitively bid contract development,* the user organization is known from the outset, but the development organization is identified only after a contract is awarded. For example, a government agency prepares a design

specification for a new computer system and awards the contract to the developer with the lowest bid. Actual practice may involve some ambiguity or complication. The user organization may have some idea who will get the contract, the user population may change before the system is completed, and contracts may be awarded in stages. But the essence of a competitively bid contract development project is that the users are identified before the developers.

In an "off-the-shelf" *product development* effort, the developers are known from the outset, but the users are often not known until the product is marketed. This is again a simplification. Development team members may change over time, and more importantly, development of any product begins with *some* idea of its intended users, whether they are an existing customer base or a new market. 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 Apple Macintosh are successful systems whose market was not initially foreseen, and the designers of countless failed products anticipated user populations that did not materialize.

Finally, in *in-house* or *custom development,* both the eventual users and the developers are known at the outset of the project. For example, the computer services division of a bank develops a system for the bank's platform managers. Again, the user population may evolve or be too large or dispersed to be dealt with individually, but the degree of initial project member identification is very high. This also occurs when external developers are engaged from the start in developing a system for a specific customer.

Projects may not be pure expressions of one paradigm. When a contract is negotiated without bidding, the eventual developer may influence its terms and thus be involved in advance, a situation that may lie somewhere between competitively bid contract development and in-house/custom development. A different merging of paradigms occurs when a development company takes on a contract for a single system with the idea of subsequently developing it into a product. Similarly, an in-house project acquires characteristics of product development if the organization has a large, diverse, and geographically dispersed user population, as when the data processing department of a bank develops a system for branches that vary in size and manner of operation.

It should not be surprising that conditions for user participation vary across these development contexts. The separations in the timing of involvement in product and competitively bid contract development projects lead to potential or even probable obstacles to active collaboration of users and developers. Many aspects of development practice have evolved in effect to "communicate across time" -- to bridge the gaps shown in Figure 1 -- as well as to bridge the physical distances that may separate

developers and users, so that one group becomes better informed about the other. 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 multi-stage contracts. Such mediation works better in some cases than in others. But keep in mind the chorus of recommendations for early and continual collaboration of developers and users in building interactive systems, and keep in mind the widening gulf between development and user environments. Intuition is becoming a less reliable guide to development, and the effectiveness of such indirect measures requires continual reexamination.

The Influence of 3 Development Paradigms on Interactive Systems

Each of the three development paradigms in Figure 1 has contributed to contemporary practices in developing interactive systems. Understanding these influences may eliminate some confusion that exists and allow us to weigh the merits of approaches based on different research and development experiences.

Contract development and the focus on software methods.

From the beginning of commercial computer development, government contracts have been a powerful force in the industry. The U.S. government has been and remains the largest computer user (Friedman, 1989). Government-initiated large-scale projects introduced a level of complexity that focused attention on software development methods. Major contributions to the stage model of systems development appeared at a 1956 Office of Naval Research Symposium and a 1966 Air Force Exhibit (Boehm, 1981; 1988). The waterfall model of the software life cycle (Royce, 1970) became the basis for most government software acquisition standards. The waterfall model describes an unvarying sequence of stages in which feasibility is established, requirements are specified, and preliminary and detailed designs are drawn up prior to coding, followed by testing, integration, implementation (or installation), operation, and maintenance. This provides minimal opportunity for the prototyping and iterative design that are the basis of "early and continual user involvement." The model restricts iteration to feedback from adjacent stages, and restricts prototyping to "build-it-twice" development, limitations recognized early in interactive systems development and a source of criticism that has gathered force with the spread of such systems. But the waterfall model is a natural fit to competitive contract development specifically, wherein the user organization determines feasibility and prepare a requirements specification, and then may issue successive contracts for design, development, installation, and maintenance.

Perhaps because the waterfall model and its refinements were particularly suited to the competitively bid contract paradigm that dominated early systems development, they underlie many of the structures and procedures

found in systems development more generally. It was the first carefully evolved software development method. It was successful for the more predictable, non-interactive systems that were then the focus of development: with less initial requirements uncertainty, the heavy emphasis on design made sense. Also, many companies were simultaneously engaged in contract and product development. However, for today's developer of interactive systems, competitively bid contract development imposes a "wall" between users and developers: the reliance on specifications documents. This wall may not be impenetrable, but it clearly impedes iterative design and significant user involvement. The other development paradigms for interactive systems face the challenge of freeing themselves from problematic organizational structures and development practices that originated in contract development.

Product development and the focus on human-computer interaction.

As hardware costs fell and the market for computer systems broadened through the 1960s and 1970s, product development came to dominate in the U.S. The discretionary nature of product acquisition meant that systems had to appeal to actual customers, rather than meet written requirements specifications. However, product development companies are buffered from "end-users" by two intermediaries: the market and the customer. The size of the market meant that a product need not appeal to any one person -- it could do very well if only a modest fraction of the market found it acceptable. In addition, especially in the beginning, the product generally had to appeal only to the customer -- the person responsible for the purchasing decision, often a manager or information systems specialist -- not the end user. These customers shared with the product development company the job of assuring system acceptance. They could hire systems administrators, provide training, establish internal development groups to tailor the product, supplement the product with third-party software, or even mandate compliance. Thus, if 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, with consequences for the field's development.

The delay in having to respond to users' needs allowed product development companies to focus on core functionality that might meet only a fraction of most actual or potential users' usability needs. Product development organizations could both discount the importance of the human-computer dialogue and develop an array of indirect methods to discover some of those needs in potential users. Now, as usability expectations in many markets increases, product development companies are entering the phase in which "user needs" replace "software constraints" as the dominant influence on development (Friedman, 1989). (Lacking these buffers between developers and users, internal systems development

entered this phase a decade earlier, as described below.) We will see that the indirect, mediating organizational structures and processes that product developers formed to bridge the gap to users may actually inhibit direct user-developer contact.

The muffling of individual users' voices meant that as usability became an issue in the 1980s, product developers could focus on the "generic" aspects of the human-computer dialogue that are shared by almost all users. This includes aspects of motor control, perception, and lower cognitive processes -- the "look and feel" of software. These were the issues explored by researchers and developers in the field of "human-computer interaction," which established an identity at the 1982 Gaithersburg Conference on Human Factors in Computing Systems, forerunner of the annual ACM Special Interest Group on Computer and Human Interaction (CHI) Conferences. Industry representation at these conferences and in the related journals (e.g., Human-Computer Interaction, established in 1985) has been predominantly from product development companies.

The product developers' focus on perceptual and cognitive aspects of usability at the expense of social or organizational concerns was further justified by declining systems costs and the arrival of multi-tasking systems and personal computing. Single-user systems and applications could be very profitable. So profitable, in fact, that American product development companies have had the resources to staff internal 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 the contract development paradigm, where usability is taking even longer to come into focus. In-house or custom development also remains relatively uninvolved. This may be partly due to relatively scarce resources, but more likely results from differing interests: internal development *must* focus on the individual and group differences and social dynamics that product developers can largely ignore -- these are central to the acceptance of a specific in-house or custom-built system.

In-house or custom development and the new focus on user participation.

In-house or internal development, in which the developers and users work under the same corporate roof, may have been the original development paradigm -- when the system developers were themselves the users. Today, it lacks some of the visibility in the U.S. that the other paradigms have achieved through their concentration of resources and association with the major computer manufacturers. But it is a major generator of development projects and in much of Europe remains the dominant paradigm. In-house and custom development afford an obvious potential advantage to user participation in design: the developers and the eventual users are known when the project is initiated. But there are potential obstacles as well. These projects are often for multi-user systems, and no

wall exists between the developers and the users -- or if one does, it comes down immediately upon completion of the system. Many of the challenges of designing interactive systems for "end users" were first fully experienced in this environment, where success requires that a pre-defined set of users accept the system. (The alternative, replacement of the users by people who can handle the system, is an undesirable outcome from everyone's perspective.) Successful contract development requires conformance to a written specification; successful product development does not depend on appealing to any one individual or group. But an internal development project must be accepted by a specific group of users.

Friedman (1989) states that by the early 1980s, "user needs" had replaced software constraints as the dominant concern within the internal systems development paradigm. It is not surprising that this occurred just as interactive systems were replacing batch processing. Design approaches based on the active involvement of users gathered strength, notably in Scandinavian projects of the late 1970s and 1980s (see e.g. Bjerknes et al., 1987). Many of these were collaborations between researcher-developers and specific user organizations; the projects belong to this paradigm in that all development partners were identified at the outset.

Cultural and political factors, including a strong trade union movement, are often credited for the rise of collaborative design in Scandinavia. They surely contributed, but the dominance of the in-house development paradigm in these countries presented precisely the right motivation and conditions for such experiments. Unlike in the United States, R&D resources were not absorbed by large government contracts and product development, contexts in which "user needs" have been slower to come into focus and in which some of the 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, but resistance to "unfriendly" systems is growing. There is greater 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.

Although some people working in the product and contract development contexts foresaw the importance of usability and considered user involvement as an approach to obtaining it, they encountered obstacles, and the overall record of achievement is not impressive. Today, researchers and developers in product and contract paradigms who wish to accelerate the pace of user involvement are turning to European researchers and developers with a track record in such collaborations. For example, at the

ACM-sponsored Conference on Computer-Supported Cooperative Work held in Portland, Oregon in September, 1988, six of the 30 papers presented were of Scandinavian origin. Thus, this third paradigm is today contributing techniques and approaches to interactive systems development.

Product and contract developers may obtain insight from the European approaches, but should bear in mind that the approaches originate largely in a different development paradigm, internal or custom development. Today, product development projects in particular have acquired the same *motivation* to involve users that in-house development projects had ten years ago. However, they experience different *conditions* within which to try to engage users in development. Roadblocks to "an early and continual focus on users" include the timing of involvement depicted in Figure 1 and organizational structures and processes that were established prior to the importance of the interaction dialogue. Adapting to the new situation may ultimately require significant organizational change. To guide that change, and to work effectively in the meantime, we can identify each paradigm's unique advantages and disadvantages for realizing successful user participation in design.

Factors Influencing Interactive Systems Development

Before exploring the opportunities and obstacles that each paradigm offers, we will identify several constraints on development projects that can influence the conditions for user involvement. One, the timing of involvement of development partners, was used to identify our three paradigms for projects: a *single development group* with a *single user organization* (internal or custom development), a *single development group* with *many potential users* (product development), and a *single user organization* for which there are *many potential developers* (competitively bid contract development). Other factors are: the size, charter, and structure of the development organization; the nature of the user population; 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 over which the partners may have little control; and changes encountered over the life of the project.

The size and charter of the development company or organization. This paper generally focuses on projects in larger organizations. Projects in small organizations may differ. 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. These permit greater flexibility, more latitude for (inexpensive) innovation, and of particular significance for user involvement, the possibility of focusing on a few potential customers (even, for example, by finding one willing to cosponsor development). In these ways, a small product development company may take on characteristics of in-house or custom development. At the other end of the size continuum lies pre-divestiture AT&T and

today's partly cooperating regional telephone companies. These large, influential organizations have characteristics of both product development and in-house development organizations, catering to the needs of large segments of the population.

Organizational charters also vary. A company may work in more than one paradigm: for example, most large product development companies will pursue government contracts for systems that can be built on their products. A separate "Federal Systems Division" may be established to manage these projects, but influences are likely to be felt across divisional boundaries. A product development company that is considering entry into a new market may experiment by custom building a system for one customer in order to obtain domain expertise. The Scandinavian UTOPIA project did the reverse experiment: methods from the in-house/custom development paradigm were adapted to a product development effort (Ehn, 1988). A similar blurring of paradigms may occur gradually, as 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.

Mediators: additional partners in the development project. Although we have considered primarily users and developers, it may be an unusual project that does not involve other parties. These include other groups within the development and user organizations, as well as external consultants, subcontractors, independent sales organizations, third-party developers, product user organizations, trade unions, and standards organizations. Such mediators may play incidental or central roles, informing developers of users' needs and informing users of technological opportunities. The adequacy of these channels for "indirect collaboration in development" is a critical question for interactive systems development.

Organizational structures and procedures within the development company. Companies doing similar work may develop similar approaches to dividing responsibility and meeting obligations. Certain job descriptions and work procedures are widespread in the industry. However, companies of similar size and charter are often organized differently, distribute power differently, and employ different approaches to system development. Some companies are driven by their marketing division, others by engineering. Some companies are hierarchic bureaucracies, others are strongly divisionalized into semi-autonomous sub-units, and some have experimented with almost ad-hoc approaches. Even where high-level approaches to organizational structure and process are shared, small variations in structure and process may greatly affect individual development projects. Reorganization is often experienced as almost continual within development companies.

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 stopped today, the current technology would not fully realize its potential for some time. The number and heterogeneity of users is a factor that is often of particular concern for internal development. Users may be "captive audiences," using systems acquired for them by others, or they may be discretionary users. At the extreme end of discretionary use, those who pay for their own systems may become involved through their personal economic stake -- an Apple developer said, "We don't have to ask our users for advice, they volunteer it." Physical separation of developers and users or the geographic dispersal of the user population may be critical, as may be barriers of class, culture, or language. The sensitivity of the users' work is another factor: a group designing a system for the CIA may quickly appreciate the potential limits to one's ability to "know the user!"

Commitments and agreements among the groups involved. These range from informal understandings between individuals or groups within one organization to binding contracts among companies. Agreements also vary in content, focus, and flexibility. The content may be restricted to technical aspects of the system or may include such commitments to the users as organizational impact statements, installation, or training. Similarly, an agreement may be restricted to the system to be developed or it may specify aspects of the development process, including the use of techniques to facilitate user involvement, such as prototyping or scheduled reviews. Contracts may vary in flexibility -- even a formal contract might specify times at which its terms could be reconsidered, again permitting design changes reached with user involvement (Grønbæk et al., 1990).

Societal conditions and change over time. Grønbæk et al. also describe dynamic influences over which the development partners may have little direct control. These include aspects of the labor market, economic considerations of supply and competition, available technology, formal standards, and legal restrictions on the use of technology or requirements for safeguards. These and the other factors described in this section are subject to change within the life of a project; it may be a rare project that enjoys static conditions from start to finish.

Focusing on Users: Opportunities, Obstacles, and Mediators

Every system development project has particular advantages and disadvantages for involving users and specific strategies to cope with incomplete communication between developers and prospective users. These are explored here at a more general level -- as they affect projects within each of the three development paradigms.

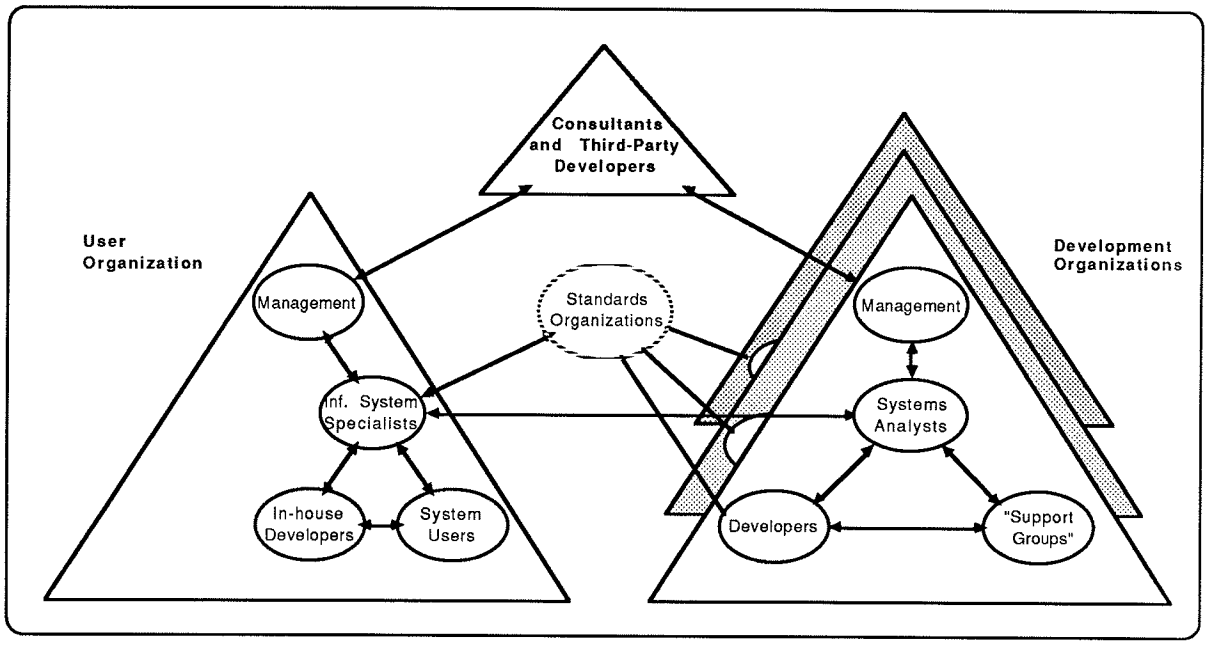


Figure 2. Competitively bid contract development: only the users are specified fully at the outset.

1. Competitively bid contract development.

User involvement faces the most formidable obstacles in this paradigm.

Opportunities. Starting with a well-defined user population is an advantage in obtaining user involvement. Of course, the users themselves typically do not compose specifications documents, but the opportunity exists to enlist their cooperation. Also, contract development projects are often large, providing substantial development resources, and slow-moving, potentially providing time for iterative development with user involvement. High-level management is committed to the success of the project, 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 (e.g., 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 typical case: first a requirements specification is prepared by the user organization, a large government agency, followed by a request for proposals to develop the system design. A contract is awarded and completed, resulting in a design specification that is the basis of a request for development proposals, leading to the development contract (Gundry, 1988). User involvement in specifying the requirements is not assured in the large bureaucracy contracting the work. Once the request for design proposals is issued, communication between potential designers and the user organization is sharply curtailed and monitored, to prevent any bidder from acquiring unfair advantage. When the contract

for design is awarded, the designers may not contact the user organization, to avoid influencing the subsequent bidding on the more lucrative development contract. The organization with the design contract does not always obtain the development contract issued later. Thus, the designers may be unable to contact the users, and the developers are not yet known. Even when the same company obtains both the design and development contracts, the development team is often distinct from the design team, who move on to other design proposals. In addition, companies often minimize risk by preparing joint bids, so development may be divided across two or more organizations. Contact with users continues to be curtailed during development, for reasons of security, geographic separation, or to avoid influencing later bidding (for example, on implementation or installation of the system, or on system maintenance, which is often the most lucrative contract of all). The designers and developers have another good reason to avoid contact with the users. Contract compliance is based on conformance to the written specifications. Programmers may even code non-functioning procedures described by a miswritten specification to avoid jeopardizing compliance. The development organization might well *prefer* the commitment to a relatively comprehensible and static document to the alternative of trying to satisfy unpredictable users. "Usability requirements" in contracts for interactive systems may consist entirely of statements such as "the system shall present information succinctly" or "the system shall be easy to operate efficiently," (Gundry, 1988). Contracts may provide for demonstrations of prototypes *to* users while legally restricted contact precludes feedback *from* the users. This has the unfortunate effect of doing little beyond alerting users to inadequacies of the system they will receive (Gundry, 1988).

Mediators. Figure 2 illustrates several groups and organizations that play mediating roles in the communication between system users and contract developers. Within the user organization, information system specialists negotiate the system requirements and internal developers may tailor aspects of the system's use. Similarly, analysts or specialists in the development organization write proposals and negotiate contracts, with developers assigned to projects after a contract is awarded. External consultants may play a "surrogate developer" role early in the requirements definition stage, providing the customer with insight into feasible technologies. Other consulting contracts may be issued as well -- for system implementation (installation), for example. In fact, contractors working on one phase of the project are in a sense consulting on subsequent phases; e.g., those writing the design specification provide guidance to the as-yet unidentified developers. Similarly, development teams that are barred from direct access to users may seek "surrogate users" -- consultants familiar with the user environment, subcontractors, or even "domain experts" hired from the customer organization to help staff the project. The user organization may 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 required by subsequent contracts. (A large enough customer organization, such as the U.S. Defense Department, can by itself promote standards development and compliance.) Gundry (1988) has proposed that the customer 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 can be viewed as a major step forward is a dramatic statement of how little contact currently exists. Finally, as noted above, the contract itself may be used to open up communication. Where legal barriers do not intervene, contracts with a process focus can define user involvement and even a formal contract may address the uncertainties of dialogue design by providing points for renegotiation of terms based on prototype testing (Grønbæk et al., 1990). Experiments have been tried with design-to-cost contracts, reward-for-effort contracts, and multi-stage contracts in which several developers build and test prototypes prior to the final development contract award (Boehm, 1988).

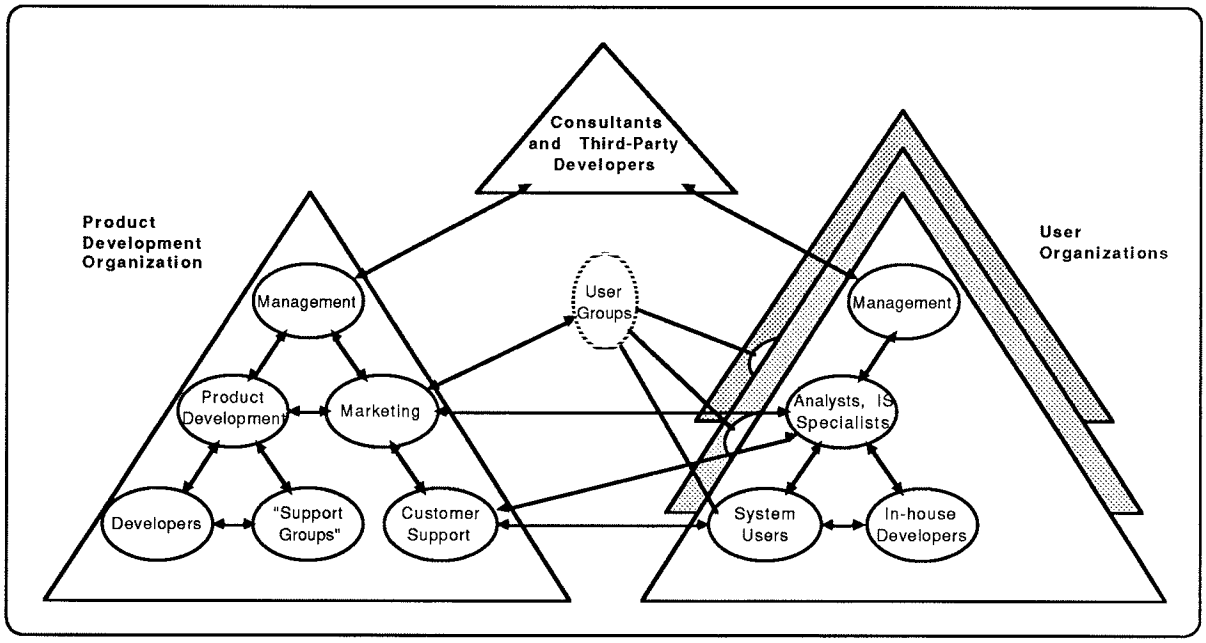


Figure 3. Product development: only the development group is fully specified prior to design.

2. Product development.

Opportunities. Because development costs are amortized over many sales, product development organizations typically have considerable resources and a large number of potential users, each of which could be drawn upon to improve usability. Competition in the marketplace provides a motive for doing so -- the attention being 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 product upgrades or new releases free some developers from "single-cycle" development: evaluation of existing practice feeds into the design of the next version, and good ideas that arrive too late for one project may be held for later use. 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 user involvement. Developers isolated in large engineering laboratories may not empathize or sympathize with users who are inexperienced, non-technical, or have different values and work styles. Even identifying the development team can be difficult: project membership changes over time and developers of different user interface components -- such as software dialogue, documentation, and training -- often communicate very little (Grudin and Poltrock, 1989). Gould's (1988) solution -- to put all aspects of usability "under one roof" -- conflicts with deeply-rooted aspects of organizational structure and process in many of these companies. Also, the difficulty of identifying future users is a major obstacle to involving them. Strategic marketing decisions may be carefully guarded by upper management lest the competition make use of the information; development teams may not even know which applications will be marketed as packages. In addition, before reaching an "end user," many products are extensively modified or tailored by other developers, either third-party or internal to the customer organization. These developers are users of the product, too. Another obstacle is the difficulty of accessing potential users, once they *have* been identified. Product development companies *try* to shield developers from the distraction of answering individual user requests. Figure 3 identifies several groups whose responsibilities include communicating user needs information, and who may therefore discourage direct developer-user contact: product management, marketing, and customer support in the development organization; IS staff in the user organization. Unfortunately, these mediators may not be effective conduits of usability information, in part due to inexperience with this newly prominent concern. Obtaining adequate time and interest from potential users may also be particularly difficult in this paradigm, where an individual user's investment is low. When contact does occur, product developers risk overgeneralizing from a few contacts and must contend with conflicting user views. Finally, information that is obtained must be worked into the development process, which is fraught with competing interests and tradeoffs. The development process may be designed to maximize predictability and error detection in developing non-interactive systems, but work less well with less predictable interactive systems development. Another obstacle is the great pressure to produce new releases of existing products -- the relatively

short development cycle favors small enhancements over substantial innovation (Poltrock, 1989) and works against providing time for users and developers to educate one another through iterative design. This perceived need for rapid development leads to attempts at routinization through early approval of specifications and schedules, again limiting flexibility.

Mediators. As shown in Figure 3 and described above, sales and marketing, management, customer support, and other groups within large product development companies mediate between users and developers. Even other development groups form part of a "corporate memory" that for usability issues may operate more effectively than formal records. External consultants serve as "surrogate users," providing information on market direction and detailed product critiques. Although consulting, market research, and competitive analysis are undertaken to support marketing campaigns for existing products, they can also provide direction to developers. Independent software vendors and value-added resellers, who adapt products to specific vertical markets, stand between development and user organizations. Customer organizations engage consultants as "surrogate developers" to advise on purchasing or installation. Product buyers rely on third-party developers, independent software vendors, or internal development groups to supplement or tailor products for their environments. Most customers exert little direct influence on product development companies individually, but user groups representing many customers can have an effect. This communication channel may be less effective if meetings are attended by "customers" or buyers rather than by users and by marketers rather than by developers.

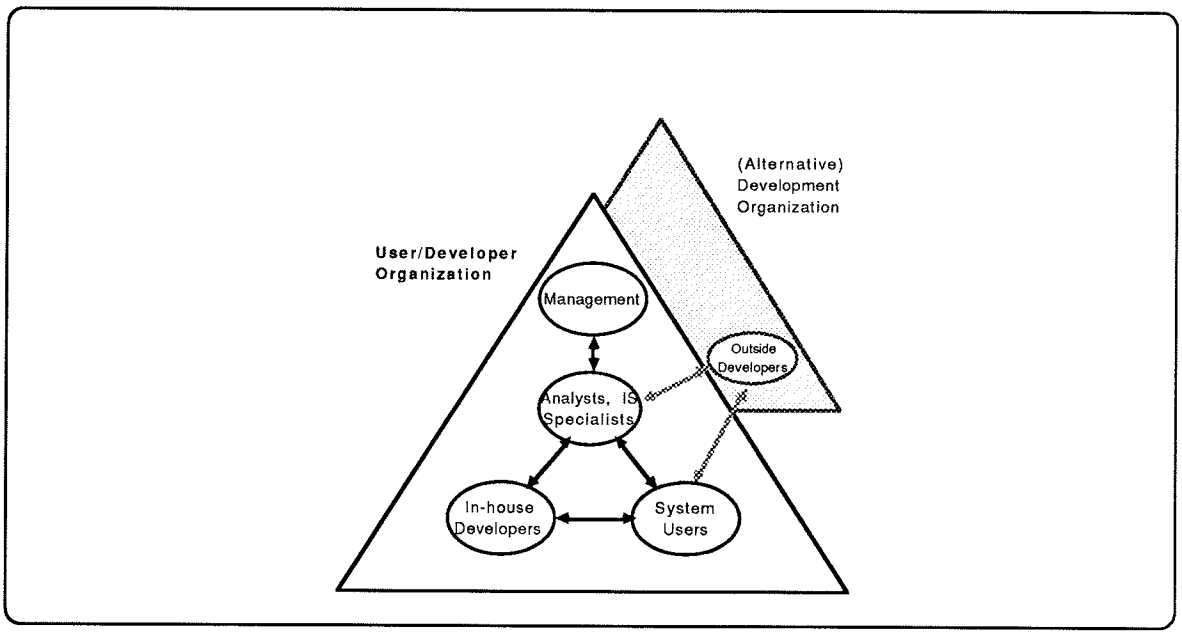


Figure 4. In-house or custom development: developer and user populations are known at outset.

3. In-house or custom development.

This paradigm appears to offer good prospects for collaboration among users and developers, but the challenges can be substantial.

Opportunities. Collaboration is logically easier when both users and developers are known from the beginning. An early relationship can lead to a better design. It can also help with system acceptance by investing participating users in the outcome -- some warn that gestures toward user collaboration may "coopt" users into accepting undesirable system features (Friedman, 1989). In "in-house" development, communication among developers and users may 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 of such development projects is that they have a particularly high level of management support, an important element in obtaining system acceptance. Finally, they may enjoy a less pressured or more flexible development schedule -- shifting from a product focus to a process focus occurs most naturally in this environment.

Obstacles. A major challenge to projects in this paradigm is that they are often systems designed to support organizations, in contrast with applications designed to support individuals. In addition, identifying future users does not insure collaboration. Conflicts of interest exist within organizations. Management may be the ultimate beneficiary of a proposed system; if its interests do not coincide with the users' interests, user alienation may be more likely than user cooperation. Conflicts also occur between different worker groups -- Ehn (1988) describes jurisdictional disputes among typesetters, journalists, and administrative workers in one project. Friction may develop between developers and users due to differing codes of values, conduct, and dress, as well as disparities in age and salary (Friedman, 1989). Selecting representative users can be a problem -- 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 potential roles as system users. In addition, some techniques must be used especially carefully in internal development -- prototyping can unduly raise users' expectations of the system's capabilities or state of completion, which is not usually an issue in product development. In-house development projects may have fewer resources than projects in the other paradigms. Individual efforts to build systems for use over many years provide fewer opportunities for evaluation, feedback, and catching missed windows of opportunity "the next time around" than product development. Prototyping and iterative design are not infinitely flexible, so the internal or custom development team must plan carefully, considering the full expected context of use and 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. As noted above, the spread of interactive systems in the early 1980s immediately confronted in-house developers with the needs of "end users" (Friedman, 1989). The other paradigms delayed their entry into this phase, developers being separated from users by the conditions of development and coping by means of the assistance of the mediating groups described above. Such mediation is less available to internal development projects, unless upper management or "Human Resources" departments are considered mediators. Friedman 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 the developer), decentralizing the information center (to bring the developer 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 merged most of them, employing direct user involvement, prototyping, and the education of developers in the domain area. In fact, these approaches can all be considered "user participation in design," if "participation" is extended to include education that precedes a particular project.

Prospects for Change in Interactive System Development

Several convergent influences are pushing systems development rapidly toward greater concern for users' needs. First, as with any new technology, reaching untapped markets requires learning more about them. The versatility of the computer enhances the likelihood of finding some way to support any group of people whose concerns are properly understood. In addition, where individuals have discretion or control over the tools they use, improving usability is an obvious key to convincing them to exercise this discretion favorably. Discretionary use underlies the recent focus on the "look and feel" of the user interface to software products. Also, understanding user needs is more important in efforts to support *groups*. Systems have focused on supporting organizations and applications have focused on supporting individuals in their work. Now, networking and powerful multitasking systems make group-level support feasible. "Groupware" product developers are confronting issues that were previously encountered mainly in internal development, such as the need to appeal to diverse sets of people. The focus of application development will shift from individual similarities (with the goal of appealing to a large, possibly homogeneous set of people) to individual differences and social dynamics (to attract all members of groups, independent of background, role, preferences). Here, application developers may be at a disadvantage relative to system developers, since

there is less corporate commitment to a groupware product than to a large system. Finally, the cost of computation may be the greatest obstacle to providing more usable systems, apart from development time. As processing time, memory, and maintenance costs fall, computational power is increasingly underutilized. Today, the cost of a better interface is not much more than what is required to build it or buy it.

In summary, vendors' interests, workers' interests, and economic factors are working in concert. Prospects for rapid advance are further enhanced by the possibilities for sharing experiences across the three developmental paradigms. CSCW '88 and the 1990 Participatory Design Conference preceding CHI'90 in Seattle brought together Scandinavian and American researchers and developers. Boehm (1988) outlines recommendations for contract development drawn from prototyping-based in-house development. Techniques developed in one paradigm may be modified and applied in others. The process focus and low-cost techniques that developed naturally in in-house development environments may have broader application (see e.g., "Designing for a dollar a day," Kyng, 1988). Experimental techniques such as "Wizard of Oz" prototyping that have been used in product development are being adapted for in-house projects. The prevalence of specific paradigms may shift with societal changes. For example, the formation of the European Community may promote competitively bid contract development as a means of insuring equal access to government projects. Europeans may profit from American experience and find ways to introduce approaches that achieve higher levels of user involvement.

The debate over the routinization of software development illustrates the different paradigmatic perspectives. Data that suggest deskilling has occurred emerge primarily from large product development organizations, where strong competition and pressure for frequent releases create a tremendous desire to control the development process and render it immune to the loss of any individual (e.g. Kraft, 1977). An absence of deskilling is observed in internal development, where competitive pressures are lower and developers may more readily acquire domain knowledge, increasing their value to their organization (Friedman, 1989). Recognition of the different conditions associated with these paradigms may enable us to apply the hard-earned lessons of one in the others. Changes in techniques, in the over-all focus (e.g. process vs. product), and in the design of organizations are among the potential benefits.

Acknowledgment

Kaj Grønbæk, Susanne Bødker, and Liam Bannon participated in the iterative design of this paper and are anticipated future users -- although, as with any product development project, who will find this useful remains to be seen. Andrew Friedman's book is an inspiration and an excellent source of information on internal development. Steve Poltrock, Henry Lahore, Craig Will, Larry Verner, and Liam Bannon contributed to my understanding of contract development, although their full understanding is not reflected here. Students at Aarhus University commented helpfully on an earlier draft.

References

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

Boehm, B., 1981. *Software engineering economics.* Englewood Cliffs, NJ: Prentice-Hall.

Boehm, B., 1988. A spiral model of software development and enhancement. *IEEE Computer, 21,* 5, 61-72.

De Marco, T., 1978. *Structured analysis and system specification.* NY: Yourdon Press, Inc.

Ehn, P., 1988. *Work oriented design of computer artifacts.* Stockholm: Arbetslivcentrum.

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

Gould, J.D., 1988. How to design usable systems. In M. Helander (Ed.) *Handbook of Human-Computer Interaction.* Amsterdam: North-Holland.

Gould, J.D. and Lewis, C.H., 1983. Designing for usability -- key principles and what designers think. In *Proceedings CHI'83 Human Factors in Computing Systems,* 50-53.

Grudin, J., 1989. Why groupware applications fail: Problems in design and evaluation. *Office: Technology and People, 4,* 3, 245-264.

Grudin, J., 1990. Obstacles to participatory design in large product development organizations. In *Proceedings Participatory Design Conference* (Seattle, April 30 - May 1).

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

Grønbæk, K., Grudin, J., Bødker, S., and Bannon, L., 1990. Improving conditions for cooperative system design - shifting from product to process focus. Manuscript submitted for publication.

Gundry, A.J., 1988. Humans, computers, and contracts. In D.M. Jones and R. Winder (Eds) *People and computers IV.* Cambridge, UK: Cambridge University Press.

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

Kraft, P., 1977. *Programmers and managers: The routinization of computer programming in the United States.* NY: Springer-Verlag.

Kyng, M. Designing for a dollar a day. *Office: Technology and People, 4,* 2, 157-170.

Norman, D.A. and Draper, S.W., 1986. *User-centered system design: New perspectives in human-computer interaction.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Peters, T. and Waterman, R.H., 1982. *In search of excellence: Lessons from America's best-run companies.* NY: Harper and Row.

Royce, W.W., 1970. Managing the development of large software systems: concepts and techniques. In *Proceedings WESCON,* August.

Suchman, L., 1988. Designing with the user. *ACM Transactions on Office Information Systems, 6,* 173-183.