The Organizational Context of User-Centered Software Designs*

By: Rob Kling

Abstract

Many computer programs that are technically well designed fail to meet the human or organizational purposes they were expected to serve. Prevailing design methodologies stress either special technical devices or policies of involving people in the design of software they will use. Several user-oriented design strategies and their limitations are discussed. An expanded conceptual framework which includes both the personal characteristics of designers and features of the organizational settings in which they work is presented. The framework is used to develop a set of policies for practitioners to help foster the development of user-centered systems.

Keywords: User-centered design, socio-technical systems, user-orientation, sociology of computing, software design
Categories: 2.11, 2.41, 3.50, 4.0

*This article is part of the URBIS Research Project: Evaluation of Information Technology in Local Government. This research is supported by a grant (APR 74-12158 AOI) from the Research Applied to National Needs Division of the National Science Foundation. An earlier version of this paper was presented to the IEEE Computer Conference, San Francisco, California, January 1975.

The Problem

This paper focuses upon users, the forgotten people in software design. It develops a framework to analyze the conditions under which software design groups are most likely to produce systems that meet the needs of their users.

Computers have been applied to an immense variety of managerial, engineering, industrial, and scientific applications during the last decade. Unfortunately, many computer programs that are well designed in terms of technical criteria, such as run-time efficiency, fail to meet the human or organizational needs they were expected to serve [38]. While the technical computer literature describes algorithms and systems that are technically effective, computer specialists have developed an informal, more private folklore of systems that were underused or abandoned because they were ineffective person/machine systems. Systems can be ineffective when they are not well understood by the people who use them [27], provide inaccurate data [13, 19], demand unusual precision and attention [3, 20], or are difficult to modify when the kinds of information users want changes [11].

These difficulties can undermine the utility of a computer system even when its users are relatively homogeneous and welcome computational assistance. When the "users" of a system have different or conflicting needs, political dynamics have an even greater impact on the use of a system than do its sheer technical qualities. Developing usable designs remains a subtle art even when the users of a system are not in conflict. The most difficult problem in many settings is acquiring an appropriate set of specifications for data and record-keeping conventions, the kinds of manipulations desired, and the way data should be available in reports or queries.

Consider the common issue of designing the interfaces for entering, retrieving, and reporting data from an information system. If one valuable byproduct of automation is the creation of jobs which are less alienating [32, 33], then gratifying interfaces between a computer system and its users are an essential ingredient of any "successful" system. Systems have largely been

designed from the "inside" out during the past decade [20]. Substantial attention is paid to efficient and elegant code. The interface with the user is usually designed last to match the "internal" system. Substantial know-how has been acquired in the construction of various efficient algorithms [15] and the majority of studies in computer science cater to such concerns [29]. We have, in contrast, relatively little systematic knowledge regarding the design of good user-oriented systems. Consequently, analysts with conventional training are biased toward neglecting the person/machine interface.

There has been increasing attention to "designing for users" during the last five years [11, 18, 19]. However, the conditions that foster designs "for users" are poorly understood. The most common prescriptions to support user-centered designs focus on involving users in design teams. The conditions, however, under which such participation will be *substantive* rather than symbolic are rarely specified. In addition, the kinds of understanding communicated by users when they participate in design teams are rarely made explicit. This paper develops a framework to predict the conditions under which designers are likely to develop user-centered systems.

Methodologies for Computer System Design

The prevailing norms of computer system design are machine-centered (Table 1). Functional designs that focus upon direct effects, such as increasing the ease of generating reports, are highly prized [13, 28], while indirect impacts tend to be ignored [11]. For example, Bjorn-Anderson [5] recently reported a study of six banking systems which had a complex pattern of impacts on the work of clerks and managers who used them. Some systems increased the variety and scope of their users' work while others led to more constrained and repetitive jobs. However, in each case the designers expected their computer systems to have no effect on the jobs of their users; the actual impacts were simply unintended. Typical systems are evaluated by their impact on information flows [33], the logical organization of the functions they perform [28], and narrowly defined cost-effectiveness [6].

Emphasizing these criteria are *policy decisions*, not simply technical decisions. Computer scientists can measure the computational efficiency of an information system; organizational psychologists can tap job satisfaction [24] and satisfaction with computing systems [17]. Both sets of criteria can be operationalized. The deeper issue is deciding which criteria are relevant to the design and implementation of computer-based systems so that the systems will be effectively utilized as well as enhance the work environment of the people who use them. Recent studies indicate that both technical and "social" criteria influence the success of computer-based systems [17, 18]. Systems which are poorly designed or do not meet the actual needs of their users are not effectively utilized, nor do they satisfy the people who use them.

During the last several years, concern for the "needs of computer users" has become commonplace. Several investigators have developed specific procedures to help insure that software designs will meet the needs of some of the people who use computing[1] [3, 17, 18, 19, 20, 23, 26]. These proposals vary in emphasis, comprehensiveness, and likely effectiveness.

Some analysts emphasize the *technical characteristics* of the interface between a computer system and its users. Both Martin [20] and Coles [3] limit their attention to interactive systems which allow some dialogue. Coles focuses upon the problems of providing users with a comfortable, natural language interface. He emphasizes in particular that novice and expert users of a particular system require substantially different interfaces. Martin's handbook [20] presents detailed guidelines for the technical development of useful and intelligible dialogues between machines and their users. He proposes specific solutions to specific problems, *e.g.*, printing back input to aid in detecting errors when data is entered. Both Martin and Coles

1. We explicitly ignore programming methodologies such as structured programming. These are of interest to the extent that they produce programs which optimize "user-centered" features such as modifiability.

Table 1
Some Characteristics of Machine and User-Centered Environments

USER-CENTERED	MACHINE-CENTERED
1. Systems are valued that increase personal competence and pride in work.	1. Efficiency is emphasized.
2. People are accepted as non-rational and error-prone.	2. Systems are designed in purely functional terms.
3. Jobs are designed to be personally satisfying. Automated procedures are designed to fit job needs.	3. Human error is not tolerated. Systems are not error tolerant and provide limited diagnostic information.
4. The burden of precision is placed on the machine. Systems are forgiving.	4. Users are forced to match the precision required by the machine.
5. Users easily obtain/create systems that meet their needs.	5. Jobs and procedures are designed to simplify machine processing.
6. Users can initiate, veto, and collaborate in system designs.	6. Human relations are ignored as long as the job gets done.
7. Designs and assumptions are intelligible to users through appropriate technique (modular structures) and clear documentation.	7. System designs are imposed on users. They initiate but never veto system designs.

attend to the different needs of naive and sophisticated users; both attend to the needs of hypothetical terminal operators.

The richest set of criteria for "humanized designs" appears in the *Stanley House Report* prepared by Sterling [34]. It explicitly considers all the people who come in contact with an information system as worthy of careful attention during design — whether they are managers, clerks, an organization's clients, or data subjects. Twenty-five guidelines for designing systems are presented that deal with users, with exceptions that arise during the course of system's use, with problems of information control, with privacy, and with aspects of a system that have a bearing on ethics. These guidelines range from cosmetic features ("Transactions with the system should be courteous") through technical requirements ("There should be procedures to correct errors," and "The system should give people choices on how to deal with it") through organizational policies ("A procedure must exist to override the system"). While these guidelines are quite general, they place a sufficiently clear set of constraints on a system that one can judge at least whether a particular system follows each guideline. In addition, these guidelines explicitly treat an information system as part of the larger organization in which it is embedded. Thus, whether or not people have choices in the ways in which they deal with an information system focuses upon what choices are available rather than whether they are automated.

Unfortunately, the approaches proposed by Martin, Coles, or the *Stanley House Report* fail to deal with the process by which the actual users of a computer system are identified and the variety of their particular needs ascertained and then negotiated into the design of a particular system, including its associated organizational arrangements.

Some analysts note that the needs of computer users vary from situation to situation. For example, Orlicky's [26] proposals are written for a business manager who is relatively unfamiliar with computers. His twelve criteria for implementing system designs include "Keep service to user uppermost in mind." This concern is elaborated in several suggested proposals such as using English instead of mnemonic programming codes on output forms. He also suggests

that users be invited or forced to participate in system designs. However, he neglects to specify the actual influence that users will have upon the resulting design. He simply doesn't distinguish between symbolic participation and substantive influence.

The most sophisticated design strategies have been developed by investigators such as Lucas [17-19], Hedberg [8], and Mumford [9] who have carefully studied the ways in which computer users may inform designers about their wants and needs with some precision and clarity. Lucas suggests that designers should regularly tap the perceptions and satisfactions of clients with the computer applications they use. His survey technique includes a questionnaire that indicates a respondent's perceptions of (a) the computer system, including its ease of use and the quality and usefulness of the information it provides, and (b) the quality of service provided by the computer staff.

Mumford reports an even more ambitious approach which is now in process in several British firms. She has elaborated a "needs-fit" theory of job satisfaction which she uses to probe for areas for computer impact that might effect the job satisfaction of employees in a computer-using department. Her theory assumes that each employee has certain needs such as to be equitably rewarded for the work performed; to use and develop his skills; to secure particular levels of achievement, recognition, responsibility, and autonomy; and to work in accord with his own ethical and social values. In contrast, different jobs offer different opportunities to satisfy each of these needs. "Job satisfaction" is measured as the quality of match a particular job provides to a particular person along these various dimensions[2]. She uses a questionnaire to tap employee job satisfaction, attitudes toward change, and attitudes toward a new computer system in a department which wishes to automate some of its procedures. This data is analyzed and used to identify those dimensions of job satisfaction that many employees are well satisfied with and that the automated system should not disturb. In addition, those dimensions with which many employees are currently dissatisfied are identified and analyzed to see how the new computer system might produce a better fit between employee needs and job characteristics. These analyses are discussed with the organizational management, department employees, and computer staff. Unfortunately, Mumford's applications are used by dozens of managers and clerks in several different work groups within complex organizations. *"The user"* is merely a convenient linguistic fiction which helps simplify syntax and implies a kind of homogeneity and consensus which makes a system design appear legitimate [14].

The actual demands that different users and consumers place upon an information system range from those that are sheerly conflicting, to others that are complementary, compatible, or indifferent. When the users of an information system are engaged in a specific social conflict, an information system should be viewed as an instrument in that conflict and the designers viewed as political agents [12]. For example, when a top manager initiates a cost-accounting or time-reporting system to help gain further control over slack resources, the resulting information may best serve *him*, even though many other managers and staff are using it to keep records and receiving specialized reports from it. Increasing the control over slack resources by one manager diminishes the extent to which lower level staff can control those resources. A designer in such a situation cannot simply meet every "user's needs" since they may be in sheer conflict[3]. The approach to software design developed in this paper presumes that the diverse users of a computer system are not engaged in a sufficiently profound and pervasive set of conflicts that cooperation is infeasible. Negotiating cooperation among computer users with incompatible interests entails some organizational politics [10].

2. This capsule summary just hints at the richness and structure of Mumford's theory. The interested reader is directed to her accounts for additional detail [23, 24].

3. Such examples are commonplace, but not dominant in many computer-using organizations. For a careful study of the ways in which automated information systems enhance the control of top executives over department heads in American local governments, see Kraemer and Dutton [16]. For an extensive account of the politics of design, see [36].

However, when there is little manifest conflict between the users of a computing system, a variety of needs may be met in a single system. Consider the example of municipal accounting and budgeting in the U.S. Most cities segregate their revenues into dozens of specially labelled funds such as property taxes, state grants, and parking fines. Some cities, such as Kansas City, have over 100 funds. The budgets of each department are drawn from specific funds. Accounting staff are particularly concerned that no fund be overdrawn and organize their ledgers and budget reports "by department, by fund." Department staff and budget officers, on the other hand, are more concerned with the total funds available in certain line-item accounts rather than the way those accounts are distributed across various funds. However, many automated municipal ledger systems produce budget summaries for departments separately for each fund. Department staff often hand tally several fund-specific reports to learn what their actual balances and expenditures are. When these ledgers were originally automated, "the user" often referred to the accounting staff, even though many other people's needs could easily have been accommodated. Even second generation machines could add across funds as well as within them.

Examples of designs which accommodate the needs of only a few people or groups that use a computer-based system are legion. They typically occur where an information system serves several departments or staff with different responsibilities and work demands. Investigators such as Hedberg, Lucas, and Mumford stress participation by a variety of users in the design of information systems to ameliorate these difficulties. However, Hedberg [8] notes that apparently broad participation may be merely symbolic. Symbolic participation is most likely when the participants are drawn from geographically dispersed work groups and have little time to devote to either travel or understanding and altering the design proposals developed by computer specialists. But even symbolic participation is still not widespread. Table 2 presents a portrait of the ways that users are currently involved in computing design in a large sample of American local governments.

According to the self-reports of computer specialists, users seldom or never review designs for computer applications in 25% of the cities and counties in the survey. Based on field studies in several dozen American local governments carried out over the last three years, we find that users are often in weak positions when negotiating with service providers. There is bias in these self-reports; computer specialists usually *overreport* the level of involvement that users play in decisions about computer use. In this sample, the median number of applications automated was 40; staff in these governments are familiar with automation. Thus we find this data particularly revealing.

It is tempting to view this data as indicating that computer specialists are more interested in serving their own interests and conceptions of computing than providing a service to their clients [4]. While there is much to be said for this view, it simply opens the question of what computer users may do to insure that software designs meet their needs.

It is further tempting to ask what kinds of incentives or organizational arrangements might promote the active participation of computer users in the designs of systems they will use. However, "user involvement" is merely one policy among many that may foster user-centered designs. We take a different approach by asking: what kinds of personal characteristics of designers and organizational characteristics foster the development of user-centered designs? Our answers to this question will help develop a set of policies that foster user-centered designs.

Framework

In this section we develop a framework to help us theorize about the conditions under which software designs will be particularly user-centered. The framework includes both characteristics of the designer, *e.g.* his values, and of the organizational context in which computer systems are designed and implemented. The framework specifies both the personal characteristics of designers and the organizational settings in which they work by a few constructs:

Table 2
Participation of Computer Users in Data Processing Activities

Question: "What is the frequency with which users of your data processing unit do each of the following?

Activities	Percentage of Installations Indicating that users participate*			
	Never	Seldom	Often	Always
A. Participate as members of a design team.	17	24	44	15
B. Perform systematic analysis of benefits and costs anticipated from a proposed computer application.	23	48	21	8
C. Review designs for a new application.	6	19	36	38
D. Provide test data for an application.	19	35	31	15
E. Sign off, accepting an application.	26	27	24	23
F. Provide informal feedback on problems with the data processing unit.	05	15	49	31

*Based on a sample on 473 cities and counties from a survey administered in 1975.

Personal Characteristics

1. Technical competence
2. Designer values

Characteristics of the Organizational Setting

3. Salience of consequences
4. Salience of responsibility
5. Management support
6. Resources: time and money.

We believe that user-centered designs are unlikely to be developed *routinely* by design groups which work in organizational settings that are not characterized by the features listed above. For example, if a technically skilled designer who wishes to develop a user-centered system has little awareness of the consequences of his design alternatives for his users and few resources to spend in finding out, then it is unlikely that the system he designs will be especially tailored to the needs of its particular set of users. This may sound obvious and even tautological; but note: the design methodologies developed by computer technologists emphasize a set of techniques for engineering appropriate data structures and an interface between a system and its users. Organizational psychologists emphasize involving users in design. Each group neglects, or takes for granted, the resources and management support they require, as well as the activities emphasized by the other. This framework makes these elements explicit. Each element is examined in this section. The next section proposes several policies to deal with the more problematic elements, *e.g.*, management support.

Technical competence

In the remainder of this article we assume that a designer is competent to carry out the technical aspects of his job. Designers of similar skill will develop different designs. We are interested in the non-technical factors that influence the shape of their designs.

Designer values

Software designs are often under-specified; designers have tremendous freedom in shaping a design in accord with their own technical preferences and understandings about who computer users are. But we know very little about the perceptions that computer specialists have of the users they serve and the ways in which they translate those perceptions into concrete designs. Hedberg and Mumford [9] report a study of the "model of man" held by a sample of Swedish and British designers. They found that designers had two quite different models of people in organizations. One model which was elicited in response to questions regarding workers in general emphasized people's desires for challenging work and self development. However, when designers were asked to characterize the people for whom they designed computer systems, they portrayed the computer users as limited, conservative people who work with a limited set of skills and who are extremely resistant to change. There were also cultural differences in the study: Swedish designers viewed both workers in general and computer users in particular as seeking more challenge and being less constricted than did British designers.

To increase user-satisfaction, should the manager of an applications programming group simply hire staff who want the systems they design to be satisfying tools? A recent study of control panel designers [21] showed that they held a set of priorities very similar to those of human factors experts. They were asked to design several simple control panels and their designs were evaluated. Surprisingly, the concerns they voiced for easy use and maintenance simply were not embodied in their designs. They felt that human factors criteria were "obvious," disregarded available data on parameters for humanly effective designs, and created products which were ill-conceived from the point of view of their ultimate users. Articulated values, and even sincere intentions seem to be inadequate to insure appropriate designs.

Salience of consequences

Are there particular conditions that would foster congruence between the values designers articulate and actual designs they produce? This question is an instance of the more general question: when will people act in accord with their beliefs? The common sense explanations of actions that are in accord with belief are framed in terms of "strength" of belief, wanting to be consistent, rewards, and courage. Recently, an experimental social psychologist has elaborated a more powerful theory to predict when a person's acts will conform with his values [30, 31]. Schwartz presents a careful set of experiments that show that the feedback a person receives about the consequences of his actions will promote congruence between his actions and his beliefs. In programming settings, organizational structures and policies which promote clear feedback from users to designers will foster user-centered designs if the designers value them and the conditions outlined in the following sections are met.

The applications programming staff in many organizations is explicitly insulated from close contact with computer users [22]. Policies in such settings which promote the participation of representative users in the design of their systems or which provide sufficient time for a designer to "get into" the work environment of the user would promote such feedback. The information that is passed on to the applications group will be highly filtered in settings where a separate consultant group handles day to day complaints. There is some evidence that officials who cannot act directly on a complaint will tend not to pass it on [35]. Similarly, a large number of complaints about a particular system may be "trapped" by the consulting group and not passed through the organization to the analyst/programmers who can act on them. Lucas' proposal for designers to directly survey user's perceptions of the satisfactions with the systems they use attempts to circumvent this difficulty [17].

Salience of responsibility

A second major factor in Schwartz's theory is the degree to which a person accepts responsibility for his actions. This may be a personal trait: some people accept more responsibility for their actions than others (ascription of responsibility), or it may be a byproduct of the job structure that particular individuals are held responsible for particular actions (salience of responsibility). The salience of responsibility in development projects is quite high since it is usually precisely clear who is responsible for which section of a design and its implementation. Programmers, in contrast, who maintain software developed by others are in an ideal situation to pass the buck on complaints. After all, they didn't design the program in question; they just have to modify it as best they can.

But the deeper issue common to both situations is: to whom, and for what, are the programming staff responsible?

Management support

Computing Staff

He who pays the piper calls the tune. While people do not respond rigidly to explicit organizational incentives, the criteria used to hire, promote, raise salaries, and fire surely impact behavior. Computing staff maintain science-related norms of preferring to work with the most sophisticated technologies available, at the "frontiers of knowledge" [24]. In addition, experience with advanced technologies aids in keeping the computing staff member from becoming technologically obsolete and in remaining attractive in the job market. Computing personnel are usually hired for technical competence, and promoted for technical and fiscal success. The latter include meeting deadlines and staying within the budget allocated. The explicit satisfaction of users with the particular applications packages they have developed in the past is, at best, usually a secondary consideration. Nor is user satisfaction systematically monitored and evaluated like a budget. It is unlikely that many systems will be designed for the actual needs of many actual users, without a set of incentives that reward those members of the computing staff who value service as much as technical sophistication.

User Staff

The preceding discussion portrays a unilateral relationship in which designers impose systems on users. Such a relationship may prevail when designers view themselves as change agents "reforming" an inefficient organization [1]. A recent study [7] of middle managers showed that many of these managers felt little need for automated information systems. While most felt that they ought to have a large role in the design of such systems, their low priority suggests that they may provide few personal or staff resources to understand them and involve themselves *meaningfully* in their design. Meaningful "user involvement" requires time and interest. The using managers of MIS may have to care about its development the way they would involve themselves with any other major project. Some of the people who use automated information systems need some time released from their routine work to engage in the design of the new system. Mumford's work in the design of user-centered systems seems to take place in departments which provide substantial support for the staff to learn about alternative designs for new systems.

Resources

There is a major trade-off between the time taken to write a computer program and its user-oriented features. Several different groups in a recent study [37] were asked to write programs to perform the same task. Each group was asked to optimize its program on a pair of different criteria chosen from minimum programming time, minimum run time, minimum storage space, minimum program size, program readability, and output readability. Program readability is a major factor in simplifying later modifications and maintenance. All the programs were ranked on all the preceding criteria. Most groups ranked first on their most important objective and often performed relatively well on their secondary objective. However, groups that performed well on some of the minimization criteria — *e.g.*, minimum programming time, performed poorly on the user-related criteria —

e.g., output readability. Weinberg claims that when a program is specified in terms of its function, programmers may design it on many different and incompatible criteria. Some of these additional criteria are often preferred by particular programmers; they contribute to "style." If these additional objectives such as efficient run time, error protection, or output readability, are included in the program specification, then programmers may meet several of them well. Otherwise, important objectives which are unspecified may be treated as "free variables" and sacrificed to satisfy other less important and idiosyncratic objectives.

The common practice of letting user-related objectives "run free" is exacerbated by their neglect in the available programming time and cost estimating procedures [2]. These procedures emphasize factors which are internal to a system, such as the number of instructions and the interdependence of subsystems, and neglect user-oriented tasks such as the time needed to negotiate a design and the training of users. These user-oriented activities are congruent with the various estimating procedures. They are often ignored and must be included for reliable estimates of overall system cost to be made.

The major resource that user-centered designs require, aside from technical skill, is time and money and respect for the people who will use a computing application. It takes time to find out what people need. It takes time to engineer graceful interfaces between a computer system and the people who use it. It takes time to produce documents which can be clearly understood by average or novice users of a new system. The scarcest resource in the computing world is time. A common script in the world of systems design emphasizes the role of time:

> *A client group wanted its system last month. The system is sold as "cost-effective" and must be cheap. A system promoter is optimistic and feels his skilled staff can produce a new system in months instead of years. The resulting schedule has little to do with the substantive complexity of the work to be done. Rather, it is tied to the schedules of related projects with which a system will interface, leasing contracts, and budget cycles. The resulting system is either poorly designed, or late.*

Whatever the variety of related causes, computing systems all too often are developed under tremendous time and money pressures. If a system is not "up," it can't be used at all. Pressures to develop a rapid design and "get something up" mitigate against careful attention to needs of the people who will use it. Caring takes too much time. Occasionally, policies of pricing computing services are tried as a way of giving users more say in what they get. Computing budgets may be given to each department rather than treating computing costs as overhead. If, however, the computing staff has a *de-facto* monopoly on the provision of service, then pricing alone will have little effect on service quality. More deeply, if the available computing service providers share common norms, then there may be little to choose between.

Policies to Foster User-centered Designs

The preceding discussion outlines key features of organizations that we believe constrain the system designs that emerge from a computing group. Existing design methods emphasize either technical features or specific structural arrangements, *e.g.*, user participation. Theoretical features in this account of organizational settings such as salience of consequences were used to make sense of the social dynamics underlying different approaches to design. Some elements of the framework developed here have been investigated in certain studies, but much remains to be done. "Management support" and "resources" in particular have been neglected as either experimental or control variables in the studies performed to date. These gaps are important to fill in developing sound theories of the impacts of organizations on computer technology and the impacts of computer technology on organizations.

Such theory development will be relatively slow. Managers, designers, and computer users need to make decisions on system development

policies in the interim. The policies listed below are intended to foster less-haphazard design settings. They take a great deal for granted. They assume that staffs are skilled and stable, and that the systems being designed are well conceived.

Designer's values

1. Allow a designer/analyst to spend ample time with users in their milieu to appreciate their needs.
2. If a consultant group is different from the design group, rotate designers through the consultant group.
3. Check with users of systems designed by a new designer when hiring new staff.
4. Recruit analysts from the computer-using group.
5. Select analysts who have some training in the social sciences. Training in the ethnographic traditions of sociology which emphasize understanding the "logic" of different subcultures is much more helpful here than the psychological traditions which emphasize the individual in social isolation and treat him as a cluster of parameters.

Salience of consequences

1. Survey user perceptions [17]: Such a survey may range from appraisals of specific features of existing systems to perceptions of and suggestions for proposed systems. It is important that the users surveyed see that their responses actually affect the systems they are using. Some complaints and suggestions may be captured as part of the data entry form.
2. Place representative users in the design group, or at least educate key users on the way a system works as well as what it does [27].
3. If a consultant group is different from the design group, have the consultants maintain a log of all complaints *and satisfactions* expressed by users. Pass this log *regularly* on to the design group.
4. Hold system critique sessions which include both designers and representative users [18, 24].

Salience of responsibility

1. Have designers place their names on systems manuals rather than simply listing group or corporate authorship.
2. Provide an ombudsman for organizational clients who have trouble using a system and have no easy way of identifying where their troubles could be resolved.

Management support

1. Include user satisfaction in the performance reviews of system designers. This is simplified if such data is regularly solicited by some of the techniques proposed to increase the consequences of alternative designs.
2. Indicate to user groups that their active involvement in system design is critical if they are to receive programs that are genuine aids.

Resources

1. When estimating costs for a new system, include the cost of implementing any of the proposals suggested above.
2. The quality of the person/machine interface should appear explicitly in the specifications for a new system. Be sure that time and cost estimates include such features.
3. Estimate costs on a life-cycle basis that includes the period from development through replacement by a new system.

Conclusions

Designing "for users" has become a commonplace slogan among software designers during the last few years. A variety of design methods to enhance the "user-centeredness" of systems have been proposed by the *Stanley House Report*, Hedberg, Lucas, and Mumford, but they have not been developed in a theoretically satisfying way.

This article develops a framework to analyze the conditions under which a software design group is likely to produce user-centered systems. The framework emphasizes structural features of the organization in which design takes place as well as personal characteristics of the designers. The production of user-centered systems according to this perspective need not be a haphazard event which is dependent of the idiosyncratic play of the particular people who develop and use each system. One can predict the likelihood that user-centered designs will be developed on a *routine* basis by understanding the organizational setting in which design takes place. The framework was used to develop a set of policies that are likely to increase the frequency with which a design group will produce software that meets the needs of its various users. These policies include both traditional approaches such as "involving users" in the design of software, and also a wider array of policies to promote sharpened responsibility and clear perceptions about the impacts of alternative software designs.

References

[1] Argyris, Chris. "Management Information Systems: The Challenge to Rationality and Emotionality," *Management Science*, Vol. 17, No. 6, 1971.

[2] Aron, J. D. "Estimating Resources for Large Programming Systems" in *Software Engineering Techniques*, ed. by J. N. Buxton and B. Randell, NATO Science Committee, Brussels, Belgium, 1970.

[3] Coles, Stephen. *Techniques for Information Retrieval Using an Inferential Question-Answering System with Natural Language Input*, Artificial Intelligence Center Tech. Note 74, Stanford Research Institute, Menlo Park, CA, 1972.

[4] Danziger, James. "The Skill Bureaucracy and Intra-organizational Control: The Case of the Data Processing Unit," Working Paper, Public Policy Research Organization, University of California-Irvine, Irvine, CA, 1977.

[5] French, Nancy. "DP Deprives Workers of Job Satisfaction, Europe Studies Show," *Computerworld*, Vol. 11, No. 34, August 22, 1977, pp. 1, 4.

[6] Gottlieb, Abe. "Management Information Systems, Public Policy and Social Change," *AFIPS Spring Joint Computer Conference*, Vol. 32, 1972, pp. 507-510.

[7] Guthrie, Arthur. *A Survey of Canadian Middle Managers' Attitudes Towards Management Information Systems*, Carlton University, Ottawa, Ontario, 1972.

[8] Hedberg, Bo. "The Design and Impact of a Real-Time Computer System," London School of Economics, unpublished manuscript, February 1976.

[9] Hedberg, Bo and Mumford, Enid. "The Design of Computer Systems: Man's vision of man as an integral part of the system design process," in *Human Choice and Computers*, E. Mumford and H. Sackman (ed.), American Elsevier, New York, 1975.

[10] Keen, Peter and Gerson, Elihu. "The Politics of Software Systems Design," *Datamation*, Volume 23, Number 11, November 1977, pp. 80-84.

[11] Kling, Rob. "Towards a Person-Centered Computer Technology," *Proceedings of 1973 ACM National Conference: Computers in the Service of Man*, Association for Computing Machinery, New York, 1973.

[12] Kling, Rob. "Computers and Social Power," *Computers and Society*, Vol. 5, No. 4, Fall 1974, pp. 6-11.

[13] Kling, Rob. "Automated Welfare Client Tracking and Service Integration: The Case of Riverville," *Communications of the ACM* (to appear in future issue).

[14] Kling, Rob and Gerson, Elihu. "The Social Dynamics of Technical Change in the Computer World," *Symbolic Interaction*, Volume 1, Number 1, December 1977.

[15] Knuth, D. *The Art of Computer Programming, Vol. 3, Sorting and Searching*, Addison-Wesley, 1973.

[16] Kraemer, K. and Dutton, W., "Technology and Urban Management: The Power Payoffs of Computing," *Administration and Society*, Volume 9, Number 3, November 1977, pp. 305-340.

[17] Lucas, Henry. "A User-Oriented Approach to Systems Design," *Proceedings 1971 National ACM Conference*, Association for Computing Machinery, 1971.

[18] Lucas, Henry. *Toward Creative Systems Design*, Columbia University Press, New York, 1974.

[19] Lucas, Henry. *Why Information Systems Fail*, Columbia University Press, New York, 1975.

[20] Martin, James. *Design of Man-Computer Dialogues*, Prentice Hall, Inc., 1973.

[21] Meister, David and Farr, David. "The Utilization of Human Factors Information," System Effectiveness Laboratory, Bunker-Ramo Corp., Canoga Park, CA, 1966.

[22] Meyer, Marshall W. "Automation and Bureaucratic Structure," *American Journal of Sociology*, LXXIV, 1969, pp. 256-265.

[23] Mumford, Enid. "Job Satisfaction: A Major Objective for the System Design Process," *Management Informatics*, Vol. 2, No. 4, 1973, pp. 191-202.

[24] Mumford, Enid. *Job Satisfaction: A Study of Computing Specialists*, Longman Ltd., London, 1972.

[25] Mumford, Enid and Sackman, Harold (ed.). *Human Choice and Computers*, American Elsevier, New York, 1975.

[26] Orlicky, Joseph. *The Successful Compter System*, McGraw-Hill Book Co., New York, 1969.

[27] Qualitz, Joseph. "The Effects of Time-sharing on the Jet-Vac Corporation: A Case Study," Unpublished manuscript, M.I.T., May 1970.
[28] Rudwick, Bernard. *Systems Analysis for Effective Planning: Principles and Cases*, John Wiley and Sons, Inc., 1969.
[29] Salton, Gerard. "What is Computer Science?" *Journal of the ACM*, Vol. 30, No. 1, January 1972, pp. 1-2.
[30] Schwartz, Shalom. "Elicitation of Moral Obligation and Self-Sacrificing Behavior: An Experimental Study of Volunteering to be a Bone Marrow Donor," Unpublished manuscript, Department of Sociology, University of Wisconsin, Madison, Wisconsin, 1970.
[31] Schwartz, Shalom and Clausen, Geraldine. "Responsibility, Norm and Helping in an Emergency," *Journal of Personality and Social Psychology*, Vol. 16, 1970, pp. 299-310.
[32] Shepard, Jon. *Automation and Alienation*, M.I.T. Press, Cambridge, Mass., 1970.
[33] Simon, H. A. "Applying Information Technology to Organizational Design," *Public Administration Review*, Vol. 33, No. 3, June 1973, pp. 268-278.
[34] Sterling, Theodor. "Guidelines for Humanizing Computerized Information Systems," *Communications of the ACM*, Vol. 17, No. 11, November 1974, pp. 609-613.
[35] Ullman, Joseph and Huber, George. *The Local Job Bank Program*, D. C. Heath, Lexington, Mass., 1973.
[36] The URBIS Group. *Computers, Bureaucrats, and Politicians: High Technology in American Local Governments*, Public Policy Research Organization, University of California, Irvine, CA, 1977.
[37] Weinberg, Gerald. "The Psychology of Improved Computer Programming," *Datamation*, November 1972.
[38] Withington, R. *The Real Computer*, Addison-Wesley, Reading, Mass., 1969.

About the Author

Rob Kling is an Assistant Professor in the Department of Information and Computer Sciences and a Research Computer Scientist in the Public Policy Research Organization at the University of California, Irvine. He received his graduate degrees from Stanford University. He is co-principal investigator of the URBIS (Evaluation of Urban Information Systems) project at the University of California, Irvine. His recent publications report on urban information systems, the politics of computing, the social impacts of electronics funds transfer systems, and the social organization of the computer world.