UC Irvine

ICS Technical Reports

Title

The social dynamics of instrumental computer use

Permalink

<https://escholarship.org/uc/item/1v39d1f0>

Authors

Kling, Rob
Crabtree, Phillip
Scacchi, Walter

Publication Date

1977-10-17

Peer reviewed

THE SOCIAL DYNAMICS OF
INSTRUMENTAL COMPUTER USE

by

Rob Kling, Walter Scacchi
and Phillip Crabtree

Technical Report #110

Department of Information and Computer Science
and
Public Policy Research Organization
University of California, Irvine

October 17, 1977

Abstract

During the last three decades, computing has far surpassed its early role as a laboratory device for scientific computation. Computers are presently used for a wide array of purposes. In most of its uses it is portrayed as a problem-solving tool and as a material or intellectual object. Despite continuing technical advances, computer use is still costly in its demands for attention and special skills by people (instrumental users) who try to use it to further their own work, whether they program or not. These problems occur because computing is also a social object. In particular, serious and continual use of computing forces users to attend to issues associated with:

1. The work setting of computer use;
2. Understanding the capabilities of computing;
3. The scope and rate of technical change;
4. Control over computing resources;
5. The overall time that attention to these social and technical issues require.

The opportunities and problems of instrumental computer use vary when users utilize different technologies and different organizational arrangements for supporting them. However, as software and hardware developments progress, the social problems of computer use will increasingly dominate the attention of users.

Acknowledgements:

Harold Sackman provided helpful comments on an earlier draft of this paper.

Introduction

Computer Technology is usually spoken of as a tool -- a helpful device used to ease the burdens and expand the flexibility of information processing. In this narrow sense, computer technologies have in fact increased the capabilities of people and organizations to carry out complex calculations, manipulate large sets of data, and access sets of data from geographically remote locations.

Many users have found, however, that these rich new opportunities generate a correspondingly large number of changes in their customary work patterns. People who use computer systems for a variety of daily tasks (e.g., recordkeeping, data analysis, automated design) must, for example, adjust to changes in computer systems, get adequate priority for their computing jobs and find and keep skilled programming staff. As a result, the very technology which was supposed to be an unobtrusive aid and timesaver can become very attention demanding and a source of continual low level conflicts.

Easing these problems has been a traditional concern of computer scientists, and a number of possible solutions have been suggested and tested. Most of these solutions, however, have assumed that computing can be seen as a fairly straightforward dialogue between a hypothetical "user" and a machine. Focus may rest on one party or another. Thus, "hardware-based" solutions (in which focus rests on expanding the flexibility and reliability of the "machines") emphasize components such as

peripheral devices, distributed computing, microprocessing, operating systems protection schemes, or computer graphics; "software-based" solutions (in which focus rests on easing the cognitive burdens of the "user") might include new programming languages or more "natural" interfaces.

These solutions, however, reduce only a selected portion of the burdens faced by computer users. For example, they do not directly address the problematic dependence of many users upon an increasingly specialized array of staff who are sophisticated in different aspects of computing. Nor do they help people who wish to use computing as a unobtrusive tool to substantially diminish the demands for attention that computer use places upon them. (Moreover, these problems may well increase with the increasing technical sophistication of computing.) While social problems of computer use are quite common, they are also poorly understood.

Based on our studies of computer use in a variety of settings [15-20], we have found that to adequately address these problems, it is necessary to expand the traditional view of computing described. Computing is not merely a "tool" or material object. Neither can its use be described as a straightforward dialogue nor computing treated solely as an intellectual object. Rather, a user of computing becomes involved in a much larger sphere of activity in order to effectively utilize the technology. This "computing world" [21] includes a combination of machines, understandings about what they are good for and how they can be used, appropriate

expectation of what constitutes "good performance," use of specialized language, the local computing milieu, and a variety of vendors, consultants, professional groups, etc. For users, computing is also a social object.

Since many of the problems experienced by computer users develop from their relationships with the "computing world," analyzing computing as a social object helps us understand the nature and source of many common problems of computer use. These problems may vary with the particular technology in use (e.g., on-line vs. batch systems) or may stem from issues external to the particular computer system in use, such as the organizational arrangements through which computing is provided. Each of these problems has associated costs.* These costs are also poorly understood and have yet to appear in the figures cited for the total systems costs [3,34].

The primary thrust of this paper is to identify the recurrent aspects of the social world of computer users which are the sources of these problems. First, we will expand our conception of computing as a social object which is a potentially problematic "tool." We will then explore some more specific

* This situation is similar to that faced by software engineers prior to the CCIP '85 study [3, 22]. The software study in CCIP'85 summarized the results of several case studies on the relative costs of software and hardware development. Prior to the publication of CCIP'85, it was known that software development costs were increasing and that they were becoming comparable to hardware costs. However, the CCIP'85 study showed that in several large systems that the software costs over the system's life cycle exceeded the hardware costs by more than two to one.

consequences that this expansion reveals. We would caution that while we list a set of issues which are problematic for computer users and computer specialists, and advance some hypotheses as to their relative and absolute costs and importance, we intend this discussion to be an introduction to the bundle of issues which warrant further investigation, articulation, and conceptualization.

Computing as a Social Object

Many people who use computing hope it will help them be more effective in their work, such as financial analysis or aircraft design. The substance of their work may have little connection with computing: it is a means to further some other end. For many such people, "instrumental users," computing is not simply a "tool" or "problem solver," [29,31] but also a source of frustration and conflict, a "problem generator."

First, instrumental users must learn about the capabilities of computers and general procedures for their use. Such learning itself may be a problem: it involves an investment in time, an acquaintance with a complex set of new concepts, a sometimes forbidding jargon, and the ability or luck to find reliable sources of information. Secondly, unless the user has programming skills and the time to use them, he will probably turn to a programmer or possibly a "systems analyst" to help define and develop (or select) a computing application suitable for his purposes. If the user shares the computer with other

groups and has it managed through some other organization (e.g. service bureau) or organizational unit (e.g. computer center), then he must learn the procedures of dealing with that organization.

In addition, details of the technology continually change. Major computing facilities seem to change machines every four-six years, operating systems about every two years, and associated supporting software (file handlers, job accounting procedures, etc.) more frequently.*

While most of these changes improve the technical quality of computing, they also demand continual attention (a social resource), and thus time on the part of users. Continual demands for attention can only be managed through satisficing or delegating work to others [30]. Users may also require growing expertise to help select new computing assistants and to deal with the on-going technical changes common in computer services organizations. All of these changes entail a renegotiation with specialists and with a technology which, for many, is somewhat foreign and puzzling.

* For example, in 1976, the UCI Computing Facility introduced 106 sets of notable software changes on one of their large computer systems. These included changing packages, replacing a timesharing monitor, adding packages, deleting facilities, and altering the mode of access to various programs. Some of these changes (e.g. bug fixes) were transparent to most users; many users were probably effected by different clusters of 6-10 of these alterations; some users were even more influenced.

Up to this point we have described a unitary "user". Further problems may arise when the local computing milieu is composed of several complex organizational units. In many actual situations in which computing is used (e.g., analyzing land-use patterns in a local government agency, preparing grant proposals in a welfare agency, or estimating the returns on investments in a stock portfolio) there may be many different people who are interested in the outcomes of a computer-based analysis and in the teams of programmers who develop and maintain the requisite software [6,13,18,20,23].

Issues in Instrumental Computer Use

In previous sections, we indicated why computing must be treated as a "social object" rather than just a computational task or device. In this section we conceptualize the consequences for instrumental users through a set of specific issues.

Computing services are produced and consumed in work settings in which the participants take on specialized roles. Some actors, such as instrumental users, usually depend on others (e.g. computer specialists), to develop, maintain, and modify their applications to better serve their needs. The demands that instrumental users and computer specialists make upon each other and of the technology hinge, in part, on their understandings of the appropriate role, capabilities and limitations of computing. Application development changes procedures and processes for users at various intervals which may be either relatively benign

or disruptive. Part of the interaction involves establishing and maintaining control over the various computing resources within the organization. Both specialists and users are dependent upon the current state of software development technology to help construct reliable programs which operate with a maximum of grace and ease. Similarly, computing creates special demands for the time and attention of users. The social aspects of computer work and computer use play a large role in shaping each computing milieu as does the particular technology in use. In the following section, we introduce the array of relevant issues which we have clustered under the categories underlined in this paragraph.

The Work Setting of Computer Use

1. The concepts users and computer specialists have of their own work and the role of computing in it.

Specialists and users have different concepts of how central computing is, should, and could be to the successful performance of their jobs. To specialists computing is central and is the *raison d'etre* of their work. Instrumental users often work on jobs in which computing is less central. For example, some urban planners see their jobs as providing analysis of social and economic data to support new urban development; computers are simply the means by which they produce these analyses. These differences of focus have substantial repercussions for the amount of effort people of each orientation are willing to spend

learning and adapting to new computer system developments.

2. The mutual perceptions of computer specialists and users.

Computer specialists can influence the involvement of users in the computing process. Shared perceptions may be important to the specialist in determining how users should be educated or to what extent users should be involved in the design and implementation of particular systems.

3. The impact of differing responsibilities among computer specialists.

Often, as a department grows and expands, the tasks of individuals within the department may become more narrowly defined and specialized [2]. In a computing department, this division of labor may create problems of coordination. For example, a user may find that to change an inquiry program he is currently using, he must coordinate his efforts with those of a programmer, a systems analyst, a database manager, and the teleprocessing specialist. Increasing the technical sophistication of a system often leads users to interactions with more specialists.

4. Doing a "good job" and being rewarded for it.

People often differ on which aspects of a job are important for satisfactory performance. Some programmers emphasize satisfying users' demands while other programmers emphasize elegant code.

Despite these individual interpretations of what constitutes doing a "good job", the organization may impose a reward system on specialists which emphasizes different activities [15]. The rewards may be for meeting schedules, for the number of coding lines produced, or for being to work on time. Whatever reward system exists in an organization for computing specialists, it may conflict with what specialists perceive to be important measures of job performance.

5. Maintaining some career mobility.

Specialists appear no different than any other employees in being concerned about their job security. Specialists may feel that a strong position in the marketplace must rest on experience with the latest technological innovations. Consequently, specialists may influence their organization to continually acquire state-of-the-art hardware and software packages [21].

Understanding the Capabilities of Computing

1. The ways people learn about computing - what computers are good for, how their particular machine might be used, etc.

Beliefs about the appropriate role and capabilities of computing vary considerably. Those who work closely with the technology often view computing as a special purpose device which is best suited for applications something like their own. Thus accountants often view computers as "accounting engines" while urban planners may view them as statistical calculators.

Coupled with beliefs about appropriate tasks for automation are beliefs about the ease of applying computing. Computer specialists often view the technology as speedy and convenient. Programmers, like planners, designers, and managers often underestimate the time required to develop and implement new projects.

Instrumental users learn about computing in many different settings: in formal classrooms, "on the job," with self-instruction materials, through the news media and advertising, at meetings of professional organizations and associations or through informal conversations. Since the "best" sources of information are often costly to find and use, most computer users rely on diffuse sources of information.

2. Getting a computational task successfully completed

When a problem is determined to be solvable with an existing computing system, users may find themselves facing a procrustean software system [29]. Rigid system designs add to the complexity which users must overcome to compute a solution to their problem. In theory, computing may be both technically and organizationally complex. In fact, it is also complicated*. Most software packages, however simple or complex, usually have idiosyncratic conventions** which arise from problems in implementation, compatibility with odd features of related systems or simply through "poor" design.

Nevertheless, an instrumental user must master and remember these conventions in order to utilize a software system.

In addition, the elapsed time to complete a computational task, from the point of view of an instrumental user, begins when the task is conceived and ends when the computed job is translated into a usable form. This time frame is larger than that of the computer specialist who counts from the time that a task is well specified until a product is delivered to the user.

* Complexity refers to substantive logic-mathematical interrelations and difficulties; complications can arise in almost any arrangement of facts, concepts and thoughts. Complication is an undesirable characteristic of any construct; complexity may be an inherent feature [6].

** For example, program runs may begin with an incantation such as //JOB=. Variables may be restricted to six alphanumeric characters and must begin with a letter. Or once a file is processed, it may not be reprocessed until a special routine is executed. Most computer users learn to use the technology despite dozens of similarly idiosyncratic conventions. However, they add undue complication to a complex technology.

And it is still longer than the "time to complete a computational job" as viewed by the computer operators. This usually exceeds the time to complete a job once it is being executed by a digital computer. Despite these "obvious" observations, the speed of computer use (and its time demands) are usually conceptualized in time frames closer to those of machine execution than to those of instrumental users.

3. The ways people deal with computing/systems jargon.

Technical jargon provides a concise means to express complex concepts to those people familiar with them. It is common for programmers to converse with non-technical clients in the technical terms with which they carry out their work. Computing jargon may be problematic for non-specialists attempting to utilize the technology. This practice is particularly troublesome when the jargon includes terms which are idiosyncratic to a particular vendor's equipment or to a specific application. The barriers created by the language may divert potential users and discourage current users from asking questions or exploring new uses of the technology.

Characteristics of Change

systems and processors with relative impunity and few users will be inconvenienced.

On the other hand, certain users often seek specific changes in both applications and support software. However, in shared systems, changes developed for one party are typically imposed upon all users of the same computational resources. Many technical changes that benefit one party may benefit others as well. However, there are also common conflicts between the technical needs of different users. It is an empirically open question how frequently technical changes are either "pareto optimal" or indicate a redistribution of computational resources. Thus, the advocacy and implementation of changes has a strong political content above and beyond the resources required to implement the change.

3. Attitudes toward Change.

A system may be altered for a variety of reasons independently of its value, indifference or inconvenience for particular users. Users will often resist changes to a system which demand unexpected resources (e.g. time, status, or money) without compensating returns in easing the burden of their work. On the other hand, system users will look favorably toward system changes in which they actively participated in specifying. Attitudes toward system changes are also effected by the perceived ease of affecting change given organizational or technical constraints (e.g. system reliability). While many changes may benefit all users in "the long run," both users and

computer-based systems in which users have easy access to expert assistance are better accepted than those in which access is more difficult [24].

3. Controlling the kinds of demands made by users.

Whenever a service is provided for users, it creates different patterns of demands. To the computer specialist, certain user demands may be expected and appear reasonable, while other demands may appear excessive, unreasonable, or uninteresting.

Computer specialists can establish certain procedures for regulating their interactions with users. These may range from the formalized rules established within the organization to the informal rules that might be implied by the specialists' behavior. In either case, specialists develop strategies for managing the behavior of their clients to help serve their own end and make their organizational life tractable.

4. The "values" sought after by those individuals and organizations which promote applications development.

Often computing systems are developed and installed when a specific person or small group of individuals actively promotes computing within an organization [20,23]. Since new computer applications are usually costly, promoters who want to resources allocated to their project must often first obtain sanctions from other organizational members. Since different actors become involved in acquiring computing resources, computing will often

serve many ends. For example, some actors may be seeking to enhance their administrative control, others seeking to cut costs, still others may be seeking to make their jobs easier or more interesting.

5. Political support within the organization.

Politics deals with the allocation of goods, services, and values. The distribution of computing is often the focus of conflicts over budgets, staff, and domain. This is not incidental, but rather an intrinsic aspect of computer use. To the extent that computing resources are valued by different actors in an organization, they will seek access to them. The resulting contention with its commonplace conflicts, bargaining, and subterfuges is similar to other kinds of organizational politics.

Also, some actors seek control over computing resources simply because it provides a relatively large number of growing staff and consequently a growing budget. There is also some evidence that overall computing arrangements can be more strongly influenced by the political access of key actors than by the technical soundness of their preferences [27].

Software Development Technology

1. Programming and Design Practices.

Since the development of software has become such a major part of the expense of computing for most organizations, considerable attention has been focused on improving software design and programming productivity. New techniques [11] and tools have been developed to assist specialists with their various tasks. "Structured" programming [9,10,35] is currently emphasized to aid specialists, as well as "Chief Programming Teams" [1] and automated design aids [4,8].

While these techniques and aids may be beneficial for the organization, they may be problematic and disruptive for specialists. They may actually make the specialists' jobs more difficult and attention demanding. They may create changes which are frustrating for specialists accustomed to previously established procedures.

2. Program testing and maintenance.

In theory, one would like to be able to automatically generate a sufficient set of test data necessary to demonstrate the probable correctness of a robust class of programs. However, for programs of even moderate complexity the set of test data to exercise all paths through a program is infeasibly large [14]. Nevertheless, some promising research is proceeding on various schemes to automate tests for special program conditions [7,28]. In contrast to the research on new tools for program testing, the

state of current practice does not rely upon much automation at all. Test data, for example, is usually selected manually by a programmer or a knowledgeable user.

In most mature installations, the fraction of the budget devoted to maintaining existing programs exceeds that which is devoted to developing new applications [3,5].

Attention

1. The kinds of attention demanded by computing.

Computing may appear to some users and specialists as a technology that demands they learn a great deal about it to effectively utilize it. Many users must (or are at least lead to believe they must) use the computer efficiently because it is a scarce resource. However, the time required by a user to prepare and successfully execute (after "debugging" runs) efficient programs often displaces any net savings in terms of completing the task at hand. That is, concern for minimal computer resource usage versus concern for minimizing the time to complete a work task often lead to conflicting demands for the user's attention.

2. The precision and detail demanded by computing.

As a tool, the computer is a fairly exacting device. It demands that procedures be followed explicitly. It does not allow loose and ad hoc procedures in handling transactions as might exist in a manual or more informal information system.

At times users complain that their jobs are actually more difficult or less interesting with computing than they had been previous to computing. Specialists complain that it takes a special person, like a "hacker," to be truly satisfied with the detail demanded by systems and application programming [33].

These issues do not exhaust those raised by the social nature of computing. But they do represent those social aspects of computing which strongly influence the patterns of computer use adopted by instrumental users. The relative importance of any of these issues is also dependent on the organizational setting where computing occurs.

The Organizational Context of Computer Use

The actual difficulties experienced in using computing depend upon the interplay between both technical and organizational arrangements. Consider, for example, the different impacts of data-base management systems (DBMS) on the time to produce a program for a user in scientific and commercial settings:

Computer specialists may assume that a scientist utilizing a DBMS will either carry out his own programming or employ a skilled research assistant who is under his supervision. This is a result of the work organization of scientific laboratories in which each research team has dedicated research assistants to help carry out a variety of laboratory chores including data collection, reduction, and analysis. If the scientist desires to change schedules or priorities in his use of the DBMS, he normally faces no bottlenecks in the process except the limitation on his own or assistant's time. Since he can regulate these alterations of priority, he is at most, buffered by one queue from access to programming.

A different situation faces the instrumental computer user in a commercial firm. In commercial firms, it is rare for staff to have their own programming assistants (programmers are usually centralized in a pool, even in user departments) and scheduled through a supervisor. The commercial user may thus be further buffered from the access to computing. He may have to negotiate with a supervisor, a special committee, or even a review board to achieve changes in schedules or priorities in dealing with a DBMS. Each of these parties has a separate queue of requests and demands with their attendant delays. Each such queue creates additional delays for the commercial user in gaining access to programming assistance. In practice, a person may wait much longer to get on the queue of a programmer and to get his work done than it takes to do the work.

Even if a DBMS reduces the time required for a programmer to write a given program, the time it takes for users to get a given program depends upon the organizational arrangements used. This example illustrates the way in which the social setting of computer use may influence users more than the technology in use.

The Costs of Negotiation

The social problems of computing may not be salient in every setting in which computing is used extensively. For example, advocates of certain technologies, such as data-base management systems, have stated objectives of making the development of ad hoc analyses easier for instrumental users [25,26]. Also, "turnkey" installation of applications and hardware may reduce the amount of disturbance caused by computing. Similarly, certain organizational arrangements are supposed to reduce difficulties encountered by instrumental users [32].

The issues identified in this paper are representative of those that arise for many instrumental users and computer specialists in their daily encounters with computing. Improving the grace or ease with which computing is used hinges on coming to grips with these issues (and thus with computing as a social object) as much as it depends upon developing new software and new hardware. In addition, a major impact on groups using computing is the increased attention to information processing -- its management and conflicts -- that negotiating these issues demands. People's time, skills and organizational resources are

involved in attending to these negotiations. The negotiation costs, in time and frustration, borne by instrumental users may become a substantial fraction (if not the largest) of the cost of a system during its life cycle.

Conclusion

To understand the dynamics of instrumental computer use, computing must be viewed as a social object. Computing is a problematic technology for many instrumental users, in part, because it raises so many social issues which continually demand attention. To date, only specific issues have been studied in particular computing settings [18,23,24,29]. Unfortunately, there is no work yet which analyzes these bundles of issues in any computing setting.

Organizing computing systems without regard to their impact on the dynamics of a computing milieu is a costly endeavor -- both socially and economically. As technical advances in hardware and software simplify some of the problems faced by computer users, the social problems of computer use will become relatively dominant. Strategies to create easy and effective environments for computer use in the coming decades will have to deal head on with computing as a social object and the attendant social problems that occur during its use.

References

1. Baker, F.T., "Chief Programmer Teams" IBM Systems Jour. 2(1), pp.56-78, 1972.
2. Blau, Peter M. "A Formal Theory of Differentiation in Organizations" Amer. Soc. Rev. 35(4), pp.201-218, April 1970.
3. Boehm, Barry W., "Software and Its Impact: A Quantitative Assessment", Datamation, 19(5), pp.48-59, May 1973.
4. Boehm, Barry W., "Software Design and Structuring," IBM Systems Journal, Vol. 11, No. 1, pp. 56-73, 1972.
5. Boehm, B., McClean, R.K., Vefrig, D.B., "Some Experience with Automated Aids to the Design of Large-Scale Reliable Software", IEEE Transactions on Software Engineering, Vol. SE-1, No. 1, March, 1975.
6. Brewer, Gary Politicians, Bureaucrats, and Consultants Basic Books, New York 1974
7. Burch, John, and Strater, Felix, Information Systems: Theory and Practice, Hamilton Publishing Co., Santa Barbara, CA., 1974.
8. Couger, Daniel, and Knapp, Robert, System Analysis Techniques, John Wiley and Sons, New York, 1974.
9. Dahl, O.J., Dijkstra, E.W., Hoare, C.A.R., Structured Programming, Academic Press, London, 1972.
10. Freeman, Peter, Software Systems Principles, S.R.A., Palo Alto, CA., 1975.
11. Freeman, P. and Wasserman, A. Software Design Techniques IEEE Press Long Beach, Calif., 1976.
12. Goldberg, J., (ed.), Proceedings of a Symposium on the High Cost of Software, Stanford Research Institute Project 3272, pp. 99-119, Sept. 17-19, 1973.
13. Greenberger, M., Matyhew A. Crenson, and Brian Crissey Models in the Policy Process Russell Sage Foundation, New York, 1976
14. Howden, W.H., "A Methodology for the Generation of Program Test Data" Info. and Computer Science Tech. Report 41, Univ. of Calif., Irvine; Irvine, CA, Feb. 1974.

15. Kling, Rob "Towards a Person-centered Computer Technology" Proc. 1973 ACM National Conference, pp.387-391.
16. Kling, Rob "Computers and Social Power" Computers and Society 5(4) pp.6-11, Fall 1974.
17. Kling, Rob "Passing the Digital Buck: Unresolved Social and Technical Issues in Electronic Funds Transfer Systems" Info. and Computer Science Tech. Report 87, (also appeared as PPRO WP-76-13, Public Policy Research Organization), Univ. of Calif., Irvine, Irvine, CA, May 1976
18. Kling, Rob "Automated Welfare Client Tracking and Services Integration: The Case of Riverville" Communications of ACM (forthcoming)
19. Kling, Rob "Information Systems in Policy Making: Computer Technology and Organizational Arrangements" Telecommunications Policy (forthcoming)
20. Kling, Rob "The Organizational Context of User-centered Software Design" MIS Quarterly (forthcoming)
21. Kling, Rob and Elihu Gerson "The Social Dynamics of Technical Change in the Computing World" Symbolic Interaction (forthcoming)
22. Kosy, D.W., "Air Force Command and Control Information Processing in the 1980's: Trends in Software Technology," R-1012-PR, Rand Corp., Santa Monica, CA, 1974.
23. Laudon, Kenneth Computers and Bureaucratic Reform Wiley Interscience New York, 1974
24. Lucas, Henry C. Why Information Systems Fail Columbia University Press, New York 1975
25. Martin, James Computer Data-Base Organization, Prentice-Hall, Inc. Englewood Cliffs, New Jersey, 1975.
26. Nolan, Richard L., "Computer Data Bases: The Future is Now", Harvard Business Review, Sept.-Oct., 1973.
27. Pettigrew, A. The Politics of Organizational Decision-Making Tavistock Press, London, 1976.
28. Ramamoorthy, C.V., Ho, S.F., "Testing Large Software with Automated Software Evaluation Systems", IEEE Trans. Software Engr., Vol. SE-1, No. 1, March, 1975.

29. Sackman, H.A. and Blackwell, F.W., "Studies in Real-World Problem-Solving With and Without Computers" R-1492-NSF, Rand Corp. Santa Monica, CA, May 1974.
30. Simon, H.A., "Designing Organizations for an Information Rich World" in Computers, Communications, and the Public Interest ed. by Martin Greeberger Johns Hopkins Press Baltimore, Md. 1971
31. Streeter, M., The Scientific Process and the Computer, Wiley-Interscience, New York, 1974.
32. Tomaszewski, L.A. "Decentralized Development" Datamation, pp.61-64, Nov. 1972.
33. Weizenbaum, J. "Science and the Compulsive Programmer" in Computer Power and Human Reason, W.H. Freeman and Co., San Francisco, CA, 1975.
34. Wolverton, R. W. "The Cost of Developing Large Scale Software" TRW Software Series TRW-SS-7201, Redondo Beach,