The Web of Computing: Computer Technology as Social Organization

ROB KLING

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

WALT SCACCHI

Department of Computer Science
School of Engineering
University of Southern California
Los Angeles, California

ADVANCES IN COMPUTERS, VOL. 21

ISBN 0-12-012121-2

1. Introduction

Most scholarly and professional examinations of the social and economic repercussions of new computing developments are based on a highly simplified conception of computing and social life. This conception focuses on certain explicit economic, physical, or information processing features of a technology. For the sake of simplicity, the social context in which the technology is developed and used, and the history of participating organizations, are ignored. This conception also assumes rationality on the part of computing developers and users which is neither very short term nor based on narrow interests. Furthermore, it sees the main repercussions of a new technology as direct translations of technical attributes into social attributes (e.g., faster data flows mean faster and better decisions). This conception informs many analyses of computing and related activities: of the payoffs of new computing developments, of the costs to be borne by developers and the broader public, of the extent to which different management strategies will yield more reliable and broadly workable computer-based systems. We call an analysis of computing which depends on these conceptions a *discrete-entity* analysis, and such a model of computing (activity) a discrete-entity model.

Examples of published analyses which rely upon a discrete-entity model of computing are very common, and they form the conventional logic for understanding the social repercussions of new computing developments. Most of the accounts emphasizing the benefits of new computer technologies rely on discrete-entity models, and criticisms of new technical developments can also be based on these models. For example, some analysts suggest that computerized systems will ensure that decision makers are aided by the selective ability of computers to sift through mounds of data, while other analysts argue that computers will rapidly increase the reams of data that decision makers must contend with. Despite

these substantial differences in the conclusions reached, if the underlying model of computer use isolates it from the actual work practices and organization of labor within which automated data systems are typically developed and used, it is a discrete-entity analysis.

Careful empirical studies of the actual outcomes of using automated technologies such as automated information systems have often found that the expected direct benefits do not materialize easily (Kling, 1978; Laudon, 1974; Colton, 1978; Kraemer and King, 1981a). In special cases, some automated systems are simply dismal failures (Dery, 1977; Brewer, 1973); they are delayed in development or sometimes abandoned late in development, and if they work they are irrelevant or have to be worked around (Comptroller General, 1980). Analysts who use a discrete-entity model explain the occurrence of difficulties in realizing the direct benefits of new computing technologies and the occurrence of unexpected negative side effects or outright failures by attributing blame to a set of discretely identifiable causes: the operating managers were inadequate; the computer used was too small or cumbersome; promised data were not available; users were disinterested; there was insufficient support from top managers; etc.

We believe that when discrete-entity models are applied to analyses of the social consequences of socially complex computer applications and the difficulties encountered in their use, the results are often misleading. This article explains some of the limits of the conventional discrete-entity models and develops a richer family of models—*web models*—which we believe help make better predictions of the outcomes of using socially complex computing developments.

In contrast to the discrete-entity models, which gain simplicity by ignoring the social context of computing developments, web models make explicit the salient connections between a focal technology and its social and political contexts. In this article we indicate that many published analyses of computing developments are based on discrete-entity or web models. Unfortunately, most social analysts of computing (or other high technologies) rarely make their models explicit. We have abstracted the discrete-entity and web models from a wide variety of professional and scholarly studies.

The contrast between discrete-entity and web models can be usefully illustrated by a simple example drawn from another technology: automobiles. Suppose a traffic analyst is asked to predict the effects of building a new freeway connecting an urban center to a rural resort and passing through a suburban ring surrounding the city. If the analyst employs a discrete-entity model, he is likely to treat demand for trips as fixed, view the freeway as a medium for expanding travel capacity, and conclude that the

new freeway will reduce congestion on existing highways. If he uses a web model, he will view the new freeway as part of a larger transportation system which is periodically congested. He is unlikely to simply assume that travel demand is fixed. Based on the history of other highway projects in which demand for short trips rather than total demand is fixed, he will consider the possibility that the new freeway will help satisfy demand for speedy trips. Net travel may increase. He may conclude that the new freeway will increase the volume of net travel between the suburbs and downtown, and between the metropolitan area and the rural resort, but that travel times will not be appreciably decreased during commuter rush hours and peak vacation times. In short, the planner who uses a web model is more likely to see a technical change (or new policy) as embedded in a larger system of activity, as having consequences which depend on peoples' actual behavior, and as taking place in a social world in which the history of related changes may influence the new change.

In this simple example the web model is an open systems model. But web models can be more far-reaching than simply being a version of "the systems approach." A politically oriented web analyst may further ask how the proposed freeway is routed and whose lands are crossed and whose avoided (Caro, 1975). He may also inquire into the interests which support the freeway and those (if any) which oppose it so as to further understand those interests it serves (Cornehls and Taebel, 1977, Chapter 5). Web analysts examine the interaction between people and technologies as part of a larger social and technical mosaic in which the development and use of the focal technology is embedded. In the case of technologies such as automobiles or electronic funds transfer systems, using the technology requires one to negotiate a complex set of institutionalized arrangements as well as deal with the equipment.

By making these models explicit, we hope to shed some light on critical assumptions which inform many published analyses of computing. The two models reveal important *emphases* of these analyses. Like any idealization, they sharpen important differences; but they do not exaggerate them. For example, no one is completely insensitive to the political context in which an organization develops a computerized model. Some analysts assume that the political relationships between an organization which uses computerized models and other groups in its environment will significantly influence the kind of model adopted, the way it is used, and the repercussions of that use (*web emphasis*) (Kling, 1978b); technical details of the model and its computer implementation will be of secondary importance. Other analysts hold that the adoption of a computerized mathematical model to replace manual or intuitive calculations has far greater influence on an organization's behavior with respect to computing

than the political environment of the adopting agency (*discrete-entity emphasis*) (Simon, 1977). We have abstracted these two different emphases into the discrete-entity and web models.

While we have reason to favor web analyses, we believe that discrete-entity models are sometimes useful. Their simplicity makes them tractable. The analyst who employs a discrete-entity model need focus only on selected technical and economic characteristics of a new computing technology or application. In contrast, the analyst who uses a web model must work much harder. He must examine the social and political context in which the technology is developed and used, the going concerns of the organization using it, and the extent to which financial, technical, and staff resources support it. Sometimes these additional data yield important and surprising findings (Kling, 1978b; Danziger *et al.*, 1982), but this is an empirical matter.

Analysts who employ discrete-entity models mistakenly assume that they are universal in their application. Researchers who employ web models usually examine socially complex technologies, but they do not have sharp, simple descriptions of the situations in which they best apply. We shall explain their nature in this article, and we hope the reader will develop some good intuitions about these models. For now, we can say that they certainly apply to situations in which many organizations participate in the development, maintenance, and use of a computerized system. Large computerized systems such as air traffic control, multiservice military command-and-control systems, and multibank electronic funds transfer are all in this class. But so are many smaller systems which are developed for organizations by outside contractors or which are simply developed within the organization through the collaboration of three or more departments.

In this article, we explain the meaning of web models for understanding the dynamics of computing development and use in organizational life. We also examine the relative explanatory power of the discrete-entity model and web model by drawing upon the existing research literature and three case studies. The main question we ask about these models is not, Which is truest? Rather, we ask, What kinds of insights does each model give into the social dynamics of computing development and use? We also inquire about the ways in which each model provides analytical power for making evaluations and predictions. It will become apparent, after we examine these models more carefully in Section 2, that they conceptualize two theoretical extremes of a wide array of intermediate analyses of computing developments. Particular analyses are likely to be based on critical assumptions closest to the axioms of one or the other of these models.

Since web analyses examine the social and economic organization of computing activities, it is helpful to introduce some analytical terms useful for social analyses. Since these concepts are relatively abstract, we believe that a short case study can be helpful for giving the reader concrete illustrations of them. This little case study—of an automated inventory control system in the manufacturing firm Audiola—is introduced in Section 3. The key theoretical concepts are explained informally in Section 4, and are developed more formally in Appendix A. In Section 5, five major propositions that web analysts make about the dynamics of computing development and use are examined and explained by reference to the Audiola case. In Section 6 two additional cases of computer development and use are introduced. Section 7 examines these cases, and Section 8 concludes.

2. Models of Computing: Discrete-Entity and Web

The contrast between the discrete-entity and web models has recently been introduced into the computing literature under a different label: *tool model* versus *package model* (Conery, 1980; Kling and Dutton, 1982; Kling and Scacchi, 1979a,b, 1980; Sterling, 1979). For example, Kling and Scacchi (1979a, p. 108) characterize the tool model in which

> one can safely focus on the device to understand its use and operation. In contrast, the package metaphor describes a technology which is more than a physical device. ... the package includes not only hardware and software, but also a diverse set of skills, organizational units to supply and maintain computer-based services and data, and sets of beliefs about what computing is good for and how it may be used efficaciously. Many of the difficulties users face in exploiting computer-based systems lie in the way in which the technology is embedded in a complex set of social relationships.

In reviewing this new literature which examines the package metaphor, we have found it useful to sharpen this characterization. We have also found it useful to change the labels to "discrete-entity" and "web" to better reflect the differences between the two models. This section defines each of these two models by five properties.

Discrete-entity models of computing assume the following:

(a) A computing resource is best conceptualized as a particular piece of equipment, application, or technique which provides specifiable information processing capabilities. (*i*) Each computing resource has benefits, costs, and skill requirements which are largely identifiable. (*ii*) Computer-based technologies are tools, and are socially neutral.

(b) Role of infrastructure[1]: (*i*) The infrastructures for supporting the focal computing resource and the organizational procedures by which it is organized and sustained are critical elements. (*ii*) Each computer-based service is provided through a set of structured computing resources and organized infrastructure. Deploying, managing, and setting procedures for these infrastructural resources is separable from deployment of the focal computer-based technology. Infrastructure, either technical or administrative, is a neutral resource. (*iii*) "Human factors" must be taken into account to ensure that people are well trained and motivated to do what is required. But "human factors" are "organizational problems"—which are separable from "technical problems."

(c) Control over infrastructure: Organizations have ample resources to support all of their computing developments and uses simultaneously. Elements of infrastructure are necessary for making the equipment or technique available to developers or users, and they can be counted on to be of adequate quality and available as necessary.

(d) The focal computing resource and any element of infrastructure can be analyzed[2] independently of: (*i*) its interactions with other computing resources; (*ii*) the social or organizational arrangements within which computer-based services are developed and provided (infrastructure and macrostructures).

(e) The formal goals of an organization are a good guide to telling what it does. The formal procedures of an organization provide a good guide to the way things are done. The formal features of a computer system are a good guide to suggesting what it can do and how it is used.

In contrast, *web models* of computing assume the following:

(a) A computer system is best conceptualized as an ensemble of equipment, applications, and techniques with identifiable information processing capabilities. (*i*) Each computing resource has benefits, costs, and skill requirements which are only partially identifiable. (*ii*) In addition to their functional capabilities as an information processing tool, computer-

[1] *Infrastructure* refers to those resources which help support the provision of a given service or product. The infrastructure for providing computer-based services includes resources such as skilled staff and good operations procedures, as well as physical systems such as reliable "clean" electrical energy and low-noise communication lines. See Section 4.2 and Appendix A for more extensive explanations.

[2] These examinations may investigate the benefits derived from alternative technical or organizational arrangements, or the payoffs of some reform to improve some infrastructural element.

based technologies are also social objects which may be highly charged with meaning.

(b) Role of infrastructure: (*i*) The infrastructure for supporting the focal computing resource and the organizational procedures by which it is organized and sustained are critical elements. (*ii*) Each computer-based service is provided through a set of structured computing resources and organized infrastructure. If this organization of essential resources is large and complex, computer-based systems are a form of social organization. Like any organization or institution, it is not necessarily neutral. (*iii*) There is no "human factor" which is specially separable from the delivery of computer-based information services. Much of the development and many of the routine operations of computer-based technologies hinge on many human judgments and actions carried out within complex, organized social settings.

(c) Control over infrastructure: (*i*) Organizations have limited resources to invest in any capital development such as computing. Not all necessary infrastructural resources are available (in adequate quality) as needed. (*ii*) Computer-using organizations rarely have complete administrative or political control over all their requisite infrastructure. Infrastructural resources may be spread across several organizational units or nominally independent organizations. Infrastructural resources also include sources of expertise which lie outside these organizations (e.g., professional associations).

(d) The information processing leverage provided by a focal computing resource, and its other costs and benefits, social and economic, are contingent upon: (*i*) its interactions with other computing resources; (*ii*) the social or organizational arrangements within which computer-based services are developed and provided (infrastructure and macrostructures).

(e) The formal goals of an organization are a fair-to-poor guide for learning what it does. The formal procedures of an organization provide a fair-to-poor guide to the way things are done. The formal features of a computer system are a fair-to-poor guide to what it can do and how it is used.

The discrete-entity and web models are both ideal types which have been abstracted from published analyses of computing developments. Illustrative examples of studies which rely upon these models are indicated at the end of this section; additional studies which rely upon web models are described or referenced in Section 5. Discrete-entity and web models rely upon a complex set of assumptions. Specific analyses will differ in how strongly they depend upon each assumption of a model. For example, many technical computing analyses investigate the interaction between technical subsystems [e.g., the demands a data base management system (DBMS) makes on an operating system]. Aside from these interactions, the remainder of the environment of computer use many be largely ignored. Such a perspective is much closer to a discrete-entity model than to the web model.

Discrete-entity analysts employ two logics of technical develoment: direct substitution and incremental aggregation. In the case of computer-based services, organizations can improve their information processing capabilities by direct substitution of a less capable technology with a better one. An old box (or set of people) exits and a better box replaces it: exit the horse and enter the horseless carriage. The second logic, incremental aggregation, accounts for the accumulation of new capabilities: the capabilities of the old box are enhanced by adding on new boxes. The addition of DBMS is one example. The "boxes" also may be managerial techniques such as cost–benefit analyses or project scheduling disciplines. In addition to these logics of technical development, web analysts employ two principles (e.g., fitting and packaging) which we will examine in Section 5.

The basic unit of analysis of the discrete-entity model is a computing resource (CR), such as an IBM 370/158, a computer terminal, or a specific application program. Systems are assumed to be loosely aggregated collections of equipment, people, organizational procedures, beliefs, etc., which may be easily broken down into relatively independent elements. The analytical virtue of this approach is its simplicity. CRs can be evaluated independently and additively. When an analyst uses a discrete-entity model to understand the computing capabilities of an organization he usually begins by asking, "What kind of equipment and facilities do they have?"

In contrast, analysts using a web model begin by asking: "What kinds of things do people do here?" While some web analysts may focus on a formal task system, most are concerned with the array of activities that people actually engage in while pursuing some task (Kling, 1982). For example, the line of work of an urban planner who examines the costs to a city of adding a new shopping center extends far beyond his making the appropriate numerical calculations. It also includes defining the problem with developers and with other city staff, finding relevant data which may be scattered in the records of various departments, etc. If this analyst uses a computerized cost-impact system to make his calculations, he may have to negotiate with the computing staff for computing time, decipher several documents which explain how to use the model and the file systems on the computer, and so forth.

These differences in perspective have tremendous significance for the

sense one makes of the dynamic aspects of computing developments in organizations. The discrete-entity analyst usually expects CRs to be used in a way which is consistent with their formal characteristics.

As noted above, the main intellectual advantage of the discrete-entity model is its analytical simplicity. This attractive advantage has two drawbacks:

(i) Predictions based on this model tend to overestimate the economic value or "success" of some computational resource or technique, and also to underestimate the requisite time and costs. These often vary with context in both their magnitude and certainty; discrete-entity models minimize context and uncertainty.

(ii) The main conceptual elements of the model constrain attention to the formal or manifest rationality attributed to a computer-based system and deflect attention from its social or political roles. Sometimes these latter roles are of greater significance than the former (Kling, 1978b).

The balance of virtues and difficulties of the web model is quite the reverse. Its primary virtue is empirical fidelity; it is organized to better match the social relations which influence the development and use of computerized technologies in complex organizations. Its primary difficulty is analytical cumbersomeness.

As developed above, both of these models are *underspecified*. First, our parlance "model of computing" is a shorthand for encapsulating many different activities which take place in and around computing: the adoption of innovations, estimating their costs and payoffs, evaluating which interests are served and how, explaining how they are integrated into the ongoing life of the social settings in which they are introduced, etc. Each one of these activities will require a different model, much in the way that physicists who rely upon classical electrodynamics still have different models for the diffraction of light in a prism and the propagation of microwaves in a circular tube. Second, web models and discrete-entity models usually make different demands for information upon the analyst. To specify a web model adequately, one must provide many more relationships and data. Discrete-entity analyses, on the other hand, are almost always difficult to amplify because they neglect the supporting infrastructure and social contexts. The web model's greater empirical accuracy represents a research cost in that this greater variety of data is needed to specify critical relationships.

Some readers may wonder whether we have stacked the deck by oversimplifying the discrete-entity model and unduly enriching the web models. We believe not. Not all web analyses are accurate or useful.

Rather, when computer-based systems and their services are socially complex, requiring the close cooperation of several diverse groups, web models are likely to provide greater analytical insight than discrete-entity models. While one is unlikely to find discrete-entity analysts who completely neglect social context, history, and infrastructure, or web analyses which minutely investigate all aspects of social context, history, and infrastructure, analyses that lean toward either extreme are easy enough to identify. For pedagogical purposes, however, we will use discrete-entity models as a foil against which to examine web models. We think this is appropriate, since discrete-entity models are well known to the reader and they frame most of the public discourse about the nature of computer-based developments.

Exemplary discrete-entity analyses may be found in research monographs (Inbar, 1979; Hiltz and Turoff, 1978), publications for computing practitioners and working managers (Simon, 1973; Kochar, 1979; Wise *et al.*, 1980), and recent MIS textbooks (Burch *et al.*, 1979, Part IV; Gessford, 1980; Kanter, 1977, Chapter 8; Hussain and Hussain, 1981, Chapter 17; Taggert, 1980, Chapter 8). Exemplary web analyses may be found in research publications such as Kling and Scacchi (1979b), King and Kraemer (1981), Markus (1979), Dery (1977), and books aimed at computing practitioners and managers (Keen and Scott-Morton, 1978; Mumford and Pettigrew, 1976; Withington, 1979). We are not aware of any MIS textbooks which are developed primarily from a perspective which emphasizes web models.

Despite our abstracting two coherent models from these analyses, they are framed in several different conceptual languages. These languages add richness and bite by giving the analyst specific concepts with which to characterize the social organization of the computing worlds he accounts for. In Section 4.2, we examine four common conceptual languages employed by social analysts of computing developments.

3. A Pedagogical Case Study: Automated Inventory Control

Certain critical concepts will help us explain how web models specify the dynamics of computing development and use. These concepts are best illustrated by concrete examples. Also, the dynamics of computing according to web models (e.g., the role of historically developed commitments in constraining choices) is best introduced by means of some good examples. The short case study that follows presents a connected set of incidents and patterns to illustrate these points.

3.1 Background of the Audiola Case

This short case study reports the development and use of an inventory control system in two different scenarios. Part I of the case describes an organization facing well-defined problems (e.g., Japanese price competition) and a well-defined technology which can help key staff solve those problems. Payoffs of different lines of action are clear. Moreover, resources such as appropriate equipment and skilled staff are either available as needed or easily obtained (e.g., by rapid purchase or training). This scenario at Audiola can be well described by discrete-entity models. Part I fits a script which can be found in many case studies of computing development and use in organizations,[3] and which has been framed within the conventions of that genre.

Discrete-entity models provide little help in interpreting the more complex and problematic dynamics of continuing computer use and devlopment described in Part II. It is not simply the case that "things went well" in Part I and badly in Part II. Rather, the circumstances that appear in Part II are attributable to a variety of conditions that are ignored in Part I—the relative wealth of the firm, changing demands on the information system, availability of key staff, ease of attracting skilled technical staff, etc. [These elements can also influence the introduction of computing into an organization (Dery, 1977), but have been ignored so that the reader can examine the kinds of dynamics which each model best accounts for.] The differences are substantial and are the focus of this article. (The information required to specify a web model also adds to the length of this section.)

This case is based on a composite of organizations which were investigated by the authors. It is introduced to help illustrate several major themes. In no single organization are all these themes simultaneously salient; therefore we have folded together certain organizational patterns and constraints.[4] This case serves to *illustrate* key concepts of the discrete-entity and web models. Because it is a composite, with an artificially rich set of constraints for any one real organization, it cannot be used to test the empirical adequacy of either discrete-entity or web models. Two empirical cases to be examined in Sections 6 and 7 can be used for testing.

[3] MIS textbooks often use cases such as these for their primary examples. See for example Hussain and Hussain (1981), pp. 436–437, 447–450; Mader and Hagin (1974), pp. 352–357.

[4] In addition, several constraints have been added which we have observed in other organizations, but not in the manufacturing firms we have studied.

3.2 Automated Inventory Control at Audiola

Audiola Corporation manufactures high-quality audio electronic products for consumer and commercial markets. It manufactures 16 products in two distinct product lines, currently grosses $60 million per year, and employs 500 people. The company was founded in the 1950s and developed a good position in the North American audio marketplace in the 1960s, but began to suffer in the face of strong Japanese competition in the early 1970s. At this time, approximately 60% of the firm's costs were in materials; the balance were in labor and overhead.

Part I: The Introduction of Manufacturing Computing Systems

In 1974, to help cut costs in the face of stiff price competition, the senior industrial engineer searched for some strong inventory control procedures. The firm maintained an active inventory of 12,000 parts, and these were difficult to control carefully by manual methods. Audiola was using a Datacrunch 1800/2 minicomputer for financial record keeping, and the senior industrial engineer discovered that Datacrunch offered a sophisticated demand-dependent inventory control system, TRACKER, which promised to help reduce Audiola's inventory carrying costs.

TRACKER was installed in 1975 over a period of four months. It required a complete and careful inventory count as well as several months of concerted coding to carefully describe such things as bills of materials and lead times for purchasing and manufacturing components. Key staff in production engineering, production control, and purchasing were introduced to TRACKER in a series of week-long seminars.

During the year after TRACKER was installed inventory accuracy improved from 80 to 94% and the figure for dollar value of on-hand inventory was decreased by 20%. In addition, five clerical positions were eliminated in inventory control and purchasing. Overall, TRACKER paid back its costs in the first year. Further stages of system expansion were planned.

In 1976, terminals were added in service, production, and the stockroom to facilitate rapid inquiry and record keeping. Audiola purchased additional equipment to upgrade their Datacrunch to a top-of-the Model 1800 line—an 1800/3, to accommodate the additional core, disk storage, and communication lines demanded by these information system improvements. (The upgrade was installed in one weekend.) While access to data in existing formats was speeded up, new reports and data formats were relatively slow to be developed.

In 1977, TRACKER was reprogrammed in DATAMAZE, Data-

crunch's data base management language, so that new reports could be more easily generated. Owing in part to his success in helping reduce inventory levels, the senior industrial engineer was promoted to vice president of manufacturing. His assistant, also an industrial engineer, was promoted to the position of senior industrial engineer.

Part II: The Continuing Use and Development of Tracker

By 1979, Audiola still faced stiff Japanese competition, sales had leveled off, and profits had fallen dramatically. In the face of rapid technical innovations in audio electronics, Audiola was introducing new audio product series every two years. The engineering department was developing a line of home video products to open up a new, large, and less competitive market. Much of the capital Audiola was able to raise was being invested in video products which were to support the "future of the company."

The firm's labor costs had risen to 60% of product costs, and the new senior industrial engineer was asked to help reduce them. He decided to extend the inventory control system to include a work in process (WIP) reporting system and a series of reports which would keep track of labor costs.

Audiola had trouble attracting and keeping good staff. Salaries were relatively low, and the production workers were turning over at 1% per month. Audiola's plant was located in a relatively smog-ridden and congested section of the Los Angeles industrial belt, and the firm had a difficult time hiring new programmers. New electrical engineers were eager to work at Audiola since the firm had interesting products and a long-standing reputation in audio innovations. For programmers, however, the software developments were routine and relatively unattractive. Technical staff were paid on an identical salary scale, which further impeded the hiring of programmers since their job market was much tighter.

In 1977 and 1978, as staff who understood and believed in TRACKER left Audiola, the quality of its data declined. (The original data processing manager and systems analyst who argued for DATAMAZE left for a better job—owing, in part, to her experience with a data base language. Senior production administrators left and were able to parlay their experience with automated inventory control into better jobs elsewhere.) These conditions contributed to new difficulties in using the inventory control system, and further complicated the work of the senior industrial engineer and his staff. The purchasing staff felt overloaded in trying to find good purchases, and delayed in updating the purchasing lead times. Consequently, purchase orders were being released too early (and ignored under other pressure), or too late (resulting in higher parts costs and parts shortages).

The senior industrial engineer found that the WIP development was very slow. The programming staff were preoccupied with improving the financial programs, and only one programmer could be assigned to developing the WIP tracking modules. This programmer, who had been recently hired, had trouble understanding the inventory system. It was not documented, and had grown from a coherent system with three major files into a loosely coupled ensemble of programs with six related files. Furthermore, since Datacrunch is a minor minicomputer vendor, Audiola had to hire programmers with experience with equipment sold by major vendors, and then allow them 6 to 18 months to become intimate with Datacrunch's equipment. The programmer assigned to WIP, who had programmed applications in an IBM shop for two years, had trouble understanding Datacrunch's systems software, file organization, and DATAMAZE. The programmer's difficulties further slowed down development. But the programmer worked full-time on the WIP modules and refused to honor requests for new reports.

After the WIP tracking system was installed, it was resented by many production supervisors. Production controllers tried to locate jobs on the shop floor using WIP reports and found that often jobs were not where they were reported to be. In many cases jobs were "ahead of the reports" since job tickets were processed a day or two after the job left a particular work center. Jobs were also "behind" the reports when they were sent back to be reworked. In addition, the labor times included in the WIP system were often inaccurate. New staff, revisions in products, and the introduction of new products all slowed the production line relative to the industrial engineer's estimates of average assembly times.

Computer runs were also delayed since the Datacrunch 1800/3 was overloaded. Several different kinds of jobs competed for machine resources: financial programs, engineering analyses, and TRACKER. TRACKER was relatively high on the priority list of programs to be run, but it was not at the top. Certain financial programs (e.g., payroll, accounts receivable) received top priority at certain times during the week. The audio engineering group had developed some special programs for calculating subtle forms of transient and phase distortion in stereo equipment. They used these programs to develop their own products. And they also sold analyses using these programs to other audio engineering firms. Furthermore, the graphic outputs of these programs were used in special demonstrations to help impress private investors and convince them that Audiola would have a good competitive position in the video market with

the new products it was developing. Analyses which could be charged to paying customers received top priority.

TRACKER ran more slowly because of changes in the material environment it was used to help manage. New product lines caused the inventory to swell to 24,000 items. Inventory control programs ran longer, and occasional processing errors would further delay reports. In addition, the new on-line terminals used for inquiry required a substantial slice of system resources (thus slowing background batch jobs), and also ran slowly themselves. During peak periods the terminals were rarely used.

Unfortunately, the Datacrunch Model 1800 series could not be expanded beyond the Model 1800/3. Audiola could have shifted to a new high-speed Datacrunch Model 2000/A at $300,000 for hardware and about $400,000 for new programming efforts, but would have experienced considerable delay in acquiring the machine and converting their programs. (Audiola owned the Model 1800/3 outright, but it was only worth about $35,000 in resale value.)

In the face of the discrepancies between the formal reports and actual production times, the production controllers resorted to scheduling workflows by hand. This further undermined support for providing accurate data for the WIP system.

Despite these difficulties, the inventory control system was visibly used by top production administrators. Reports were prepared close to their scheduled times and disseminated to appropriate staff. Some of the reports were used as originally planned, others were used with handwritten modifications, and still others were ignored. Inventory levels were monitored through TRACKER, and lines of action were undertaken to help meet shipping dates through transactions and reports generated by TRACKER.

4. A Brief Introduction to Conceptual Elements for Web Models

This section introduces some key concepts for describing the structure of computer-based systems support and operations:

1. Lines of work and going concerns
2. Infrastructures of computing
3. Production lattices
4. Macrostructures of computing environments

These are introduced in Section 4.1 and developed in greater detail in Appendix A.

Any analyst who uses a discrete-entity or web model employs some conceptual vocabulary for describing and explaining the social events pertinent to computing development and use. Should organizations be viewed primarily as noisy decision-making systems or as battlegrounds in which participants struggle for the control of critical resources? Should one assume that rules are relatively clear and often followed, or that rules are flexible and often subject to on-the-spot negotiation and interpretation? Conceptual choices such as these can be viewed as a vocabulary of description. Several theoretical vocabularies have been employed by social analysts of computing. While most analysts do not rigidly restrict their conceptions to the terms of one vocabulary, they often lean strongly to one approach rather than drawing equally on all. The role these vocabularies play in filling out analyses based on discrete-entity and web models is explained in Section 4.2.

4.1 Structural Concepts of Computing

4.1.1 Lines of Work and Going Concerns

If one wants to predict how people will integrate computer-based systems into their organizational activities, it helps to know what people actually do and care most about when they act in organizations. Formal job descriptions provide a useful first approximation. Similarly, one can learn something useful about an organization from knowing its formal charter and goals. But formal tasks rarely define what people actually do and formal goals do little to describe what activities organizations pursue most forcefully.

Behaviorally oriented analysts treat formal tasks (or job descriptions) and organizational goals as possible points of departure, but not as behaviorally sharp descriptions *a priori*. In contrast to formal self-descriptions, they attend to the actual behavior of organizational actors and organizations, which may often differ sharply from public descriptions. In this section *line of work* (Gerson, 1982) indicates what people actually do in a job. Similarly, *going concerns* (Hughes, 1971), indicates the multiplicity of overlapping and sometimes conflicting activities which characterize collective lines of action in many organizations. There is a long history of these contrasts between the formal conception of organizations and less formal, more dynamic and behaviorally grounded accounts of them. Classical accounts are nicely described in Selznik (1957) and Dalton (1959); Strauss (1978a) and Perrow (1979) provide more contemporary treatments. To keep this article self-contained, these terms are also elaborated further in Appendix A. Here it

is simply useful to indicate how they enrich a web analysis of computing.

First, the line of work as an organizing concept helps shed more light on how things are actually done. Formal descriptions of how things are done are often tinged with an admonition that they be done only as stated. At the very least, a vocabulary which expands the formal conception of a job allows us to examine what happens as well as what a formalist thinks should happen (e.g., the purchasing agents at Audiola should be updating TRACKER, the investment counselor should be more of a neutral advisor than a highly self-interested salesman). Second, a focus on the formal job description misses many of the emerging and dying patterns in work life. This is particularly important for explaining technical and social change, where formal descriptions ossify outdated conceptions of what the analyst thinks should be happening.

Third, the formal goals of an organization are often insufficient to explain why certain critical patterns of computing develop. One can argue that good material handling was critical to Audiola's profitability. If one simply assumes that Audiola is a private firm and its goal is to maximize profits, then its not providing TRACKER with adequate computing resources seems foolish. By viewing Audiola as a going concern, one is led to ask: "What concerns are developed here and how do they facilitate and constrain TRACKER's development?" In answering this question, an analyst would find that the firm was short of funds and that new money was being heavily invested in new video product lines. Moreover, a revenue-producing activity such as selling engineering analyses or running the accounts receivable programs was given priority over money-saving programs such as TRACKER. Last, the fancy engineering analyses were also employed to impress potential investors, and thus played a symbolic role as well as a narrow instrumental role in the firm's development.

In summary, a web analyst who wants to understand how computing developments are integrated into an organization will ask: "What do people do and value here?" He will not expect single answers to such questions, and he will not expect that descriptions of formal jobs and goals will provide any but the most rudimentary leads. "Lines of work" and "going concerns" are more apt expressions than the latter for examining the ecology of actions (Long, 1958) which compose organizational life and into which computing developments are grafted.

4.1.2 *Infrastructure*

Infrastructure refers to those resources which help support the provision of a given service or product. The term is often used by urban

planners to refer to such basics as roads, sewers, and utilities, which support social and business activities. The infrastructure for providing computer-based services includes resources such as skilled staff and good operations procedures, as well as physical systems such as reliable "clean" electrical energy and low-noise communication lines.

Discrete-entity analysts of computing often take infrastructural support for granted (Inbar, 1979). Other times, the development of infrastructure is treated as a separate and incidental topic, whereas the development of new computer applications or technologies is given center stage (Kochar, 1979). But however much care they devote to the development of infrastructure, discrete-entity analysts make certain common assumptions: (*i*) infrastructural resources can be developed to as high a quality as needed with little delay; (*ii*) the computer-using organization has substantial control over groups providing infrastructural support; (*iii*) groups which provide infrastructural support are socially and politically neutral agents.

These assumptions are not explicit. We infer them from the extent to which the development of adequate levels of infrastructure is treated as a problem only insofar as it needs attention and resources; other difficulties such as the interests of infrastructural support groups and the constraints placed by labor markets shortages, organizational career structures, and the difficulties of coordinating groups across organizational lines are ignored (Kanter, 1977; Taggart, 1980; Hussain and Hussain, 1981). We also deduce the first assumption when the analyst pays no attention to limitations in the quality of computer-based products when the available infrastructural support is less than adequate (Kanter, 1977; Gessford, 1980; Hussain and Hussain, 1981).

The case of Audiola is instructive. The production staff had trouble using and extending TRACKER, although its information processing architecture appeared sound. Problems occurred because there was inadequate programming and machine support. In addition, production administrators who understood the systems left the firm, and certain staff, such as purchasing agents, delayed updating critical parameters.

Conventional analyses of computer use recommend that organizations evaluate the appropriateness of a new mode of computer use or application independent of the infrastructure which is likely to be available (Gessford, 1980; Inbar, 1979; Kochar, 1979). Such practice can cause a variety of problems. Many organizations still learn the extent of required infrastructure for new computing technologies *after* purchasing them. The avid computing promoter can take advantage of analyses which minimize the role of infrastructural costs; overall system costs will be underestimated and consequently will appear more affordable. After a new system is purchased, the organization will find a

way to use it or to improve its infrastructure rather than declare the initial investment as lost. Usually infrastructure costs are not fully accounted for when the costs of computing developments are calculated.

In the case of Audiola, shifting the implementation of TRACKER from COBOL to DATAMAZE was unwise given that their infrastructure was inadequate to support the shift, but was rationalized in terms of the technical advantages of DATAMAZE. These advantages could have been realized if the firm had had programmers who were skilled in manipulating DATAMAZE and adequate machine resources to support it. In 1977, optimistic assumptions concerning this infrastructure appeared warranted. But by 1979 Audiola was in a tight fiscal situation and reliance on DATAMAZE compounded the firm's problems. While the firm's top managers were trying to attract new venture capital to invest in new video products, they faced the prospect of also having to invest substantial funds in programmers' salaries and new computing equipment in order to maintain their level of operations.

Furthermore, the programmers at Audiola had few good documents to support their work. This is a common difficulty in organizations. Again, discrete-entity models assume that resources, such as the knowledge represented in software documents, will be available exactly as needed and when needed.

We find the term infrastructure useful to denote these supporting resources for computing development. They are often taken for granted, and priority is placed on the leverage to be gained by new computing developments. For organizations with limited resources, the ability to provide adequate infrastructure, rather than the ability to purchase new equipment, may be the element which most influences the organizational value of computing technologies.

4.1.3 *Production Lattices*

Computing resources are not simply provided by "people and machines." People and machines are organized through a division of labor. Different groups in different organizational locations provide different elements which contribute to some final product. These contributing elements constitute a production lattice.

The case of Audiola provides a simple illustration. A material shortage list used by a production scheduler is not simply produced "by the computer." It draws upon data from several departments, including purchasing, receiving, and the stockroom. The data are keypunched by clerks who can be located in yet another department (e.g., data processing, finance, or even a service bureau). TRACKER was designed

and programmed by several people in production and data processing. It was maintained by a programmer in data processing who consulted with the new senior industrial engineer. The material shortage list was produced as a byproduct of TRACKER runs overseen by a computer operator. The production lattice for this report was constituted by these chains of dependency.

Suppose that the material shortage report is delayed by three days and when delivered, some of the data are clearly inaccurate. Where should a production scheduler go to remedy the difficulties? If a meeting is held by the relevant parties, how many people, from how many different organizational units, might have to attend? The group that attends the meeting provides a rough map of the production lattice for that report. (A different report may be produced through a slightly different production lattice, and a meeting about its content may be composed of a slightly different group.)

The production lattice is a key social element in the development and provision of reports and records through computer-based systems. From the perspective of consumers of these products, whether organizational participants or the broader public as clients of computer-using organizations, production lattices serve a peculiar behind-the-scenes role. When "things go well," the credit belongs to an appropriate organization of tasks and skilled staff to do them. These are often invisible to the consumer of computer-based products, and the particular computer system is often foregrounded as *that which is completely responsible* for the useful reports, funds transfers, electronic mail, etc. Conversely, when computing developments or operations are troubled, the production lattice is more visible. But attendant problems are then attributed to "people" rather than "the computer." Web analyses avoid these dramaturgical manipulations by treating the production lattice for a computing product as an essential element of "the computer system" (Scacchi, 1981; Kling, 1982).

4.1.4 *Macrostructures*

Participants in organizational units are not completely free to act as they wish. Important constraints are imposed by their parent organizations and the organizational ecologies in which they act. We call these constraints *macrostructures* since they derive from outside the computer-using organizational unit and often affect many other subunits at the same organizational level and higher.

At Audiola, for example, users of the TRACKER system were affected by the use of a Datacrunch computer because the tight labor

market for Datacrunch programmers made programming support a particularly scarce resource. (In Sections 5.4 and 7.4, we examine how the character of the production lattice associated with a particular computer-based service is shaped by these broader social worlds and markets.) If Audiola had trouble attracting programmers, then perhaps it should have begun paying larger salaries. But salary scales for technical staff were set organizationwide, so that audio engineers and programmers with similar professional maturity were paid equivalently.

Other critical constraints can be found in the equipment procurement policies set by organizations. Most public agencies must select equipment based on competitive bids, even when the lowest bidders may not provide a systems environment which is the best for the price once labor costs are included (Comptroller General, 1979b). Other organizations will set equipment or technical standards to which all organizational units must comply, except in special circumstances. A simple example is when the top officers of a firm decide to standardize their equipment by purchasing all systems from one vendor such as IBM, DEC, or Burroughs. Any department can select systems it can justify, but they must be purchased from IBM (or DEC or Burroughs). Policies like these aim to minimize overall organizational costs. But they do limit the choices each organizational unit can make, and they mean that computing developments will often be suboptimal in the small. Conversely, developers or users who insist on getting close to optimal computing resources will try to alter or circumvent organizationwide rules which constrain their choices.

4.1.5 *Summary*

This section introduced five key concepts: lines of work, going concerns, the production lattice, the infrastructure of a computer-based system or service, and the macrostructures within which production lattices are situated. These concepts are of particular value to web theorists because they help describe the critical social and technical relationships from which webs of computing development and use are spun.

The discrete-entity analyst can ignore the social organization of computing development and use, and make simplifying assumptions which are usually optimistic: (*i*) users have effective control over all resources and services on which they depend; and (*ii*) suitable resources of adequate quality can be counted on to be available as needed.

The production lattice helps identify the various stages in which CRs are built up in providing a specific computer-based service. When the CRs are spread across several organizations, no one actor is likely to have complete control over the production chain.

Infrastructural investments help a computer user increase his productivity. Often these investments are minimized. Consequently, computer use is more problematic than had been expected, development and use take longer, and overall system costs are higher. Richer organizations can afford to develop infrastructure, buffer themselves from delays, and avoid minor failures, all of which would be "luxuries" to poorer organizations. Thus, by identifying some of the exogenous elements which influence the character of specific computerized information services, we can identify additional contingencies which delay productive computing activities or which prevent "optimal" systems from being developed, implemented, or operated.

4.2 Theoretical Languages and Models of Computing

In analyzing the dynamics of computing developments and use, the analyst relies upon a conceptual vocabulary, however implicit, for making sense of the technical and social worlds he examines. Such a vocabulary expresses concepts of a particular theoretical perspective. Each perspective also includes central ideas for applying it.

Four perspectives[5] predominate in the literature of social analyses of computing:

1. Formal–rational
2. Structural
3. Interactionist
4. Political

Each of these perspectives casts a different light on the significant aspects of computing development and use. Each relies upon a different root metaphor for interpreting social life as well. Organizations, for example, are viewed by formal–rational analysts as relatively well-defined machines, while structural analysts view them as slowly adapting, noisy, rigid, and sluggish task systems. Interactionists view them as arenas in which people enact important social meanings, and political analysts view them as battlegrounds in which participants continually struggle for control over valuable resources.

[5] Kling and Scacchi (1980) and Kling (1980) identify six theoretical perspectives used by analysts of computing. Two of these, human relations and class politics, will not be examined here because they influence only small segments of the literature and we do not find them as helpful as the four other perspectives for relatively wide-ranging analyses. See Appendix B for a brief introduction to the four perspectives examined here.

These theoretical perspectives provide useful vocabularies and organizing concepts fleshing out particular discrete-entity or web models. Most discrete-entity analysts employ the idiom of the formal–rational perspective. Its reliance on the formal goals of organizations, formal procedures, and formal properties of technical systems mates nicely with the organizing concepts of the discrete-entity model delineated in Section 2.

Sometimes discrete-entity models of computing are embedded in larger structural or political analyses. A common example is when an organization is viewed as a noisy communication system, and computerized systems are held to help ease problems of communication and information overload. If the computing technology is viewed as a context-free object which itself is not subject to uncertainties, we have a discrete-entity analysis (Simon, 1973). Similarly, if computerized information systems are viewed simply as instruments of organizational politics, but themselves reflect no political decisions for their internal arrangements, we also have a discrete-entity analysis.

Because they concern the connections between a given computing resource, the lines of work within which it is used, and infrastructure developments and macrostructural patterns, web models cannot be framed in the formal–rational perspective. They can be framed, however, in any of the other three theoretical perspectives. In short, web analyses are framed in theoretical terms which allow for social and economic context, social conflict, and uncertainty in the character of the supporting social organization. This framing is examined in Sections 5 and 7.

5. The Dynamics of Computing Development and Use

What affects the development and use of computing in an organizational setting? What story best accounts for the evolution of computing arrangements at Audiola? As we saw in the preceding sections, discrete-entity and web analysts tell stories framed with different central organizing concepts focusing on different aspects of computing. In this section, we examine the factors that each model specifies as affecting the development and use of computing in organizations.

As we explained in Section 2, discrete-entity analysts employ two logics of technical development: direct substitution and incremental aggregation. According to this dual logic, Audiola improved its ability to track inventory and production through several substitutions and incremental accumulations. Manual records were replaced with TRACKER

(substitution); terminals were added to improve data accuracy and data access (incremental aggregation); flexible report writing was supported by DATAMAZE (incremental aggregation) and work-in-process tracking (incremental aggregation). After all of these alterations, the discrete-entity analyst would argue, Audiola's production administrators should have been better able to manage their inventory and production schedules than when they relied upon manual records.

Unfortunately, Audiola was in worse shape, not better shape. Many of the firm's difficulties were due to exogenous market conditions, but these conditions also made its computing support problematic. The Audiola case reflects a dynamic which is at odds with a simple logic that equates technical substitution and incremental aggregation with overall improvements in organizational effectiveness. But many other such examples can be found, particularly among organizations whose services are based on large systems or old ones. The General Accounting Office of the United States Congress routinely reports bureaucratic bungling, poor planning, political infighting, and bad project management—these factors lead to poor system implementations, cost overruns, and other problems of major proportions for many large, publicly financed computing projects (e.g., Comptroller General, 1976, 1978a, 1979a,b, 1980, 1981).

Discrete-entity models do provide some partially accurate descriptions of TRACKER's use at Audiola. In the idiom of these models, one can say that TRACKER was "used," and that it met some of the information needs of Audiola's production staff. Its "potential" was not exploited, and Audiola was having "people problems" in expanding TRACKER. Also, "better planning" might have led to avoiding DATAMAZE entirely. While valid, these observations miss the heart of the payoffs of TRACKER to the staff at Audiola and the difficulties in using and expanding it. TRACKER is not best understood through the categories of "success" or "failure." It was "sucessful" in that it was used routinely by the production staff at Audiola. It still fails to meet their current needs and expectations. It is typical of many automated information systems advertised as "successes"—it is used and useful, but imperfect and sometimes problematic.

Discrete-entity analyses unfortunately provide only limited terms to explain these recurrent difficulties. They are relatively ahistorical and asocial. Lines of action that did not work well are primarily viewed as the products of bad decisions and inappropriate procedures. Web analysts are more likely to view difficulties in developing and using complex computerized systems which cross technical boundaries as the byproducts of larger patterns.

We find that the following five propositions which are consistent with

web models of computing portray a richer view of the development and use of computing than that of discrete-entity analysis:

1. Computer-based service provision is specialized.
2. History of commitments constrains choice.
3. Narrow incentives and opportunities motivate choice.
4. Macrostructural patterns influence local computing.
5. Computing systems evolve through fitting and packaging.

These propositions are not exhaustive and they do not provide unique answers to the initial questions, but we believe they lead to more astute observations of the conditions, qualities, and problems of complex computing arrangements. (See Section 5.6 for several additional propositions.) We use structural, interactionist, and political perspectives to examine these five propositions.[6]

5.1 Computer-Based Service Provision Is Specialized

The digital computer used by an organization may be general-purpose, but that does not tell the whole story. Local computing organizations and their supporting infrastructure organize to provide some services well and other services badly, if at all.

This principle is best illustrated by an example.[7] A common example is when staff who wish to undertake statistical data analyses attempt to use a computer operated and programmed by a staff who primarily engage in routine data processing. While statistical packages and algebraic language processors may be easily available for the machine, the analysts often will experience considerable difficulty in having their analyses programmed and run in a timely manner.[8] Most discrete-entity analysts would emphasize the capabilities of the underlying computing equipment, and would not predict difficulties as long as the equipment is suitable.

Web analysts would be less sanguine. It is useful to see how web analysts who employ structural, interactionist, or political perspectives make sense of computer-based services.

[6] See Appendix B for a description of these three theoretical perspectives.

[7] For a different kind of example, see James Fallows' account of the way Army specialists redesigned the AR-15 rifle so that it lost its field effectiveness in Vietnam (Fallows, 1981).

[8] In 1976, the first author and colleagues at the University of California at Irvine studied the patterns of computing support provided by 42 local governments for a variety of applications and departments, including finance, police, and urban planning (cf. Danziger *et al.*, 1982). He studied planning applications in Brockton, Mass., Flourisant, Mo., Montgomery, Ala., Newton, Mass., Kansas City, Mo., New Orleans, La., St. Louis, Mo., and Tulsa, Okla. In Flourisant and Brockton, the planners were few in number and made no requests for analytical computer runs from the municipal data processing organization. In St. Louis, the planners received adequate service from the municipal data processing center. In all the other cities listed above, the planners received poor service from the municipal data processing center and turned to organizations outside the city government for computer service to support analysis. Universities were used by planners in New Orleans and Newton for locally developed applications; some planners also used special models developed and run by state government computing facilities.

Structural analysts, who view organizations as routinized production and communication systems, would be less surprised (Cyert and March, 1963). Organizations which have developed routine data processing as their primary product have developed special skills and standard operating procedures for increasing the efficiency of standard batch file manipulation and scheduled report production. These include investments in appropriate language processors (e.g., COBOL and IMS) and staff who are skilled in their use. Such an organization will also have standardized its operations so that jobs are run at preset times against specified files. An instrumental computer user who wants a new report is requesting a service which such an organization has tuned itself to best provide. An analytical computer user, on the other hand, makes a set of demands which cut across the standardized procedures and skills that characterize this subunit. The most common languages still used for analysis (e.g., FORTRAN) are infrequently used for data processing, and vice versa. Thus the user of an analytical report is unlikely to find a programmer in the data processing department with appropriate skills to consult on ways to set up and organize jobs and files, etc. Data analysts are often exploring data, and will execute a program some indefinite number of times until they obtain a satisfactory result. That style of use does not mesh easily with the standard procedures of a shop where all programs are run a specific number of times and a fixed number of reports are prepared at the end of each run.

Structural analysts might also wonder whether the equipment were specially configured to best support data processing jobs, which are often input–output (I/O)-bound, or data crunching jobs which are often CPU-bound. If the system resources (e.g., memory, I/O channels) are selected for one job type and the operating system is also tuned to perform best with jobs of that type, then adding jobs that make complementary demands on the machine will also appear to consume disproportionately large system resources and probably will decrease overall throughput unless the system is reconfigured and retuned for the broader mix. (This is expensive and time consuming, and demands expertise often not available in small computer centers—it is avoided when possible.)

An *interactionist* analyst would pay particular attention to the way in

which data processors and data analysts who work together bring different, and incompatible, meanings of "good and responsible" computation to their work setting. For the data processor, standard, predictable reports are most sensible. Reports that are immediately thrown away are "wasteful" and shouldn't have been generated in the first place. For the data analyst, one may not know what an analysis should look like until several trials have been run. Responsible computing exploits the computer to search through hundreds or thousands of possible relationships to help find a few important ones. These stereotypic positions characterize the orientations of computer specialists who are immersed in data processing and computer users whose interests are primarily analytic (Kling and Gerson, 1978). For a discrete-entity analyst, an "obvious solution" for enhancing the computing support of the data analysts is simply to hire appropriate computing specialists. Such a strategy makes good sense, but is not trouble-free. In particular, interactionists would expect similar tensions between those specialists who support routine data processing and those who support analytical applications.

A *political* analyst would examine the difficulties in service primarily in terms of control over scarce resources and autonomy. According to Danziger (1979), the data processing staff form a "skill bureaucracy" which defines its own standards, operates so as to maximize its autonomy, and seeks to expand its domain of administrative control. Facing demands for a type of data analysis that his staff find unusual, the data processing manager may lean toward one of two strategic extremes: either ignoring the request, since if his staff do not perform well they could be labeled incapable, or attempting to provide good analytical support even if that requires hiring new staff, so that the data processing facility maintains a local monopoly over the provision of computing.

Clearly, there are also intermediate alternatives. At any time, the data processing center is specialized. It is organized and staffed so that it can provide some computing services much better than others. In the face of service demands that are not easy to accommodate, the computing staff will make strategic choices which increase their autonomy and administrative control. Expansion and retrenchment can each be a sensible choice, depending upon the salient social elements. For example, no computing center will service all requests for software alterations with equal speed: the power of the customer, the technical interest of the job, the possibility of new revenue will all be positively correlated with speedier service.

Computing resources which are relatively general-purpose at the "source" of a production chain are effectively specialized through a variety of processes which cannot be instantaneously altered. Users will get

less than they expect if they assume that service providers that sit between them and a particular resource will be neutral agents.

5.2 History of Commitments Constrains Choices

According to discrete-entity models, the historical evolution of computing services in an organization is irrelevant to an understanding of current arrangements or to developing new arrangements.

Discrete-entity analysts sometimes do appreciate (and bemoan) the way in which historical precedents and early commitments have shaped the current practices and technology of an organization, including its computing arrangements. But they believe that historical choices do not usually meet "current needs," and decisions about choices should be ahistorical. In normative administrative theory, the budgetary reform movements such as zero-base budgeting best exemplify this sensibility.

It would be hard to label web analysts as "antireform." At the very least, however, web models attempt to provide sharp concepts for understanding the dynamics of social and technical change. According to web models, at any given time the ensemble of computing arrangements that support a particular application represent a capital investment in equipment and support materials (Withington, 1979) and a political investment in both technical experts and user clientele. Feasible choices for the "next version" of a system are both supported and constrained by the costs these historical commitments represent. Current commitments represent a fragile platform on which new capabilities may be built. (At Audiola it was easier to add DATAMAZE after the computing staff had some experience with TRACKER.) But new systems and enhancements also represent unique compromises in choices of equipment and infrastructure which are optimized to minimize direct costs. At the very least, ahistorical proposals will incur relatively high indirect costs as local infrastructures are reshaped, political commitments are altered, etc.

Web models differ in the places on which they shed their conceptual light. *Structural* analysts view organizations as networks of specialized production units and communicators. Audiola in 1979, for example, had developed TRACKER on a Datacrunch 1800/3 in DATAMAZE. Suppose that an alternative package such as IBM's COPICS were technically a better product for the job. Substituting COPICS and an IBM machine for TRACKER and the Datacrunch would be a laborious and expensive job. Equipment costs set a conservative lower bound. While equipment compatibilities are the best understood historical constraints in the computing world, there are other kinds of commitments which often constrain the choice of one actor in a production chain about the kind of equipment

to acquire, procedures to institute, or services to provide. Making radical changes of equipment indicates a kind of infrastructural constraint, but other examples can occur from outward on the production lattice. Organizational subunits historically develop clienteles among certain departments. They are rarely free to abruptly alter the kinds of services they provide without providing a graceful way for their previous clients to find service elsewhere.

Interactionists would amplify this last observation by underlining the way in which organizations, including those that provide some kind of computing service (e.g., CPU cycles, software products, data analysis, consulting) develop images which coalesce shared meanings about what the organization is. These are difficult to change.

Political analysts pay particular attention to the coalitions of interests served by current arrangements and proposed new ones. For them, history is not some abstract label, but a catchword for indicating the way a group has allocated its resources—in our cases through technical investments and related services. Alterations are likely to redistribute benefits, and stronger interests are likely to fight hard to avoid big losses.

5.3 Narrow Incentives and Opportunities Motivate Choices

According to discrete-entity models, CRs can be characterized by a set of attributes (e.g., costs and information processing capabilities). New CRs are attractive when their information processing capabilities and cost provide additional leverage for users. For many computing specialists, technically sophisticated systems motivate people to use them in an effective manner. In this way, interactive systems are preferable to batch systems, as distributed are to centralized and personal are to shared systems. Similarly, the more elegant features, concurrent computations, and processor cycles a system offers and the fewer keystrokes it requires for use, the more seductive it will be. In economically focused discrete-entity analyses, technical sophistication comes at a price, and the smart adopter selects a set of features where the marginal benefit equals or exceeds its marginal cost.

Web analysts do not have such sharply defined positions on the incentives for adopting new CRs. They believe that discrete-entity explanations hold in certain cases, but are limited in (a) assuming that decision makers employ broad and long-range rather than narrow and short-range criteria; (b) characterizing the adoptability of CRs by technical and economic attributes only; and (c) assuming a single logic of adoption.

Structural analysts find that organizational actors do not bring a panoramic rationality to critical situations and decisions. Their framing of

issues, selection of information, and evaluation of alternatives are influenced by their occupational perspective (Dearborn and Simon, 1958; Cyert and March, 1963). Moreover, the conflicting pressures of their own jobs and commitments outside the organization strongly influence their priorities. Pettigrew (1973), for example, studied the acquisition of computer equipment by a British manufacturing firm and found that the contribution of equipment to the overall profitability of the firm played a small role in the actual process by which equipment was selected.

In practical terms, one finds organizational subunits seeking CRs which best serve their own interests. Sometimes, common computing arrangements or equipment can jointly serve many groups well, and no serious compromises need to be made. However, when organizational subunits engage in radically different lines of work (e.g., scientific analysis and purchasing, medical laboratory analysis and billing), joint solutions will be farther from optimal for some parties.

Interactionist analysts observe that people bring their own incentives to a setting and find opportunity in their interpretation of what resources are available. Accordingly, people strive to become more certain about what computing arrangements are preferable and act to realize that interpretation. For many computing specialists, developing or using "state-of-the-art" systems is important and an indicator of professional status. Development and use of such systems often brings peer recognition of technical mastery and prowess.

Furthermore, this recognition can open up new career opportunities for these specialists, as gossip about their achievements spreads inside or outside their employing organization. In the case of Audiola, computing staff who gained experience with DATAMAZE found good jobs elsewhere. In addition, the industrial engineers who developed TRACKER were promoted within the firm. (It is not the case that computer specialists only find career mobility outside the organization and users within it, although this is common.) The skills and recognition that are attained in this way can thus take on symbolic and capital value which specialists can develop through working with state-of-the-art CRs. Similarly, instrumental users of computing such as insurance actuaries or chemical engineers may find that their occupational experience with newer or more complex computing systems has payoffs for their own careers—and serves accordingly as an incentive for action.

People act to serve their own interests. People who control decisions about how to develop or use CRs are likely to determine whose interests are served by them. For analysts of *organizational politics*, the opportunity to control computing resources to serve one's interests is a common incentive for developing or using CRs.

Interesting examples can be readily found in the scholarly literature. Kling (1978b) reports the case of an information system which was highly touted by its developers and users as improving the efficiency and effectiveness of welfare operations in a medium-sized southern city. Close inspection revealed that the system had few direct payoffs for organizational efficiencies. It was, however, an important instrument in bargaining with Federal auditors and HEW in receiving additional funds. The presence of the system rather than its use was most critical.[9]

5.4 Macrostructural Patterns Influence Local Computing

According to discrete-entity models, near-optimal arrangements for computing can be developed independent of the organizational or social mosaic within which they are to operate. This is similar to saying that computing systems can be developed or provided in a "context-free" manner. In contrast, according to web models, the structural arrangements of computing reflect the patterning of organized interests in the host organization. This is similar to saying that the development or provision of computing systems in an organization is "context dependent."

At Audiola, users of the TRACKER system were affected in several ways by the company's use of a Datacrunch computer. The labor market for Datacrunch programmers made programming support a particularly scarce resource. The use of DATAMAZE, a language unique to Datacrunch, compounded Audiola's difficulties. The problem of expanding a computer center which owned one full-scale model 1800/3 further compounded these difficulties. Audiola's low salary scale for programmers further weakened the company's ability to compete in the job market for Datacrunch programmers with DATAMAZE experience. In addition, the relative priorities given to financial and manufacturing applications were higher than those assigned to engineering and planning applications, which are technically more challenging and interesting, and which therefore are likely to appeal to highly qualified potential employees. All of these arrangements undermine Audiola's ability to develop systems that fit well and work dependably.

The *structural* perspective provides a good basis for analyzing four elements that plague the development and provision of computerized services: "inappropriate procedures," delays, skilled labor shortages, and difficulties of communication. Delays are pedestrian, but are often treated

[9] For other examples see Sections 4.1 and 4.3 of Kling (1980), and Markus (1979), Pettigrew (1973), and Albrecht (1979).

as unexpected. They are much like system costs: "cost overruns" is a commonplace term while "cost underrun" is not. "Unexpected delay" is a common term while "unexpected advance" is not.

The model of organizations as networks of processors which produce and transport materials or information, and as noisy communication networks, helps make delays "normal" (Carzo and Yanouzis, 1967; Cyert and March, 1963). In particular, efficient processors should have a low idle time and nonempty queues of waiting jobs. (Municipal firemen who are organized to respond rapidly to any emergency often sit idly and so are "inefficiently" used much of the time.) However, in the case of software development and maintenance, staff who are so efficiently organized that they are always busy cannot attend to all requests as they come in, or even as they are "needed." Some requests for service will be queued for later service. If the service unit (e.g., maintenance programmers) is very efficient or understaffed, its queues should rarely be short. The queue discipline can be dynamically reordered to accommodate "emergencies" (thereby delaying other requests), or the priorities of best customers (thereby risking charges of "politics").

Further delays come as a result of the time required to make a decision about a "needed" resource. Unless a manager is prescient and proactive, his perception that some computing resources are needed will develop as bottlenecks arise. Requests for equipment (e.g., terminals or software) or staff usually have to be approved if they exceed some set amount. In wealthy organizations or on high-priority projects, such approvals may come rapidly. In poorer organizations or on low-priority projects requests are more carefully scrutinized. They may even be batched for periodic committee review. Creating justifications, getting approvals, possibly recycling, seeking goods or services in the market, completing a workable contract, getting delivery of goods, training new staff, etc.—all take time. Some organizations have additional procedures which add additional delays. Public agencies usually require competitive bidding on expensive equipment and affirmative action searches for filling job openings. While each has merit, each also delays the acquisition of staff, labor, or materials.

Another factor producing constraints is the relative availability of resources needed by the organizational units on the production chain for specific computer based services. Organizations and their subunits vary considerably in their abilities to compete for scarce resources. To compete for technically skilled labor, the relevant resources are salaries, location, interesting jobs, high morale, and so forth. In the 1970s there has been a very tight labor market for computer specialists, and it will proba-

bly continue into the 1990s. Organizations which are poorer in their ability to attract skilled computer specialists will simply suffer a lack of ability to maintain relatively sophisticated systems.

In our pedagogical example, Audiola's location in an old industrial section of Los Angeles was an impediment to attracting technical specialists, including programmers. Had Audiola been rich enough, it might have compensated for its location through salaries or other perquisites (or possibly even moving). Sometimes, identifiable classes of organizations will be relatively disadvantaged. During the late 1970s public agencies faced sharp fiscal reductions which have made them less able to compete with private firms for skilled technical staff.[10]

An interesting example of macrostructural influence which lends itself to web analysis can be found in evaluation of the capabilities of Soviet technologists and industrialists to develop sophisticated computer systems. Discrete-entity analysts examine which technologies are least well developed (e.g., high-density magnetic disk storage), and argue that if the Soviets could develop them, their technical capacity would be dramatically enhanced. Web analysts take a different approach and reach different conclusions. Goodman examined the social system in which the Soviets produce, disseminate, and maintain software (Goodman, 1979b). In addition, Goodman broke further ground by examining Soviet software through the system life cycle. This turned out to be particularly apt, since the Soviet institutional arrangements support some activities (e.g., design) similar to Western countries, while they are particularly troublesome for other stages of the life cycle (e.g., testing, maintenance). Generalizing from his analyses of hardware, software, and economic applications, Goodman examines the ways in which the effective computing developments depend on institutional relations which are responsive to users and which cut across institutional sectors (e.g., commercial and military, academic and industrial).

The standard model of the Soviet economy treats it as a two-tier system segmented into military and consumer subeconomies. In the Soviet Union, most production and distribution is planned, and there are relatively few market mechanisms for adjusting allocations of resources. In the planning system, the military subeconomy receives top priority. The civilian subeconomy receives lower quality materials and talent, and its supply of resources can be preempted by organizations in the military

[10] Even the United States military has trouble retaining specially trained staff. For example, in the last quarter of 1980, 33 military personnel who were specialized in the use of a computer for diagnosing electronic failures in F-15 fighter planes were eligible for reenlistment. None reenlisted (Fallows, 1981, p. 48).

subeconomy. In each sector, the industrial order is organized primarily as a hierarchy, with relatively few cross-cutting organizations, committees, trade associations, professional groups, etc. These features of the Soviet economy are well known, but have not yet informed analyses of computing developments in the Soviet Union.

Goodman argues that this two-tier model and its hierarchical organization has posed particular problems for computing developments. It works best for developing and disseminating products which can be neatly segregated into one of the subeconomies (e.g., toasters or submarines), and can be adequate for disseminating products which require little fine-grained feedback from their consumers.

Computer technologies do not have these characteristics. Computing developments easily spill over from one sector to another. Most hardware is relatively general-purpose. Moreover, computer systems, particularly software, are best developed when there is close contact between consumers and developers. Such close contact is essential for maintaining software so that it works under changing technical configurations and so that it can be finely tuned to changing organizational conditions.

Goodman shows that the Soviets have had systemic problems in developing computer technology, and that they have had markedly greater success with hardware technologies than with software support.[11] He analyzes how many of these problems derive from the two-tier, hierarchically organized economic system. This leads him to conclude that technical progress in software developments relative to Western countries will be impeded by the two-tier economic segmentation, regardless of their advances in some specific component technology such as mass storage media.

Goodman's web analysis differs considerably from those put forth by discrete-entity analysts (Dale, 1979) who ignore the structural character of the Soviet economy. The conventional discrete-entity analyses assume that advances in certain key technologies, such as magnetic disk storage media, will make substantial differences in the Soviets' ability to develop useful software technologies such as DBMS. Goodman examines the role of critical component technologies in estimating the Soviets' computing capabilities as well as examining their routinized structures for developing and using new technical systems. He simply finds that major advances in component technologies cannot easily increase the Soviets' technical capacity by providing a technical fix which compensates for their institutional constraints.

[11] There are variations in their technical success in each area. Soviet CPU technologies appear to be more reliable than magnetic disk storage technologies, for example.

Goodman's analyses of the Soviet computing industry go far beyond the largely technology-centered analyses which have been previously published in the West. These tend to miss the different institutional requirements for hardware and software production as well as the way in which different stages of the software development life cycle require different social infrastructures for support.

Interactionists also notice that the meanings of computing are not inherent in a technology or setting. They are often developed by people through their interactions with computing and with others both in the focal social setting and in the larger world. Organizations which attempt to attract good staff by adopting "state-of-the-art" equipment exploit one class of social meanings. Conversely, organizations which have trouble finding technical staff to work on older equipment, routine projects, or documentation are "victimized" by a set of social meanings defined outside their walls.

5.5 Computing Systems Evolve through Fitting and Packaging

According to discrete-entity models, coherent systems can in practice be developed and regularly used. Coherent systems emerge through modular development, standardized interfaces, and ease of extension. Older, less integrated and more clumsily developed systems will naturally be displaced when newer, more sophisticated systems arrive. Better systems will drive out poorer systems. New system components offer more performance or additional capacity and in turn facilitate system growth. Adoption of a DBMS to enhance an organization's ability to manage data spread over many different computer-based files is a common example. Systems thus evolve through the substitution or accumulation of advanced system components.

In contrast, real automated information systems and information processing resources are rarely coherent and completely consistent "systems," according to some web analysts. Further, such systems are unlikely to become completely coherent. Real-world computing systems are often developed ad hoc and incrementally. Small alterations are grafted onto existing systems in ways which may compromise a modular architecture and lead to a more skewed coupling. Maintaining a coherent system configuration takes effort, adequate resources, and few constraints on their use.

More typically, people develop their software systems layer upon layer. The layering that emerges reveals systems as obscurely coupled ensembles whose interactions are sometimes troublesome to manage or comprehend (Palme, 1978). People regularly tinker with their systems to

get them to operate as they have in the past or as they should under present idiosyncratic circumstances. Furthermore, similar but incompatible systems proliferate under these circumstances, or in settings where user groups differ in work demands and control over substantial programming resources.

People shape the evolution of computing systems by the ways in which they (a) develop new systems or add enhancements to existing systems and (b) assimilate these "computing innovations" into their daily routines (Scacchi, 1981). The people who produce a computing innovation determine its contours and intended fit into the local computing infrastructure. Each innovation may be developed by participants according to the social conditions and technological configuration they assume (e.g., eager users and abundant processor cycles). However, these assumptions may not be fully met by existing computing arrangements. Users may be preoccupied and disinterested and processor cycles are often scarce. These arrangements and the conditions and resources they assume outline the local package of a computing innovation.

Computing innovations not only come as a package, but the way they are assimilated into local computing arrangements constitutes a repackaging of local computing infrastructure. The computerized accounting arrangements described in Stewart (1977, Appendix A) is a clear example of this process. This repackaging (or simply, packaging) reflects a redistribution of (1) access to computing resources, (2) the structure of computing tasks, (3) the configuration of system components, and (4) the appropriateness of prior experiences and present understandings of the most manageable ways to conduct computing tasks. Packaging characterizes one process through which local computing arrangements are innovated and evolved.

In *structural* terms, computing innovations that "fit well" closely conform to the interests, resources, and constraints of participants, thus effectively serving organizational interests. The appropriateness of an innovation's fit can be identified according to (a) the existing social and technological arrangements in which it will be embedded and (b) the commitment of resources necessary to sustain its fit. But the window for smooth entry and integration of an innovation into the local computing infrastructure changes shapes and location over time.

Embedded innovations can become obsolete when they no longer fit the contours of the local computing infrastructure. The resources, inputs, and skills required by outdated computing innovations may no longer be available or their output no longer needed. This may be due to technical or organizational improvements elsewhere. The efficiency of a particular innovative technique depends not only on its own performance and that of

its immediate substitutes, but also on its surrounding technology. This means that a technique may survive from the past as operationally effective and yet be obsolete when compared to the technology in use.

For *interactionists*, the actions of participants attempting an innovation together with the loose (available) elements in the local computing package "define" which innovations are appropriate and fit well. A new computing innovation is not appropriate a priori, no matter how capital-intensive, ugly, small, or beautiful. Rather, local computing arrangements, how participants embed their work in it, and what they perceive as the (dis)advantages of a new resource distribution all shape how computing innovations will fit and be evolved.

Similarly, for interactionists the local computing infrastructure is like a multidimensional jigsaw puzzle being assembled by many people who interpret its always partial labyrinth of pieces as images ultimately becoming coherent in the near future. Computing innovations serve as new or additional pieces that, when fitted to existing pieces, reveal new or slightly altered but still incomplete assemblages. The recurring packaging and fitting together of new system pieces—by the way participants develop, use, and evolve computing innovations through their work—are basic social processes which animate the local computing infrastructure.

Computing innovations may reinforce, stabilize, or subvert existing organizational computing arrangements—according to analysts of *organizational politics*. People in organizations seek control over their own resources (e.g., their time, attention, and skill). They will at times attempt to innovate their computing arrangements to create marginal differences in their resource control. Innovating computing arrangements is a familiar and routine way participants alter the patterns of computing use and distribution of resources under their control. Whether the innovations reinforce or subvert is determined by the distribution and configuration of system resources, by those actors participating in the development of the innovation, by the participants who use and evolve them, and by the conditions existing when they were packaged.

5.6 Synthetic Observations

Discrete-entity analyses color our understanding of computing developments by overestimating "rational" payoffs and providing an impoverished conceptual vocabulary for understanding the conditions of less than complete "success" or "failure." They certainly do little to explain how success for some is failure for others.

Discrete-entity models can be employed in "pessimistic" analyses

which assume that computerized systems will be oppressive instruments of surveillance (Rule, 1974) as easily as they can inform analyses of broad social boons created by new computing developments (Hiltz and Turoff, 1978).

In practice, discrete-entity analysis is more often turned to promoting new computing developments rather than demoting them. Its conceptual vocabulary is not terribly useful for understanding the actual dynamics of computing developments. A web analyst takes as given certain commonplace difficulties: misspecified systems, cost overruns, and rigid systems. If one attempts to explain or predict their occurrence, the discrete-entity analysis does not provide much help: these problems are perceived to occur because (stupid) people used the wrong technology (Ackoff, 1967), developers and users did not plan properly (Comptroller General, 1979b) or they mismanaged their resources.

We find the explanatory terms of the three theoretical perspectives within which web analyses are framed to be much more satisfactory. If cost overruns are common across many projects, many organizations, and many technologies, the web analyst looks for systemic causal features of the environments rather than characteristics of individuals. Structurally induced delays, conflicts over goals which also produce delays, and incentives for increasing costs all provide useful points of departure.

In this section we have examined many themes which distinguish web models of computing developments from discrete-entity models:

1. Computer-based service provision is specialized.
2. History of commitments constrains choice.
3. Narrow incentives and opportunities motivate choice.
4. Macrostructural patterns influence local computing.
5. Computing systems evolve through fitting and packaging.

These are hardly exhaustive. We could have added the following:

6. Adoption is selective (Laudon, 1974; Pettigrew, 1973; Danziger *et al.*, 1982; Perry and Danziger, 1980).
7. Innovation is continuous rather than discrete (Kling and Gerson, 1977; Scacchi, 1981).
8. Costs are often underestimated and economic payoffs overestimated (King and Kraemer, 1981).
9. Different technical arrangements reflect political and social value choices as well as "technical rationality" (Kling, 1978c,d; Danziger *et al.*, 1982).
10. Weak infrastructure often impoverishes the quality of computer-based services and systems actually provided (Goodman, 1979c).

11. The infrastructure of computing services is often unevenly developed in organizations. The quality of infrastructure will also vary across organizations, across applications within an organization, and across modes of computer use.
12. Outcomes of computer use and strategies for computing management are context-sensitive (Kling and Scacchi, 1979b; Danziger *et al.*, 1982).

Underlying these claims is the conception that the computer-based systems they best describe are a form of social organization. Discrete-entity models indicate what may happen under the best of circumstances. The web models are best suited to ask what is likely.

Our exposition in this section rested in large part on the pedagogical case of Audiola, research reported by others, and our own perceptions. In the next section, we report two new cases of computing to examine the mileage we gain from the propositions developed here.

6. Two Empirical Cases: Office Automation and DBMS

The following cases examine computing arrangements in two real organizations, CSRO and INSURE, to help ground the preceding analyses. CSRO is a computer science research organization and INSURE is a commercial insurance company.[12] Both cases are computing arrangements which are currently receiving much attention: office automation and data base management. Further, these cases describe "showcase" or "leading-edge" computing settings. However, we focus our attention on the routine use of computing systems as performed by the local participants, rather than just on the novelty of the computing systems themselves. These cases exemplify what a local computing milieu looks like, how it is experienced by participants, how it shapes their work and vice versa, and how it can be understood in terms of discrete-entity and web models. This should help ground the preceding analyses.

[12] CSRO and INSURE are pseudonyms, as are the names of the software processors described in the cases. The cases are based on intensive observational field studies and structured interviews with more than 30 participants in each organization. Data for these cases were collected by the second author during 1977–1979. The data collection scheme emphasized interactionist and structural themes (Kling and Scacchi, 1980, Appendix A). See Scacchi (1981) for details of the data collection and additional information about both organizations and Kling and Scacchi (1980) for additional details about CSRO.

6.1 Office Automation at CSRO

CSRO is a university-based research organization established during the late 1960s to coordinate research activities between computer scientists and researchers in other scientific fields. The staff at CSRO consists of about 60 researchers, including principal researchers, research associates, graduate students, and research programmers supported by computer facility and secretarial personnel. Researchers group according to projects or common interests. This clustering reflects the divisions of system resources collectively allocated by CSRO managers (e.g., the allocation of processor time and pages of disk storage).

In 1974, principal researchers at CSRO and their collaborators acquired a large-scale interactive computer system with funds from a joint research grant to support their investigations. Their computer system was a DECsystem-10 with TENEX operating system utilizing more than 2 megabytes of main memory and more than 400 megabytes of on-line disk storage. The system could support more than 50 interactive users during peak use times. Some users develop very large interactive application systems written in the INTERLISP language and programming environment as part of their research (Sandewall, 1978; Teitelman, 1978). Most participants make extensive and frequent use of the office automation facilities on this system. The system is also linked to outside organizations and other computing facilities through the ARPANET computer network. The DECsystem-10 computer system, the TENEX operating system, and the INTERLISP programming environment were all requisite hallmarks of CSRO's location within the world of computer science research (Kling and Gerson, 1978). In sum, at the time of our investigation (1977–1979), the computing environment at CSRO was quite sophisticated.

In 1976, CSRO's single-processor computer was upgraded to a dual-processor configuration. The upgrade was intended to improve overall system performance, but the gains were smaller than expected. The "overhead" computations of the computer's operating system increased substantially, thus slowing the upgraded system (staff estimated 35% overhead increase). The computer facility staff attributed the behavior of the system to the computer's memory (size) limitation. But prior to the dual-processor upgrade, memory capacity was not seen to be the limiting factor in increasing the system's performance. About a year later, the capacity of the computer's memory was doubled through the addition of a faster second memory unit.

From 1975 to 1979, CSRO's on-site users and the computing facility staff repeatedly exceeded their allocations for disk storage. To meet the

increased demand for storage, the computer's disk storage capacity was doubled by adding more disk units. Also during this period, the average number of regular CSRO system users doubled. Total processor utilization increased by a factor of six although peak daily system load average remained steady.[13] Though the computer's processor, memory, and disk storage capacity were augmented, total computer system usage far outstripped this augmentation. Measurements of system load averages suggested that users were extending their system usage by increasing their consumption of system resources. Most users agreed that this was the case. But these users also indicated that there were noticeable fluctuations in system usage after each of these hardware enhancements was introduced.

Many users indicated that the level of computer use continued to increase. In order to avoid high system loads during regular work hours (8 A.M. to 5 P.M.), research users shifted their periods of use to off-hours; evenings, graveyard hours (12 A.M. to 8 A.M.), and weekends. But when the processor and memory enhancements were made, many users of off-hours computing shifted back to regular work hours. They chose to run large programs normally relegated to nighttime hours during the day because of the apparent increase in computer resources. In both instances, system use increased so dramatically in the first few months that users once again were faced with shifting their computer usage back to nighttime hours in order to avoid system congestion. The steady growth of the number of system users also reinforced this pattern of use of computing resources.

The increase in disk storage capacity revealed similar trends. Before the addition of more disk units, user demand or comsumption of disk storage space regularly exceeded its allocation. When storage allocations were exceeded, users had to either archive or delete storage files. But with doubled storage capacity, users were able to keep more disk files on-line. Though capacity was doubled, the rate of disk storage consumption after the addition was so steep that CSRO managers acted to take some of the additional disk units off-line. This reduced total available storage. CSRO systems programmers indicated that this was done to ensure the availability of additional storage in the near future.

The facilities for office automation were regularly used by all participants at CSRO. The components for text processing (four text editors and

[13] System load average is a measure of the level of system utilization computed and recorded by the TENEX operating system. Five-year plots of system load averages which denote the reported increase across the user communities appears in a CSRO annual report.

the NEAT text formatting system),[14] file archiving (ARCHIVE), bulletin board (BB), electronic mail (NETMAIL), and computer use monitoring (MONITOR) had been automated but not integrated into a single system (Scacchi, 1981). Together with system utilities, these components served as the facilities for office automation. We were not surprised that CSRO's participants would be among the first to acquire or develop such capabilities. Each of these system components was operational and in routine use prior to our inquiry in 1978. We therefore examined this collection of software system components which participants use in support of their research work.

There had been no formal plan at CSRO calling for the progressive adoption of this computerized work environment. Each of these system components was either acquired from other organizations (the NETMAIL subsystems) or developed locally by users (the BB facility), staff specialists (the file ARCHIVE utility), or project managers (the MONITOR program). No one person was specifically responsible for defining, organizing, using, or integrating these components into a single unified system. Each appeared to be developed independent of the others. However, the adoption and development of these components were consistent with the local organizational climate which supported and encouraged the creation of sophisticated computing environments. All computer users at CSRO were aware of and used (or encountered) most of these components on a daily basis. Yet there was no single source of information or documentation that described the uses of these systems. Instead, system users would learn these facilities through interactive system use or through interaction with other computer users.

Each of the system components was used to support either research work or communication within and among project staffs. The text processing facilities provided the production facilities for reports prepared by CSRO participants: technical reports, internal memos, research proposals, graduate theses, or system documentation. The BB facility was used as a public repository of project communications or other "public" notices. The ARCHIVE utility was used to automatically store unused computer files (e.g., old documents or source program codes), especially when on-line disk storage space allocations were near or exceeding their assigned limit. The NETMAIL programs were used to send or read electronic messages whether generated inside or outside of CSRO. Finally, the MONITOR facility was used by the computer facility manager as a

[14] The development and evolution of this text processing system is described in Kling and Scacchi (1980).

means to keep users from exceeding their allocated amounts of computer time or storage space. But participants' use of these components reflected peculiar kinds of workplace interactions mediated by the computer.

Users find one particular aspect of the computerized work environment essential for its use: a user needs a computer terminal to access and compute with system resources. All users did not have computer terminals readily accessible. Computer terminals were necessary for interactive computing and the office components were "useful" only when used on-line. Computer terminals were provided by individual research projects within CSRO according to their funding arrangements. Bigger projects had larger staffs and more terminals than did smaller projects. Project members with terminals not in use were encouraged to share them with other users. This arrangement was successful as long as there were enough terminals to go around. Thus, users in small or unfunded projects were particularly pressed when many users in the large projects wanted to compute. They might not in that case have access to system resources. In fact, after a few heated arguments over terminal use erupted between members of different projects, CSRO project managers formulated a policy establishing priorities for sharing terminals among users.[15]

The BB utility was a convivial user community facility for exchanging notes, recipes, and reviews. Everything stored in this facility (e.g., system documentation, recipes, and book reviews) was accessed in the same convenient manner. Few users reported any problems in using BB; the few complaints centered on occasionally encountering frivolous messages or having to respond to other participants who were reacting to outdated bulletins. Users appeared to be quite satisfied with this facility.

Most users reported spending 15–60 minutes a day reading or sending NETMAIL messages to each other. It was a widely used facility and was regarded as essential to the work of many participants. However, if a user received too many messages (i.e., too many message files) and exceeded his or her on-line disk storage allocation, then the ARCHIVE facility might automatically start to archive the least recently used files regardless of their content or importance.[16] To avoid this, users who received many messages (e.g., project managers) had to manage their file stores. Auto-

[15] Since not all participants might have a terminal to access their NETMAIL, this policy was circulated to everyone in *paper* form.

[16] Active participants sometimes produce or interact with 10–30 files a day. But if the space allocated to a user is exceeded, the program removes and archives the least recently used files until under allocation. ARCHIVE was programmed to visit user file directories once a day. Once archived, it took a full day to get files retrieved and restored onto the user's disk area.

matic archiving was more likely to occur to users who broke an extended pattern of system use by leaving for vacation or a conference. Upon their return, they would sometimes find many messages present and critical files missing but archived. Users who had active files unintentionally archived subsequently attended closely to their use of NETMAIL and file management.

The NETMAIL subsystems also had a few technical problems (system bugs) and were not as reliable as users expected. Certain "normal" sequences of interaction with NETMAIL would result in messages being either lost or garbled en route. For participants accustomed to communicating short messages through the NETMAIL facilities, the degraded system performance was surprising. Many tried to figure out what was wrong with the system until a common diagnosis of the difficulty emerged. Their understanding was that they were using a recently enhanced version of a NETMAIL subsystem that was not completely compatible with the current version of the computer's operating system. But with no contract service agreement with NETMAIL's developers, users chose to accommodate themselves to its erratic, mysterious, and sometimes frustrating performance in order to use it until it could be fixed.

The MONITOR program was designed to ensure that system resources allocated to each user community would be available when system use (load average) was high. This program was implemented to (1) warn users of excessive use of system resources per unit time, (2) disconnect users from the system who failed to heed such warnings, and (3) automatically send NETMAIL messages to users exceeding their allocation of disk storage warning them of pending file archiving. (Some users in turn identified these sorts of messages as bothersome, electronic junk mail.) But users on small projects with small resource allocations would not even be able to read or send NETMAIL, let alone do any serious computing during high-use periods. Users with sufficient access to the system's resources discovered a simple way to slip by the MONITOR without being removed from the system when they exceeded allocated limits. Further, they could easily slip back after a few minutes to their system resource-demanding usage and continue their work. As serious researchers sharing critical resources, they did not generally abuse this slipping capability. Instead, they would try to choose a time to compute that would allow them to get adequate system resources without being bothered by the MONITOR. Unfortunately, users in different projects or communities did not generally coordinate a schedule for using extensive system resources. Thus, for all but small project users, the operation of the MONITOR effectively served as a reminder to try to compute (hence work) during periods when

system load average is low. But for small project users, the MONITOR was a nemesis keeping them from access to system resources during peak use periods.

The computerized work environment facilitated the research activities of participants at CSRO. Participants in every project or user community reiterated this claim many times. This was a strong, pervasive belief throughout CSRO. But these automated office components often amplified the demand for a user's time, skill, and attention for managing the storage of files on disk. However, users were not dissatisfied with the conception, use, or upkeep of these components. Instead, they would sometimes go to great lengths to explain the sophistication of their computing work environment, treating the difficulties that emerged as unrelated or isolated problems (except for documentation, "which could always be better"). The difficulties that did emerge were downplayed, whereas the richness of the environment and the importance of the research was highlighted. Many participants indicated that the dilemmas that did occur could or would be easily remedied by some alternative (i.e., future) computing arrangement: it would just be a matter of time, adequate resources to do the job, and people with the skill necessary to realize it.

6.2 Data Base Management at INSURE

At the time of our field research, INSURE was one of the top 25 mutual insurers competing nationally in the insurance industry. It employed about 1100 people at its central office, the site under examination.[17] INSURE has been using computers for processing transactions related to insurance policies since 1955, and it was one of the first insurance companies to computerize. INSURE, like most insurance firms, was quite committed to computing to support the processing of high volumes of policy transactions. INSURE's commitment to computing was reflected in their allocating 7% of their total annual operating budget for computing operations, approximately $4.5 million during 1978.

An "automation committee" composed of managers from several divisions served to formulate and implement both short-term (quarter-to-year) and long-term (one to five-year) plans for computing systems at INSURE. (Often top managers did not attend these meetings, but sent their computer liaisons instead.) These plans for corporate systems comprised a portfolio of high-risk/high-payoff development projects together with low-risk projects for enhancing operational systems. These plans were integrated with those for the development of new "products" for sale—that is, insurance policy packages. The plans indicated the current priorities and negotiated settlements between the different interests represented in the automation committee.

[17] This central office is about 20 times the size of CSRO, but less than 25% of the size of the university in which CSRO is situated.

The plans also served as the directions for the Information Systems Department (ISD) for system development and evolution. However, the major settlements achieved within the automation committee often led to one division getting the support necessary for developing some new system together with some assignment of ISD staff to the project. This meant that ISD representatives on the automation committee were always influential in decisions to reallocate support staff from current projects to new ones or else to hire additional staff for the new projects. Further, if additional staff could not be hired readily, then committee members, especially in the affected user division, had to choose how to have the ISD-based support services divided between the new development project and the existing operational systems.

The principal activities at INSURE included developing insurance policy packages, selling insurance, selecting insureds, fixing premium rates, writing insurance policies, investing money, keeping accounts and statistics, paying losses, and in the course of these activities, dealing with legal problems and regulatory agencies. These activities were common to most insurance companies (cf. Riegel *et al.*, 1976). Each department was responsible for duties arising under each activity category and all were coordinated through their use of a central computer-based master file.

Computing at INSURE was used to produce analytic findings, insurance policy packages, and internal and external reports and to manage investment portfolios. Of these, insurance policy packages were the main product which INSURE agents sold to customers. Developing new products was an increasingly important line of work of actuarial staff that helped keep INSURE a going concern, competitive within the insurance industry.

By 1978, seven computing systems were in use at INSURE. Most computing services were provided by the ISD with its staff of 180. This staff supported the operation of a modern large-scale computer system, an Itel AS/5,[18] within a central computer facility. The other systems—including four different minicomputer systems and two remote data-sharing services—were decentralized and located outside of INSURE's central computing facility. Various managers reported that acquiring or developing new systems was one way to retain skilled computing staff: "keeping current with the technology," as one said. But scheduling system

[18] National Advanced Systems now sells and services these computers.

usage was a recurring problem associated with all the in-house computing systems. Processing time on the central computer was most scarce and the problem managers faced was how to minimize contention for this high-demand resource. Thus, acquiring additional computing systems or system capacity was a frequently tried strategy.

Most computing activities at INSURE centered around the operation and upkeep of the "master file." The master file was the most important computer-based resource at INSURE; it was initiated around 1960 when the firm was using a first-generation Univac-II computer. Since then, many staff have worked to expand it and maintain its integrity. Though it was called a "file," it was actually a very large data base organized as a collection of many files operating on the central Itel computer system. It contained the insurance policy records on more than 400,000 insureds.[19] This data base was primarily used in the processing of insurance policy transactions and in the preparation of actuarial studies. However, department managers also used analyses drawn from the master file to help plan and control activities in their lines of work.

Managing the master file was an important set of activities organized to handle routine policy transactions for the growing volume of business. Many of these transactions were supported by on-line (interactive) applications. Many different kinds of summary reports were generated from this data base at different times. Special software facilities were required to restructure it when new insurance products were introduced and new reports needed, or when system capacity was exceeded. Interactive facilities for selectively accessing or aggregating data were regularly used to support actuarial computations of premium rates for new product development. All our respondents reported that obtaining valid data to and from their expected location in the master file was important, and costly if done wrong. Thus, determining who would have "read" or "write" access to the master file and what software facilities would be used to interact with it was a common concern of department managers, actuarial users, and ISD staff. (While their concerns were shared, their proposals sometimes differed.)

The staff in the ISD, the automation committee, the user department–ISD liaisons, and instrumental users in each division were all interested in controlling different aspects of data management at INSURE. The managers in the ISD wanted to be able to develop and deliver on schedule systems that supported new insurance products. Delivering an operational system on schedule was one way ISD staff and managers could display their technical competence. To ensure their ability to demonstrate competence, ISD managers adopted a (two-volume) set of procedures for project planning and control in an attempt to routinize the development of these systems. However, unexpected or rapid turnover of ISD personnel could jeopardize scheduled completion of a system's development. These managers learned that adopting current computing technologies such as a new DBMS helped attract and retain skilled programming staff by keeping current with the technology.

[19] Each customer record contains about 2000 characters of information. Thus, the master file contains nearly one billion characters (or bytes) of stored information.

The automation committee decided what systems were to be built and whether they would be cost-effective when operational. Committee members were well aware that changes to the structure of the master file could be costly if many operational programs had to be converted to remain compatible. Sometimes systems intended to support new products required such alterations. The ability to systematically restructure the master file was one reason why the committee chose to adopt a modern DBMS. Another reason was to support the development of new insurance products. However, the insurance agents in the field together with INSURE's top managers actually initiated the demand for new products—and thus for new application systems.

The liaisons advocated the adoption and selection of particular systems or technologies that they believed would best serve the interests of users in their division. If the development of a new product line did not depend upon its software being integrated into the master file, then the liaisons had a prime opportunity to use their clout and take responsibility for choosing the new system and overseeing its implementation. The liaisons were quite influential in decisions to adopt new minicomputer systems for their user divisions.

Finally, the actuarial users were responsible for specifying the functions (i.e., the design) of new or evolving application programs. Actuarial users developed the products that were marketed by the agents in the field. They relied upon their analyses of data drawn from the master file to determine the premium rates and profitability of INSURE's products. Their application designs were implemented by ISD staff either as enhancements to operational systems or as new systems. These programs would access data, perform the necessary computations, and generate the reports from data located in the master file. Thus these users had to coordinate the necessary support of other actuarial staff, the automation committee, the ISD staff, and the master file with the existing operating procedures in order to analyze the requirements for these programs.

Since the early days of computing at INSURE, four kinds of software systems had been adopted. These included many stand-alone application programs, an integrated file management system, at least two user-

oriented facilities for (limited) query processing and report generation, and most recently a DBMS. The first production applications were originally developed to run on the Univac-II computer in machine code. Some of these applications still exist and operate on a regular basis.[20] The remaining systems were implemented in assembly or higher level languages (e.g., COBOL) native to IBM S/360 or S/370 computers. These software systems represented the variety of application programs for managing master file processing.

In production, many of these applications are streamed together so that the operation of one program depends on the successful operation of its predecessor(s). The failure of some of these programs could potentially stop or disrupt the work of hundreds of people at INSURE. (This was said to happen at least once or twice a year.) Programs could fail if they accessed either the wrong data set or one with previously scrambled data. ISD specialists attempted to prevent or catch these failures by incorporating extensive data validation checks into newer systems. While newer programs had these checks built into them, some of the older Univac-II programs did not. Thus, many ISD specialists reported occasionally coming to work in the middle of the night to revive a program that failed so that other production programs could run.

Automated data validation emerged as a necessary condition for ensuring the proper operation of production systems. Data had to be checked by each program to determine if they were appropriate and whether they should be treated as regular or exceptional. The major portion of most of the modern production systems at INSURE had been designed and implemented specifically to handle exceptional or errant cases. As one software systems manager noted, "90% of our code is devoted to handling 5% of the cases."

Maintaining data base applications programs was a central activity attendant to managing the master file. These systems "wore out" when they could no longer be understood and maintained by ISD staff. For example, at least one-third of the production systems for one user division (including those for policy contract administration, underwriting, and premium rate book) were coded in machine language for the antiquated Univac-II computer. The only documentation for some of these systems were source code listings four to five years old. Further, the "comments" annotating the program were sparse, few in number, and penciled in by their maintainers. Therefore, the knowledge necessary to understand and

[20] These old programs were made compatible with the present large computer by means of a series of systems programs which either emulate or simulate the operation of the earlier generation computers.

maintain these systems remains available only to the two specialists sustaining them.[21]

In order to accommodate the evolution of these systems, additional programs were built about core production systems. Although the internal structure of an old system might not be well understood by most specialists, its inputs and outputs generally were. These inputs and outputs could therefore be pre- or postprocessed to obtain some desired net computation. Old systems were treated as black boxes which could be embedded within newer programs whose remaining computations (e.g., for exceptional case handling) were understandable. When system alterations were requested, they were generally made to the periphery, rather than the core, of an older production system. Accordingly, system structure primarily evolved at its periphery rather than through its core. However, some system changes should have been made to the core system. Otherwise, computations done in periphery routines could essentially undo or work around the computations executed in the core. Staying on top of these fragmenting systems remained manageable as long as a system's behavior was understood by the responsible specialist. But his understanding depended on familiarity with how these systems were used and the quality of their documentation.

Managers at all levels had come to recognize that replacing these embedded systems was increasingly costly. Further, there were barriers to system conversion within organizational operations and in the implementations of computing systems, which in turn drove up the cost of these systems.

Specialists trained in either assembly or higher level programming languages were not willing to learn the obscure machine language systems. This professional skill was useless for them outside of INSURE: there was no market for Univac-II programmers. Similarly, within the organization, Univac-II programming skills would not be useful when these systems were replaced. The obsolete program code for these systems was effectively incomprehensible to all but two specialists in INSURE, yet these systems generally perform as expected when run (Rosenberg, 1980). The two specialists who possessed the dated skills wanted to gain programming experience with COBOL and with the contemporary DBMS. This was part of an agreement they established with their ISD supervisor to ensure (1) their position in the firm and (2) the upkeep of the antiquated

[21] Most of these systems have recently been redeveloped and coded in COBOL, the programming language in which most ISD programmers were skilled. The few remaining old systems are scheduled for replacement by summer 1981, 10 years since their conversion was initiated.

systems. These specialists wanted to develop contemporary programming skills that ensured their career options both inside and outside of INSURE.[22] Thus, they limited their commitment to working with these vintage systems.

The conversion and upgrade of the old Univac-II programs had been under way since 1971. This was in line with the automation committee's long-term plan calling for the "accelerated redesign of these systems ... for more flexibility and more modern technology." The first step taken was to acquire and modify an integrated file management system, ALIS. (ALIS was actually a package of assembly-level subroutines developed for the insurance industry that required modification in order to fit into INSURE's operations.) This system services applications for only the individual division, its end-user community. In the three years necessary to make this system compatible with the existing systems, over $2,000,000 was spent to realize its integration. Most of the applications then developed to run with this software package were originated for this system rather than being conversions of older Univac-II systems. This new system did not resolve the difficulties of data base management that had arisen with the older systems. Instead, it mitigated the occurrence of those problems for the newly developed applications.

Since ALIS was adopted, decisions to adopt other major software systems were more closely bound to the opportunities and constraints facing ISD staff. In particular, their judgment regarding the redevelopment and file conversion necessary to make an off-the-shelf system usable was central to adoption decisions. However, they invested limited amounts of time, attention, and interest in such analyses because of commitments to other, more pressing aspects of daily work (e.g., fixing system bugs, implementing user-requested system enhancements). As a result, many of these analyses were ad hoc or casual. But they served to demonstrate the analyst's conformity to organizational procedure (e.g., in conducting cost studies) as well as substantiating the decision of the involved participants for or against the adoption of some new system.

Finally, the DBMS was adopted in 1975 in accord with the automation committee's plan and the interests of ISD staff. Most of the ISD staff were subsequently trained to work with the DBMS. However, in the first three years following its adoption, only three applications were (re)developed to operate with the DBMS. ISD staff explained this low rate of development of DBMS based applications in terms of (1) the reliability of the older production systems and the skills of staff familiar with them, (2) the effective organizational policy to primarily maintain operational systems before developing new systems, and—to a much lesser extent—(3) the deterioration of their unused skills for working with the DBMS. Further, most systems specialists were assigned to maintain two systems and limited to participating in the development of only one other at a time. To ensure the spread of expertise on individual application systems among specialists as a backup measure, specialists were periodically rotated among systems. The implementation of the DBMS was only a partial success by 1978, although it offered many opportunities to the firm—including retaining talented ISD staff.

Our summary observation of data base management at INSURE is this: Even within a firm that has major organizational and financial commitments to data base management (as well as computing), there still are not enough skilled people, much less, sufficient time, inclination, and budget to convert old production systems, maintain currently operational systems, and develop new systems as desired.

[22] ISD actively pursued a policy providing that 5% of staff time be committed to training and skill improvement in system development.

7. Case Analysis

We can examine the explanatory power of the five propositions which were introduced in Section 5 by using these two cases. We believe that these cases usefully illustrate many elements of web models, but we do not suggest that these cases cover all significant aspects of computer use in complex organizations. A few special features of these cases should be kept in mind:

1. Both CSRO and INSURE are relatively wealthy firms and invest substantial resources in their computing developments (unlike Audiola, for example).
2. Within these organizations, we investigated applications whose use is relatively discretionary. Moreover, these modes of computing are much more loosely coupled than for applications which share data in real time —such as airline reservation systems or military command-and-control systems. At INSURE these are analytical programs and packages for producing actuarial analyses, while at CSRO they are a set of facilities for producing textual reports and electronic mail.
3. Most of the staff who were selected for study are highly skilled semiprofessionals (e.g., insurance actuaries) and computer specialists who support the packages they use.
4. Data were collected to shed light on certain specific questions: "How is computing embedded in the routine work of computer users?"

"How do organizational participants initiate and respond to computing innovations?" etc.

Our focus in analyzing these cases will be an activity common to many lines of computing work: using a computing system to produce reports. (These are research documents at CSRO and actuarial reports for new insurance products at INSURE.) In each case, local computing arrangements influence the ease. We first identify the lines of work and resource dependencies that participants undertake in producing their reports. Then we examine the five dynamics of computing development and use for each case.

Organizational participants encounter different elements of the local computing infrastructure in the course of their computing work. But how is the work of various participants linked to this infrastructure? Participants enact their computing work as a series of tasks, some formal and routine, others ad hoc and circumstantial. Although their work does not occur in some mechanistic manner, participants' work is marked by a sequence of events through which they consume available resources until some valued object is produced. By examining the "task chains" of various participants, we can identify how their work with computing is embedded in local computing arrangements.

Task chains link participants' instrumental work and their use of computing resources. These chains denote the commitments that individuals, groups, or organizations make to working with the available computing resources. (Task chains are not rigid. People often try to arrange these commitments in a way they prefer.) Thus, task chains denote the routine fragments of work in a production lattice local to an organization.

Consider, for example, text processing at CSRO and the computing activities in which a participant engages in producing a document:

> To produce a document of professional quality a user needs to get a computer terminal; gain access to the system; create or alter text files with a text editor, text formatter, and a spelling checker, then iterate as desired; extract document test pages for formatting verification; gain access to a suitable letter-quality lineprinter; make sure that the lineprinter has suitable printing materials; compute with system utilities to print the document; examine the printed document to assure quality "standards"; and repeat any of these should system bugs or glitches be found.

This chain outlines a recurring set of tasks and computing arrangements that must be traversed to produce a report.[23] It is assumed that, in an excursion down a document preparation task chain the user knows the various system operations or else consults system documentation or other

[23] *Computing innovations* in turn alter the sequence of elements and tasks that must be traversed.

system users for assistance. The specific order of activities performed is a matter of circumstance and practical action rather than formal procedure (Suchman, 1980). A "beaten path" down this chain may be traversed frequently when documents styled for journal publication are produced. Further, it is assumed that the task chain is performed by a person who already has a rough draft of the document in hand or in mind. We will examine task chains as we analyze the cases, especially in Section 7.5

We now turn to review the cases of INSURE and CSRO. We use the five propositions from web models examined in Section 5 and the lines of work for producing reports as our foci. Our analysis will be partial and suggestive rather than exhaustive in covering the concepts and case data introduced earlier.

7.1 The Provision of Computing Services Is Specialized

At CSRO, each system user is almost always a computing specialist. Even clerical and administrative staff are specialists in the use of local automated office facilities. In organizations whose business is computer science, this is not uncommon (cf. Gruhn and Holh, 1979). The office automation facilities are used in producing computer based reports. Each user is often solely responsible for text generation, editing, formatting, typesetting, file managing, printing, and locating and circumventing system bugs when preparing computer-based documents. The widespread routinization of these skills and task chains minimizes situations where document preparation services require skills not available nearby. This highly homogeneous distribution of specialized skills in automated document preparation results in many fewer problems of access to skilled expertise than are found in other organizations, such as INSURE. The most salient difficulties at CSRO are primarily those of access and control over local computing equipment: terminals, memory, CPU cycles.

At INSURE, access to the master file data base is often necessary in producing actuarial reports. The production lattices necessary to produce a report are diffuse and often include participants who work in various organizational subunits. A plethora of software facilities and specialists in the ISD support the routine production of these reports. However, most routinely produced reports are tied to particular data base management facilities and those specialists fluent with them. For example, the Univac-II programs selectively access the master file in producing the firm's annual report. These programs are operational only once or twice a year, and thus they have low visibility in the firm. These programs are maintained by specialists who would rather work with contemporary systems to upgrade their professional skills and career opportunities. Other spe-

cialists in the firm are disinterested in working with these antiquated systems. Though conversion of these systems is now imminent, ten years have elapsed since their conversion was initiated. This conversion was protracted due to (a) difficulties in understanding how the programs work and how they might be coupled to other systems, (b) finding and retaining specialists fluent in both antiquated machine languages and modern data base management techniques, and (c) fluctuating shortages of skilled staff, who were frequently required to support other, more visible systems which also had to be kept operational.

The distribution of report production skills at INSURE is less integrated and uniform than at CSRO, though both organizations have several computing systems. At INSURE, the provision of computing services was specialized as a necessary and sufficient condition for the professional upward mobility of many participants.

Finally, at both CSRO and INSURE, the configuration of computing hardware and software in use closely fits the routine operations in each organization as well as the professional interests of local participants. Although both office automation and data base management represent general-purpose computing technologies, the manner in which they are developed, used, and evolved is centrally dependent upon the organizational setting in which they are embedded. Thus, the system for office automation at CSRO emerged and was used in a way closely aligned to the interests of local participants. But the same automated system might be awkward to use and maintain in a setting like INSURE. Conversely, the data base management facilities at INSURE which support many routine operating procedures in that organization would likely be quite arcane, unworkable, and downright distasteful to use in a setting like CSRO.

7.2 The History of Commitments Constrains Present Choices

At CSRO, choices for how to configure an adequate computing environment hinged on the amount (and kind) of system resources that participants held were necessary for the continued growth and advance of their research work. The common interest among participants was that when the available system resources were in great demand, more system resources should be added to relieve resource contention. However, how the additional system resources would be used or how they might interact with the prior system configuration was not well understood. The choice of which resources to add were limited to those upward-compatible with the existing system configuration as well as what local participants held would be the most effective addition under present circumstances (i.e., patterns of system use). Thus, upgrading the central processor was not effective until memory capacity was doubled. But before the dual processor configuration was implemented, the memory was adequate. When both the processor and memory capacity were enhanced, the demand for disk storage soared as users pursued a more aggressive style of system resource use.

The use of multiple text editors among participants reflects a reliance on prior routine skills and familiar facilities not available in the newer editing systems. Since older and contemporary systems are concurrently in use, both must be maintained. Similarly, as reported in Kling and Scacchi (1980), the multiple versions of the NEAT text formatting system in use leads to many users becoming system maintainers in order to keep this version current with their use of it. Hence, some duplication of effort throughout the organization is necessary to sustain or upgrade operational compatibilities of these systems. This of course increases the amount of attention and effort that must be directed at maintaining a working knowledge of local computing arrangements.

At INSURE, the implementation of the family of data base management facilities reveals a different set of historical constraints. As we saw above in Section 7.1, the specialization of data base management services is bound up with the computing staff who work with them. Similarly, though generation after generation of technically improved software facilities were installed, the patterns of use of older embedded facilities and the layered embedding of these facilities beneath other systems reflect prior commitments to certain directions of system development and use. The old Univac-II programs, ALIS, and DBMS facilities were added to support the growth and diversity of applications interacting with the master file. But altering one set of master file processing procedures might affect others operating at different layers. Thus, changes to existing applications as well as the implementation of new systems were subject to an increasing array of bureaucratic procedures in an attempt to minimize possible disruptions.

The adoption of the newer ALIS and DBMS facilities was also predicated on the availability of suitably skilled staff to work with them. Such accommodations were necessary at INSURE in order to compensate for work contingencies and to retain skilled specialists in the firm. Thus, the choices for how the existing master file applications could be altered or extended were limited by the availability of specialists who understood how the existing systems operated as well as how the systems came to assume their current configuration.

At both CSRO and INSURE, participants acted to increase their control over local computing arrangements and their use in producing reports. The ways in which participants got control over local computing

arrangements were marked by commitments to acquire particular systems and to develop certain professional skills. These commitments, when fulfilled, served to increase the control of local participants over their computing setting as well as realizing additional constraints on how existing computing arrangements could be changed in the future. Such actions were taken to sustain and extend both the capabilities of their ensemble of computing systems and the operating capacities of local participants. In addition, the history of commitments in both settings was such that participants invested significant amounts of social resources (e.g., time, skill, and sentiment) in order to mobilize the mix of technological resources for use in their work.

7.3 Narrow Incentives and Opportunities Motivate Choices

At CSRO, the computing system was clearly sophisticated at the time of our investigation. We found that most participants chose to use nearly all of the automated office facilities as they became available. Part of their incentive was to be compatible with the style and pattern of work done by others in the organizations. For example, NETMAIL was the only reliable medium for sending messages to participants who worked at all hours of the day or night. Through the practical use of this facility, participants learned that they could always receive and respond to messages delivered in their electronic mailbox. In so doing, they acquired or renewed a shared meaning of a particular arrangement for interaction with other CSRO participants. Nonetheless, their regular use of the NETMAIL facilities required their attention and skill when it came to managing the contents of their mailbox and disk storage.

The expansion of the computer's processing capabilities followed a similar track. CSRO managers believed that sophisticated and abundant computer resources were necessary to retain and attract computer science researchers. In turn, the researchers at CSRO came to expect that sophisticated and abundant computing resources would be at their disposal. The level of availability of processor cycles, memory space, and disk storage was, however, frequently lower than they expected. These were less available due to the researchers' pattern of use of these same resources. When contention for system resources was continually high, researchers would alter their work periods to find a less congested time. If they were producing a report rather than developing a large program, they could easily accommodate these conditions. Conversely, if these conditions persisted, they could shift the focus of their work to report production as appropriate. But when the second processor was added, when the

memory capacity was doubled, and when disk storage capacity was enlarged, they acted to use these again abundant resources by reverting to former patterns of system use. Thus, when project leaders identified this routine of behavior, they chose to implement policies ensuring that most users would have some opportunity to use some sizable portion of available system resources: sizable, that is, to those participants working on funded projects, and less sizable for the remaining few.

At INSURE, actuarial staff sought to have their systems accommodate the development of new insurance products, while ISD staff sought to develop or maintain systems that were close to their professional ambitions. But both had to work together in order that each could pursue their own interests. When interactions between actuarial staff and ISD staff led to inconsistent or conflicting system enhancements, interdepartmental liaisons were brought in to mitigate the troublesome situations. This interactional arrangement eventually brought local authority and control over certain computing resources to the liaisons.

The liaisons also served to help regulate ongoing enhancement and growth of the facilities for managing the master file. They were in a position to aid the production of actuarial reports only to the extent that they could motivate improved service from the ISD staff or lobby members of the automation committee for additional system resources. Although they might expedite production of a report through ISD staff services, their interstitial coordinating role added to the chain of tasks that actuarial users had to negotiate in producing or modifying their reports. Subsequently, ISD specialists and actuarial users sometimes took occasion to bypass the liaison in order to produce a slightly different report. These end-runs were taken only when the concerned participants felt that this was the fastest or easiest way to complete their task. As a result, the liaisons further narrowed and expanded the ways that actuarial users could produce reports in their interest and the set of systems ISD staff would work with in pursuing their professional careers.

In both settings, participants often acted to gain more of the available system resources: to get a bigger share of a bigger pie. Increased access and use of a complex ensemble of system components facilitates new career options for participants in the two organizations. Participants' demand for access and use of system resources often exceeded the rate at which they were provided. In response, managers acted to regulate access in an attempt to achieve some near-optimal arrangement. But these arrangements did not serve everyone equally well. Thus, local computing arrangements help provide reasons for certain participants to seek ways to alter their use of available computing resources, to seek alternative

means of gaining access to these resources, or otherwise to seek ways to increase their control over these resources so that they can work with them as they desire.

7.4 Macrostructural Patterns Influence the Character of Local Computing

At CSRO, service delays, skilled staff, and communications were active concerns for many participants. Due to the homogeneity of service users and service providers, service delays were primarily distributed by project affiliation. Users in small or unfunded projects found their access to system resources was delayed, whereas users on large funded projects had much easier access. Users in this second class effectively became the providers of system access to users in the first class at times when consumption of system resources was high. Further, this relationship was reinforced when the MONITOR facility was implemented to regulate use of system resources.

The hardware and software configurations in use were salient to the professional interests and communications of local participants. Additional processing equipment was acquired as part of a strategy by CSRO project leaders to help retain and attract better computer scientists to CSRO. The BB and NETMAIL systems and the multicomponent text processing system were adopted by local specialists eager to demonstrate their technical competence and professional allegiance to peers in other settings. Further, users of the text processing facilities indicated that such facilities were necessary in order to produce documents of "professional quality." For these users, professional quality meant the appearance of the final document (i.e., formatting, type font) as well as its substantive literary and intellectual merits. The audience for these documents was of course computer science researchers in other organizations: organizations where these participants might eventually seek employment. Well-received, professional-quality reports could then be instrumental in facilitating career opportunities elsewhere.

At INSURE, the provision of computing services was clearly linked to career opportunities for both specialists and actuarial users. As software facilities for managing transactions and report generation became more embedded in organizational procedures and new facilities were adopted, finding specialists interested in working with old cost-effective applications became harder. There are few professional rewards for maintaining older systems simply to keep local users satisfied. On the other hand, in developing a portion of a new system a specialist could easily become more highly regarded both inside and outside the firm. The specialists responsible for maintaining the vintage Univac-II programs were aware of this, and thus they negotiated an arrangement whereby they would receive training and spend at least half of their time working with the contemporary facilities. Actuarial users also came to value professional computing experience.

Part of the work of actuarial users is to develop new insurance products. The design of these products is derived from computations on data stored in the master file. In addition, the development of these products requires a set of computational routines—an application program—to implement the coinciding transaction processing and report generation. Since developing new insurance products is part of the job that actuarial users perform and are rewarded for, being skilled in specifying, designing, and even coding the necessary computations is a professional asset. Many eager junior and midcareer actuaries actively pursue these activities as part of their strategy for advancement within the firm. But this experience and familiarity with computing specialist skills also helps them to better comprehend and work around the kinds of service delays that ISD specialists frequently encounter. ISD specialists, in turn, prefer to interact with those actuarial users (via liaisons) more fluent in system development than with those who are less eager to become involved with computing operations in putting together new products. Subsequently, this distribution of computing expertise across actuarial users and ISD specialists helps raise the quality of computing services and the ease with which certain actuarial reports will be produced at INSURE.

At both CSRO and INSURE, the configuration of hardware and software systems is linked to the career options of local participants. Not all participants are equally closely bound to local computing arrangements. When opportunities for professional advancement exist in the labor market, and when experience in developing or using is a major path to this advancement, the local computing systems will be further developed and used to that end. However, if participants see few or no such opportunities, then innovation, as well as the ease with which local computing systems will be used and kept operational, will decline, and the systems will become less well understood, more demanding to use, and more costly to sustain.

7.5 Computing Systems Evolve through Fitting and Packaging

Structurally, computing work is an ensemble of computing tasks. There are many task chains that intersect different participants' work. But computing task chains denote kinds of work activities that depend upon a variety of computing resources.

Existing task chains can become tangled and knotted when they interlink. This disordering occurs when participants alter, plug up, or bypass established workflows. Administrative discretion in handling an "exceptional situation" might entail skipping regular events or states in a task chain. At CSRO, an organizational fight emerged among different users over who was to have terminals at his disposal. The contention over terminals surfaced at a time when particular users hoarded terminals while trying to finish a report by a scheduled deadline. All of the affected users (about 10 individuals across three project groups at the time) utilized CSRO's text processing system to produce their reports. Project managers then intervened with a policy directive that assigned terminal-sharing conditions among all users. In affecting all system users, this policy-directed rearrangement represents an unplanned excursion around the previously established production stream for text document generation. But such a bypass was deemed necessary by project managers to ensure the continuity of work and the regular flow of reports, and to restore cooperation among users. In contrast, at INSURE, problems in handling or verifying customer transaction data occurred with sufficient frequency that most of the data management application software was developed to routinize what were previously exceptional situations.

A task chain can be unclogged through congenial, covert, or coercive negotiation between the affected participants. Another important way to unclog recurrently congested work flows is through computing innovation, be it with a new system development or an enhancement to some existing system. At both INSURE and CSRO, participants regularly untangled knotted task chains through incrementally innovating their local computing arrangements.

At CSRO, the growth of the computer's configuration did not follow a long-term plan and coherent design. Rather, the central computer system was expanded to achieve circumstantial relief from regularly congested use. As system capacities were expanded, an increased level of system resource utilization soon followed. Similarly, the office automation facilities were developed with system components either provided by vendors, imported from other research organizations, fabricated by local specialists, or enhanced from local versions that had proved reliable in regular use. This ensemble of software components evolved in a piecemeal manner rather than through a well-conceived implementation plan.

Each office automation innovation, when assimilated into the production lattice, lengthens and links the repertoire of computing tasks and computing resources in which users become versed. For example, as users worked with the text system to produce their documents, they acquired a style of working whereby text in one section or document is duplicated or relocated into another. Instead of developing new text for each section or document, many users followed this style of document generation via text replication because it is easy to do with available computing facilities. Though this may help some users produce more documents than otherwise possible for them (hence increasing their productivity), the documents are produced via very many short-lived intermediate drafts. This reflects a style of work that users follow in producing reports. Thus, the form and content of reports produced with local computing arrangements were both the cause and consequence of the practical use of automated office components.

At INSURE, facilities for data base management accumulated for more than 20 years. As each new facility was added, prior facilities became more embedded in routine operations. Similarly, as each facility became assimilated into organizational routines, it became increasingly difficult and costly to add newer systems into the assembly of embedded components. As newer components were added, specialists treated their familiarity and skill with the older system as undesirable and something to get away from in order to maintain professional mobility. But as the labyrinth of software components was meshed and coupled in new ways, actuarial users became more involved in further developing and changing local system capabilities. Developing new insurance products was their business, and reports generated by some new system configuration signified their success in getting those products developed. In sum, new computing arrangements for data base management appear to be necessary for continuing INSURE's operations, while at the same time the actions local participants take in performing these operations make the development and appearance of new computing arrangements for managing data, generating reports, and retaining skilled staff necessary.

In both CSRO and INSURE, computing innovations have to fit the career contingencies and work style of local participants as well as system configurations and organizational procedures. As computing innovations were assimilated into the background infrastructure for computing, the fit and packaging of other system components and work arrangements could shift. In these situations, the regular flow of system enhancements became both necessary and sufficient for improving the fit and packaging of local computing arrangements to the flow of work.

7.6 Synthetic Observations

Dealing with local computing arrangements is sometimes problematic, yet necessary and enriching for instrumental use. Various features of the local infrastructure support, constrain, and facilitate opportunities for

computing use (cf. Gasser and Scacchi, 1979). The technical base on which a particular computing system rests, and the networks of service organizations, vendors, and knowledgeable users nearby are supporting features of the computing environment. These are necessary; without them users don't continue to compute for very long. Similarly, these features limit how an organization's computing system is used as well as how easily it can be used.

The computing infrastructure makes possible new uses of computing. It facilitates system development and maintenance services, access to system and computing know-how, and other resources costly to readily acquire. However, these same arrangements often give rise to recurring procedural or computational bottlenecks in the production of reports or other lines of work. These obstructions are usually relieved through some computing innovation. Participants try to alter their computing arrangements in ways they find make their systems more useful and fulfill their interests. Established or embedded computing arrangements cannot be altered at random. Yet, computing innovations have to conform to or be squeezed into an existing computing system, participants' work flow, and those organizational operations that are affected.

Similarly, the distribution of computing skill plays a significant role in shaping the effectiveness of local computing arrangements. The career contingencies that are activated by increased experience with specialized computing services cannot be ignored as a factor influencing the development of local computing arrangements. Career contingencies play a major role in mobilizing bias in favor of adopting state-of-the-art systems, in avoiding working with "obsolete" systems, and in avoiding equipment from minor vendors. Thus a computing infrastructure is inherently necessary, though it concurrently constrains, supports, and enriches the uses of computing.

The computing web is always part of the context of computer use in any organizational setting. It is both cause and consequence of the actions people take in developing and using their computing systems. Any direct or indirect use of computing will be mediated by the constraints and opportunities provided through the web of people, services, resources, and organizational setting. The preceding two cases and the investigations of other analysts of computing packages indicate that a significant variety of computing arrangements does occur. Variation in these arrangements shapes the ease with which local participants pursue alternative lines of action. In all, the local computing web is a structural context in which participants innovate their computing arrangements, accomodate these innovations, and in turn evolve them into a new web.

We have used a web model for analyzing the cases of CSRO and

INSURE presented in Section 6. Earlier, we asked what insights we might gain in following a web model rather than a discrete-entity model when examining computing in an organizational setting. Seven observations can be drawn from this analysis beyond the assumptions of the web model and the five propositions examined above:

1. Computer systems are dynamic entities. Organizations continually alter their computing arrangements and "the computer system" is a linguistic fiction. Today's configuration may not have been in place yesterday, and may only last a few months.
2. The distribution of computing resources and system configurations is tied to career contingencies of local participants.
3. Control over access to computing resources is central to the productive activities of participants, though continually in flux.
4. Access to computing resources shapes the complexity of computing tasks for participants.
5. Computing innovations can either increase or decrease the complexity of local computing arrangements, depending upon how they are packaged and fit.
6. Participants negotiate significant amounts of resources in trying to develop and use local computing arrangements in a way they prefer.
7. The outcomes of negotiations in which participants engage determine (a) the growth and manageability of the local computing infrastructure, (b) what affects the costs and benefits of computing in a complex organization, and (c) how these costs and benefits will be distributed among participants.

None of these observations follows from a discrete-entity model of computing in organizations. Terms such as career contingencies and resource negotiations are not part of the conceptual vocabulary of discrete-entity models. Yet these terms help to better define the conditions to be found when people become involved with computing in their organizational settings. These seven observations are consistent with web models of computing, but they are not exhaustive. Moreover, they also do not easily align with any single analytic perspective. They draw from the three theoretical perspectives and vocabularies. But this seems appropriate in order to develop sensible accounts of the actions of participants in each of the two settings discussed.

7.7 Theoretical Perspectives Revisited

In examining these two cases we have not paid special attention to the slice of computing development and use best explained by each of the

three theoretical perspectives which can be used to frame sharp web models: structural, interactionist, and political. While we believe that workable theories will probably be framed in terms of careful models which draw from two or more perspectives (Scacchi, 1981), at times we find it useful to keep them distinct. One reason for this distinctness is that analysts of each persuasion will focus on different elements as problematic.

For structural analysts (as for discrete-entity analysts), the overall effectiveness of computing developments is a central concern in these cases. While some facilities at CSRO (e.g., BB) and INSURE work pretty much as planned and officially described, others are problematic. Sometimes expected outcomes do not materialize because the technical systems being altered are not easily modeled and there is substantial uncertainty about their behavior under dynamic conditions (e.g., upgrading memory and processors at CSRO to gain improvements in DECsystem-10 performance).

Other times, software packages will have untoward side effects (e.g., ARCHIVE erased files needed by major NETMAIL users when they left CSRO for a few days). Other facilities, such as MONITOR at CSRO could be worked around by sly users. Overall, the rich computational ecology is composed of elements which are often loosely or moderately coupled, but which can interact in complex ways under special circumstances. Moreover, computer users can influence these dynamics by the way they organize their own activities within the computational ecology.

But structural approaches are relatively mute about the forces which drive the behavior of people such as computer users and system developers. The users who work around MONITOR at CSRO or those who began to work by day when an improved DECsystem-10 performance was promised are not following some deterministic plan.

A key structural element which influences the overall effectiveness of these computing arrangements is the kinds of delays faced by computer users. The kinds of delays, their repercussions, and their sources differ substantially at CSRO and INSURE. At INSURE, access to expertise was scarcer than at CSRO and delays in (re)developing computing applications were more common. In contrast, at CSRO, some staff faced delays in getting access to terminal time and CPU cycles. In neither organization were these delays randomly distributed. At INSURE, users of older or less visible programs faced delays in having programs altered or repaired. At CSRO, the staff of smaller or less well-funded projects were more likely to face delays in obtaining computer access. The structural analyst faces an easier time in explaining the distribution of delays at CSRO than at INSURE. In short, larger projects attract critical portions of extramural funds which are essential to keep CSRO as a going concern.

Their productivity is most critical, and by providing them with the best access, CSRO's managers intelligently distribute delays in computing so as to minimize organizational uncertainty (Hickson *et al.*, 1971).[24] The political analyst has a less arcane formulation of the same phenomenon: groups with more extramural funding are more powerful, and access to computing power follows organizational power (Pfeffer, 1981).

Explaining the differential attention given to the programs running on the DBMS and those written in the obsolete Univac Autocoder at INSURE requires a richer conceptual approach. First, computer skills are highly differentiated across types of applications, programming languages, machines, and modes of computing (e.g., graphics versus networking). Computer specialists usually prefer to work with newer systems and "state-of-the-art" technologies. Career mobility, providing richer job opportunities for those with highly marketable skills, plays a strong role in encouraging programmers to avoid obsolete technologies. In addition, there is a mobilization of bias to prefer newer technologies to older ones, on aesthetic and ideological grounds.

There is relatively little behavior in these cases which requires a strongly political perspective. Since we have found political themes central in other investigations of computing in organizational life (Kling, 1978a,b,d; Danziger *et al.*, 1982), we do not believe that a political perspective is useless. Conflict is most salient when valued resources are limited, or social arrangments cannot be organized so as to satisfy the preferences of all actors simultaneously. The second of these conditions is more likely to be found in the deployment of information systems of large social scope, such as TRACKER in the Audiola case (Albrecht, 1979; Markus, 1979). At the time our data were collected, there were few sharply defined battles over scarce resources at either INSURE or CSRO. At other times, the development and use of computing may have been more bound up in intra- or interorganizational conflicts. For example, at the time data were collected, several minicomputer systems were located in departments outside of the central computer facility (ISD). We suspect that deploying the first few of these was very likely a center of conflict involving the ISD managers, managers in the computer-using department, the computer steering committee, and possibly managers elsewhere in the firm. If this decision was the center of conflict, it was long resolved by the time of our field study. The politics of computer deployment were no longer active

[24] Uncertainty absorption appears to be a necessary, but not a sufficient condition for intraorganizational power. Another critical condition met by the major projects at CSRO is that the group's work cannot be easily subcontracted (Hinings *et al.*, 1974). See Pfeffer (1981, pp. 101–115) for a careful discussion of these and other structural bases for the power of organizational subunits.

and salient for users or developers of the computing arrangements for supporting actuarial analyses.

It is also possible that there are political conflicts in other arenas which influence local computer use, but which are unknown, unreported, or taken for granted by our respondents. For example, the funding decisions within the Federal agencies which supported CSRO may have had strongly manifest politics for an analyst observing the annual budgeting for research centers. Or future computing decisions may become a more manifest center of controversy and coalitional behavior at INSURE or CSRO.

It would be extremely useful to be able to predict where and when conflicts over computing arrangements will be extremely salient, and when they will be of little significance to participants.[25] Unfortunately, the research literature is mute on many of these matters. Most analysts simply adopt a particular theoretical perspective (or mixture of perspectives) and develop what findings they can frame within it. In the literature on organizational decision making, Allison's (1971) study stands alone in examining a common body of data from more than two perspectives. But his pioneering work sheds little light on the comparative bite and relative bearing of the perspectives he examines. The literature which examines computing in particular is also rather mute on these matters. Markus (1979, pp. 238–245) suggests conditions under which formal–rational, sociocultural,[26] and political perspectives might have the greatest explanatory power for explaining the acceptance of computing innovations. This useful contribution is, however, limited in scope—since it focuses on one issue and uses one variable (e.g., locus of control) as an explanatory factor.

Rather than simply "horse racing" perspectives, we believe that it is most useful to learn what kinds of explanatory power each allows. As a research strategy, the use of several perspectives in parallel allows a richer array of behavior and potential lines of explanation to be tapped (Kling, 1982). In addition, it is still an open and useful question how the different perspectives bear on each other. For example, at CSRO, we found that the distribution of delay in access to computing (a structural concept) paralleled the distribution of power of the project groups. Similarly, a concern for career development (an interactionist element) led to differentials in the delays of programming support at INSURE. These are minor but useful illustrations of a line of analysis which merits further development.

[25] Observing the presence of substantial conflict is simply the beginning, not the end of a political analysis. See, for example, Danziger *et al.* (1982).

[26] Her model mixes elements of the human relations and interactionist perspectives (Kling, 1980).

8. Conclusions

This article articulates the assumptions of web models of computing and examines their conceptual elements.[27] Web models emphasize the way in which some focal computing resource is produced by a network of producers and consumers which we call a production lattice. The production lattice for a particular computer-based system or service is a social organization which is itself embedded in a larger matrix of social and economic relations ("macrostructures") and is dependent upon a local infrastructure. According to web models, the macrostructures of broader social relations and the local infrastructure shape the kind of computer-based service made available at each node of the production lattice. Since workable computing arrangements are bound up with the infrastructures available, and since these evolve over time, computing developments are shaped by a set of historical commitments. In short, web models view computing developments as complex social objects constrained by their context, infrastructure, and history.

Web models are most applicable to complex computer-based systems and services which couple many different groups in their development, operation, and use. Their infrastructure is inseparable from "the computer system." Huge systems such as the World-Wide Military Command and Control Systems (WWMCCS) (Comptroller General, 1979a) and the Air Traffic Control system (Wise *et al.*, 1980) most readily come to mind. But one need not search for examples at the extremes of scale or complexity to find situations where web models have more explanatory power than the conventional discrete-entity models. In this article, we examined automated information systems in two medium-sized organizations (Audiola and INSURE), and office automation in a scientific laboratory (CSRO). In these three cases, web models helped shed new light on the social activities that shape the development and use of relatively mundane computing arrangements.

Audiola, CSRO, and INSURE varied considerably in their wealth and the kinds of technology used. CSRO and INSURE were rich, and Audiola, the pedagogical case, was relatively poor. But staff in all three organizations spent considerable energy fiddling with their computing resources—reorganizing them, fitting in new resources, repackaging, etc. Each organization's current choices were influenced by the history of commitments they made. The systems we examined in these cases are not well described by the pallid labels "success" or "failure." CSRO and

[27] Precursors of the web model formulation can be found in Kling and Scacchi (1979) and have been implicit in studies done by others (Alter, 1980; Coulam, 1977; Fallows, 1981; Goodman 1979c; King and Kraemer, 1981; Kraemer and King, 1981a,b; Mumford and Pettigrew, 1976).

INSURE are both viewed as very "successful" computer-using organizations by observers inside and outside (Scacchi, 1981). Their systems are useful and used. But they are also problematic: their operation and incremental enhancement is embedded in a much larger matrix of technical and social relationships which make reliable use, purposive changes, and expected stability all uncertain.

We believe that web models shed greater light on socially and technically complex, embedded computing developments than do discrete-entity models. At best, discrete-entity models account for the differential (dis)advantage provided by a new technology, organizational arrangement, or technique. Since they are context-free, discrete-entity models can be used to describe the results of many simple experiments. In discrete-entity analyses, all things being equal is the rule, while the social setting of technical development and use is largely ignored. That neglect is usually untenable when the organizational setting or the technology itself is complex. Even simple technologies may be compromised by complex, demanding settings. Fallows, for example, reports the bizarre case of a simple flashlight developed in the United States Air Force which was expanded to a multipurpose, unwieldy, and unworkable triservice flashlight (Fallows, 1981, pp. 50–51).[28]

In this article, we examined how socially complex technologies are also "compromised" from an abstract ideal when the production lattice spans several organizational subunits and draws upon staff with different occupational specialties and lines of work. We have not organized this article to test the relative explanatory power of the discrete-entity and web models. Rather, we have used the discrete-entity model as a foil and have spent most of our attention in explaining the conceptual elements of web models. We have focused on the development of and use of computing within organizations and have ignored the interaction between the public and computer-using organizations. Web models are useful here also (Kling, 1981; Laudon, 1980; Sterling, 1979, 1980). Unfortunately, the interactions between the public and computer-using organizations have been little studied. In this section we have emphasized those empirical studies which shed sharp light on the dynamics of computing development and use, and have unfortunately reflected a significant gap in the research literature.

Despite our enthusiasm for the value of web models, they are not carefully articulated in the research literature. Many of the studies we have cited strongly bear on these models, or use them implicitly. This article advances the literature by coherently articulating the character of these

[28] See also the case of a joint Air Force–Navy tactical fighter–bomber, the F-111, reported by Coulam (1977).

models and examining some important propositions which are consistent with them. These analyses can be developed further by: (a) developing specific web models for particular aspects of computing development and use (e.g., negotiations over design, consequences of use in organizational life); (b) carefully testing web models against sharply drawn alternatives; and (c) translating web model descriptions into useful normative guidelines.[29]

Appendix A. The Structure of Computing

Section 4.1 introduced some key concepts for describing the structure of computer based systems support and operations: (1) lines of work and going concerns; (2) production lattices; (3) the structure and infrastructure of computing; and (4) the structure of computing environments. They were introduced with some simple illustrations drawn from the Audiola case. Here, their theoretical role and rationale are explained more carefully.

A1. Lines of Work and Going Concerns

If one wants to predict how people will integrate computer-based systems into their organizational activities, it helps to know what people actually do and care most about when they act in organizations. Formal job descriptions provide a useful first approximation. One expects a detective working a homicide detail and a purchasing agent for Audiola to spend their workdays doing radically different things. Purchasing agents do not wander into a city police department investigating homicides (by day), and detectives do not spend their days negotiating the purchase of electrical parts and checking on late shipments. But formal job descriptions provide little insight into the ways in which people go about their work, and even less insight into their ways of negotiating with others in their work worlds. Actual patterns of computer use depend as much on these work styles, over which many computer developers and users have substantial discretion.

If one wants to know whether a purchasing agent at Audiola will update the lead times in TRACKER, it is not sufficient to know that it is "a part

[29] Web models treat infrastructure as essential for complex technical systems. In prescriptive terms, they link new developments to their necessary infrastructural support. The normative guidelines, for example, would differ from discrete-entity guidelines by suggesting early developments of infrastructural support and strategies to circumvent specific macrostructural constraints.

of his job." Most jobs have many parts. When people have some discretion over what parts of their work to emphasize, they usually emphasize some tasks over others. Some people, such as the senior industrial engineer at Audiola, may go out of their way to play with computers. Others, such as the overloaded purchasing agents, may prefer to spend their time negotiating purchases rather than updating lead times in TRACKER.

One may advance many different explanations for differences in people's style of computer use. Differences are common and styles of computer use cannot be easily predicted a priori by knowing a person's formal responsibilities.

Line of work (Gerson, 1982) connotes what people actually do in a job, in contrast with their formally defined activities. It helps us express such ideas as: "Purchasing agents see their line of work as negotiating sales rather than feeding computers data." "Some engineers find developing new computer applications to be an entertaining and absorbing activity which they can justify in their lines of work." Or: "Trust investment counselors don't simply act as passive advisors. Their lines of work require that they convince their clients that they are making sound investments. Any device that helps their negotiations, whether a catchy economic theory or a nice computer graphic display, is likely to be tried to keep turning over clients' funds and making the commissions."

The activities undertaken by an organization, and their relative emphases, also differ from those that are described by a list of formal goals or defining principles. For example, elementary schools do not just educate children. They socialize, baby sit, and provide a place for kids to meet other kids. They serve as, among other things, places of employment for teachers, and consumers for the textbook and school supply industry. It is not enough to say that schools serve many roles, as if there were a fixed list. In the terms of Everett Hughes (1971), schools are *going concerns*. This term indicates the overlapping activities that enable schools to provide a wide array of participants with value enough to perpetuate the activity. Because most organizations, like schools, encompass an array of overlapping and sometimes conflicting lines of action, programs for planned social change which are narrowly conceived often fall far short of their announced goals.

The formal goals of an organization are often insufficient to explain why certain critical patterns of computing development and use occur. For example, in Sections 6 and 7, we reported how the staff of a research institute which employs automated text processing became quite concerned with the typographical quality of their papers in addition to their scientific and literary merits. Publishing reports with attractive type fonts developed into an index of professional quality, and helped the institute maintain an image of high professional quality within its research community. "Lines of work" and "going concerns" are more apt expressions than "formal tasks" and "official goals" for examining the ecology of actions (Long, 1958) which compose organizational life and into which computing developments are grafted. Additional depth and detail may be found in Strauss (1978) and Perrow (1979).

A2. Production Lattices

When people interact with some narrowly defined computing resource (e.g., waiting for a computer-generated report, altering a program, using a text editor), they often deal with a wider array of computing resources and social relations within which the focal resource is embedded. Users of computer-based information systems differ considerably in the extent to which others provide their critical information and computing resources.

Near one extreme is the person who depends upon many different groups and organizations to provide appropriate services.[30] In an organizational setting, where the labor essential for collecting, organizing, coding, and auditing data is spread over many subunits, and computing operations are handled in another group, some users simply could not take over all these activities. Even to take on some of these activities would compromise their own time devoted to their current lines of work.

Some examples may help illustrate this common and important situation. Suppose that the production scheduler at Audiola[31] wants a new report format. He is unlikely to simply reprogram the TRACKER system and autonomously develop and receive reports in the new format. In typical situations, he is likely to negotiate the system change with several parties: other schedulers (to get a consensus in favor of the change), the material control manager, the data processing manager, and the programmer working on TRACKER. Other parties could also be involved if the change were large enough to require the approval of the data processing manager and a data processing steering committee.

The list of potentially active parties does not end there. *Other departments* such as purchasing or receiving could be involved if the new report

[30] Closer still to this extreme is the client of a computer-using organization who depends upon some computer-based record for service. Not only does he depend upon the staff of the service-providing organization to keep accurate records, render services as contracted, etc., but also he depends upon them to correct errors and even set up adequate procedures for ensuring that errors are easily detected and corrected. Sterling's study of the difficulties that consumers have in correcting errors in automated billing systems suggests that these latter lines of action are often difficult (Sterling, 1979).

[31] See Section 3 for the Audiola case.

included new data which they would have to provide. *Other organizations* could also be involved, less directly: the firm which originally developed TRACKER didn't provide adequate software documentation, thereby making it hard for the programmer to make sense of some parts of the code and consequently increasing the time and cost of certain alterations; Datacrunch's sales representative might have been encouraging the data processing manager to upgrade to a Datacrunch 2000/A, and temporarily to suspend alterations being made to programs written in DATAMAZE. *Other actors,* more remote and less interested in the activities at Audiola can also be influential: local colleges teach programming using machines from better known vendors such as IBM or DEC. They help create a labor pool of programmers who are experienced in IBM or DEC computing environments and to whom Datacrunch machines may appear alien and less interesting.

In short, the production scheduler may have to interact with a relatively large set of individuals, groups, and organizations to make even a "simple" report change.

At the other extreme is the person who provides most computing services for himself. The person who balances his checkbook with his own pocket calculator using data he enters exemplifies this extreme. Close to this extreme is the person who uses a personal computer for some "self-contained" activity such as writing reports or making simple financial calculations using data which he has provided and entered. To the extent that all critical computing resources are controlled by the user (e.g., access to equipment, programming, data entry, maintenance), the computing world dramatically collapses.[32] Of course, no one is wholly isolated, and even the one-man computer center may depend upon others for advice about how to best use his equipment, and upon equipment vendors for hardware repairs and additional equipment. But, for this one user, many of the negotiations over access to computing resources depend on taking into account the advice of others and making up his own mind—rather than having to convince a widely dispersed array of actors with other interests at heart.

[32] The particular fascination of microcomputers is that they allow more developers and users to exercise larger self-control over what has heretofore been one of the scarcest and most expensive computing resource: CPU cycles. These developments, which are often hailed as the "wave of the future," are misperceived when it is assumed that the use of a microprocessor automatically collapses the rest of the computer users' dependencies into almost complete self-sufficiency (Garrett and Wright, 1981).

Access to computing resources is differentiated and organized. Even access to personal computing resources is organized through the market arrangements for selling systems, expertise, and software (Gasser and Scacchi, 1979). Most automated applications—such as information systems, statistical analysis, and text processing—depend on access to a variety of CRs: data, applications, support software (e.g., operating systems and software tools), and various kinds of expertise. Some applications require little sharing of data. Computational resources may then be segmented so that individuals can use their own machines. For other applications, which require the sharing of large amounts of data among users, such as air traffic control (Wise *et al.*, 1980) or TRACKER at Audiola, the introduction of microcomputers will be less significant in simplifying the social worlds of computer use.

Between the two extremes of users—those who are implicated directly and inextricably in a vast web of social and technical arrangements and those who can isolate themselves—are many computer users (and software developers): (*i*) those who depend on others for varying fractions of their critical CRs; and (*ii*) those for whom negotiating new resources depends upon a smaller or larger collection of resource controllers and interested parties.

We use the terms "production chain" and "production lattice" in characterizing the way in which a user of computer based services receives critical computing-related resources from others. Consider a chain of provider–client relationships which connects the end-users of information service to providers of data and raw computing (CPU cycles). The user of an information service (such as a person who receives a computer-printed check or report) as he looks into the underlying computer system sees a sequence of groups who: (a) gather and prepare the data for routine processing; (b) develop and operate the computer application(s) which process the information; and (c) produce and provide raw CRs, such as CPU cycles, compilers, etc. We call such a chain of providers and clients who support the production of some specific computer-based service a *production chain*. The production chain for a specific computing service, such as receiving monthly status reports or credit card bills, becomes particularly visible when a person or group which is outward on this production chain tries to alter information or computing arrangements requiring changes several steps down the chain (Kling and Gerson, 1977).

In fact, this production chain is better modeled as a lattice of social relations since links (a)–(c) can be complex patterns of groups which may be located in several distinct organizational units. For example, in automated information systems such as TRACKER, data are collected from several departments, audited by other departments, and processed in still other departments. Computing services may even be provided by a third party, and the applications software by a fourth party. Under these conditions, a "computing system" depends so critically upon a social organization that it is at least as appropriate to call it a social organization as it is to call it a set of information flows and algorithms or a machine.

These groups in a production lattice may be bound together through different kinds of social relations. Two common arrangements are for the

groups to be[33]: (*i*) subunits of the same organization sharing a common locus of authority; or (*ii*) customer and client with a contractual obligation for specific services.

Examples of alternatives (*i*) and (*ii*) are commonplace: data gathered by one group are keypunched by another; applications used by one group are provided by another (such as a local programming group or an outside vendor), and so forth.

These arrangements can be particularly troublesome for software products which depend upon extremely specialized expert knowledge for their construction and maintenance, and which are often used long after the designers have left the scene.

New computerized technologies have no unequivocal relationship to the length and complexity of production chains in which they are embedded. Stand-alone microprocessor applications which are operated by their users clearly shrink the length of the chains they face. But other technologies such DBMS and networks provide additional data manipulation and communication leverage at the expense of lengthening the production chains. A DBMS requires a data base administrator to help organize data and files. Networks add new protocols that facilitate internode communication. They also add third parties who provide network communications and interfacing.

A3. The Structure and Infrastructure of Computing

The sequence from information system user to provider of raw computing is layered into the nodes of a production lattice. Each actor, looking inward toward the basic CRs on which he depends, sees a structuring of particular resources to which he has direct access as well as other CRs which help produce his own resources, but which lie out of reach.

We find the term *infrastructure* useful to describe certain critical resources, whether close at hand or more remote. Urban planners and economists treat the "infrastructure of a city" as a basic physical plant which helps support other valued activities. Typically, the physical infrastructure of a city includes its utilities and transportation system. Ilchman and Uphoff (1969, p. 35) have usefully expanded the term to include a wider variety of administrative, social, and political investments to make a service more efficient and more responsive. We apply their expanded conception to the case of computing developments.

[33] Either kind of relationship may work well on the average. Difficulties occur for (*i*) when the common locus of authority is so far up the organizational hierarchy that authoritative action cannot be relied upon to resolve relatively minor but frustrating problems. Difficulties occur for (*ii*) when the contract does not carefully specify the range of necessary services in adequate detail.

In general, each actor sees a structured set of computer-based services or resources which are directly available. Supporting these are other resources which we label as infrastructure. In practical terms, a user of a computer-based service has access to certain focal CRs. These differ, depending on where in the lattice of service provision he sits and on the specifics of his setting. A purchasing agent at Audiola, for example, may have both printed reports and a computer terminal, while a production controller may have only a set of printed reports. These resources are structured insofar as they are available at certain times, are placed in certain locations, may be altered in a specific manner, etc.

Those investments and procedures which support the provision of these reports we treat as "infrastructure for computer use." This infrastructure includes documents and training aids to explain report formats. It also includes experts inside or outside the organization (e.g., in professional associations) to explain strategies of computer use and to build morale.

More important, infrastructure also denotes the procedures and institutionalized values for providing these supporting resources. This includes administrative procedures for ensuring that systems are adequately documented, that specialists and users are trained in new technical skills as necessary, that errors can be routinely corrected, that strong contracts can be written to ensure that outside vendors deliver what they promise on time and within budget, and that resources may be efficiently gained or relinquished (e.g., having a report canceled). Moreover, the set of CRs that are one layer removed, which support, for example, TRACKER, are also infrastructure investments for the information systems user: the Datacrunch 1800/3, the DATAMAZE language, the programmers, etc.

One step away in the production lattice are those groups which support the application, such as TRACKER at Audiola. For a programmer supporting TRACKER, the available CRs include TRACKER, DATAMAZE, the Datacrunch 1800/3, documents which describe TRACKER's programs and data, other staff who understand TRACKER'S operation, etc. The infrastructure of computing support for him includes any software development tools, documents which describe the operations of DATAMAZE and other computing equipment, professional associations and trade journals, procedures for taking outside seminars to improve his skills, provisions for hiring consultants, etc.

The most obvious way that equipment vendors support the infrastructure of local computing is by providing more flexible equipment. But equipment vendors also exert more subtle, but nevertheless significant in-

fluence through other means. Good field support when equipment fails is critical for a smooth-running computer-based system, but vendors vary considerably in the care with which they provide rapid service. At the high end, large main frame manufacturers will sometimes station a full-time person at sites using their equipment on a scale which tallies to millions of dollars a year. At the lower end, new vendors may have a weak nationally organized service setup. However, some customers of even better established vendors will at times have trouble with reliable service support. But troubles also occur when the computer using organization cannot find adequately skilled staff to operate or maintain local equipment.

Each technique or device within a technology is designed to be used within a particular technological system. There may be technical linkages between different parts which must go together to operate efficiently. Technological innovations then must fit in and link up with existing system parts. This is a basic explanation accounting for the development and evolution of a technological system. Stewart (1977, p. 108) makes this point with the following example:

> An organization adopts new accounting techniques that involve computerization. Once introduced on any scale, this provides an incentive for improved computer techniques. These in turn may lead to other changes in managerial methods, including computerized payments and stock control. Simultaneously, the accounting innovation imposes requirements for changes in training requirements. Once these changes in training are achieved, the new work skills permit other innovators to take for granted a particular type of trained labor force and design new techniques against this background.

The skills which may be "taken for granted" are often strung out across the lattice of social relations which support the production of a specific computerized service. As Stewart indicates, investment in some kinds of skills precedes the development of other innovations and their requisite skills. Troubles occur, however, when the earlier skills cannot be taken for granted. At Audiola, for example, the industrial engineer who was introducing the work-in-process (WIP) system would have liked to take for granted the presence of TRACKER, DATAMAZE, the Datacrunch 1800/3, and the skills of certain personnel—of programmers to maintain this equipment and of the production administrators to use it. By 1979, these skills could not be easily taken for granted at Audiola because of staff turnover. Moreover, since the lattice for producing reports from TRACKER or the new WIP system was strung out over two organizational units and several occupational specialties, no one person knew all the relevant computing and production management technologies to institute new training courses in all the relevant skills.

This discussion introduces the concept of "infrastructure" and indicates how computer based services depend upon a wide variety of infra-

structural investments for smooth operation. These investments are often invisible. For example, the time that users spend learning how to use a new computerized system almost never shows up in an organization's accounting of computer costs; the infrastructural investment in good documents and training aids is more likely to show up. Infrastructural developments are often overlooked. They are distributed along the production chain and usually are not under the control of one organizational actor. They are often underdeveloped, and consequently increase the rigidity of computing and costs of computer use for others down the production chain.

A4. The Structure of Computing Environments

When the production lattice which supports a particular computerized service crosses an organizational boundary, or its provision requires the support and collaboration of people with different occupational specialties, then that computer based service is embedded in a larger mosaic of social worlds and markets. These include relevant labor markets, and the organizations within which the production lattice is embedded. Their activities influence the practice of computing, the products desired, and the products which can reliably be delivered.

It is helpful to have a term denoting this larger mosaic of social worlds, social relations, markets, etc. The theoretical vocabulary of sociology provides us with two interesting candidates: *environment* and *macrostructure*. "Environment" is the more common term; it has been extensively used by organizational theorists (Thompson, 1967). It denotes something "outside" the focal social organization or CR. While theorists often refer to certain properties of an organization's environment, such as its "uncertainty" or "turbulence," the term "environment" connotes little structuring. Some analysts such as Withington simply segment the environment of computing development (e.g., "a user environment") (Withington, 1979). This segmentation helps, but we still find the resulting usages understructured.

In contrast, "macrostructure" denotes the set union of broader organizational arrangements, and extraorganizational arrangements which influence the production lattice. The main limitation of this term is the possible presumption that these social and economic arrangements are very well ordered when they rarely are.

Our solution is to employ the term macrostructure to denote these broader relationships and to suggest that they are not amorphous. Some of the key macrostructures which have important influences on the character of computing developments and use in an organization are:

Labor Markets. During the 1970s (and now the 1980s), there has been a severe shortage of skilled computer specialists. To develop and support computerized systems, one needs skills which are often specific to the kind of service provided and the equipment used. While these skills may be learned on the job (and some skills almost always are), training people with "near" skills may delay for several weeks or months their ability to work with adequate skill and independence. If an organization attempts to hire a person with the exact skills it seeks, it will be searching in labor markets of different density. There are many more programmers with experience programming or using equipment from popular vendors such as IBM, DEC, or Honneywell than there are with experience with less popular vendors such as Harris, Data General, or Cray. Similar observations apply to the kinds of applications experience sought (e.g., banking administration versus actuarial computations versus real-time satellite communications), programming languages known, kind of managerial experience, etc. (Kling and Gerson, 1978). Some of these combinations are so common that in any metropolitan region many people will be found in the applicant net cast by a computer-using organization. Other combinations will be so rare that the organization will have limited choice of skilled applicants and may have to bear a large portion of the requisite training costs and forego very skilled staff support in the interim.

Organizations also differ in their ability to attract skilled labor. Organizations vary in their geographical location, pay scales, "image and organizational climate," and the opportunity structure within the organization. Some organizations are simply more competitive than others in providing their staffs with better resources. It is unlikely that all organizations will be equal in their appeal, and those that are poorer will suffer in the competition for skilled staff. There may be little additional penalty for being less competitive unless the poorer organizations adopt technologies whose infrastructural costs exceed their resources.

Organizational Procedures and Policies. Organizational subunits are not masters of their own destinies. Many of their policies for handling expensive resources are negotiated with the central administration of the larger organization. In particular, policies for hiring staff, setting salaries, purchasing equipment, letting consulting contracts, and overall budgeting influence the development of local computing resources.

Appendix B. Four Theoretical Perspectives

A theoretical perspective is a collection of key theoretical concepts for analyzing the role of computing in social life or organizational activities (Table I). Each perspective also includes central ideas for applying it.

Four perspectives dominate the literature.[34] Each casts a different light on the significant aspects of computing development and use, as well as on the terms in which they should be understood. (Table I lists the key concepts used by each perspective to characterize computing arrangements and the organizational settings in which they are developed and delivered.)

The *formal–rational* perspective focuses upon the formal goals of organizations and the publicly sanctioned means to achieve them. Rational analysts usually examine computer applications and their underlying technologies as tools to meet relatively clear goals. The provision of computing is also viewed as a set of clearly defined tasks which depend upon specific technologies (e.g., data base systems) and personal skills (e.g., programming, requirements analysis, training). The major values of computing are better decisions or faster production of routine transactions and reports. Problems of computer use are viewed as being relatively individual, and best solved by separation into independent, individually solved subproblems.

Scholars who adopt a *structural* (or cybernetic systems) approach view organizations and their subunits as entities pursuing multiple (and possibly conflicting) goals in uncertain environments. In contrast with the analysts applying the formal–rational perspective, who tend to look for universally effective strategies of computer use, structural analysts view organizational activities as actions which may be most effective only under certain carefully defined conditions (contexts).

Structural analysts view organizations as imperfectly coupled task systems which are bound together by noisy channels for communicating and transferring various materials. The transmission of information and transportation of goods, as well as the making of relevant decisions, will entail delays. Individuals are viewed as having limited cognitive capabilities. To compensate for the limited information processing capabilities of their individual members, organizations develop standard operating procedures so that recurring situations do not have to be continually redecided.

Both formal–rational and structural analyses have a strong normative thrust, and usually link preferred computing arrangements to organizational efficiency, effectiveness, or flexibility.

Interactionist and *political* analyses usually aim at descriptive accuracy rather than normative prescription. In addition, interactionist and political analyses do not assume the possibility of consensus on goals or effective arrangements among groups or individuals. Rather, both perspectives view organizations as aggregations of differentiated groups with parochial interests and isolated views—groups which are often in conflict.

[34] Kling and Scacchi (1980) and Kling (1980). See also footnote on p. 23.

TABLE I

THEORETICAL PERSPECTIVES ADOPTED BY SOCIAL ANALYSTS OF COMPUTING

	Systems rationalism		Segmented institutionalism	
	Rational	Structural	Interactionist	Organizational politics
Technology	Equipment as instrument	Equipment as instrument	"Package" as milieu	Equipment as instrument
Social setting	Unified organization: 1. The user 2. Tasks and goals 3. Consistency and consensus over goals (assumed)	Organizations and formal units (e.g., departments): 1. Formal organizational arrangements 2. Hierarchy of authority, reporting relationships	Situated social actors: 1. Differentiated work organizations and their clientele 2. Groups and overlapping and shifting interests	Social actors in positions: Individuals/groups and their interests
Organizing concepts	Rationalization Formal procedures Individual ("personality") differences Authority Productivity	Organizational structure Organizational environment Communication Standard operating procedures Organizations resources and rewards	Defining situations Labeling events as a social construction Work opportunities/ constraints Package	Work opportunities/ constraints Power Social conflict Legitimacy Elites
	Need Cost–benefit Efficiency Task	Uncertainty absorption Rules Authority/power Information flow	Career Legitimacy Social world Social conflict Interaction Role Negotiations	Coalitions Political resources Bargaining Power reinforcement
Dynamics of technical diffusion	1. Economic substitution—"Meet a need" 2. Educate users 3. A good technology "sells itself"	The fit between attributes of the innovation, organization, and environment enable diffusion	Accepted technologies preserve important social meanings of dominant actors	Accepted technologies serve specific interests
Good technology	1. Effective in meeting explicit goals or "sophisticate" use 2. Efficient 3. Correct	Helps organizations adapt to their environments	Does not destroy social meanings important to lower level participants, public, and underdogs	Serves the interests of all legitimate parties and does not undermine legitimate political process
Workplace ideology	Scientific management	Scientific management	Individual fulfillment through evocation of valued social meanings	[Several conflicting ideologies]

The *interactionist* perspective focuses upon the generation and maintenance of shared meanings in organizations, and the ways in which shared meanings determine the character of computing arrangements. Groups in organizations are differentiated partly by the meanings they ascribe to objects and actions. Within groups, common meanings and ideologies develop.

The prevalence of particular computing arrangements or a particular set of beliefs about computing is not necessarily predictable in the view of interactionist analysts. Instead, the interactionist view holds that the "negotiated" outcomes of social interactions determine computing arrangements and beliefs. However, certain people or groups with greater or lesser power to define the situations in which interaction takes place, or to establish the terms of argument, may bias the outcomes of interactions in their favor (Strauss, 1978a). In addition, people's participation in a variety of reference groups and social worlds, both within and outside the organization, informs the meaning computing has for them.

Preferred computing arrangements, from an interactionist perspective, are those which align with prevailing beliefs about computing or which provide arenas for productive social interaction, rather than those which are, for example, demonstrably most efficient or easiest to use. Accordingly, problems with computing are defined and prioritized based on the ideologies of computing and work which prevail in the setting at a particular time.

The primary focus of the *organizational politics* perspective is upon the power relationships in organizations and their ramifications for computing development and use. This perspective, like the interactionist perspective, presumes no underlying harmony of goals or processes in organizations. Instead, organizations are seen as aggregates of groups with diverse interests, relatively unequal power or authority, and shifting alliances. The basic aims in organizations, from this perspective, are the survival, self-aggrandizement, and expansion of key groups. Both conflict and cooperation often become important strategies for action in organizations. Prevailing computing arrangements are not predetermined, but are the outcomes of intergroup struggles.

Organizational politics analysts and interactionists differ, however, on some critical matters. Interactionists are preoccupied with the ways in which actors define their situations. For interactionists, formal control over certain resources means relatively little, whereas for organizational political analysts it often means a great deal.

Access to computing resources will most likely be unequal, from an organizational politics perspective, and some groups will receive better, faster, or greater amounts of computing services than others. Preferred computing arrangements will be those which support the agendas of the most powerful participants. An important role for computing, from an organizational politics perspective, is the use of computing to enhance key actors' control over resources and to establish or consolidate their power. Computing problems are prioritized depending upon the relative power of those affected by the problems. Thus minor technical problems may still create major political problems, and receive high priority.

Finally, within each of these four perspectives, one may frame many different specific theories or models—integrated sets of propositions. Thus, one rational theory of the persistence of computing arrangements may be based on the assumption that arrangements will persist if they continue to yield organizational benefits which exceed their operating costs. An alternative rational theory may be based on the assumption that computing arrangements will persist if they continue to provide benefits, and any alternative would be more costly to install and operate. All rational theories, nevertheless, may share some common conceptual elements and assumptions.

Similar observations apply to each of the other theoretical perspectives. Common conceptual elements and premises define each perspective, and many different and specific theories may be framed within each perspective.[35]

ACKNOWLEDGMENTS

Preparation of this article was supported in part by NSF grant MCS 77-20831. We wish to thank Steve Franklin, Peter Freeman, Les Gasser, Seymour Goodman, Suzi Iacono, John King, Charles Perrow, Rob Rittenhouse, Dave Sheldon, and Lee Sproull for providing helpful comments on earlier drafts of this article. Eli Gerson, Sara Kiesler, and Ken Kraemer participated in some specially useful discussions as critical sections of this article developed.

REFERENCES

Ackoff, R. (1967). Management misinformation systems. *Manage. Sci.* **14**, B147–B156.
Albrecht, G. (1979). Defusing technical change in juvenile courts. *Sociol. Work Occup.* **6**(3), 259–282.

[35] For example, Danziger *et al.* (1982) examine three political theories of computing use in organizations: pluralist, elite control, and reinforcement politics. Each of these shares the common conceptual elements of organizational political theories (e.g., conflict, control, interests served, coalitions). They also share certain assumptions of all political theories: organizations are best viewed as coalitions of groups which act to improve their control over critical organizational resources. The three theories do, however, provide different answers to the question of who controls the major decisions about computer use and automated data analyses in organizations: No one group (pluralist), technical experts (elite-control), or groups which are already powerful (reinforcement politics).

Alter, S. (1980). "Decision Support Systems: Current Practice and Continuing Challenges." Addison-Wesley, Reading, Massachusetts.
Anderson, R. M., and Coover, E. R. (1972). Wrapping the package: Critical thoughts on application software for social data analysis. *Comput. Humanities* **7**, 81–95.
Bremer, J. W. (1976). Hardware technology in the year 2001. *Computer* (Dec.), pp. 31–36; also in Podell and Weiss (1980).
Brewer, G. (1973). "Politicians, Bureaucrats, and the Consultant: A Critique of Urban Problem-Solving." Basic Books, New York.
Broad, W. J. (1980). Computers and the U.S. military don't mix. *Science* **207**, 1183–1187.
Burch, J. G., Strater, F. R., and Grudnitski, G. (1979). "Information Systems: Theory and Practice" (2nd ed.). Wiley, New York.
Burns, J. C. (1981). The automated office. *In* "The Microelectronics Revolution" (T. Forester, ed.). MIT Press, Cambridge, Massachusetts.
Caro, R. (1975). "The Power Broker: Robert Moses and the Fall of New York." Random House, New York.
Carzo, R., and Yanouzis, J. (1967). "Formal Organization: A Systems Approach." Dorsey, Homewood, Illinois.
Colton, K. (1978). "Police Computer Systems." Lexington Books, Lexington, MA.
Comptroller General (1976). "Improvements Needed in Managing Automated Decision-Making by Computers throughout the Federal Government." Rep. No. FGMSD-76-5. U.S. General Accounting Office, Washington, D.C.
Comptroller General. (1978a). "The Air Force Continued to Develop the Advanced Logistics System—A Program It Was Directed to Cancel." Rep. No. LCD-78-108. U.S. General Accounting Office, Washington, D.C.
Comptroller General. (1978b). "Inadequacies in Data Processing Planning in the Department of Interior." Rep. No. FGMSD-78-41. U.S. General Accounting Office, Washington, D.C.
Comptroller General. (1979a). "The World-Wide Military Command and Control System—Major Changes Needed in Its Automated Data Processing Management and Direction." Rep. No. LCD-80-22. U.S. General Accounting Office, Washington, D.C.
Comptroller General. (1979b). "Database Management Systems—Without Careful Planning There Can Be Problems." Rep. No. FGMSD-79-35. U.S. General Accounting Office, Washington, D.C.
Comptroller General. (1980). "The Navy's Computerized Pay System is Unreliable and Inefficient—What Went Wrong?" Rep. No. FGMSD-80-71. U.S. General Accounting Office, Washington, D.C.
Comptroller General. (1981). "Federal Agencies' Maintenance of Computer Programs: Expensive and Undermanaged." Rep. No. AFMD-81-25. U.S. General Accounting Office, Washington, D.C.
Conery, J. (1980). "Metaphors of Instrumental Computer Use: A Case Study" (working paper). Public Policy Research Organization, University of California, Irvine.
Cornehls, J. V., and Taebel, D. A. (1977). "The Political Economy of Urban Transportation." Kennikat Press. Port Washington, New York.
Coulam, R. (1977). "Illusions of Choice: The F-111 and the Problem of Weapons Acquisition." Princeton Univ. Press, Princeton, New Jersey.
Cyert, R., and March, J. (1963). "A Behavioral Theory of the Firm." Prentice-Hall, Englewood Cliffs, NJ.
Dale, A. (1979). Database management systems development in the Soviet Union. *ACM Comput. Surv.* **11**(3), 213–226.

Dalton, M. (1959). "Men Who Manage." Wiley, New York.
Danziger, J. (1979). The "skill bureaucracy" and intraorganizational control. *Sociol. Work Occup.* **6**, 204–226.
Danziger, J., Dutton, W., Kling, R., and Kraemer, K. (1982). "Computers and Politics: High Technology in American Local Governments." Columbia Univ. Press, New York.
Dearborn, D. C., and Simon, H. A. (1958). Selective perception: A note on the departmental identification of executives. *Sociometry* **21**, 140–144.
Dery, D. (1977). The bureaucratic organization of information technology: Computers, information systems, and welfare management. Ph.D. Dissertation, Dept. of Public Policy, University of California, Berkeley.
Ellis, C. A. and Nutt, G. J. (1980). Office information systems and computer science. *ACM Comput. Surv.* **12**(1), 27–60.
Evans, L. B. (1981). Industrial uses of the microprocessor. *In* "The Microelectronics Revolution" (T. Forester, ed.), pp. 138–151. MIT Press, Cambridge, Massachusetts.
Fallows, J. (1981). "National Defense." Random House, New York.
Farberman, H. (1975). A crimonogenic market structure. *Sociol. Q.* **16** (Autumn), 438–457.
Forester, T. (ed.). (1981). "The Microelectronics Revolution." MIT Press, Cambridge, Massachusetts.
Garrett, J., and Wright, G. (1981). Micro is beautiful. *In* "The Microelectronics Revolution" (T. Forester, ed.), pp. 488–496. MIT Press, Cambridge, Massachusetts.
Gasser, L., and Scacchi, W. (1979). "Toward a Social Framework for Understanding Personal Computing." Tech. Rep. No. 142. Dept. of Information and Computer Science, University of California, Irvine.
Gerson, E. (1982). Scientific work and social world. *Knowledge* (in press).
Gessford, J. (1980). "Modern Information Systems." Addison Wesley, Reading, Massachusetts.
Goodman, S. E. (1978). The Soviet bloc's unified system of computers. *ACM Comput. Surv.*
Goodman, S. E. (1979a). Computing and the Soviet economy. *In* "The Soviet Economy in a Period of Transition" (John Hardt, ed.). Joint Economic Committee, U.S. Congress, Washington, D.C.
Goodman, S. E. (1979b). Software in the Soviet Union: Progress and problems. "Advances in Computers" (M. C. Yovits, ed.), Vol. 18, pp. 231–287. Academic Press, New York.
Goodman, S. E. (1979c). Soviet Computing and technology transfer: An overview. *World Politics* **31**, 539–579.
Gotdiener, M. (1981). "Computer Innovation in a Powerful Organization: The Case of the FBI." Department of Sociology, University of California, Riverside. (Unpublished.)
Gruhn, A. M., and Hohl, A. C. (1979). A research perspective on computer-assisted office work. *IBM Syst. J.* **18**(3), 432–456.
Hickson, D. J., Hinings, C. R., Lee, C. A., Schneck, R. H., and Pennings, J. M. (1971). A strategic contingencies theory of intraorganizational power. *Admin. Sci. Q.* **16**, 216–229.
Hiltz, R. S., and Turoff, M. (1978). "The Network Nation: Human Communication Via Computer," Addison-Wesley, Reading, Massachusetts.
Hinings, C. R., Hickson, D. J., Pennings, J. M., and Schneck, R. E. (1974). Structural conditions of intraorganizational power. *Admin. Sci. Q.* **19**, 22–44.
Hughes, E. (1971). Going concerns: The study of American institutions. *In* "The Sociological Eye: Selected Papers on Institutions and Race," pp. 249–327. Aldine, Chicago, Illinois.

Hussain, D., and Hussain, K. M. (1981). "Information Processing Systems for Management." Dorsey, Homewood, Illinois.
Ilchman, W. F., and Uphoff, N. T. (1969). "The Political Economy of Change," pp. 208–255. Univ. of California Press, Berkeley.
Inbar, (1979). "Routine Decision-Making: The Future of Bureacracy." Sage Publ., Beverley Hills, California.
Kanter, J. (1977). "Management-Oriented Information Systems" (2nd. ed.). Prentice-Hall, Englewood Cliffs, New Jersey.
Keen, P., and Scott-Morton, M. (1978). "Decision-Support Systems: An Organizational Perspective." Addison-Wesley, Reading, Massachusetts.
Kleinjen, (1979). "Computers and Profits." Addison-Wesley, Reading, Massachusetts.
King, J. L., and Kraemer, K. L. (1981). Cost as a social impact of computing. *In* "Telecommunications and Productivity" (Mitchell Moss, ed.). Addison-Wesley, Reading, Massachusetts.
Kling, R. (1978a). Information systems and policymaking: Computer technology and organizational arrangements. *Telecommun. Policy* **2**, 22–32.
Kling, R. (1978b). Automated welfare client tracking and service integration: The political economy of computing. *Commun. ACM* **21**(6), 484–493.
Kling, R. (1978c). Value conflicts and social choice in electronic fund transfer system development. *Commun. ACM* **21**, 642–656.
Kling, R. (1978d). Information systems as social resources in policy-making. *Proc. Natl. ACM Conf., 1978*, 666–674. Washington, D.C.
Kling, R. (1980). Social analyses of computing: Theoretical perspectives in recent empirical research. *ACM Comput. Surv.* **12**(1), 61–103.
Kling, R. (1981). "An Empirical Approach to Studying the Citizen-Orientation of Automated Information Systems." Department of Information and Computer Science, University of California, Irvine. (Unpublished.)
Kling, R. (1982). "Visible Opportunities and Hidden Constraints: Engagements with Computing on a Social Terrain." Department of Information and Computer Science, University of California, Irvine. (Unpublished.)
Kling, R., and Dutton, W. (1982). The dynamics of the local computing package. *In* "Computers and Politics: High Technology in American Local Governments" (J. Danziger *et al.*, eds.), pp. 22–50. Columbia Univ. Press, New York.
Kling, R., and Gerson, E. M. (1977). The social dynamics of technical innovation in the computing world. *Symbolic Interact.* **1**(1), 132–146.
Kling, R., and Gerson, E. M. (1978). Patterns of segmentation and intersection in the computing world. *Symbolic Interact.* **1**(2), 24–43.
Kling, R., and Scacchi, W. (1979a). Recurrent dilemmas of computer use in complex organizations. *Proc. Natl. Comput. Conf., New York, 1979* **48**, 107–116.
Kling, R., and Scacchi, W. (1979b). The DoD common high-order programming language effort (DoD-1): What will the impacts be? *SIGPLAN Not.* **14**(2), 29–41.
Kling, R., and Scacchi, W. (1980). Computing as social action: The social dynamics of computing in complex organizations. *In* "Advances in Computers," Vol. 19. Academic Press, New York.
Kochar, A. K. (1979). "Development of Computer-Based Production Systems." Arnold, London.
Kraemer, K. L., and King, J. L. (1981a). "Comparative Study of Computing Policies and Impacts in Cities." University of California, Irvine. (Unpublished.)
Kraemer, K. L., and King, J. L. (1981b). Computer technology in local governments in the 1980's: A U.S. forecast. *Int. Rev. Admin. Sci.* **47**(2), 155–125.

Laudon, K. (1974). "Computers and Bureaucratic Reform." Wiley (Interscience), New York.
Laudon, K. (1980). Privacy and federal data banks. *Society* **17**(2), 50–56.
Long, N. (1958). The local community as an ecology of games. *Am. J. Sociol.* **44**, 251–261.
Mader, C., and Hagin, R. (1974). "Information Systems: Technology, Economics, and Applications." Science Research Associates, Palo Alto, California.
Markus, M. L. (1979). "Understanding Information System Use in Organizations: A Theoretical Explanation." Ph.D. Dissertation, Department of Organizational Behavior, Case Western Reserve University, Cleveland, Ohio.
Mumford, E., and Pettigrew, A. M. (1976). "Implementing Strategic Decisions." Longman, New York.
Palme, J. (1978). How I fought with hardware and software and succeeded. *Software—Pract. Exper.* **8**, 77–83.
Perrow, C. (1979). "Complex Organizations: A Critical Essay" (2nd. ed.). Scott-Foresman and Co., Glenview, Illinois.
Perry, J. L., and Danziger, J. N. (1980). The adoptability of innovations. *Admin. Soc.* **11**(4), 461–492.
Perry, J. L., and Kraemer, K. L. (1978). Innovation attributes, policy intervention, and the diffusion of computer applications among local governments. *Policy Sci.* **9**, 179–205.
Pettigrew, A. (1973). "The Politics of Organizational Decision-Making." Tavistock, London.
Pfeffer, J. (1981). "Power in Organizations." Marshfield Publ. Co., Pittman, Massachusetts.
Podell, H., and Weiss, M. (eds.) (1980). "Introduction to Business Data Processing." Computer Society Press, Silver Springs, Maryland.
Riegel, R., Miller, J. S., and Williams, C. A. (1976). "Insurance Principles and Practices: Property and Liability" (6th ed.). Prentice-Hall, Englewood Cliffs, New Jersey.
Rosenberg, R. L. (1980). "Incomprehensible Computer Systems: Knowledge Without Wisdom." Rept. No. MIT/LCS/TR-227. Massachusetts Institute of Technology, Cambridge, Massachusetts.
Rule, J. (1974). "Private Lives and Public Surveillance: Social Control in the Computer Age." Prentice-Hall, Englewood Cliffs, New Jersey.
Sandewall, E. (1978). Programming in an interactive environment: The "LISP" experience. *ACM Comput. Surv.* **10**(1), 35–71.
Scacchi, W. (1981). "The Process of Innovation in Computing: A Study of the Social Dynamics of Computing" Ph.D. Dissertation, Department of Information and Computer Science, University of California, Irvine.
Selznik, P. (1957). "Leadership in Administration." Harper and Row, New York.
Simon, H. A. (1973). Applying information technology to organizational design. *Publ. Admin. Rev.* **33**, 268–278.
Simon, H. A. (1977). What computers mean for man and society. *Science* **195**(4283), 1186–1191.
Sterling, T. D. (1979). Consumer difficulties with computerized transactions: An empirical analysis. *Commun. ACM* **22**(5), 283–289.
Sterling, T. D. (1980). Computer ombudsman. *Society* **17**(2), 31–35.
Stewart, F. (1977). "Technology and Underdevelopment." Macmillan, New York.
Strauss, A. (1978a). "Negotiations: Varieties, Contexts, Processes, and Social Order." Jossey-Bass, San Francisco, California.
Strauss, A. (1978b). A social world perspective. *In* "Studies in Symbolic Interaction" (N. Denzin, ed.), Vol. 1, pp. 99–128. JAI Press, Chicago, Illinois.

Suchman, L. (1980). "Office Work as Practical Action" (working paper). Xerox Palo Alto Research Center, Palo Alto, California.
Taggart, W. M., Jr. (1980). "Information Systems: An Introduction to Computers in Organizations." Allyn and Bacon, Boston.
Teitleman, W. (1980). "INTERLISP Reference Manual." Xerox Palo Alto Research Center, Palo Alto, California.
Thompson, J. D. (1967). "Organizations In Action." McGraw-Hill, New York.
Wise, K., Chen, K., and Yokely, R. (1980). "Microcomputers: A Technology Forecast and Assessment to the Year 2000." Wiley (Interscience), New York.
Withington, F. G. (1979). "The Environment for Systems Programs." Addison-Wesley, Reading, Massachusetts.